UNIVERSIDAD HISPANOAMERICANA
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA INFORMÁTICA
SEDE HEREDIA

TÍTUTO:

MICROARQUITECTURA & DESARROLLO: UN PROYECTO DE EXPLORACIÓN PARA EL MONITOREO DE LAS UNIDADES FUNCIONALES DEL PROCESADOR DURANTE LA EJECUCIÓN DE APLICACIONES DE USUARIO, 2019.

POSTULANTE:

ELISA M. ARGUETA RUIZ

COMITÉ ASESOR:

AGUSTÍN FRANCESA ALFARO (TUTOR)
JORGE IVAN ARROYO TORRES (LECTOR)

2019

DECLARACIÓN JURADA

Yo Elisa María Argueta Ruiz, mayor de edad, portadora de la cédula de identidad número 402080404, egresado de la carrera de Bachillerato en Ingeniería Informática de la Universidad Hispanoamericana, hago constar por medio de éste acto y debidamente apercibido y entendido de las penas y consecuencias con las que se castiga en el Código Penal el delito de perjurio, ante quienes se constituyen en el Tribunal Examinador de mi trabajo de tesis para optar por el título de Bachillerato en Ingeniería Informática, juro solemnemente que mi trabajo de investigación titulado: **Microarquitectura & Desarrollo: Un proyecto de exploración para el monitoreo de las unidades funcionales del microprocesador durante la ejecución aplicaciones de usuario**, es una obra original que ha respetado todo lo preceptuado por las Leyes Penales, así como la Ley de Derecho de Autor y Derecho Conexos número 6683 del 14 de octubre de 1982 y sus reformas, publicada en la Gaceta número 226 del 25 de noviembre de 1982; incluyendo el numeral 70 de dicha ley que advierte; articulo 70. Es permitido citar a un autor, transcribiendo los pasajes pertinentes siempre que éstos no sean tantos y seguidos, que puedan considerarse como una producción simulada y sustancial, que redunde en perjuicio del autor de la obra original. Asimismo, quedo advertido que la Universidad se reserva el derecho de protocolizar este documento ante Notario Público. en fe de lo anterior, firmo en la ciudad de Heredia, a los veinticinco días del mes de julio del año dos mil diecinueve.

Elisa Argueta

Firma de la estudiante

Cédula: 402080404

CARTA DEL TUTOR

Cartago, 24 de julio de 2019

Sra. María Isabel Losilla Barrientos
Directora de carrera
Ingeniería en Informática
Universidad Hispanoamericana

Estimada señora:

La estudiante Elisa Argueta Ruiz, cédula de identidad número 402080404, me ha presentado, para efectos de revisión y aprobación, el trabajo de investigación denominado ***Microarquitectura & Desarrollo: un proyecto de exploración para el monitoreo de las unidades funcionales del procesador durante la ejecución de aplicaciones de usuario,*** el cual ha elaborado para optar por el grado académico de Bachillerato en Ingeniería Informática.
En mi calidad de tutor, he verificado que se han hecho las correcciones indicadas durante el proceso de tutoría y he evaluado los aspectos relativos a la elaboración del problema, objetivos, justificación; antecedentes, marco teórico, marco metodológico, tabulación, análisis de datos; conclusiones y recomendaciones.

De los resultados obtenidos por el postulante, se obtiene la siguiente calificación:

a)	ORIGINAL DEL TEMA	10%	10
b)	CUMPLIMIENTO DE ENTREGA DE AVANCES	20%	20
C)	COHERENCIA ENTRE LOS OBJETIVOS, LOS INSTRUMENTOS APLICADOS Y LOS RESULTADOS DE LA INVESTIGACION	30%	30
d)	RELEVANCIA DE LAS CONCLUSIONES Y RECOMENDACIONES	20%	20
e)	CALIDAD, DETALLE DEL MARCO TEORICO	20%	20
	TOTAL		100

En virtud de la calificación obtenida, se avala el traslado al proceso de lectura.

Atentamente,

Ing. Agustín Francesa Alfaro
113470859
CPIC #6233

CARTA DE LECTOR

San José, 23 de septiembre de 2019

Universidad Hispanoamericana
Sede Heredia
Carrera de Ingeniería Informática

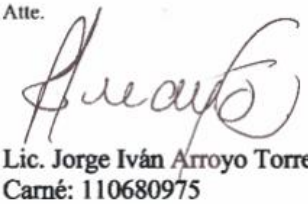
Estimado señor

La estudiante ELISA M. ARGUETA RUIZ, cédula de identidad 4-0208-0404, me ha presentado para efectos de revisión y aprobación, el trabajo de investigación denominado " MICROARQUITECTURA & DESARROLLO: UN PROYECTO DE EXPLORACIÓN PARA EL MONITOREO DE LAS UNIDADES FUNCIONALES DEL PROCESADOR DURANTE LA EJECUCIÓN DE APLICACIONES DE USUARIO, 2019", el cual ha elaborado para obtener su grado de BACHILLERATO EN LA CARRERA DE INGENIERÍA INFORMÁTICA.

He revisado y he hecho las observaciones relativas al contenido analizado, particularmente lo relativo a la coherencia entre el marco teórico y análisis de datos, la consistencia de los datos recopilados y la coherencia entre éstos y las conclusiones; asimismo, la aplicabilidad y originalidad de las recomendaciones, en términos de aporte de la investigación. He verificado que se han hecho las modificaciones correspondientes a las observaciones indicadas.

Por consiguiente, este trabajo cuenta con mi aval para ser presentado en la defensa pública.

Atte.



Lic. Jorge Iván Arroyo Torres.
Carné: 110680975

Cartago, 01 de octubre de 2019

Señores (as):
Universidad Hispanoamericana

Estimados señores (as):

Yo, María Fernanda Sanabria Coto, cédula de identidad 1-1429-0780, bachiller en Filología española, perteneciente a la Asociación Costarricense de Filólogos, carné 225 y al Colegio de Licenciados y Profesores en Letras, Filosofía, Ciencias y Artes de Costa Rica, código 75402, hago constar que he revisado el proyecto titulado:

Microarquitectura & Desarrollo: un proyecto de exploración para el monitoreo de las unidades funcionales del procesador durante la ejecución de aplicaciones de usuario, 2019.

Dicho documento fue elaborado por Elisa Argueta Ruiz, cédula de identidad 4-0208-0404. El proyecto fue realizado con el fin de optar al grado de Bachillerato en Ingeniería Informática. He revisado y corregido aspectos tales como construcción de párrafos, vicios del lenguaje trasladados a lo escrito, ortografía, puntuación y otros relacionados con el campo filológico. Por lo tanto, con los cambios aplicados, considero que está listo para ser presentado.

Atentamente,

Fernanda S. Coto.

María Fernanda Sanabria Coto
Asociación Costarricense de Filólogos. Carné nro. 225
Colypro. Código 75402
fernanda.sanabria@filologos.cr

UNIVERSIDAD HISPANOAMERICANA
Entregado por:
Recibido por: Jessica
Fecha: 30/09/2019

UNIVERSIDAD HISPANOAMERICANA
CENTRO DE INFORMACION TECNOLOGICO (CENIT)
CARTA DE AUTORIZACIÓN DE LA AUTORA PARA LA CONSULTA, LA REPRODUCCION PARCIAL O TOTAL Y PUBLICACIÓN ELECTRÓNICA DE LOS TRABAJOS FINALES DE GRADUACION

Heredia, 05 noviembre 2019

Señores:
Universidad Hispanoamericana
Centro de Información Tecnológico (CENIT)

Estimados Señores:

El suscrito (a) Elisa María Argueta Ruiz con número de identificación 402080404 autor (a) del trabajo de graduación titulado **Microarquitectura & Desarrollo: un proyecto de exploración para el monitoreo de las unidades funcionales del procesador durante la ejecución de aplicaciones de usuario** presentado y aprobado en el año 2019 como requisito para optar por el título de Bachillerato en Ingeniería Informática; SI autorizo al Centro de Información Tecnológico (CENIT) para que con fines académicos, muestre a la comunidad universitaria la producción intelectual contenida en este documento.

De conformidad con lo establecido en la Ley sobre Derechos de Autor y Derechos Conexos N° 6683, Asamblea Legislativa de la República de Costa Rica

Cordialmente,

Elisa Argueta

Firma y Documento de Identidad

402080404

RESUMEN.

La mejoras en el rendimiento de los procesadores son una preocupación constante para Intel ya que el rendimiento no sólo depende de las características del procesador sino también del software que explota sus recursos. El proyecto nace como una iniciativa para conocer la forma en que los recursos del procesador son ejercitados desde las aplicaciones de usuario por lo que se propone *Desarrollar un prototipo de software capaz de monitorear de forma no intrusiva la manera en que los recursos del procesador Intel core i7 para computadoras de uso general son ejercitados.*

El problema planteado se resuelve al dividir el proyecto en seis etapas. La selección de las etapas se basó en el enfoque de caracterización de sistemas computacionales propuesto por Jain, R. (1991), entre ellas la definición del problema, selección de métricas, discusión sobre parámetros y factores, diseño de experimento, análisis e interpretación de resultados y presentación de resultados del experimento. En el caso del proyecto, la investigación preliminar permitió delimitar los alcances, las fases de prueba y visualización de los resultados obtenidos.

Como parte de los resultados obtenidos, se demuestra por ejemplo que en de las aplicaciones de *Open Office* se da cuenta de que, en el caso del procesador en estudio hay más de un 40% de instrucciones retiradas, denotando una utilización casi óptima del sistema de segmentación utilizado. Además, se concluye que los códigos que soportan las aplicaciones son sumamente largos, por lo que saturan el secuenciador de microcódigo del procesador.

En los próximos capítulos se documentan las etapas recurridas para la construcción del prototipo.

Palabras clave: procesador, análisis de rendimiento, métricas de rendimiento, arquitectura de computadoras.

INTRODUCCIÓN.

En el siguiente proyecto se desarrolla un prototipo funcional para Intel Costa Rica que permite monitorear las unidades funcionales de un procesador de propósito general. Este prototipo se enmarca en los esfuerzos de mejora continua de Intel, por medio de los cuales la empresa se plantea la reducción de costos y optimización de procesos para la normalización del éxito en los proyectos emprendidos. Entre las acciones de mejora continua en Intel se incluye el diagnóstico de posibles eventos que obstaculicen en alguna medida la normalización del éxito, como lo son pocas herramientas o automatizaciones especializadas, escasez de tiempo del personal para realizar actividades de desarrollo complementarias, limitada socialización sobre estándares de arquitectura para la construcción y migración de las piezas de software desarrolladas y carencia de documentación del software especializado existente para su perfeccionamiento e integración al desarrollo de otras piezas de software.

Solucionar estos puntos es prioritario para la empresa cuando se analiza desde la óptica del monitoreo de las unidades de microarquitectura ya que las herramientas auxiliares para análisis de rendimiento permiten conocer la forma en que los programas de usuario hacen uso de todas las unidades funcionales del procesador, lo que repercute necesaria y directamente en el desempeño de los procesadores. El proyecto realizado contribuye entonces a la solución de estos puntos al plantear el desarrollo de un prototipo de software capaz de monitorear la manera en que los recursos de los procesadores Intel para computadoras de uso general están siendo ejercitados.

Para el desarrollo del prototipo se aplican conocimientos sobre principios de arquitectura de computadoras, arquitectura de los procesadores Intel, técnicas para mejorar el rendimiento de los procesadores contemporáneos, relación del procesador con otros componentes, nociones acerca de CISC, RISC, el ciclo de vida de las instrucciones, registros, procesos de segmentación, selección de técnicas y métricas en relación con análisis de rendimiento, tipos de monitores existentes, ciclo de vida del software y modelos de desarrollo.

Asimismo, se implementa una metodología inspirada en las técnicas de caracterización planteada por Jain, R. (1991), dicha metodología culmina con la definición de seis etapas:

i. Etapa I: Definición del problema. En esta primera etapa se definen los requerimientos funcionales y no funcionales.

ii. Etapa II: Selección de métricas de rendimiento. En esta etapa se define la herramienta de monitoreo a bajo nivel que se utilizará, unidades que se monitorearán y métricas asociadas a esas unidades.

iii. Etapa III: Discusión sobre parámetros y factores. En esta etapa se discute la relación entre parámetros, factores y métricas.

iv. Etapa IV: Diseño de experimento. Se elaboran diagramas UML del prototipo en desarrollo.

v. Etapa V: Análisis e interpretación de datos. Se describen las pruebas y los análisis sobre las pruebas realizadas.

vi. Etapa IV: Presentación de resultados. Se describen las decisiones tomadas en cuanto a la presentación de las síntesis de resultados.

ÍNDICE DE CONTENIDO

RESUMEN. 8
INTRODUCCIÓN. 9
ÍNDICE DE CONTENIDO 11
ÍNDICE DE ILUSTRACIONES. 15
ÍNDICE DE TABLAS 19
CAPÍTULO I: PLANTEAMIENTO DEL PROBLEMA 23
1.1 Antecedentes y Justificación del proyecto. 25
1.1.1 Marco de Referencia Empresarial y Contextual 25
1.1.2 Justificación del proyecto 31
1.2 Definición del problema 33
1.2.1 Problemática 33
1.2.2 Problema general 34
1.2.3 Problemas específicos 34
1.3 Objetivos del proyecto 35
1.3.1 Objetivo General 35
1.3.2. Objetivos específicos. 35
1.4. Alcances y limitaciones. 37
1.4.1 Alcances 37
1.4.2. Limitaciones. 38
1.5 . Cronograma de trabajo. 39
1.5.1 Resumen de cronograma de trabajo de proyecto. 39
CAPÍTULO II: MARCO TEÓRICO 41
2.1 Generalidades. 43
2.1.1 Principios de arquitectura de computadoras. 43
2.1.2. Un acercamiento a los procesadores Intel. 47
2.2. Los microprocesadores y la competencia por el rendimiento. 51
2.2.1. La ley de Moore. 51
2.2.2. Técnicas para mejorar rendimiento en procesadores contemporáneos. 54
2.2.3. Afectaciones en el rendimiento del procesador en relación con sus componentes. 55
2.3. Contexto del procesador. 57

2.3.1. Generalidades sobre los sets de instrucciones. 58
2.3.2. El dilema entre CISC vs RISC. 59
2.3.3. Una vista al ciclo de una instrucción. 59
2.3.4. Otras posibilidades para el flujo de instrucciones y datos. 62
2.4. Estructura y funcionamiento de un procesador. 67
2.4.1. Registros visibles para el usuario. 70
2.4.2. Registros de estado y control. 71
2.4.3. Una mirada más profunda a los procesos de segmentación en los procesadores. 73
2.5. Análisis de rendimiento. 78
2.5.1. Aspectos importantes en la selección de métricas y técnicas. 82
2.5.2. Consideraciones finales en relación con los análisis de rendimiento. 88
2.6. Técnicas de caracterización por cargas de trabajo. 90
2.7. Consideraciones sobre el tipo de análisis que se realizará. 91
2.8. Notas sobre los monitores de tareas. 93
2.9. Clasificación de los monitores existentes 94
2.9.1. Monitores de software. 94
2.9.2. Monitores de hardware. 95
2.9.3. Firmware y monitores híbridos. 95
2.10. Ciclo de vida del software y modelos de desarrollo. 96
2.10.1. Modelo en cascada. 96
2.10.2. Prototipado evolutivo. 97
2.10.3. Modelo en espiral. 98
2.10.4. Modelo orientado a objetos. 98
2.11. Relación entre microarquitectura, análisis de rendimiento, monitoreo y software. 98
CAPÍTULO III: MARCO METODOLÓGICO. 101
3.1. Tipo de investigación. 103
3.2. Enfoque de investigación. 105
3.2. Fuentes y sujetos de información. 107
3.2.1. Fuentes de información 107
3.2.2. Sujetos de información. 109
3.3. Técnicas y herramientas de recolección de datos. 109
3.3.1. Técnicas de recolección de datos. 110
3.3.2. Herramientas de recolección de datos. 113

3.4. Variables de investigación. 118
3.5. Diseño de la investigación. 120
3.5.1. Etapa I: Definición del problema. 120
3.5.2. Etapa II: Selección de métricas de rendimiento. 121
3.5.3. Etapa III: Discusión sobre parámetros y factores. 122
3.5.4. Etapa IV: Diseño de prototipo. 122
3.5.5. Etapa V: Análisis e interpretación. 122
3.5.6. Etapa VI: Presentación de resultados. 123
3.6. Matriz de coherencia. 123
CAPÍTULO IV: DIAGNÓSTICO DE LA SITUACIÓN ACTUAL. 127
4.1. Visión, misión, valores y estrategias empresariales. 129
4.2. Diagnóstico administrativo. 130
4.2. Diagnóstico técnico. 133
4.4. Diagnóstico de percepción. 135
4.3. Consideraciones finales del diagnóstico. 140
4.3.1. Proyecto en relación con principios empresariales. 140
4.3.2. La propuesta técnica de Intel en relación con la propuesta de proyecto. 140
4.3.3. Recomendaciones a partir de diagnóstico de percepción. 141
CAPÍTULO V: PROPUESTA DEL PROYECTO. 143
5.1. Etapa I: Definición del problema. 145
5.1.1. Requerimientos funcionales. 146
5.1.2. Requerimientos no funcionales. 151
5.1.3. Logros obtenidos durante la Etapa I del proyecto. 154
5.2. Etapa II: Selección de métricas. 154
5.2.1. Un acercamiento al rendimiento a través de las métricas. 154
5.2.2. Anotaciones sobre VTUNE Amplifier. 155
5.2.3. Instalación y ejecución de la herramienta seleccionada. 157
5.2.4. Detección y selección de unidades de microarquitectura del procesador seleccionado. 158
5.2.5. Logros obtenidos durante la etapa II del proyecto. 164
5.3. Etapa III: Una discusión sobre parámetros y factores. 164
5.3.1. Relación entre parámetros, factores y métricas. 165
5.3.2. Logros obtenidos en la etapa III del proyecto. 166
5.4. Etapa IV: Diseño del prototipo. 166

5.4.1. Casos de uso. 167
5.4.2. Diagramas de secuencia. 173
5.4.3. Diagrama de clases. 178
5.4.4. Diagrama de colaboración. 179
5.4.5. Diagramas de actividad. 182
5.4.6. Plan de pruebas. 186
5.4.7. Logros obtenidos durante la etapa IV del proyecto. 194
5.5. Etapa V: Análisis e interpretación. 195
5.5.1. Descripción de pruebas preliminares realizadas. 196
5.5.2. Análisis de las pruebas preliminares realizadas. 196
5.5.3. Logros obtenidos durante la etapa V del proyecto. 199
5.6. Etapa VI: Presentación de resultados. 200
5.6.1. Acerca de la sistematización de los reportes para su visualización. 200
5.6.2. Resultados del plan de pruebas. 201
5.6.3. Logros obtenidos durante la etapa VI del proyecto. 202
CAPÍTULO VI: CONCLUSIONES Y RECOMENDACIONES DEL PROYECTO. 204
6.1. Conclusiones. 206
6.2. Recomendaciones. 207
REFERENCIAS BIBLIOGRÁFICAS. 210
GLOSARIO. 216
ANEXOS. 218
Anexo 1: Cronograma detallado. 220
Anexo 2: Detalle de comparación para métricas de rendimiento. 232
Anexo 3: Unidades del procesador y su relación con métricas de rendimiento. 234
Anexo 4: Descripción de métricas de rendimiento seleccionadas para la elaboración del prototipo. 236
Anexo 5: Información del sistema destino. 240
Anexo 6: Descripción de las pruebas preliminares. 242
Anexo 7: Precondiciones y utilización del prototipo realizado. 248
Precondiciones para el uso. 248
Utilización del prototipo. 248
Anexo 8: Diagrama causa-efecto. 260
Anexo 9: Información sobre sistema destino. 261

Anexo 10: Pruebas de aceptación. 262

ÍNDICE DE ILUSTRACIONES.

ILUSTRACIÓN 1. COMPUTER. TOP LEVEL STRUCTURE. EN COMPUTER ORGANIZATION AND ARCHITECTURE (P.5), POR STALLINGS, W, 2010,CONNECTICUT, NEW ENGLAND: PEARSON EDUCATION. DERECHOS DE AUTOR [2010] POR PEARSON. REIMPRESIÓN AUTORIZADA. 44
ILUSTRACIÓN 2.SIMPLIFIED VIEW OF MAJOR ELEMENTS OF A MULTICORE COMPUTER. EN COMPUTER ORGANIZATION AND ARCHITECTURE (P.7), POR STALLINGS, W, 2010,CONNECTICUT, NEW ENGLAND: PEARSON EDUCATION. DERECHOS DE AUTOR [2010] POR PEARSON. REIMPRESIÓN AUTORIZADA. 46
ILUSTRACIÓN 3. SEMICONDUCTOR RENEW. EN FIFTYYEARS OF MOORE ' S LAW (P.205), POR A. MACK, C, 2011, NEW JERSEY,EEUU: IEEE. DERECHOS DE AUTOR [2011] POR IEEE. REIMPRESIÓN AUTORIZADA..... 53
ILUSTRACIÓN 4. LEY DE OHM. EN PRINCIPIOS DE CIRCUITOS ELÉCTRICOS (P.73), POR L. FLOYD, T., 2008, MÉXICO: PEARSON. DERECHOS DE AUTOR [2008] POR PEARSON. REIMPRESIÓN AUTORIZADA..... 57
ILUSTRACIÓN 5. PROCESSOR TRENDS. EN COMPUTER ORGANIZATION AND ARCHITECTURE (P.51), POR STALLINGS, W., 2010,CONNECTICUT, NEW ENGLAND: PEARSON EDUCATION. DERECHOS DE AUTOR [2010] POR PEARSON. REIMPRESIÓN AUTORIZADA. 57
ILUSTRACIÓN 6.COMPUTER. TRANSFER OF CONTROL VIA INTERRUPT. EN COMPUTER ORGANIZATION AND ARCHITECTURE (P.92), POR STALLINGS, W, 2010,CONNECTICUT, NEW ENGLAND: PEARSON EDUCATION. DERECHOS DE AUTOR [2010] POR PEARSON. REIMPRESIÓN AUTORIZADA..... 60
ILUSTRACIÓN 7. INSTRUCTION CYCLE WITH INTERRUPTS. EN COMPUTER ORGANIZATION AND ARCHITECTURE (P.92), POR STALLINGS, W, 2010,CONNECTICUT, NEW ENGLAND: PEARSON EDUCATION. DERECHOS DE AUTOR [2010] POR PEARSON. REIMPRESIÓN AUTORIZADA. 61
ILUSTRACIÓN 8. BUS INTERCONECTION SCHEME. EN COMPUTER ORGANIZATION AND ARCHITECTURE (P.101), POR STALLINGS, W, 2010, CONNECTICUT, NEW ENGLAND: PEARSON EDUCATION. DERECHOS DE AUTOR [2010] POR PEARSON. REIMPRESIÓN AUTORIZADA. 63
ILUSTRACIÓN 9. COMPUTER HARDWARE AND SOFTWARE STRUCTURE. EN COMPUTER ORGANIZATION AND ARCHITECTURE (P.277), POR STALLINGD, W, 2010, CONNECTICUT, NEW ENGLAND: PEARSON EDUCATION. DERECHOS DE AUTOR [2010] POR PEARSON. REIMPRESIÓN AUTORIZADA..... 65
ILUSTRACIÓN 10. THE MEMORY HIERARCHY. EN COMPUTER ORGANIZATION AND ARCHITECTURE (P.125), POR STALLINGS, W, 2010, CONNECTICUT, NEW ENGLAND: PEARSON EDUCATION. DERECHOS DE AUTOR [2010] POR PEARSON. REIMPRESIÓN AUTORIZADA. 67

ILUSTRACIÓN 11. THE CPU WITH THE SYSTEM BUS. EN COMPUTER ORGANIZATION AND ARCHITECTURE (P.490), POR STALLINGS, W, 2010, CONNECTICUT, NEW ENGLAND: PEARSON EDUCATION. DERECHOS DE AUTOR [2010] POR PEARSON. REIMPRESIÓN AUTORIZADA. 68
ILUSTRACIÓN 12.INTERNAL STRUCTURE OF THE CPU. THE CPU WITH THE SYSTEM BUS. EN COMPUTER ORGANIZATION AND ARCHITECTURE (P.491), POR STALLINGS, W, 2010, CONNECTICUT, NEW ENGLAND: PEARSON EDUCATION. DERECHOS DE AUTOR [2010] POR PEARSON. REIMPRESIÓN AUTORIZADA..... 69
ILUSTRACIÓN 13. THE INSTRUCTION CYCLE. THE CPU WITH THE SYSTEM BUS. EN COMPUTER ORGANIZATION AND ARCHITECTURE (P.497), POR STALLINGS, W, 2010, CONNECTICUT, NEW ENGLAND: PEARSON EDUCATION. DERECHOS DE AUTOR [2010] POR PEARSON. REIMPRESIÓN AUTORIZADA..... 72
ILUSTRACIÓN 14. DATA FLOW, FETCH CYCLE. THE CPU WITH THE SYSTEM BUS. EN COMPUTER ORGANIZATION AND ARCHITECTURE (P.498), POR STALLINGS, W, 2010, CONNECTICUT, NEW ENGLAND: PEARSON EDUCATION. DERECHOS DE AUTOR [2010] POR PEARSON. REIMPRESIÓN AUTORIZADA..... 73
ILUSTRACIÓN 15. TIMING DIAGRAM FOR INSDTRUCTION PIPELINE OPERATION. EN COMPUTER ORGANIZATION AND ARCHITECTURE (P.502), POR STALLINGS, W, 2010, CONNECTICUT, NEW ENGLAND: PEARSON EDUCATION. DERECHOS DE AUTOR [2010] POR PEARSON. REIMPRESIÓN AUTORIZADA..... 74
ILUSTRACIÓN 16. THE EFFECT OF A CONDITIONAL BRANCH ON INSTRUCTION PIPELINE OPERATION. EN COMPUTER ORGANIZATION AND ARCHITECTURE (P.503), POR STALLINGS, W, 2010, CONNECTICUT, NEW ENGLAND: PEARSON EDUCATION. DERECHOS DE AUTOR [2010] POR PEARSON. REIMPRESIÓN AUTORIZADA..... 76
ILUSTRACIÓN 17. THE POSSIBLE OUTCOMES OF A SERVICE REQUEST. EN THE ART OF COMPUTER SYSTEMS PERFORMANCE ANALYSIS (P.33), POR JAIN, R., 1991, NEW YORK: WILLEY. DERECHOS DE AUTOR [1991] POR WILEY. REIMPRESIÓN AUTORIZADA. 84
ILUSTRACIÓN 18. SYNTHETIC WORLOAD GENERATION PROGRAM. EN THE ART OF COMPUTER SYSTEMS PERFORMANCE ANALYSIS (P.51), POR JAIN, R., 1991, NEW YORK: WILLEY. DERECHOS DE AUTOR [1991] POR WILEY. REIMPRESIÓN AUTORIZADA. 87
ILUSTRACIÓN 19. THE SUT AND CUS. EN THE ART OF COMPUTER SYSTEMS PERFORMANCE ANALYSIS (P.61), POR JAIN, R., 1991, NEW YORK: WILLEY. DERECHOS DE AUTOR [1991] POR WILEY. REIMPRESIÓN AUTORIZADA..... 88
ILUSTRACIÓN 20.PIPELINE SLOT CATEGORIZATION. EN TOP-DOWN MICROARCHITECTURE ANALYSIS METHOD (P.4), POR MARUSARZ, J. & RYABTSEV, D., 2019,EEUU: INTEL. DERECHOS DE AUTOR [20109] POR INTEL. REIMPRESIÓN AUTORIZADA..... 92
ILUSTRACIÓN 21. PROCESO DE PRODUCCIÓN DE CONOCIMIENTO. EN INVESTIGACIÓN APLICADA. (P.35), POR LOZADA,J., 2014, ECUADOR: CENTRO DE INVESTIGACIÓN EN MECATRÓNICA. DERECHOS DE AUTOR [2014] POR UNIVERSIDAD TECNOLÓGICA INDOAMÉRICA. REIMPRESIÓN AUTORIZADA..... 104
ILUSTRACIÓN 22. ETIQUETAS INTEL PROCESADORES GENERACIONES NEOSTUFF. EN DIFERENCIAS ENTRE GENERACIONES DE PROCESADORES INTEL (P.01), POR CARVAJAL,

H., 2018, MÉXICO: NEOSTUFF. DERECHOS DE AUTOR [2018] POR NEOSTUFF. REIMPRESIÓN AUTORIZADA..... 108
ILUSTRACIÓN 23. FASES DE LA ENTREVISTA. EN LA ENTREVISTA, RECURSO FLEXIBLE Y DINÁMICO (P.164), POR L. DÍAS, B.; TORRUNCO G.,M.; MARTÍNEZ H. ET AL, 2013, MÉXICO D.F.: INVESTIGACIÓN EN EDUCACIÓN MÉDICA. DERECHOS DE AUTOR [2013] POR REVISTA DE EDUCACIÓN MÉDICA. REIMPRESIÓN AUTORIZADA. 112
ILUSTRACIÓN 24. INTEL ENTERPRISE DIGITAL TRANSFORMATION UNDERWAY. EN DRIVING THE DIGITAL ENTERPRISE TRANSFORMATION (P.4), POR INTEL., 2019, EEUU: INTEL. DERECHOS DE AUTOR [2019] POR INTEL. REIMPRESIÓN AUTORIZADA..... 133
ILUSTRACIÓN 25. OPERATING MODEL. EN DRIVING THE DIGITAL ENTERPRISE TRANSFORMATION (P.5), POR INTEL., 2019, EEUU: INTEL. DERECHOS DE AUTOR [2019] POR INTEL. REIMPRESIÓN AUTORIZADA. 134
ILUSTRACIÓN 26. IDENTIFY AND MANAGE TECHNICAL DEBT. EN DRIVING THE DIGITAL ENTERPRISE TRANSFORMATION (P.6), POR INTEL., 2019, EEUU: INTEL. DERECHOS DE AUTOR [2019] POR INTEL. REIMPRESIÓN AUTORIZADA. 135
ILUSTRACIÓN 27. FRECUENCIA DE UTILIZACIÓN DE MÉTRICAS DE RENDIMIENTO, ELABORACIÓN PROPIA..... 136
ILUSTRACIÓN 28. HERRAMIENTAS MÁS UTILIZADAS PARA OBTENCIÓN DE MÉTRICAS DE RENDIMIENTO, ELABORACIÓN PROPIA. 137
ILUSTRACIÓN 29. FUNCIONALIDAD DE LAS HERRAMIENTAS PARA OBTENCIÓN DE MÉTRICAS DE RENDIMIENTO, ELABORACIÓN PROPIA. 138
ILUSTRACIÓN 30. TIPO DE UTILIZACIÓN DE MÉTRICAS DE RENDIMIENTO, ELABORACIÓN PROPIA. 138
ILUSTRACIÓN 31. MÉTRICAS DE INTERÉS EN TIEMPO REAL, ELABORACIÓN PROPIA..... 139
ILUSTRACIÓN 32.SILICON DIE LAYOUT FOR AN INTEL® CORE™ M PROCESSOR. EN FOR AN INTEL® CORE™ M PROCESSOR (P.2), POR INTEL CORPORATION, 2014, EEUU: INTEL CORPORATION. DERECHOS DE AUTOR [2010] POR INTEL CORPORATION. REIMPRESIÓN AUTORIZADA. 159
ILUSTRACIÓN 33. AN INTEL® CORE™ M PROCESSOR SOC AND ITS RING INTERCONNECT ARCHITECTURE. EN FOR AN INTEL® CORE™ M PROCESSOR (P.3), POR INTEL CORPORATION, 2014, EEUU: INTEL CORPORATION. DERECHOS DE AUTOR [2010] POR INTEL CORPORATION. REIMPRESIÓN AUTORIZADA. 159
ILUSTRACIÓN 34. PRIMER DISEÑO PRELIMINAR DE PROCESADOR, ELABORACIÓN PROPIA.... 160
ILUSTRACIÓN 35. COMPUTER. TOP LEVEL STRUCTURE. EN COMPUTER ORGANIZATION AND ARCHITECTURE (P.10), POR STALLINGS, W, 2010,CONNECTICUT, NEW ENGLAND: PEARSON EDUCATION. DERECHOS DE AUTOR [2010] POR PEARSON. REIMPRESIÓN AUTORIZADA. 161
ILUSTRACIÓN 36. UNIDADES FUNCIONALES PROCESADOR INTEL CORE I7, ELABORACIÓN PROPIA. 163
ILUSTRACIÓN 37. DIAGRAMA DE CASOS DE USO, ELABORACIÓN PROPIA. 167
ILUSTRACIÓN 38. D1, ID DE CASO DE USO: C1, ELABORACIÓN PROPIA. 173
ILUSTRACIÓN 39. D2, ID CASO DE USO: C2.1, ELABORACIÓN PROPIA. 174
ILUSTRACIÓN 40. D3, ID CASO DE USO: C2.2, ELABORACIÓN PROPIA. 175

ILUSTRACIÓN 41. D4, ID CASO DE USO: C3, ELABORACIÓN PROPIA. 176
ILUSTRACIÓN 42. D5, ID CASO DE USO: C4.2, ELABORACIÓN PROPIA. 177
ILUSTRACIÓN 43. DIAGRAMA DE CLASES, ELABORACIÓN PROPIA. 178
ILUSTRACIÓN 44. DIAGRAMA DE COLABORACIÓN 1, ELABORACIÓN PROPIA..... 179
ILUSTRACIÓN 45. DIAGRAMA DE COLABORACIÓN 2, ELABORACIÓN PROPIA..... 179
ILUSTRACIÓN 46. DIAGRAMA DE COLABORACIÓN 4, ELABORACIÓN PROPIA..... 180
ILUSTRACIÓN 47. DIAGRAMA DE COLABORACIÓN 3, ELABORACIÓN PROPIA..... 180
ILUSTRACIÓN 48. DIAGRAMA DE COLABORACIÓN 5, ELABORACIÓN PROPIA..... 181
ILUSTRACIÓN 49. DIAGRAMA DE ACTIVIDAD 1, ID CASO DE USO: C1, ELABORACIÓN PROPIA. 182
ILUSTRACIÓN 50. DIAGRAMA DE ACTIVIDAD 2, ID CASO DE USO: C2, ELABORACIÓN PROPIA. 183
ILUSTRACIÓN 51. DIAGRAMA DE ACTIVIDAD 3, ID CASO DE USO: C3, ELABORACIÓN PROPIA. 184
ILUSTRACIÓN 52. DIAGRAMA DE ACTIVIDAD 4, ID CASO DE USO: C4, ELABORACIÓN PROPIA 185
ILUSTRACIÓN 53. FUNCIONAMIENTO DE MEMORIA CACHÉ. EN ESTRUCTURA DE COMPUTADORES, CAP.6 (P.3), POR UNIVERSIDAD COMPUTENSE DE MADRID, 2019, UCM: CURSO 11-12. DERECHOS DE AUTOR [2019] POR UCM. REIMPRESIÓN AUTORIZADA. 198
ILUSTRACIÓN 54. POBLACIÓN DE CANARIAS POR ISLAS 1-1-2006. EN GRÁFICOS ESTADÍSTICOS (P.3), POR GIL ARMAS & MARTÍN GONZÁLEZ, 2010, INSTITUTO CANARIO DE ESTADÍSTICA. DERECHOS DE AUTOR [2010] POR INSTITUTO CANARIO DE ESTADÍSTICA. REIMPRESIÓN AUTORIZADA. 201
ILUSTRACIÓN 55. DETALLE DE CRONOGRAMA, PARTE 1, ELABORACIÓN PROPIA. 220
ILUSTRACIÓN 56. DETALLE DE CRONOGRAMA, PARTE 2, ELABORACIÓN PROPIA..... 221
ILUSTRACIÓN 57. DETALLE DE CRONOGRAMA, PARTE 3, ELABORACIÓN PROPIA..... 222
ILUSTRACIÓN 58. DETALLE DE CRONOGRAMA, PARTE 4, ELABORACIÓN PROPIA..... 223
ILUSTRACIÓN 59. DETALLE DE CRONOGRAMA, PARTE 5, ELABORACIÓN PROPIA..... 224
ILUSTRACIÓN 60. DETALLE DE CRONOGRAMA, PARTE 6, ELABORACIÓN PROPIA..... 225
ILUSTRACIÓN 61. DETALLE DE CRONOGRAMA, PARTE 7, ELABORACIÓN PROPIA..... 226
ILUSTRACIÓN 62. DETALLE DE CRONOGRAMA, PARTE 8, ELABORACIÓN PROPIA..... 227
ILUSTRACIÓN 63. DETALLE DE CRONOGRAMA, PARTE 9, ELABORACIÓN PROPIA. 228
ILUSTRACIÓN 64. DETALLE DE CRONOGRAMA, PARTE 10, ELABORACIÓN PROPIA..... 229
ILUSTRACIÓN 65. DETALLE DE CRONOGRAMA, PARTE 11, ELABORACIÓN PROPIA..... 230
ILUSTRACIÓN 66. DETALLE DE CRONOGRAMA, PARTE 12, ELABORACIÓN PROPIA..... 231
ILUSTRACIÓN 67. VISTA GENERAL, ELABORACIÓN PROPIA..... 249
ILUSTRACIÓN 68. SELECCIÓN DE LIBREOFFICE, ELABORACIÓN PROPIA. 249
ILUSTRACIÓN 69. INICIAR RECOLECCIÓN LIBREOFFICE, ELABORACIÓN PROPIA. 250
ILUSTRACIÓN 70. INICIANDO RECOLECCIÓN LIBREOFFICE, ELABORACIÓN PROPIA..... 250
ILUSTRACIÓN 71. PESTAÑA NUEVA DE LIBREOFFICE, ELABORACIÓN PROPIA..... 251
ILUSTRACIÓN 72. EJEMPLO DE ACCIONES SOBRE LIBREOFFICE, ELABORACIÓN PROPIA..... 251
ILUSTRACIÓN 73. FINALIZACIÓN DE SESIÓN LIBREOFFICE, ELABORACIÓN PROPIA. 252

Ilustración 74. Mensaje de finalización de recolección en LibreOffice, elaboración propia. 252
Ilustración 75. Inicio de recolección desde Demo, elaboración propia. 253
Ilustración 76. Validación de inicio de recolección desde Demo, elaboración propia. 253
Ilustración 77. Iniciando recolección desde Demo, elaboración propia. 254
Ilustración 78. Finalización de recolección desde Demo, elaboración propia... 254
Ilustración 79. Selección de reportes, elaboración propia. 255
Ilustración 80. Ejemplo de gráfico de barras, elaboración propia. 256
Ilustración 81. Guardar reporte, elaboración propia. 256
Ilustración 82. Pestaña de ayuda, elaboración propia. 257
Ilustración 83. Guía rápida recolección en tiempo real, elaboración propia. ... 258
Ilustración 84. Guía rápida, recolección desde demo, elaboración propia. 258
Ilustración 85. Guía rápida, vista de reportes, elaboración propia. 259

ÍNDICE DE TABLAS

Tabla 1. Resumen de cronograma de trabajo, elaboración propia. 39
Tabla 2. 1970s Processors. En Computer Organization and Architecture (p.26), por Stallings, W, 2010,Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada. 48
Tabla 3. 1980s Processors. En Computer Organization and Architecture (p.26), por Stallings, W, 2010,Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada. 48
abla 4. Recent Processors. En Computer Organization and Architecture (p.27), por Stallings, W, 2010,Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada. 48
Tabla 5. 1990s Processors. En Computer Organization and Architecture (p.26), por Stallings, W, 2010,Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada. 49
Tabla 6. Comparación entre CISC y RISC, elaboración propia. 59
Tabla 7. Ejemplo de Mix de instrucciones de Gibson (elaboración propia)..... 85
Tabla 8. Los procesos del ciclo de vida del software según ISO 12207-1. En Una metodología para el diseño, desarrollo e implementación de software educativo (p.36), por Cataldi, Z., 2000, Uruguay: Universidad Nacional de la Plata. Derechos de autor [2000] por Universidad Nacional de la Plata. Reimpresión autorizada. 96
Tabla 9. Definición de sujetos de información, elaboración propia. 109
Tabla 10. Solución técnica para técnica seleccionada: Análisis de Rendimiento, elaboración propia. 110
Tabla 11. Muestra de cuestionario, elaboración propia. 115
Tabla 12. Ejemplo de guía de entrevista, elaboración propia. 116

TABLA 13. VARIABLES ASOCIADAS A LOS OBJETIVOS, ELABORACIÓN PROPIA. 119
TABLA 14. RELACIÓN ENTRE OBJETIVOS, ENTREGABLES, ETAPAS, TÉCNICAS, HERRAMIENTAS Y TEMAS DESARROLLADOS EN EL MARCO TEÓRICO..... 124
TABLA 15. CÓDIGO DE CONDUCTA DE INTEL, ELABORACIÓN PROPIA. 130
TABLA 16. DESGLOSE DE LA PRIMERA ETAPA DEL PROYECTO, ELABORACIÓN PROPIA. 145
TABLA 17. R1: MOSTRAR ESPECIFICACIONES DEL SISTEMA, ELABORACIÓN PROPIA. 146
TABLA 18. R2: RECOLECCIÓN DE MÉTRICAS, ELABORACIÓN PROPIA. 147
TABLA 19. R3: SISTEMA DESTINO, ELABORACIÓN PROPIA. 147
TABLA 20. R4: REGISTRAR MEDICIONES REALIZADAS, ELABORACIÓN PROPIA. 148
TABLA 21. R5: INICIO Y FINALIZACIÓN DE RECOLECCIÓN DE DATOS, ELABORACIÓN PROPIA. 149
TABLA 22. R6: SÍNTESIS DE RESULTADOS, ELABORACIÓN PROPIA. 150
TABLA 23. R7: GUARDAR SÍNTESIS DE RECOLECCIÓN, ELABORACIÓN PROPIA 150
TABLA 24. R8: PARADIGMA DE PROGRAMACIÓN, ELABORACIÓN PROPIA. 151
TABLA 25. R9: GUÍA Y DOCUMENTACIÓN, ELABORACIÓN PROPIA. 152
TABLA 26. R10: AMBIENTE Y LENGUAJE DE PROGRAMACIÓN, ELABORACIÓN PROPIA. 152
TABLA 27. R12: FORMATO DE REPORTES, ELABORACIÓN PROPIA. 153
TABLA 28. DESGLOSE DE LA SEGUNDA ETAPA DEL PROYECTO, ELABORACIÓN PROPIA. 154
TABLA 29. DESGLOSE DE LA TERCERA ETAPA, ELABORACIÓN PROPIA. 164
TABLA 30. DESGLOSE DE LA CUARTA ETAPA DEL PROYECTO, ELABORACIÓN PROPIA. 166
TABLA 31. C1: SOLICITAR CARGA DE TRABAJO, ELABORACIÓN PROPIA. 168
TABLA 32. C2.1: EJECUTAR CARGA DE TRABAJO DESDE USUARIO, ELABORACIÓN PROPIA.. 168
TABLA 33. C2.2: EJECUTAR CARGA DE TRABAJO SINTÉTICA, ELABORACIÓN PROPIA. 169
TABLA 34. C3: SOLICITAR REPORTE, ELABORACIÓN PROPIA. 170
TABLA 35. C4.1: BUSCAR REPORTE DEL MENÚ DE AYUDA, ELABORACIÓN PROPIA. 171
TABLA 36. C4.2: BUSCAR REPORTE DE CARGA DE TRABAJO DEFINIDA POR EL USUARIO, ELABORACIÓN PROPIA. 172
TABLA 37. P1: PRESENTAR ESPECIFICACIONES DEL SISTEMA, ELABORACIÓN PROPIA. 186
TABLA 38. P2: PROCESAR INFORMACIÓN VTUNE 1, ELABORACIÓN PROPIA. 186
TABLA 39. P3: PROCESAR INFORMACIÓN 2, ELABORACIÓN PROPIA. 187
TABLA 40. P4: PROCESAR INFORMACIÓN VTUNE 3, ELABORACIÓN PROPIA. 187
TABLA 41. P5: PROCESAR INFORMACIÓN VTUNE 4, ELABORACIÓN PROPIA. 188
TABLA 42. P6: PROCESAR INFORMACIÓN VTUNE 5, ELABORACIÓN PROPIA. 189
TABLA 43. P7: PROCESAR INFORMAVIÓN VTUNE 6, ELABORACIÓN PROPIA. 190
TABLA 44. P8: MOSTRAR RESUMEN, ELABORACIÓN PROPIA 191
TABLA 45. P9: DESPLEGAR RETIRING, ELABORACIÓN PROPIA. 191
TABLA 46. P10: DESPLEGAR LÍMITE DE INTERFAZ 192
TABLA 47. P11: MOSTRAR ESPECULACIÓN FALLIDA, ELABORACIÓN PROPIA 192
TABLA 48. P12: DESPLEGAR LÍMITE DE BACKEND, ELABORACIÓN PROPIA. 193
TABLA 49. P13: GUARDAR SÍNTESIS, ELABORACIÓN PROPIA. 193
TABLA 50. TRAZABILIDAD DE CASOS DE PRUEBA, ELABORACIÓN PROPIA. 194
TABLA 51. DESGLOSE DE LA QUINTA ETAPA, ELABORACIÓN PROPIA. 195

TABLA 53. DESGLOSE DE LA SEXTA ETAPA DEL PROYECTO, ELABORACIÓN PROPIA. 200
TABLA 54. COMPARACIÓN DE MÉTRICAS, ELABORACIÓN PROPIA. 232
TABLA 55. CRUCE ENTRE UNIDADES FUNCIONALES Y MÉTRICAS ASOCIADAS, ELABORACIÓN PROPIA. 234
TABLA 56. DESCRIPCIÓN DE MÉTRICAS DE RENDIMIENTO PARA PROYECTO, ELABORACIÓN PROPIA. 236
TABLA 57. INFORMACIÓN SOBRE SISTEMA DESTINO Y PROCESADOR, ELABORACIÓN PROPIA. 240
TABLA 58. DESCRIPCIÓN DE PRUEBAS PRELIMINARES, ELABORACIÓN PROPIA..... 242
TABLA 59. INFORMACIÓN SOBRE CARGA DE TRABAJO SINTETICA, ELABORACION PROPIA... 261
TABLA 60. DESCRIPCIÓN DE PRUEBA REALIZADA 1, ELABORACIÓN PROPIA. 262
TABLA 61. DESCRIPCIÓN DE PRUEBA REALIZADA 2, ELABORACIÓN PROPIA. 262
TABLA 62. DESCRIPCIÓN DE PRUEBA REALIZADA 3, ELABORACIÓN PROPIA. 262
TABLA 63. DESCRIPCIÓN DE PRUEBA REALIZADA 4, ELABORACIÓN PROPIA. 263
TABLA 64. DESCRIPCIÓN DE PRUEBA REALIZADA 5, ELABORACIÓN PROPIA. 263
TABLA 65. DESCRIPCIÓN DE PRUEBA REALIZADA 6, ELABORACIÓN PROPIA. 263
TABLA 66. DESCRIPCIÓN DE PRUEBA REALIZADA 7, ELABORACIÓN PROPIA. 264
TABLA 67. DESCRIPCIÓN DE PRUEBA REALIZADA 8, ELABORACIÓN PROPIA. 264
TABLA 68. DESCRIPCIÓN DE PRUEBA REALIZADA 9, ELABORACIÓN PROPIA. 264
TABLA 69. DESCRIPCIÓN DE PRUEBA REALIZADA 10, ELABORACIÓN PROPIA. 265
TABLA 70. DESCRIPCIÓN DE PRUEBA REALIZADA 11, ELABORACIÓN PROPIA. 265
TABLA 71. DESCRIPCIÓN DE PRUEBA REALIZADA 12, ELABORACIÓN PROPIA. 265
TABLA 72. DESCRIPCIÓN DE PRUEBA REALIZADA 13, ELABORACIÓN PROPIA. 266

CAPÍTULO I: PLANTEAMIENTO DEL PROBLEMA

1.1 Antecedentes y Justificación del proyecto.

El siguiente segmento tiene como finalidad presentar el contexto en el que se llevó a cabo el proyecto, por lo que se exponen los antecedentes y las motivaciones que dieron origen al proyecto.

1.1.1 Marco de Referencia Empresarial y Contextual

El siguiente apartado tiene la finalidad de ofrecer un panorama general de la compañía en la que se realizó el proyecto. Su historia y tendencias en la actualidad, delimitación geográfica, y ubicación dentro del marco socio-político y legal costarricense.

1.1.1.1 Contexto

Según Noyce & Hoff (1981) Intel es la corporación más grande del mundo dedicada al diseño, manufactura y suplencia de microprocesadores para PCs, laptops, servidores y estaciones de trabajo. Según estos autores, su éxito se debe en parte a la compatibilidad de sus microprocesadores con sistemas basados en Windows y a sus alianzas con los fabricantes originales de los equipos, quienes adaptan sus sistemas a los requerimientos de los procesadores de Intel.

La realidad es que desde que el procesador entró en los planes de Intel, esta pieza se convirtió en una unidad transformadora tanto de la compañía como de la competencia y revolucionaría lo que se consideraba para entonces como las fronteras de las computadoras.

Los orígenes de los procesadores pueden rastrearse a la década de 1960, cuando la complejidad de los diseños de lógica aleatoria hacía cada vez más evidente las limitantes de las arquitecturas utilizadas hasta el momento, que suponían la utilización de cada vez más transistores y con ello, aumentos en los costos espaciales y de manufactura. De esta forma, se puede deducir que el microprocesador como invención, se plantea no tanto como un hecho aislado y aleatorio, sino como una necesidad.

Por su parte, en lo que se refiere a Intel, justo antes de la invención del procesador, la empresa ya se había conformado como corporación y fabricaba Memorias de Direccionamiento Aleatorio (RAMs) y Semiconductores de Óxido Metálico (MOS), contando con cerca de 500 empleados. De hecho, hacia 1969, cuando Busicon, empresa japonesa especializada en la fabricación de calculadoras, le pide a Intel producir un conjunto de chips para una familia de calculadoras programables de alto desempeño para lo que le entrega un diseño tan complejo que el proyecto se convierte en la materia prima perfecta para la elaboración de una propuesta más innovadora y eficiente. Hoff, el ingeniero a cargo para entonces, propone que reducir la lógica aleatoria e incrementar los requerimientos de memoria daría como resultado un procesador de uso general. De esta idea nace un diseño de tres chips que incluirían entre sus componentes: una Unidad Central de Procesamiento (CPU), Memoria de solo lectura (ROM) y RAM. Cuando Federico Faggin se hizo cargo del proyecto, el modelo ya había sido refinado en repetidas ocasiones; aun así, se hicieron más refinamientos y se construyeron varias muestras de los chips.

En sus inicios, todavía en la década de 1960, los chips fabricados se vendieron a Busicon para con quien Intel tenía obligaciones contractuales, pero a partir de 1971, se abrió su comercialización a otros mercados particulares.

Aun así, para inicios de la década de los setentas, el mercado de los microprocesadores seguía siendo pequeño, al considerarse un reemplazo para las micorcomputadoras programables. Por lo que no es sino hasta la llegada de Ed Gelbach, quien ve en la invención no solo un reemplazo para las microcomputadoras programables sino como una forma de introducir inteligencia en artefactos por primera vez en la Historia. Producto de esta premisa se da la publicación del MCS-4 en noviembre de 1972, tras largos procesos de experimentación y con la EPRON de Frohman incorporada. Para 1975, el Supervisor de Implementación de sistemas Intel (ISIS por sus siglas en inglés) y el Emulador Embebido (ICE por sus siglas en inglés), permitieron la depuración del hardware y el software en tiempo de ejecución de los microprocesadores existentes, impulsando la liberación de una competencia campal por quien haría el procesador con mejor rendimiento. De esta lucha participan, además de Intel, Teledyne Systems, Texas Instruments and General Automation. Algunos de los Hitos de este periodo son: el 8080 de Intel, que fue rápidamente

convertido en un mecanismo estándar para computadoras de 8 bits, el 6800 de Motorola, primero que necesitó +5 voltios para funcionar desde una única fuente de alimentación, el 8048 de Intel, primer microcomputador de 8 bits con un solo chip, el primer microprocesador de un solo chip y 16 bits de National Semiconductor y el CP 1600 de Texas Instruments, también de 16 bits.

Tal y como lo menciona Monge-González (2017), se puede decir que el auge de los microprocesadores se alimenta de la popularización de las computadoras en los noventas. Este crecimiento en la demanda convierte a Intel en una de las empresas más rentables a nivel internacional con un 85% del total de ventas de microprocesadores a nivel mundial en 1995, año en el que muestra sus primeros rasgos de transnacionalidad al fundar el concepto de *fábrica virtual*, que implicaba que todas las decisiones importantes relacionadas con tecnología, capacidad de producción, entre otros, serían el resultado de negociaciones entre gerentes pertenecientes a diversos países del mundo.

Este carácter de transnacionalidad se concreta en el año 1996, cuando la corporación toma la decisión de establecer una planta para ensamblaje y prueba de microprocesadores, entre los países candidatos se encuentran Indonesia, Brasil, Argentina, Chile, México y posteriormente Costa Rica. Tras un extenso proceso de negociación con CINDE, institución privada sin fines de lucro fundada en 1982 para facilitar la inversión extranjera directa en Costa Rica, y, tras evaluar Intel la estabilidad política de los países candidatos, estructura de costos, interés gubernamental para impulsar la inversión extranjera, logística, tiempo de fabricación y permisos de construcción para la planta, el 13 de noviembre de ese mismo año, se anuncia que Intel iniciará operaciones en Costa Rica bajo el Régimen de Zonas Francas (RZF). Tras lo que se construye una planta en el país de 400.000 pies cuadrados, que a su vez albergaría 2.000 personas para ensamblar y probar los procesadores de las generaciones más recientes.

Para 1999 la corporación ya había invertido en Costa Rica más de 390 millones de dólares y contaba con más de 2.200 trabajadores en el área de manufactura. Dejando en el periodo 2004 – 2015 casi un 0,61% del Producto Interno Bruto (PIB) en la economía costarricense.

Sin embargo, entre los años 2014 y 2016 Intel cierra sus operaciones de manufactura en Costa Rica, migrándolas a Asia. Este periodo se ve marcado por una nueva etapa de negociaciones de Intel junto con instituciones gubernamentales costarricenses, como el Ministerio de Comercio Exterior (COMEX), Ministerio de Ciencias, Tecnología y Telecomunicaciones (MICITT) y otras instituciones privadas como el CINDE con el fin de ampliar la inversión de la compañía en territorio costarricense, en las áreas de Investigación y Desarrollo (I&D) y Servicios Compartidos. Subsecuentemente y como producto de estas negociaciones se da un aumento en las actividades de I&D y Servicios Compartidos de la empresa en Costa Rica, lo que trae como consecuencia, una menor necesidad de personal de fábrica, operativo, técnicos de manufactura, ingenieros de fábrica y soporte administrativo de fábrica, y más ingenieros eléctricos, electrónicos, de software, Tecnologías de la información (IT), así como personal contable, financiero y bilingüe.

A partir del 2016 Intel comienza un nuevo proceso de transformación a nivel mundial en el que incorpora a su nicho de mercado el desarrollo de memorias e incursionando en internet de las cosas, por lo que internamente la empresa se divide en unidades de negocio, según su línea productiva particular.

1.1.1.2 Diagnóstico

En este apartado se describe la relación de Intel como organización en relación con su proyección en Costa Rica desde un punto de vista político, económico y empresarial.

1.1.1.2.1 Zonas Francas y IED en Costa Rica

Según Monge González et al. (2012), a partir de la mitad de la década de 1980 en Costa Rica nace el Régimen de Zona Franca (RZF) en forma de un conjunto de incentivos para empresas multinacionales (MNCS) para incentivar la Inversión extranjera directa (IED) en Costa Rica y con ello promover oportunidades de empleo y la diversificación de la base productiva del país.

Estas iniciativas tuvieron repercusiones significativas localmente, ya que entre 1986 y 1995, la IED en Zonas Francas (ZF) implicó un 20% del PIB. Además, dichas zonas, para el 2009, ya generaban alrededor de 50.000 empleos en el país.

Esta nueva realidad de Costa Rica, se constituyó en una oportunidad para *derrames* y/o *transferencias de conocimiento* desde Intel hacia empresas costarricenses. Dicho en otras palabras, la inversión que Intel realiza en el país, se convierte en la coyuntura para que, a través de la interacción, las empresas locales incorporen en sus procesos conocimientos provenientes de las MNCS, en este caso Intel, y con ello aumenten su productividad y capacidad de ventas reales.

De hecho, según un estudio realizado por Monge-González & González-Alvarado (2007), en MNCS tales como Intel, Cisco y Microsoft, los derrames de conocimiento se evidencian en las capacitaciones recibidas por la población empleada de estas empresas, quien, no sólo recibe entrenamiento técnico específico propio de sus tareas dentro de la organización, sino también, entrenamientos que le serán de utilidad a lo largo de la vida, *lifelong learning process*, o capacitaciones en el área que la persona empleada elije para su propio desarrollo personal, estas capacitaciones pueden estar relacionadas con la administración de negocios, los idiomas, comercio, diseño gráfico, entre otros.

Además, tal y como lo apuntan Monge González et al. (2012), la evidencia sugiere una externalidad positiva en lo que se refiere a la movilidad laboral desde MNCS hacia empresas locales en Costa Rica, ya que las empresas locales que han contratado a personas que previamente se han desempeñado en una MNCS muestran un más alto desempeño en las áreas de crecimiento de ventas y empleo, en relación con sus competidores.

1.1.1.2.2 Derrames y transferencias de conocimiento de Intel en Costa Rica

Desde la perspectiva de Monge-González (2017), a pesar de que, entre las externalidades positivas que Intel ha impulsado, se pueden mencionar, estándares en el manejo del medio ambiente, seguridad ocupacional e inversión en educación, los derrames y las transferencias de conocimiento hacia empresas locales todavía es reducido y ha comenzado a revertirse en los últimos años a través de encadenamientos con universidades públicas y empresas locales del país para fortalecer la sección de I&D e innovación. "… durante el año 2016 (luego del ascenso en la CGV) las compras locales de servicios

especializados en tecnologías de la información e I&D representaron el 17%, mientras que este mismo rubro representaba el 5% en el año 2013, cuando Intel Costa Rica trabajaba principalmente en ensamblaje y prueba de microprocesadores". (Monge-González, 2017, pág.53)

En este mismo texto, se menciona que las dificultades para la incorporación de conocimiento desde Intel hacia empresas locales se explican por una baja capacidad de las absorción de empresas locales, asociada a una baja productividad, pocas personas calificadas participando en los procesos de producción y comercialización, bajos niveles de exportación e innovación. Al lado de factores estructurales como lo son limitaciones en el acceso a financiamiento, tecnología en telecomunicaciones, recursos humanos, políticas de inversión e industriales.

1.1.1.2.3 Tendencias de Intel 2019

A pesar de que en sus primeros años de existencia, Intel se concentró en la producción y el posicionamiento de procesadores para el mercado de las Computadoras Personales (PCs); durante los últimos años, se ha propuesto una baja en este tipo de inversión, que se ha concretado en un disminución del equivalente a dos millones de unidades enviadas para su comercialización en el cuarto trimestre de 2018. Concentrándose más bien en el mercado de las computadoras portátiles y servidores. (Lopez, 2018)

Además, según Alejandro Alonso (2018) Intel se plantea el fortalecimiento de los centros de datos con el fin de obtener la mayor cantidad de información de estos con la mayor rapidez posible por medio de mejoras en la conectividad y las redes. Esto de la mano de la optimización de las jerarquías de memoria con lo que se espera lograr una mayor capacidad de almacenamiento a un costo menos elevado. A esta tendencia responde el lanzamiento de la memoria *persistente Intel Optane DC*, ubicada entre la RAM y los Discos Duros (SSDs).

Las mejoras en el rendimiento y optimización de los microprocesadores es una venta crucial en la compañía. Esta preocupación se ha visto plasmada en el lanzamiento de nuevas generaciones de procesadores *Intel Xeon*, como lo son los procesadores *Cascade Lake*, *Cooper Lake y Ice Lake* y las *FPGA Intel* para la inteligencia artificial (IA).

1.1.2 Justificación del proyecto

A continuación, se expone la relevancia del proyecto para la organización en la que se realiza desde un punto de vista económico, estratégico y de decisión, explorando las razones que le dieron origen y las necesidades a las que contribuye como solución.

En un mundo enmarcado por la globalización y la informática, en el que las necesidades y exigencias de quienes consumen son el motor primordial en el rumbo de los negocios, surge la necesidad de las empresas por mantenerse a la vanguardia de su sector comercial, compitiendo no sólo con otras empresas del mismo sector sino a nivel interno, mejorando la calidad de sus productos.

En este sentido, a partir del razonamiento de Lossada & Robles (2014), los procesos de mejora continua resultan ser un mecanismo fundamental en las actividades de cada negocio. De esta forma, la mejora continua se entiende como los esfuerzos para estandarizar mejoras graduales, identificando causas, ideas y proyectos de mejora, que puedan ser a su vez controlados y evaluados.

Con la mejora continua, por tanto, se pretende optimizar la calidad de los procesos de producción en las empresas, por medio de la planificación y la toma de medidas remediales en los casos en los que resulta necesario o en otras palabras, verificando la efectividad de los procesos controlados. Se desprende, por tanto, que estos procesos estarán mediados por la retroalimentación de todas las partes involucradas: clientes, colaboradoras y colaboradores de la empresa, entre otros.

A través de los procesos de mejora continua, será posible enfrentarse a dos situaciones, la primera de ellas, que se alcance el objetivo de mejora, en cuyo caso, se buscará la normalización del éxito alcanzado en toda la organización; o que, en caso contrario, que el objetivo no sea alcanzado, en este último caso será necesario iniciar un nuevo proceso de gestión de mejora continua.

El proyecto que se plantea en este trabajo, se enmarca dentro de los procesos de mejora continua de la organización P2CA, Intel, Costa Rica, ya que se propone monitorear las tareas de los procesadores Intel, con el fin de detectar las unidades de hardware que están siendo utilizadas de acuerdo al programa en ejecución, potenciando así un mejor aprovechamiento del hardware a través de la detección de posibles cuellos de botella o unidades desaprovechadas.

Asimismo, la elaboración de este proyecto proveerá a la empresa de información útil desde el punto de vista de la reducción de costos. En este sentido, tal y como lo mencionan Marulanda Grisales, Hincapié Pizza & Echeverry Correa (2016), la Teoría de las Restricciones (TOC por sus siglas en inglés), dicta que, aunque las organizaciones tienen metas y objetivos variados, estos tienen como finalidad común el aumento de su rentabilidad y utilidades. El cumplimiento de estos objetivos está necesariamente ligado a fluctuaciones, puntos débiles, que afectan el desempeño global de la organización, usualmente llamados *´cuellos de botella´*. Según la TOC, es vital identificar las restricciones de una organización (cuellos de botella) y/o desperdicios para que posteriormente sean eliminados.

Desde el punto de vista de este texto, con la ejecución del proyecto en cuestión se plantea la detección, no sólo de cuellos de botella en los microprocesadores, sino también la detección de unidades ociosas durante la ejecución de ciertos programas.

Tal y como lo evidencian Isaza-González, Serrano-cases, Restrepo-calle, Cuenca-asensi, & Martínez-Álvarez (2017), los cuellos de botella y las unidades ociosas en los procesadores han sido tema común en la literatura especializada e implican un recurso desperdiciado en los procesos de segmentación, así como una posibilidad de mejora para las compañías. La detección oportuna de unos y otras, es un punto significativo de competitividad, ya que se constituye en una medida de control que puede llevar a mecanismos correctivos, desembocando finalmente en una mejor experiencia de usuario y con ello, un factor de valor para la empresa.

En este proyecto, se encuentra en el ámbito de estas tendencias, ya que se plantea la elaboración de una herramienta que permita monitorear las unidades del microprocesador, y con ello, detectar oportunamente el desperdicio o sobrecarga en los recursos de este.

1.2 Definición del problema

Todo proyecto nace para solucionar un problema y se ancla en una problemática específica que usualmente es más amplia que contribuirá a solucionar. Los siguientes párrafos se expone la problemática sistematizada por medio de un diagrama causa-efecto ampliada luego a través de su descripción. Además, aborda el problema general y los problemas específicos que soluciona el proyecto, como aporte a la empresa en su contexto específico.

1.2.1 Problemática

En el presente apartado se discutirá el problema, sus causas y la forma en que el proyecto presentado es un aporte a su solución. Para su mejor comprensión, el diagrama causa-efecto del que se deriva la descripción de la problemática se encuentra en el Anexo 7 del proyecto.

1.2.1.2 Descripción de problemática

La necesidad de conocer la forma en que se distribuyen los recursos del procesador al ejecutar cualquier software de usuario, como lo son la suite de *Master Collection* o el paquete de *Microsoft Office* es una preocupación vigente para Intel. Esta problemática se ve alimentada por las siguientes causas:

a. Pocas herramientas: en la actualidad existen herramientas de software y hardware especializadas en el monitoreo de los recursos de los microprocesadores que permiten conocer a través de mediciones específicas, posibles cuellos de botella o deficiencias presentes tanto durante los procesos de diseño, como en la fabricación o postimplementación de procesadores. Estas deficiencias se detectan por medio de un diagnóstico a través de indicadores relacionados con componentes de micro y macro arquitectura que afectan directa o indirectamente su desempeño. Sin embargo, el número de herramientas especializadas es escaso, lo que **limita la capacidad de recopilar datos y con ello generar la información necesaria para la fiabilidad del diagnóstico,** dificultando así conocer el margen de error existente entre las mediciones.

b. Escasez de tiempo del personal para realizar actividades de desarrollo complementarias: aunque existe personal altamente capacitado en el área de la microarquitectura. Éste se

encuentra principalmente dedicado a actividades que la compañía considera más prioritarias, **dejando poco espacio para el desarrollo y trayendo como consecuencia pocos programas diseñados a la medida de acuerdo a las necesidades del equipo**.

c. Limitada socialización sobre estándares de arquitectura para la construcción y migración de las piezas de software desarrolladas: Los procesadores Intel se construyen a partir de estándares. Sin embargo, la falta de socialización acerca de la forma en que se encuentran construidos favorece **un desconocimiento general sobre aspectos que podrían ser utilizados para la migración o adaptación de software de una plataforma a otra.**

d. Carencia de documentación del software especializado existente para su perfeccionamiento e integración al desarrollo de otras piezas de software: el conocimiento sobre el código, la funcionalidad de los algoritmos, así como el flujo esperado se adquiere, en su mayoría, de forma intuitiva, lo que desemboca en un **desaprovechamiento del potencial de las herramientas de software existentes dentro de la organización**.

1.2.2 Problema general

Las tendencias del mercado al día de hoy exigen sistemas cada vez más veloces y con ello, procesadores cada vez más eficientes en la prestación los servicios que brindan. Esta realidad tae consigo una problemática para Intel, ya que, las constantes mejoras en el hardware no resultan útiles o visibles para el usuario final a menos que el software haga uso de los recursos existentes en los procesadores. Es por esto que es una pregunta constante: ¿Cómo pueden monitorearse as unidades funcionales del procesador durante la ejecución de aplicaciones de usuario?

1.2.3 Problemas específicos

Para responder a este cuestionamiento se plantean los siguientes problemas específicos:

a. ¿Qué requerimientos tanto funcionales como no funcionales debería de tener una aplicación que permita en monitoreo de las unidades funcionales del procesador?

b. ¿Cómo se puede establecer una relación entre la microarquitectura del procesador en estudio, las unidades funcionales que lo componen y el ejercicio de éstas unidades al ejecutarse la aplicación de un tercero?

c. ¿Qué relación lógica podría existir entre las distintas capaz de hardware y software para llevar a cabo la propuesta de forma exitosa?

d. ¿Qué mecanismo se podría utilizar para mostrar la forma en que las unidades funcionales de un procesador son utilizadas durante la ejecución de la aplicación de un tercero?

1.3 Objetivos del proyecto

A continuación, se exponen los propósitos que se cumplirán durante el desarrollo del proyecto. En síntesis, se expondrá el objetivo general del proyecto, seguido de los objetivos específicos por medio de los cuales se concretará.

1.3.1 Objetivo General

Construir un mecanismo para el monitoreo de las unidades funcionales del procesador durante la ejecución de aplicaciones de usuario, 2019.

1.3.2. Objetivos específicos.

a. Identificar los requerimientos de software para la elaboración de un prototipo funcional que monitoree la utilización de las unidades funcionales de los microprocesadores Intel core i7.

b. Analizar la relación entre la arquitectura del procesador, sus unidades funcionales y el ejercicio de éstas unidades al ejecutarse la aplicación de un tercero.

c. Diseñar una propuesta de software que permita monitorear la forma en que las unidades de los procesadores Intel core i7 están siendo ejercitadas desde el nivel de usuario.

d. Desarrollar un prototipo de software para mostrar la forma en que las unidades funcionales de un procesador core i7 son utilizadas durante la ejecución de la aplicación de un tercero.

1.4. Alcances y limitaciones.

En este apartado se encuentra una descripción más amplia de los objetivos planteados en el proyecto. Asimismo, se describen los entregables que se esperan de acuerdo con cada una de las fases que involucran la elaboración de este.

1.4.1 Alcances

En esta sección se explicitan los alcances con sus respectivos entregables por medio de un EDT.

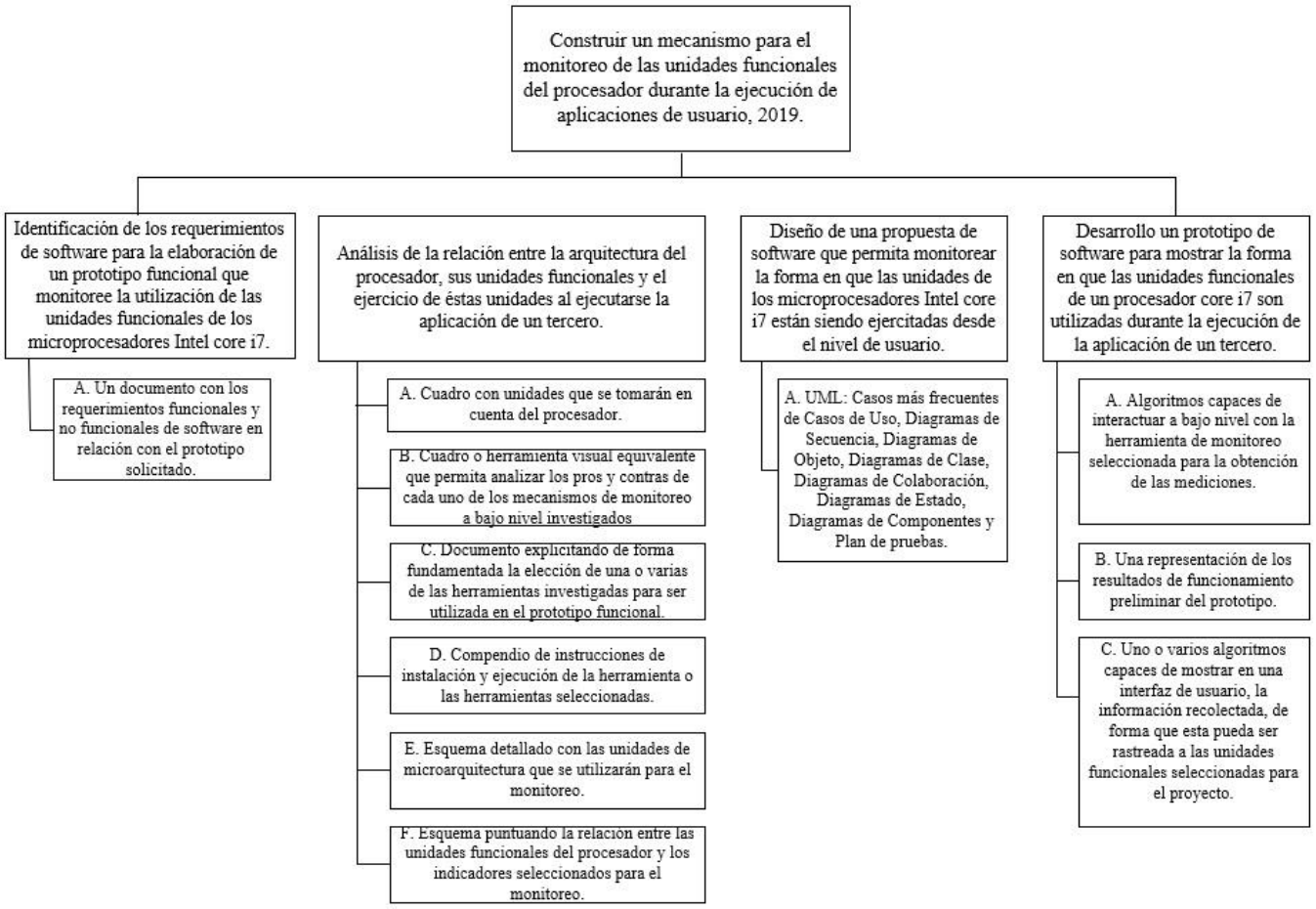


Ilustración 1. EDT sobre alcances, elaboración propia.

1.4.2. Limitaciones.

Los proyectos están enmarcados en contextos sociales, históricos y políticos que los sujetan a un tiempo y espacio específicos. En este apartado se comentan las principales limitaciones que se desprenden del contexto específico en el que se desarrollará.

1.4.2.1. Acerca de la arquitectura.

Debido al periodo de tiempo definido para el desarrollo del proyecto se limitó la arquitectura a los procesadores Intel core i7 para computadoras de uso general. A pesar de esto, se aportará documentación para facilitar el proceso de adaptación para que pueda ser utilizado en otras arquitecturas.

1.4.2.2 Acerca del software a ser desarrollado

Se espera como producto final un prototipo funcional, una herramienta capaz de demostrar que es posible rastrear las unidades funcionales del procesador y arrojar datos acerca de la forma en que estas unidades son utilizadas. Sin embargo, no será un software totalmente funcional por lo que será susceptible a errores.

1.4.2.3 Sistema operativo

Debido a limitantes de tiempo, el prototipo se desarrollará en Ubuntu, Linux. Sin embargo, se aportarán el código y la documentación necesarias para que pueda migrarse a otras plataformas.

1.4.2.4. Sistema destino

El prototipo estará diseñado para funcionar de forma local, por lo que no se tomarán en consideración accesos remotos.

1.5. Cronograma de trabajo.

En el presente apartado se presentan los esfuerzos realizados de acuerdo a los objetivos, por fechas y pesos. El desglose de tareas con sus respectivas fechas se detalla en el Anexo 1 de este trabajo.

1.5.1 Resumen de cronograma de trabajo de proyecto.

Tabla 1. Resumen de cronograma de trabajo, elaboración propia.

Task Name	Duration	Start	Finish	Predecessors
Problema del proyecto	**19 days**	**2/4/19**	**2/28/19**	
Marco teórico	**22 days**	**3/1/19**	**4/1/19**	**1**
Generalidades	**2 days**	**3/1/19**	**3/4/19**	
Los procesadores y la competencia por el rendimiento	**2 days**	**3/5/19**	**3/6/19**	**16**
Contexto del procesador	**2 days**	**3/7/19**	**3/8/19**	**19**
Estructura y funcionamiento de un procesador	**3 days**	**3/11/19**	**3/13/19**	**22**
Análisis de rendimiento	**2 days**	**3/14/19**	**3/15/19**	**26**
Técnicas de caracterización por cargas de trabajo	1 day	3/19/19	3/19/19	32
Notas sobre los monitores de tareas	1 day	3/20/19	3/20/19	33
Clasificación de los monitores existentes	**4 days**	**3/21/19**	**3/26/19**	**30**
Ciclo de vida del software y modelos de desarrollo	**4 days**	**3/27/19**	**4/1/19**	**35**
Marco metodológico	**22 days**	**4/1/19**	**4/30/19**	**15**
Tipo de investigación	1 day	4/1/19	4/1/19	
Enfoque de investigación	1 day	4/2/19	4/2/19	45
Fuentes y sujetos de investigación	**3 days**	**4/3/19**	**4/5/19**	
Técnicas y herramientas	**2 days**	**4/8/19**	**4/9/19**	**47**
Variables	1 day	4/10/19	4/10/19	53
Diseño de la investigación	**7 days**	**4/11/19**	**4/19/19**	**51**
Matriz de coherencia	1 day	4/22/19	4/22/19	55
Diagnóstico de empresa	**8 days**	**4/20/19**	**4/30/19**	**44**
Visión, misión, valores empresariales y estrategias empresariales.	1 day	4/20/19	4/20/19	
Diagnóstico admnistrativo	**6 days**	**4/22/19**	**4/27/19**	
Diagnóstico técnico	1 day	4/29/19	4/29/19	65
Diagnóstico de percepción	1 day	4/30/19	4/30/19	71
Consideraciones finales	**3 days**	**4/29/19**	**5/1/19**	**72**
Diseño y desarrollo del proyecto	**39 days**	**4/30/19**	**6/24/19**	**63**
I. Definición del problema	**8 days**	**5/1/19**	**5/10/19**	
II. Selección de métricas	**4 days**	**5/13/19**	**5/16/19**	**78**
III. Una discusión sibre parámetros y factores.	**1 day**	**5/17/19**	**5/17/19**	
IV. Diagramas de UML	**14 days**	**5/20/19**	**6/6/19**	
V. Análisis e interpretación	**10 days**	**6/6/19**	**6/19/19**	
VI. Presentación de resultados	**3 days**	**6/20/19**	**6/24/19**	
Conclusiones y recomendaciones	**2 days**	**6/25/19**	**6/26/19**	
Conclusiones.	1 day	6/25/19	6/25/19	
Recomendaciones.	1 day	6/26/19	6/26/19	105
Bibliografía consultada	**1 day**	**6/27/19**	**6/27/19**	
Glosario	**3 days**	**6/28/19**	**7/1/19**	**107**
Anexos	**1 day**	**7/5/19**	**7/5/19**	**108**

CAPÍTULO II: MARCO TEÓRICO

2.1 Generalidades.

Los párrafos a continuación describen generalidades necesarias para situar el proyecto en su contexto. Por ello, describen terminología general acerca de arquitectura y organización de computadoras, pasando luego a la vista de un procesador.

2.1.1 Principios de arquitectura de computadoras.

Según Stallings, W. (2010), cuando se discute sobre computadoras, es indispensable conocer la diferencia entre Arquitectura de Computadoras, también llamada Arquitectura de las instrucciones x86 (ISA, por sus siglas en inglés) que se refiere a todos los atributos que se encuentran visibles para quien programa y por tanto tienen un impacto directo sobre la ejecución lógica de un programa y la Organización del Computador, que se refiere a las unidades operacionales y sus interconexiones según especificaciones de arquitectura dadas. Por esto, cuando se habla de la organización del computador, se hace referencia a los detalles de hardware que son transparentes para quien programa, como lo son señales de control, interfaces entre los distintos componentes y periféricos, así como la tecnología de memoria utilizada. Esta diferenciación entre la arquitectura de una computadora y la organización de esta permite entender como una misma arquitectura puede ser implementada de formas diversas por medio de su organización, como sucede con las familias de microprocesadores de muchas compañías en la actualidad.

Además, los sistemas actuales hacen necesaria la diferenciación entre a forma en que los componentes se encuentran interconectados (estructura) y la operación que cumple cada componente individualmente dentro del sistema (función).

En cuanto a la función, las computadoras, en cualquiera de sus niveles, pueden efectuar cuatro operaciones básicas:

a. Procesamiento de datos: los datos pueden ser procesados de diversas formas y según una cantidad gigantesca de requerimientos.

b. Almacenamiento de datos: el procesamiento constante de datos implica que la computadora debe ser capaz de almacenar datos tanto a corto como a largo plazo.

c. Movimiento de datos: los dispositivos de almacenamiento de información pueden convertirse, dependiendo del proceso, tanto en origen como en destinatarios de datos.

d. Control: la unidad de control en una computadora maneja y ordena el comportamiento de las diferentes partes funcionales y la forma en que responden a as instrucciones.

En relación con su estructura, se pueden distinguir cuatro partes básicas:

a. Unidad Central de Procesamiento (CPU, por sus siglas en inglés): procesa las funciones de datos, y se conoce usualmente de forma abreviada como procesador.

b. Memoria principal: almacena datos.

c. E/S: Puente entre los datos en el interior de la computadora y el exterior.

d. Sistema de interconexión: Permiten la comunicación entre CPU, memoria principal, y E/S.

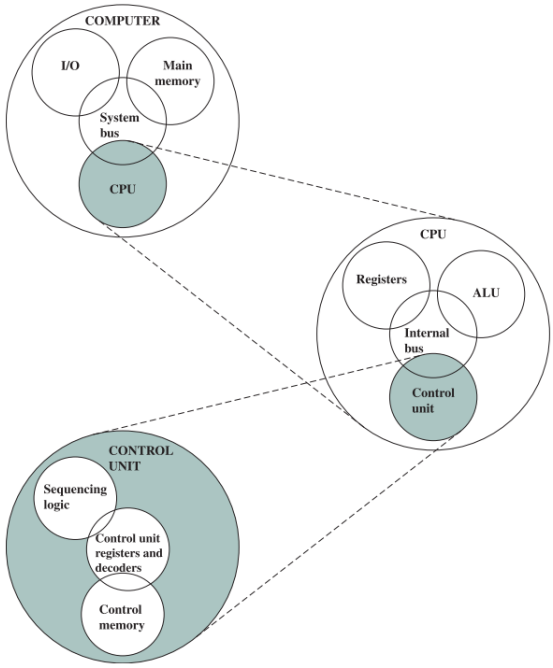


Ilustración 2. Computer. Top level Structure. En Computer Organization and Architecture (p.5), por Stallings, W, 2010,Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

Aparte de esto, el componente más suscitado es el procesador, cuyos componentes se explicitan a continuación:

a. Unidad de control: controla las operaciones de la computadora.

b. Unidad Aritmético- Lógica (ALU, por sus siglas en inglés): realiza las funciones de procesamiento de datos de la computadora.

c. Registros: Proveen almacenamiento a lo interno del procesador.

d. Interconexiones del CPU: Proveen comunicación entre la ALU, la unidad de control y los registros.

En la actualidad, en un solo chip pueden haber muchos procesadores. Cuando esto sucede, puede hablarse de computadoras de multinúcleo, más comúnmente conocidas como *multicore*. De esta forma, cada núcleo o *core*, tendrá su propia CPU, que a su vez tendrá una unidad de control, ALU, registros y en algunos casos cada vez más frecuente, caché. A continuación, se explica de una forma más específica, el funcionamiento de estas unidades en cada núcleo:

a. CPU: recupera instrucciones de la memoria principal o la caché y las ejecuta.

b. Núcleo: Unidad de procesamiento, es equivalente a CPU en una computadora de un solo procesador.

c. Procesador: Una pieza (chip) de silicio que alberga uno o varios núcleos.

Dicho de forma muy general, el elemento indispensable para el ensamblaje del procesador como tal es la tarjeta madre, que es un circuito impreso (PCB, por sus siglas en inglés). Sobre la tarjeta madre se ubican varios chips de silicio. Cada chip está construido de material semiconductor y contiene tanto circuitos electrónicos como compuertas lógicas. Estos chips estarán encargados de realizar funciones de hardware

controles y acceso a memoria, buses y E/S. La tarjeta madre contendrá una abertura para el chip del procesador, que, de ser *multicore*, a su vez estará conformado por varios núcleos individuales.

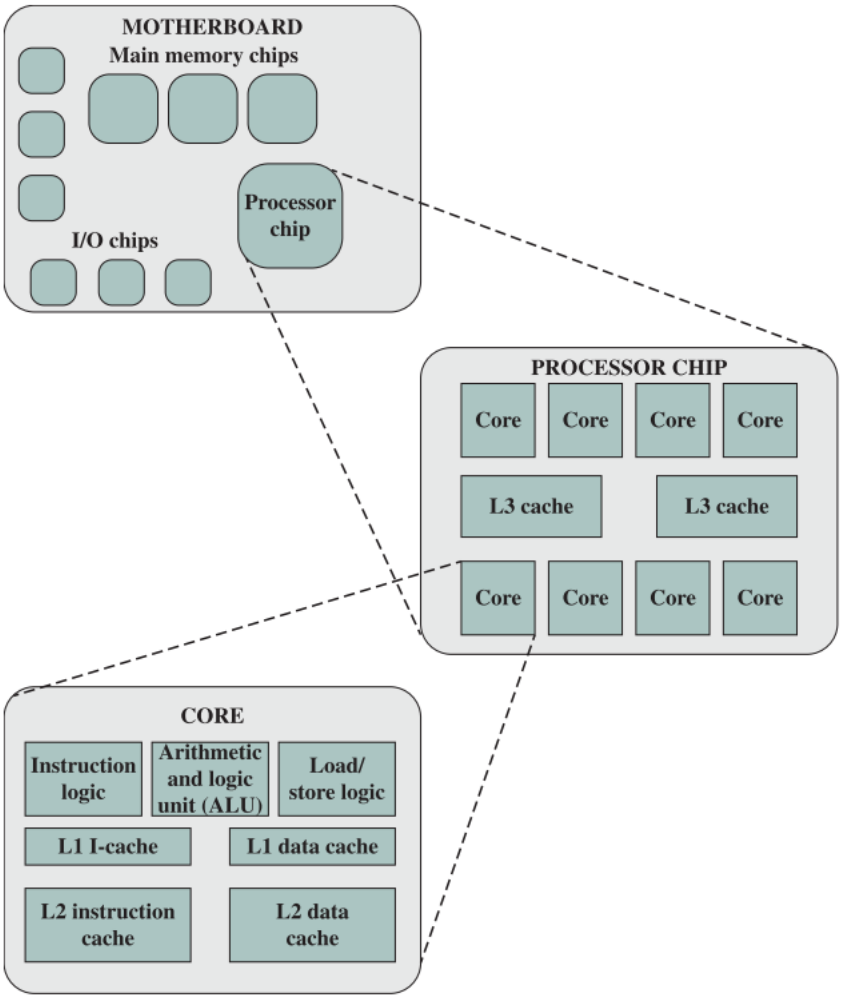


Ilustración 3.Simplified View of Major Elements of a Multicore Computer. En Computer Organization and Architecture (p.7), por Stallings, W, 2010,Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

Visto a gran escala, los elementos funcionales de cada uno de los núcleos son:

a. Lógica de instrucción: Involucra recuperación de instrucciones desde la memoria y la decodificación de cada una de las instrucciones con el fin de determinar los operandos que serán utilizados para su procesamiento, así como su localización dentro de la memoria.

b. Unidad Aritmético-Lógica: Ejecuta la operación especificada por medio de una instrucción.

c. Lógica de carga y almacenamiento: se encarga de la transferencia de datos desde y hacia la memoria caché o la memoria principal.

d. Caché: Los núcleos tienen en su interior una memoria caché para acelerar el acceso a datos. Usualmente los niveles de caché utilizados son:

- L1: se divide entre **caché de instrucciones**, que se utiliza para transferir instrucciones desde y hacia memoria, y **caché de datos**, que se utiliza para transferencia de operandos y resultados.
- L2: Usualmente este nivel también se divide entre caché de datos y caché de instrucciones.
- L3: Es la memoria caché más lejana dentro de la CPU.

2.1.2. Un acercamiento a los procesadores Intel.

A continuación se muestra a modo de resumen el recorrido de los procesadores Intel a partir de la visión de Stallings, W (2010) a través de la historia hasta la generación del *core i7*.

Tabla 2. 1970s Processors. En Computer Organization and Architecture (p.26), por Stallings, W, 2010,Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

(a) 1970s Processors

	4004	8008	8080	8086	8088
Introduced	1971	1972	1974	1978	1979
Clock speeds	108 kHz	108 kHz	2 MHz	5 MHz, 8 MHz, 10 MHz	5 MHz, 8 MHz
Bus width	4 bits	8 bits	8 bits	16 bits	8 bits
Number of transistors	2,300	3,500	6,000	29,000	29,000
Feature size (μm)	10	8	6	3	6
Addressable memory	640 bytes	16 KB	64 KB	1 MB	1 MB

Tabla 3. 1980s Processors. En Computer Organization and Architecture (p.26), por Stallings, W, 2010,Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

(b) 1980s Processors

	80286	386TM DX	386TM SX	486TM DX CPU
Introduced	1982	1985	1988	1989
Clock speeds	6–12.5 MHz	16–33 MHz	16–33 MHz	25–50 MHz
Bus width	16 bits	32 bits	16 bits	32 bits
Number of transistors	134,000	275,000	275,000	1.2 million
Feature size (μm)	1.5	1	1	0.8–1
Addressable memory	16 MB	4 GB	16 MB	4 GB
Virtual memory	1 GB	64 TB	64 TB	64 TB
Cache	—	—	—	8 kB

Tabla 4. Recent Processors. En Computer Organization and Architecture (p.27), por Stallings, W, 2010, Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

T

(c) 1990s Processors

	486TM SX	Pentium	Pentium Pro	Pentium II
Introduced	1991	1993	1995	1997
Clock speeds	16–33 MHz	60–166 MHz,	150–200 MHz	200–300 MHz
Bus width	32 bits	32 bits	64 bits	64 bits
Number of transistors	1.185 million	3.1 million	5.5 million	7.5 million
Feature size (μm)	1	0.8	0.6	0.35
Addressable memory	4 GB	4 GB	64 GB	64 GB
Virtual memory	64 TB	64 TB	64 TB	64 TB
Cache	8 kB	8 kB	512 kB L1 and 1 MB L2	512 kB L2

Tabla 5. 1990s Processors. En Computer Organization and Architecture (p.26), por Stallings, W, 2010,Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

(d) Recent Processors

	Pentium III	Pentium 4	Core 2 Duo	Core i7 EE 4960X
Introduced	1999	2000	2006	2013
Clock speeds	450–660 MHz	1.3–1.8 GHz	1.06–1.2 GHz	4 GHz
Bus width	64 bits	64 bits	64 bits	64 bits
Number of transistors	9.5 million	42 million	167 million	1.86 billion
Feature size (nm)	250	180	65	22
Addressable memory	64 GB	64 GB	64 GB	64 GB
Virtual memory	64 TB	64 TB	64 TB	64 TB
Cache	512 kB L2	256 kB L2	2 MB L2	1.5 MB L2/15 MB L3
Number of cores	1	1	2	6

Tal y como lo anota Gomez Vazquez, J. (2012) y puede observarse en las tablas anteriores, la generación actual de procesadores *core i7* de Intel es el resultado de varias décadas de perfeccionamiento del Set de Instrucciones Complejas (CISCs, por sus siglas en inglés).

Asimismo, puede observarse la siguiente evolución:

a. 8080: Primer microprocesador de uso general de la Historia, se construyó para una computadora de 8 bits con 8 bits en las rutas de memoria.

b. 8086: computadora de 16 bits, buses de datos más grandes y caché de instrucciones.

c. 80286: modelo similar a la 8086 con una memoria de 16MB en vez del 1MB de la 8086.

d. 80386: primera computadora Intel con 32MB. Fue la primera computadora Intel en permitir multitarea.

e. 80486: implementó el uso de una caché más sofisticada, segmentación de instrucciones (*pipelining*) y un coprocesador matemático, capaz de efectuar tareas matemáticas complejas.

f. Pentium: Introducción en los microprocesadores Intel de tecnología superescalar, para la ejecución de instrucciones en paralelo.

g. Pentium Pro: se dio una especialización aun mayor de técnicas superes calares por medio del renombramiento de registros, predicción de saltos (*branch predincting*), análisis del flujo de datos y ejecución especulativa.,

h. Pentium II: Se incorporó la tecnología Intel MMX, para el procesamiento eficiente de video, audio y gráficos.

i. Pentium III: en esta generación se incorporaron setenta instrucciones nuevas de punto flotante, *The Streaming SIMD Extensions (SSE)* de Intel, para set utilizadas en los casos en que la misma operación debe ser ejecutada en múltiples objetos.

j. Pentium 4: incluyó más instrucciones de punto flotante, así como mejoras en multimedia.

k. Core: Nace con el x86 y se refiere a la implementación de varios núcleos en un mismo chip.

l. Core 2: Permitió la existencia de cuatro núcleos en un mismo chip, para una computadora de 64bits. Se introdujo *the Advanced Vector Extensions Set*, para mejorar el procesamiento de datos vectoriales.

2.2. Los microprocesadores y la competencia por el rendimiento.

A partir del análisis de Stallings, W. (2010), a través de los años ha habido una demanda cada vez mayor de computadoras menos costosas cuya capacidad de rendimiento sea cada vez mayor. Las formas de medir esta eficiencia varían. Visto de una forma general, se busca mejorar el rendimiento en procesos que son comunes para quien utiliza la computadora, estos pueden observarse desde parámetros tales como: procesamiento de imágenes, representaciones tridimensionales, reconocimiento de voz, videoconferencia, multimedia, ficheros de voz y video y modelado de simulación.

Además, el rendimiento de los microprocesadores debe valorarse al lado del costo, el tamaño, la seguridad, confiabilidad y hasta su consumo de electricidad. De hecho, el rendimiento bruto de un procesador va a depender también de la aplicación que se ejecuta, y con ella el set de instrucciones que ejecuta, el lenguaje en el que se implementa y la eficiencia del compilador para decodificar dichas instrucciones.

En las siguientes secciones se discuten en forma general el rendimiento desde las perspectivas física y lógica.

2.2.1. La ley de Moore.

Según Mack, C.A. (2011), G. Moore, fundador del semiconductor *Fairchild,* nota en 1965, que desde 1959 el número de componentes por chip se duplicaba cada año, llegando, para 1965, a alrededor de 64 componentes por chip. Siguiendo este análisis, Moore prefijó que para 1975, cada chip tendría 65000 componentes.

En sus orígenes, el número de componentes por chip descrito por Moore incluía transistores, capacitores y resistores. Posteriormente, con la llegada de la era digital, la circuitería analógica perdió su preponderancia inicial, transformando la visión de Moore, que reduciría la ley de Moore al número de transistores. Tras varios refinamientos, Moore redefinió el número de componentes por chip, que pasó a significar el número de componentes que minimizarían el costo de un circuito dado.

Para 1975, y nuevas observaciones guiadas por un dispositivo de memoria de carga acoplada de Intel, que nunca fue comercializado y que a pesar de tener alrededor de 32 000 componentes solo 16000 de ellos eran transistores, la tendencia de crecimiento, se redujo de doblar el número de componentes por chip cada año, a doblarse cada dos años. Estas conclusiones fueron guiadas por una serie de análisis. Primeramente, Moore divide los avances en la complejidad de los chips en tres subáreas: aumento del área del chip, disminución del tamaño de sus características y mejores diseños de dispositivos y circuitos. Según sus cálculos, para 1975, el tamaño de los chips había disminuido aproximadamente un 10% por año, lo que a su vez implicó que los transistores fueran un 21% más pequeños, lo que a su vez conllevó a un 25% de más transistores por área. Finalmente, el área de los chips estaba aumentando un 20% cada año también. Con estos cálculos concluyó que el número de transistores por chip aumentó un 50% cada año.

Esta predicción resultó pesimista para la época, ya que el número de componentes por chip definido por Moore, se dobló un 60% cada año desde 1975, lo que equivaldría a doblarse completamente cada dieciocho meses aproximadamente. En las décadas de 1980 y 1990 se popularizó la ley de Moore, bajo esta última acepción. En otras palabras, se aceptó la ley de Moore como el doblamiento de la cantidad de transistores por chip se daría cada dieciocho meses, aunque Moore nunca hizo tal afirmación.

De hecho, a finales de la década de 1970, se dio la bifurcación de la Ley de Moore en dos tendencias: memoria y lógica. De esta forma, aunque los chips de memoria continuaron avanzando en complejidad al lado de la Ley de Moore, los chips lógicos, como es el caso de los microprocesadores, avanzaran a un paso menos acelerado. Esta bifurcación se debe, entre otras cosas, a que la capacidad de colocar más transistores por chip superó la capacidad del mercado de diseñar los chips que albergarían a tantos transistores. De esta forma, mientras

para el año 2000 los chips de memoria utilizaban diez mil millones de transistores, los microprocesadores más avanzados tenían entre veinte y cuarenta millones de transistores, lo que equivaldría a un aumento en el caso de los chips de memoria de 1.58 por año y de 1.38 en el caso de los microprocesadores.

La Ley de Moore alcanzó tal popularidad que se utilizó para hacer predicciones sobre la industria, así como para la elaboración de planes a futuro. Una de estas iniciativas es el Plan Nacional de Tecnología para Semiconductores (NTRS, por sus siglas en inglés), que fue desarrollado en primera instancia por la Asociación Industrial de Semiconductores (SIA, por sus siglas en inglés) den 1994, con el fin de estandarizar el crecimiento de la industria en semiconductores. A partir de este plan, se predice que los tamaños mínimos para la producción de chips DRAM de 64Gb sería de 70mn. Después de varias actualizaciones, en 1999, se transforma en el Plan Internacional de Tecnología de Semiconductores (ITRS, por sus siglas en inglés).

El plan de las DRAM de 64Gb no resultó seguir el paso que se esperaba, en parte porque la experiencia ha demostrado que entre uno y dos Gb son suficientes en una DRAM para cubrir las necesidades de los productos de gran escala. Esto no quiere decir que las mejoras en las DRAM se hayan detenido, sino que los esfuerzos se enfocan en lograr disminuir su tamaño y costo monetario. El dominio de más bits en un mismo chip ha pasado a ser parte de las metas relacionadas con las memorias Flash, que se están produciendo acorde con la Ley de Moore.

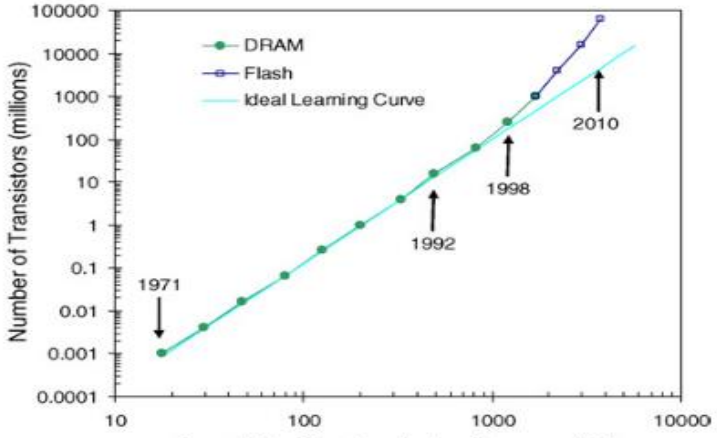


Ilustración 4. Semiconductor Renew. En FiftyYears of Moore ' s Law (p.205), por A. Mack, C, 2011, New Jersey,EEUU: IEEE. Derechos de autor [2011] por IEEE. Reimpresión autorizada.

Todo esto resulta significativo, ya que permite notar que la Ley de Moore ha sido un aporte para aumentar la densidad y rendimiento de los transistores en los chips, y de ahí su limitante, el no tomar en cuenta el incremento de la funcionalidad de cada chip. Dicho en otras palabras, si bien la memoria Flash saca provecho de todos los transistores que se puedan fabricar de forma económica, los demás chips en la industria no necesitan del máximo de transistores que se puedan fabricar de la misma manera. Siguiendo con el caso de DRAM, la SIA pronosticó que para 1997, el tamaño de los chips sería un poco mayor de 11 cms, a pesar de esto, desde hace más de una década, los chips se han mantenido en alrededor de 2 cms.

2.2.2. Técnicas para mejorar rendimiento en procesadores contemporáneos.

Según Parahmi, B. (2007), la lucha por el rendimiento no sólo ha conllevado mejoras físicas, sino que también ha implicado la necesidad de construir sets de instrucciones capaces de sacar provecho de las tecnologías de arquitectura de los procesadores.

Algunas técnicas utilizadas para mejorar el rendimiento de microarquitectura en la actualidad son:

a. Segmentación: la ejecución de instrucciones involucra varias fases que incluyen la obtención de instrucciones y la realización de cálculos. La segmentación permite que varias instrucciones puedan ejecutarse de forma simultánea, ya que el procesador traslapa operaciones a través del movimiento de datos o instrucciones. Dicho en otras palabras, su éxito radica en dividir las instrucciones en subfases que pueden ejecutarse por separado hasta completar la fase. Una división simple involucra la obtención de la instrucción (ciclo de fetch), decodificación de la instrucción, obtención de operandos, ejecución de instrucciones según operaciones y finalmente el almacenamiento de los resultados. En este caso sería posible tener hasta cinco instrucciones ejecutándose en cada una de las subfases, reduciendo el tiempo de latencia en la ejecución de dichas instrucciones. (Abd-El-Barr,M. & El-Rewini, H., 2005)

b. Predicción de saltos: por medio de este mecanismo, el procesador observa las instrucciones que han sido obtenidas de la memoria con el fin de predecir cuáles saltos o instrucciones serán procesadas posteriormente.

c. Ejecución superescalar: es la capacidad de los procesadores de ejecutar más de una instrucción en un ciclo de reloj.

d. Análisis de flujo de datos: es el proceso mediante el cual el procesador analiza las dependencias entre las instrucciones y sus respectivos resultados. De forma tal que éstas se ejecutan de acuerdo a cuando están listas, independientemente del orden del programa original, evitando así retrasos innecesarios.

e. Ejecución especulativa: algunos procesadores utilizan la predicción de saltos y el análisis de flujo de datos para predecir las instrucciones que se necesitarán sin que estén en cola, almacenándolas luego en ubicaciones temporales de memoria.

2.2.3. Afectaciones en el rendimiento del procesador en relación con sus componentes.

Según Stallings, W. (2010), a pesar de que el poder de los procesadores ha aumentado exponencialmente durante los últimos años, otros componentes no han tenido la misma curva evolutiva, lo que ha implicado la generación de mecanismos de balance que tienen como fin compensar estas diferencias. Esta situación puede observarse en la velocidad por medio de la que los datos son transferidos desde memoria hasta el procesador y viceversa. Dicho de otra forma, la interfaz entre el procesador y la memoria principal es crucial en el rendimiento de la computadora, ya que es responsable de un flujo constante de instrucciones de programa y datos entre ambas partes. Si los buses no logran satisfacer las demandas del procesador, el procesador permanece en un estado de espera en los que sus recursos son desaprovechados. Otra área que usualmente necesita ser observada son las peticiones de E/S. En la actualidad las computadoras trabajan con cada vez más dispositivos E/S, lo que hace necesarias cada vez más rápidas y sofisticadas técnicas para el manejo de estos pedidos.

Además de cuestiones pendientes por solucionar en relación con la memoria y los pedidos E/S, la demanda del mercado exige procesadores cada vez más veloces. Desde una perspectiva tradicional, este aumento en la velocidad puede seguirse a través de varios enfoques:

a. La reducción del tamaño de las compuertas lógicas en los chips del procesador, lo que permite empaquetar más compuertas juntas y a su vez aumentar la velocidad de los pulsos de reloj. Al estar las compuertas más cerca unas de otras, el

tiempo de propagación de las señales de reloj, disminuye, permitiendo que las instrucciones individuales se ejecuten más rápidamente.

b. El aumento en el tamaño y la velocidad de las memorias caché que se encuentran entre el procesador y la memoria principal. En general, si la memoria caché se encuentra cercana en el mismo chip que el procesador, el tiempo invertido en los viajes a memoria disminuye notoriamente.

c. Cambios en la organización y arquitectura del procesador para lograr la ejecución efectiva de instrucciones por medio de mecanismos como paralelismo o segmentación.

En síntesis, los aumentos más significativos en el rendimiento de los microprocesadores pueden rastrarse a mejoras en la velocidad de los pulsos de reloj, por otra parte, al aumento de la densidad lógica. A pesar de esto, otros aumentos en su rendimiento están relacionados con el aumento en la capacidad de la memoria caché o la implementación de tecnologías Multinúcelo (*Multicore*), según las cuales es posible colocar varios procesadores en un mismo chip. Usualmente estos procesadores comparten una misma memoria caché, y permiten casi el doblamiento del rendimiento del computador. Finalmente, algunos procesadores incluyen unidades de procesamiento gráfico (GPUs), que consisten en unidades especializadas para operaciones vectoriales.

Varios factores han dificultado en alguna medida el aumento en el rendimiento de los procesadores. Entre ellos pueden mencionarse:

a. Potencia: la densidad de potencia medida a través de watts/cm2 ha hecho evidente la dificultad por disipar el calor generado por los chips.

b. Latencia de los circuitos Resistor & Condensador (RC): la velocidad a la que los electrones pueden fluir a través de los chips y de los transistores está limitada por la resistencia y la capacidad de los cables metálicos que los conectan. Con el advenimiento de chips cada vez más pequeños, los cables se encuentran más

$$I = \frac{V}{R}$$

donde: I = corriente en amperes (A)
V = voltaje en volts (V)
R = resistencia en ohms (Ω)

juntos y las interconexiones se vuelven más delgadas, aumentando de esta forma, tanto la capacidad como la resistencia. Esta latencia se encuentra guiada por la Ley de Ohm.

c. Rendimiento y latencia en las memorias: como se mencionó anteriormente, si la memoria, o sus buses no logran procesar a tiempo los pedidos del procesador, sus recursos se ven a su vez desaprovechados.

Los temas discutidos anteriormente pueden sintetizarse en la siguiente gráfica:

Ilustración 5. Ley de Ohm. En Principios de circuitos eléctricos (p.73), por L. Floyd, T., 2008,

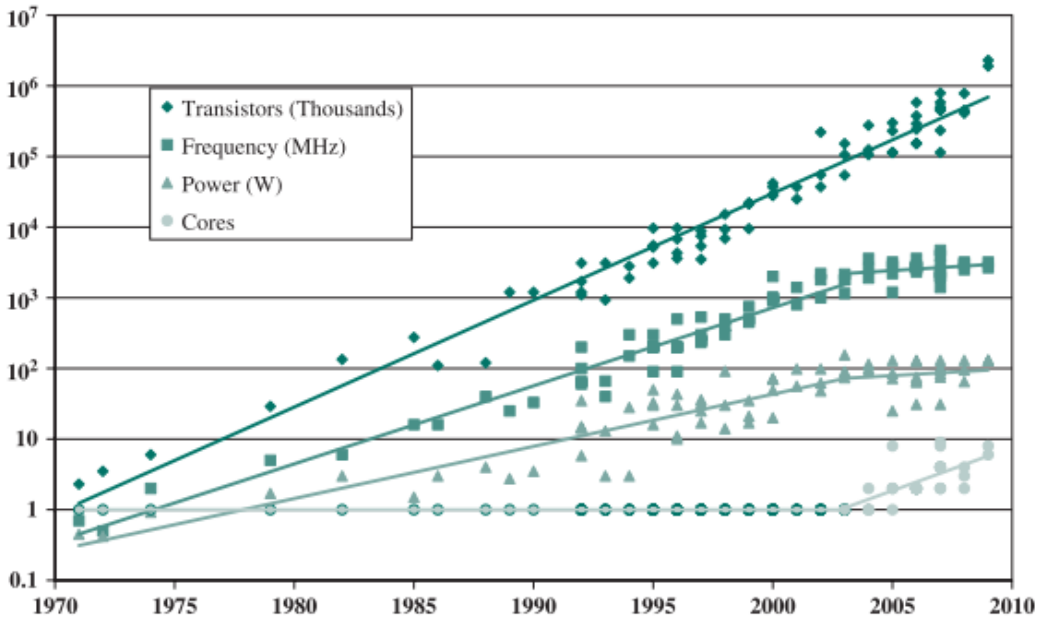


Ilustración 6. Processor Trends. En Computer Organization and Architecture (p.51), por Stallings, W., 2010,Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

2.3. Contexto del procesador.

Los procesadores ejercitan sus diferentes componentes por medio de instrucciones. Estas instrucciones deben estar acorde con el diseño del procesador a nivel de microarquitectura. Los siguientes apartados exploran esta relación.

2.3.1. Generalidades sobre los sets de instrucciones.

Según Morris Mano, M. (2005), la frontera existente entre el diseño de una computadora y su programación reside en el set de instrucciones que utiliza, este set corresponde al conjunto de instrucciones que un microprocesador puede comprender. Estas instrucciones se encuentran escritas en formato binario. Sin embargo, usualmente se utilizan nemónicos para facilitar su utilización. ADD para sumar, SUB para restar, MUL para multiplicar, DIV para dividir, LOAD para traer datos de la memoria principal y STOR para almacenar datos. A diferencia de los lenguajes de alto nivel que utilizan operaciones algebraicas para manipular los datos, el lenguaje máquina consiste en movimientos de datos desde o hacia diversos registros.

De hecho, el diseño de un set de instrucciones define gran parte de las funciones que realizará el microprocesador y tiene un efecto directo sobre la forma en que será posible implementar el procesador. Por esto, al diseñar un set de instrucciones deberán tomarse en cuenta aspectos como:

a. El repertorio de operaciones posibles: cuántas, cuáles y qué tan complejas serán.

b. Registros: número de registros del procesador y el uso que tendrán.

c. Formas de direccionamiento: modo por medio del que se especifica el lugar donde se encuentran los operandos.

d. Tipos de datos: se deberán definir los tipos de datos que podrán manipularse a través de operaciones. Entre los más utilizados se encuentran: números, caracteres, datos para efectuar operaciones lógicas y direccionamientos.

Asimismo, los microprocesadores deben ser capaces de soportar operaciones tales como de transferencia de datos, aritméticas, lógicas, de conversión (E/S), control y transferencia de control. El siguiente cuadro describe los tipos de operaciones usualmente soportadas por los procesadores.

2.3.2. El dilema entre CISC vs RISC.

El tamaño del set de instrucciones está relacionado también con el tamaño de la memoria, su organización, estructura de buses y la complejidad del procesador mismo.

De hecho, según D. George (1990), por muchos años, la tendencia en el diseño de microprocesadores estuvo marcada por una cantidad pequeña de registros, modos de direccionamiento e instrucciones complejas y largas. Estas tendencias dieron origen al Set de Instrucciones Complejas de Computadora (CISC, por sus siglas en inglés), esta tendencia de diseño puede rastrearse hasta el 80486 y al 8080 de Intel.

Sin embargo, los avances en memorias semiconductoras, registros, las optimizaciones en la tecnología de los compiladores se ha hecho cada vez más posible el diseño de microprocesadores construidos para comprender instrucciones simples, mejorando, en muchos casos, el tiempo de ejecución y el rendimiento de estos. Esta tendencia es conocida como Set Reducido de Instrucciones de Computadora (RISC, por sus siglas en inglés).

El cuadro que se expone a continuación muestra las principales diferencias entre ambos abordajes.

Tabla 6. Comparación entre CISC y RISC, elaboración propia.

CISC	**RISC**
Número reducido de registros.	Registros de memoria grandes.
Instrucciones de tamaño variable.	Casi todas las instrucciones se ejecutan en un único ciclo de reloj.
Más formas e instrucciones de acceder para memoria.	Las operaciones para el tratamiento de memoria LOAD/STORE son simples.
Muchos y sofisticados modos de direccionamiento.	Pocos modos de direccionamiento a memoria.
Set de instrucciones más grande.	Pocas instrucciones.
Lógica microprogramada	Lógica de cableado
Instrucciones de tamaño variable que pueden sobrepasar el tamaño de una palabra.	Instrucciones no sobrepasan el tamaño de una palabra.

2.3.3. Una vista al ciclo de una instrucción.

Para Gómez, J. (2012), el procesamiento de una instrucción regular consiste básicamente de la búsqueda de una instrucción en memoria, seguida de la ejecución de dicha

instrucción. Usualmente el ciclo comienza cuando el Contador de Programa (PC, por sus siglas en inglés) carga la siguiente instrucción que se buscará en memoria, el contador de programa incrementará una instrucción a la vez. La instrucción recuperada de memoria se albergará en el Registro de Instrucciones (IR, por sus siglas en inglés), posteriormente, el procesador interpretará la instrucción y efectuará la acción que se requiera.

2.3.3.1. Interrupciones.

Según Plaza Carrasco & Cozar (2001), en algunas ocasiones se dan eventos llamados interrupciones que modifican el flujo normal del procesador.

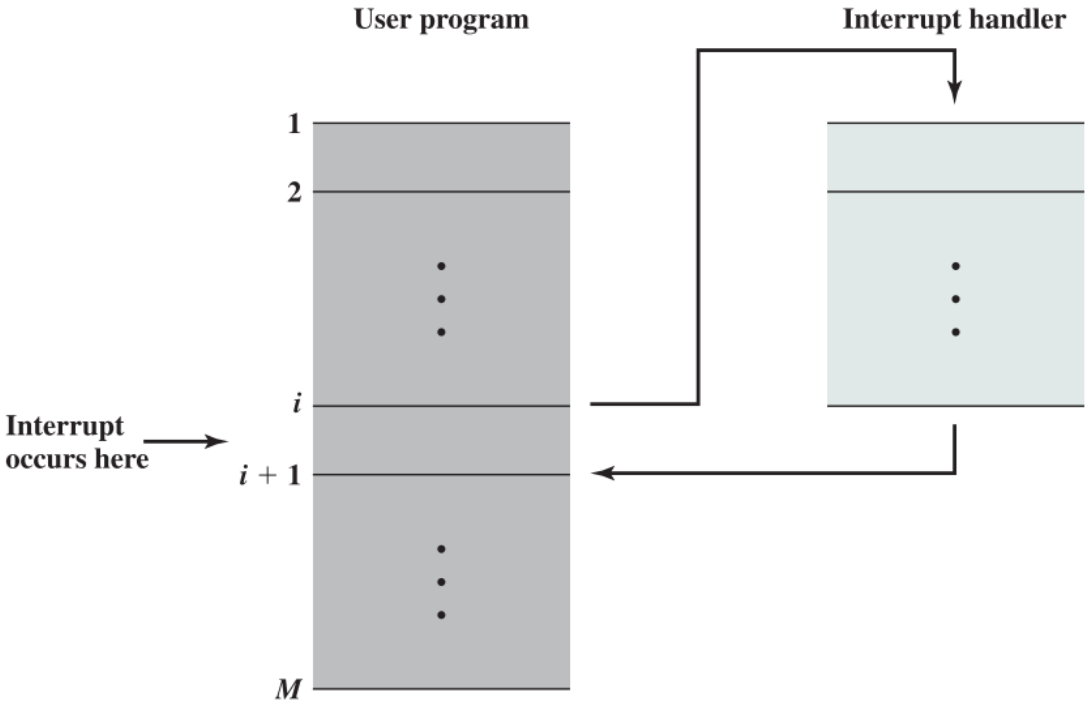


Ilustración 7.Computer. Transfer of control via interrupt. En Computer Organization and Architecture (p.92), por Stallings, W, 2010,Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

Estas interrupciones pueden ser de varios tipos:

a. De programa: Se dan durante la ejecución de instrucciones como lo son divisiones entre cero, intentos por ejecutar instrucciones no permitidas y desbordamientos aritméticos.

b. De tiempo: Generadas por temporizadores desde el procesador.

c. Se generan a través de los controladores de operaciones E/S.

d. Fallos de hardware: fallos en la alimentación y paridad de memoria.

Cuando ocurre una interrupción, se suspende el programa en ejecución, se salva la dirección de la siguiente instrucción que se ejecutará del programa, se ejecuta la rutina de interrupción y finalmente, continúa con el flujo de ejecución del programa guardado.

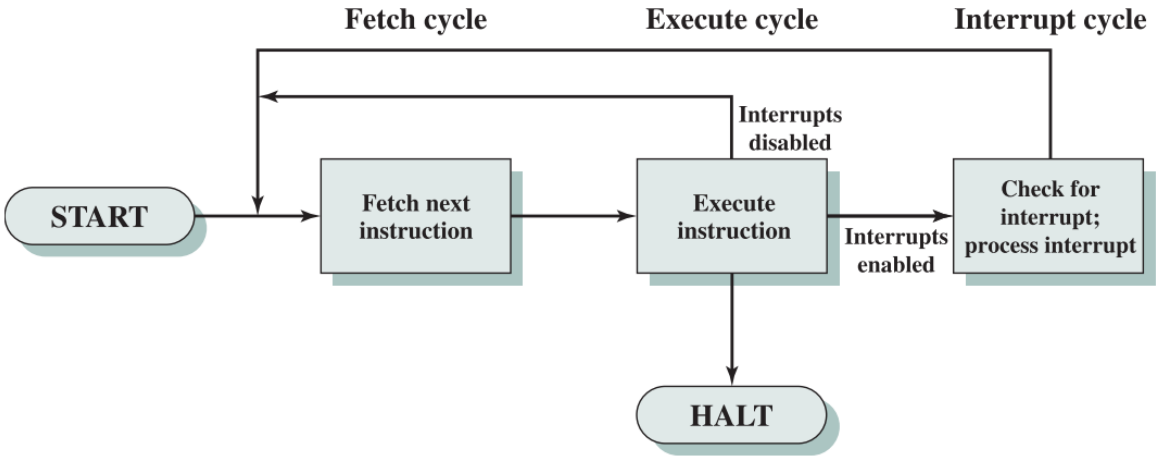


Ilustración 8. Instruction Cycle with Interrupts. En Computer Organization and Architecture (p.92), por Stallings, W, 2010,Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

2.3.3.2 Ciclo indirecto

Según Stallings, W. (2010), en algunos casos, la ejecución de una instrucción necesita de uno o más operandos adicionales que se encuentran en la memoria principal, en estos casos por lo que se debe realizar lo que se conoce como un ciclo indirecto. Este ciclo consiste en alternar la carga y la ejecución de una instrucción. De esta forma, justo después de que una instrucción es obtenida de memoria, se observa en caso de que su ejecución conlleve algún acceso a otros operandos en memoria principal. De ser así, estos son obtenidos por medio de direccionamiento indirecto antes de ejecutar la siguiente instrucción.

2.3.4. Otras posibilidades para el flujo de instrucciones y datos.

En la sección anterior se hizo referencia a las interacciones entre el procesador y la memoria. Sin embargo, hay otros tipos de interacciones posibles. En este apartado se discuten otras posibles interacciones en relación con el procesador.

2.3.4.1. Funciones E/S.

Stallings, W. (2010), Los módulos E/S pueden intercambiar datos directamente con el procesador. Esto quiere decir que, así como es posible leer y escribir datos desde y hacia memoria, el procesador también es capaz de leer y escribir datos desde los módulos E/S. En este caso, el procesador es capaz de identificar un dispositivo a través del módulo E/S e iniciar un ciclo de instrucciones similar al ciclo de memoria desde ahí.

2.3.4.2. Estructura de interconexión.

Desde el punto de vista de Stallings, W. (2010), a partir de los tres módulos básicos de una computadora: procesador, memoria y E/S, se puede rastrearla forma en que se comunican entre sí los distintos componentes. Una estructura de interconexión debe soportar los siguientes tipos de trasferencias de datos:

a. Memoria a procesador: Necesarios cuando el procesador necesita hacer lecturas desde memoria.

b. Procesador a memoria: Se utilizan cuando el procesador necesita escribir datos en memoria.

c. E/S a procesador: Cuando el procesador necesita leer datos desde un dispositivo a través de un módulo de E/S.

d. Procesador a E/S: Cuando el procesador envía datos a un módulo de E/S.

e. E/S a o desde memoria: los módulos de E/S pueden intercambiar datos directamente con memoria, sin necesariamente tener comunicación con el procesador.

Para hacer efectiva esta comunicación el sistema usualmente cuenta con interconexiones de buses y estructuras de interconexión punto a punto.

2.3.4.2.1. Interconexiones de buses.

Según Barrachina Mir et al. (2018), los buses son vías de interconexión entre los dispositivos y un medio compartido para la transmisión de datos. Usualmente, hay muchos dispositivos conectados a un mismo bus y sus señales son accesibles para cualquiera de los dispositivos conectados a éste. Sin embargo, si varios dispositivos emiten señales al mismo tiempo, éstas se traslapan, sólo un dispositivo puede transmitir exitosamente señales a la vez.

A pesar de que el número de líneas depende de la arquitectura. Hay tres tipos básicos de buses que los sistemas comparten:

a. Buses de sistema: son los buses que conectan a los componentes principales de la computadora (procesador, memoria y E/S).

b. Buses de datos: permiten el movimiento de los datos a través de los módulos del sistema.

c. Buses de direcciones: Transportan las fuentes de los datos y su destino.

d. Buses de control: regulan el acceso a los buses de datos y de direcciones.

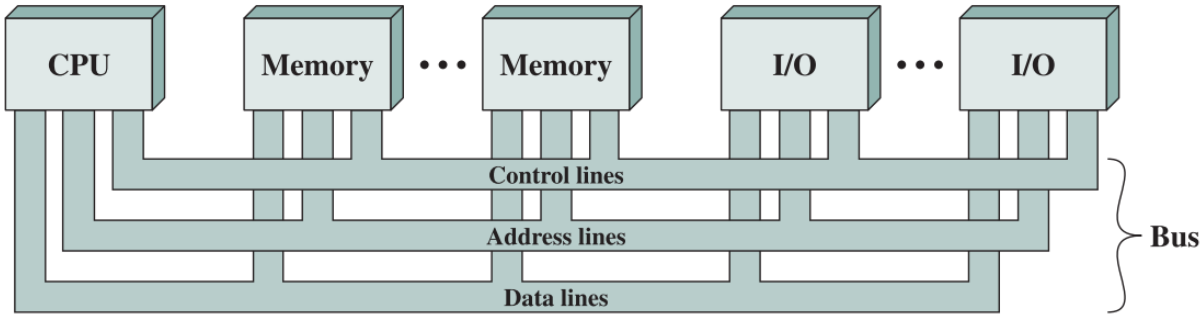


Ilustración 9. Bus Interconection Scheme. En Computer Organization and Architecture (p.101), por Stallings, W, 2010, Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

2.3.4.2.2. Interconexión punto a punto.

Para Stallings, W. (2010), a pesar de que las interconexiones de buses fueron a forma de interconexión dominante durante varias décadas, el incremento de la dificultad para sincronizar y arbitrar funciones cuando se necesitan son el escenario para la aparición del

Multicore, que, así como la posibilidad de aumentar la cantidad de memoria en un mismo chip, marcó el auge de las intercomunicaciones punto a punto. Tres de sus principales características son las siguientes:

a. Múltiples conexiones directas: en múltiples casos, los componentes pueden comunicarse entre ellos sin necesidad del arbitraje antes proporcionado por los buses de control.

b. Arquitectura de múltiples capas: permite la división de transporte de datos por capas. Un ejemplo de este tipo de arquitectura es el *Self Defined Communication Protocol*, que utiliza una capa de transporte, responsable de la transmisión de mensajes de extremo a extremo (Software – Hardware). (Li, Ruan, Li, Li, & Liao, 2009)

c. Transmisión de datos empaquetados: en vez de enviar los datos en secuencias de bits, esto se hace por medio de secuencias de paquetes.

2.3.4.2.3. Interconexión de componentes periféricos (PCI).

Para Morris Mano, M. (2005), la interconexión de componentes periféricos es un bus muy utilizado con mucho ancho de banda. Este bus es independiente del procesador y puede funcionar como un bus periférico, ofreciendo un más alto rendimiento del sistema cuando se trabaja con dispositivos de E/S de alta velocidad, como es el caso de adaptadores de pantalla gráfica e interfaz de red.

2.3.5. Consideraciones acerca del Sistema Operativo.

Según Milenkovic, M. (1996), el sistema operativo es un programa que controla la ejecución de aplicaciones, funcionando como una interfaz entre las aplicaciones y el hardware de la computadora. Asimismo, el Sistema Operativo procura que los recursos de la computadora sean utilizados de forma más eficiente.

Las computadoras en la actualidad delegan el manejo del hardware en aplicaciones de sistema llamadas bibliotecas y utilidades. Estos programas funcionan como auxiliares en la ejecución de diversas tareas recurrentes. En este sentido el Sistema Operativo, es la aplicación de sistema más utilizada.

El sistema operativo es auxiliar en las siguientes tareas:

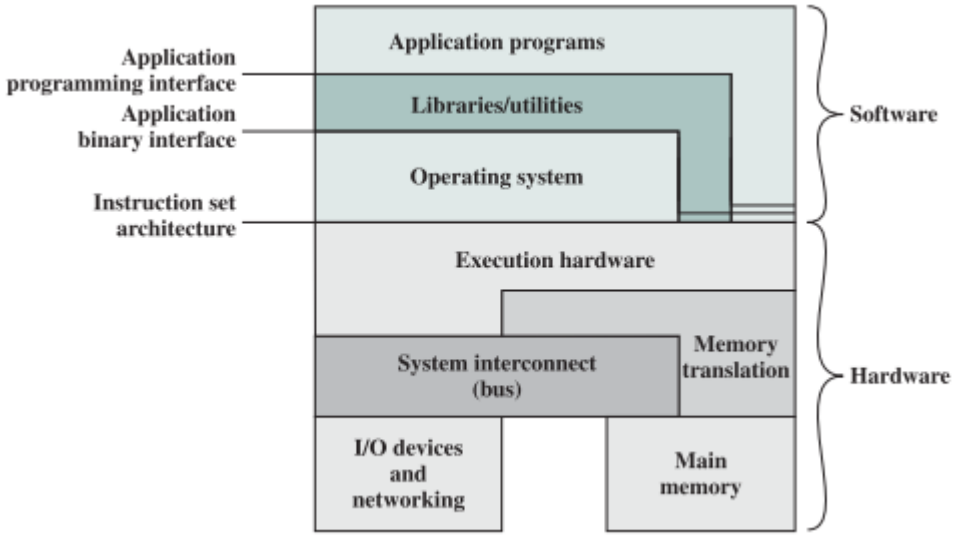


Ilustración 10. Computer Hardware and Software Structure. En Computer Organization and Architecture (p.277), por Stallingd, W, 2010, Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

a. Creación de programas: provee de editores y depuradores.

b. Ejecución de programas: manipulación de datos e instrucciones desde y hacia memoria, dispositivos de E/S y archivos.

c. Acceso a dispositivos de E/S: Cada dispositivo de E/S tiene su propio set de instrucciones y señales de control.

d. Controla el acceso a archivos: el formato de los archivos y permisos de acceso.

e. Acceso al sistema: protege los datos de forma que no sean accesados por usuarios no identificados.

f. Detección y respuesta a errores: control de errores de hardware, software, memoria, E/S.

g. Contabilización: recolecta estadísticas y monitorea parámetros de rendimiento.

El Sistema Operativo es un programa ejecutado desde el procesador. Además, cede el control al procesador y depende de éste para obtenerlo de nuevo. De esta forma, el Sistema Operativo le indica al procesador la forma en que debe hacer uso de los recursos de la computadora, luego, deja en control al microprocesador para que ejecute otros programas y finalmente retoma el control para indicarle la siguiente labor a realizar.

De esta forma, una porción del Sistema Operativo se encuentra en la memoria principal. En esta poción se encuentra el kernel, que contiene las funciones que el Sistema Operativo necesita más frecuentemente y las partes del Sistema Operativo que están siendo utilizadas. El Sistema Operativo decide el momento en el que un programa en ejecución puede utilizar un dispositivo de E/S y controla la utilización de archivos. Visto desde esta perspectiva, el procesador es un recurso más que el Sistema Operativo debe controlar, de forma tal que debe decidir cuánto tiempo dedicará a la ejecución de un programa de usuario.

2.3.6. Sistema de memoria

Para Stallings, W. (2010), el dilema existente en los diseñadores radica en una negociación entre la capacidad de la memoria y el costo de ésta. Por este motivo se ha recurrido a la construcción de un modelo de memoria que, visto de forma descendente, permite visualizar el siguiente comportamiento:

a. Disminuye el costo por bit.
b. Se incrementa su capacidad.
c. Se incrementa el tiempo de acceso a memoria.
d. Disminuye el número de accesos de procesador a memoria.
e. Existe mayor cercanía al procesador.

Dicho en otras palabras, cuanto más pequeña es una memoria, más rápida resulta; por el contrario, cuanto más grande, más incrementa el tiempo necesario para acceder a ésta.

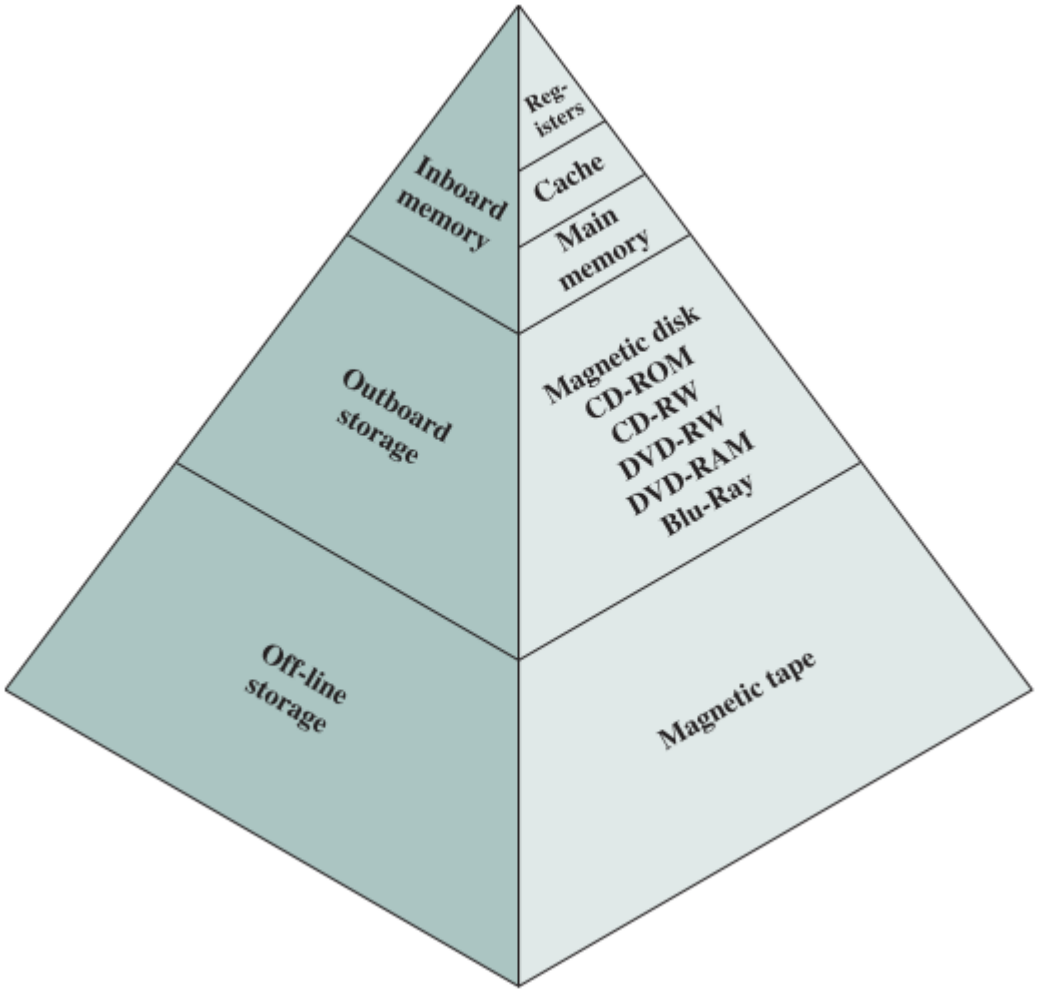


Ilustración 11. The Memory Hierarchy. En Computer Organization and Architecture (p.125), por Stallings, W, 2010, Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

2.4. Estructura y funcionamiento de un procesador.

Tal y como lo menciona Stachniak, Z. (2013), un procesador cualquiera debe ser capaz de cargar instrucciones de memoria, interpretarlas, procesar los datos y escribir en memoria. Para lograrlo cuenta necesariamente con una ALU, encargada del procesamiento de datos y una UC, que controla el movimiento de datos e instrucciones hacia y desde el microprocesador, así como las operaciones de la ALU. Además, cuenta con registros para

almacenar datos en su interior. Estos componentes se conectan con el exterior por medio de buses.

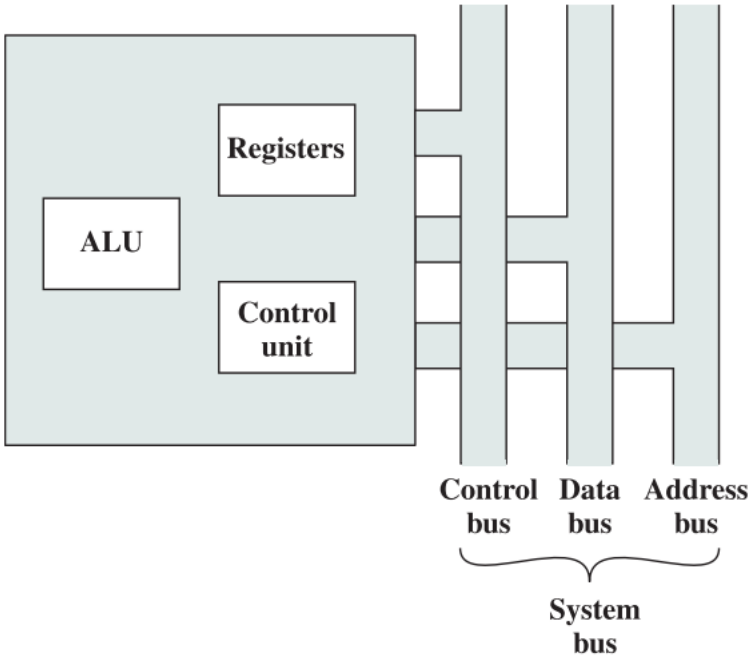


Ilustración 12. The CPU with the System Bus. En Computer Organization and Architecture (p.490), por Stallings, W, 2010, Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

Uno de estos buses se conoce como *Bus Interno del CPU,* este bus tiene la función de transferir datos entre los registros y la ALU, ya que ésta sólo puede operar con datos que se encuentren en la memoria interna del procesador. Algunos componentes básicos de la ALU son las banderas de estado, variador, complementador y una unidad de lógica tanto aritmético-lógica como booleana. Los elementos de E/S, memoria, unidad de control y ALU se conectan entre sí por medio de rutas de datos.

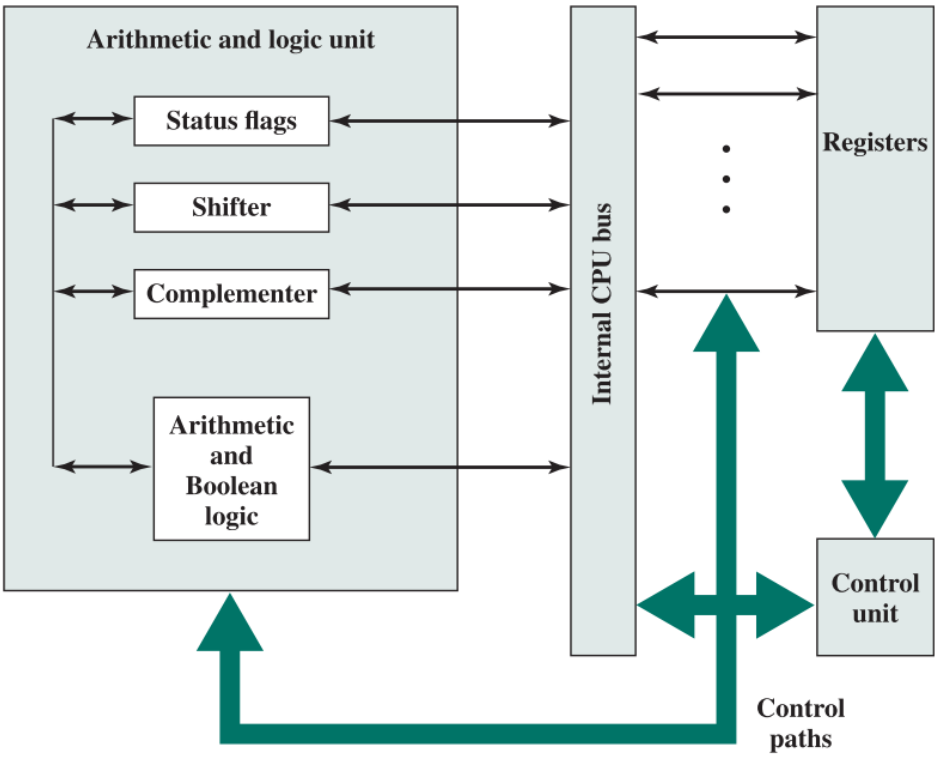


Ilustración 13.Internal Structure of the CPU. The CPU with the System Bus. En Computer Organization and Architecture (p.491), por Stallings, W, 2010, Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

Por otra parte, los registros en un microprocesador pueden tener dos roles:

a. Registros visibles para el usuario: permiten que la computadora o quien programe en lenguaje ensamblador minimice las referencias hacia la memoria principal por medio de la optimización del uso de los registros.

b. Registros de control y estatus: son utilizados por la UC para controlar el comportamiento del microprocesador y controlar la ejecución de programas a través del sistema operativo.

2.4.1. Registros visibles para el usuario.

Desde el análisis de Stallings, W. (2010), los registros visibles para el usuario pueden a su vez dividirse en:

a. Registros de propósito general: con algunas excepciones, le puede ser asignado un amplio espectro de funciones. En algunos casos estos registros se dividen en *Registros de Datos*, que pueden ser utilizados únicamente para guardar datos y *Registros de Direcciones*, que se destinan a un modo particular de direccionamiento, como es el caso de los punteros de segmento, registros de índice y punteros de pila.

b. Código de estado: también conocidos como banderas, son un set de bits que se fijan desde el hardware del procesador como resultado de una operación. Estos códigos son especialmente útiles en operaciones de ciclos condicionales

En el diseño de microprocesadores, se debe decidir entre usar solo registros de propósito general, ya que, de utilizarse registros especializados, se hace implícito a qué tipo de registro se refiere un especificador de operando determinado, de tal forma que la búsqueda del especificador de operando se reduce a un conjunto de registros especializados, pero a su vez, esta especialización obliga a quien programa a utilizar solamente registros especializados.

Otro dilema que debe enfrentarse resulta del número de registros. Más registros implicarán a su vez más bits de dirección de operando (entre 8 y 32 parece ser el número más adecuado según las investigaciones hasta la fecha), pero menos registros conllevan a su vez más referencias a memoria.

Por último, también hay decisiones que tomar en relación con el tamaño de los registros. En el caso de los registros que contienen instrucciones, estos deben ser al menos lo suficientemente grandes como para albergar la instrucción más grande.

2.4.2. Registros de estado y control.

Según Morris Mano, M. (2005), los registros de estado y control son los registros utilizados por la CU con el fin de regular las operaciones del procesador. Los registros de estado y control más utilizados son:

a. Contador de programa (PC, por sus siglas en inglés): Contienen la siguiente dirección de memoria a ser ejecutada.

b. Registro de Instrucción (IR, por sus siglas en inglés): contiene la instrucción más reciente cargada de memoria.

c. Registros de direccionamiento a memoria (MAR, por sus siglas en inglés): Contienen una dirección de memoria.

d. Registro de búfer de memoria (MBR, por sus siglas en inglés): Contiene una palabra de datos que será leída o la que fue leída más recientemente.

Los procesadores que no tienen registros de tipo MAR y MBR tienen mecanismos similares para llevar a cabo los mismos procesos.

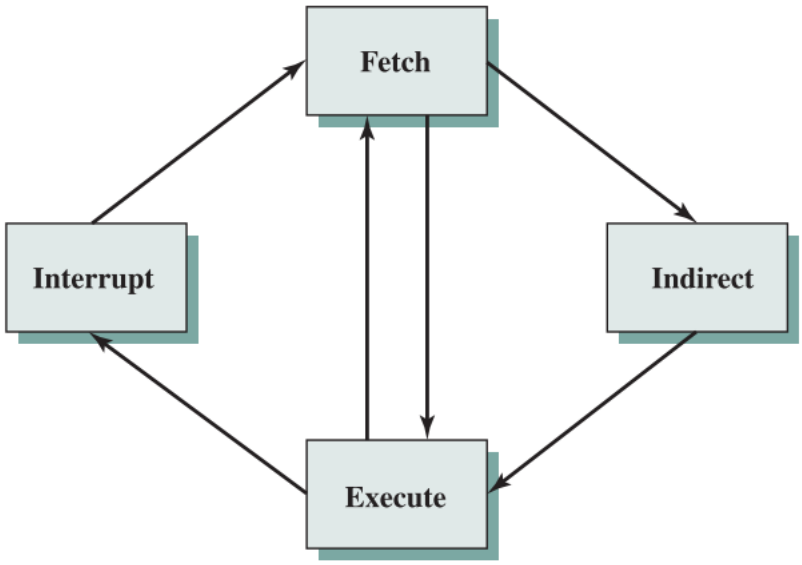


Ilustración 14. The instruction cycle. The CPU with the System Bus. En Computer Organization and Architecture (p.497), por Stallings, W, 2010, Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

En un escenario cotidiano, el flujo entre los registros mencionados anteriormente sería el siguiente: el microprocesador actualiza el PC después de cada dirección obtenida desde memoria, de forma tal que el PC siempre apunte a la siguiente instrucción a ser ejecutada. La instrucción obtenida de memoria es cargada en el IR, en donde el *opcode* y el especificador de operando son analizados. Posteriormente hay intercambios con memoria a través de MAR y MBR. En un sistema organizado a través de buses, MAR se conecta directamente con el bus de direcciones y MBR se conecta directamente con el bus de datos para poder efectuar estos intercambios. Asimismo, los registros visibles para el usuario intercambian datos con el MBR. En el interior del procesador, la ALU también debe tener acceso a los MBR y a los registros visibles para el usuario para el intercambio de datos.

Visto de otra forma, aunque la secuencia exacta de eventos depende en gran medida del diseño del procesador, en términos generales, al emplearse un MAR, un MBR, un PC y un IR, durante el ciclo de captación la PC contendrá la próxima dirección a ser captada de memoria, esta dirección será movida al MAR y será colocada en el bus de direcciones. La UC solicitará una lectura de memoria cuyo resultado se colocará en el bus de datos, copiado en el MBR y trasladada al IR. En tanto que la PC se incrementará en 1 preparándose para la siguiente captación. Una vez que finaliza el ciclo de captación, la unidad de control revisa

el contenido de IR con el fin de determinar si contiene algún especificador de operando que necesite de un direccionamiento indirecto.

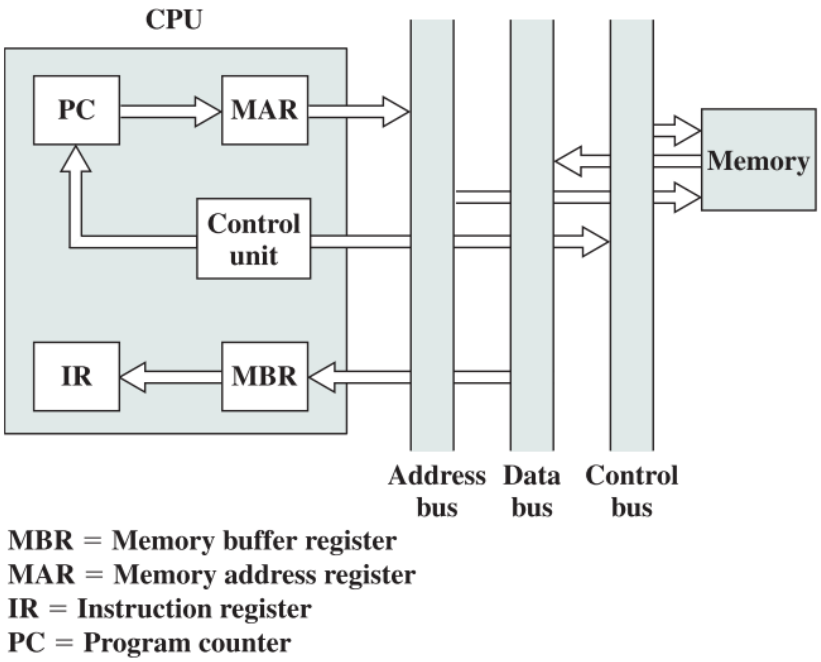


Ilustración 15. Data Flow, Fetch Cycle. The CPU with the System Bus. En Computer Organization and Architecture (p.498), por Stallings, W, 2010, Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

2.4.3. Una mirada más profunda a los procesos de segmentación en los procesadores.

Según Parhami, B. (2007), la segmentación puede compararse con una línea de ensamblaje en una planta de manufactura. Según este razonamiento, así como en las líneas de ensamblaje, los productos pasan por una línea de producción que contiene varias etapas, la ejecución de instrucciones también comprende una serie de etapas, que de forma muy general son la captación de la instrucción y su ejecución.

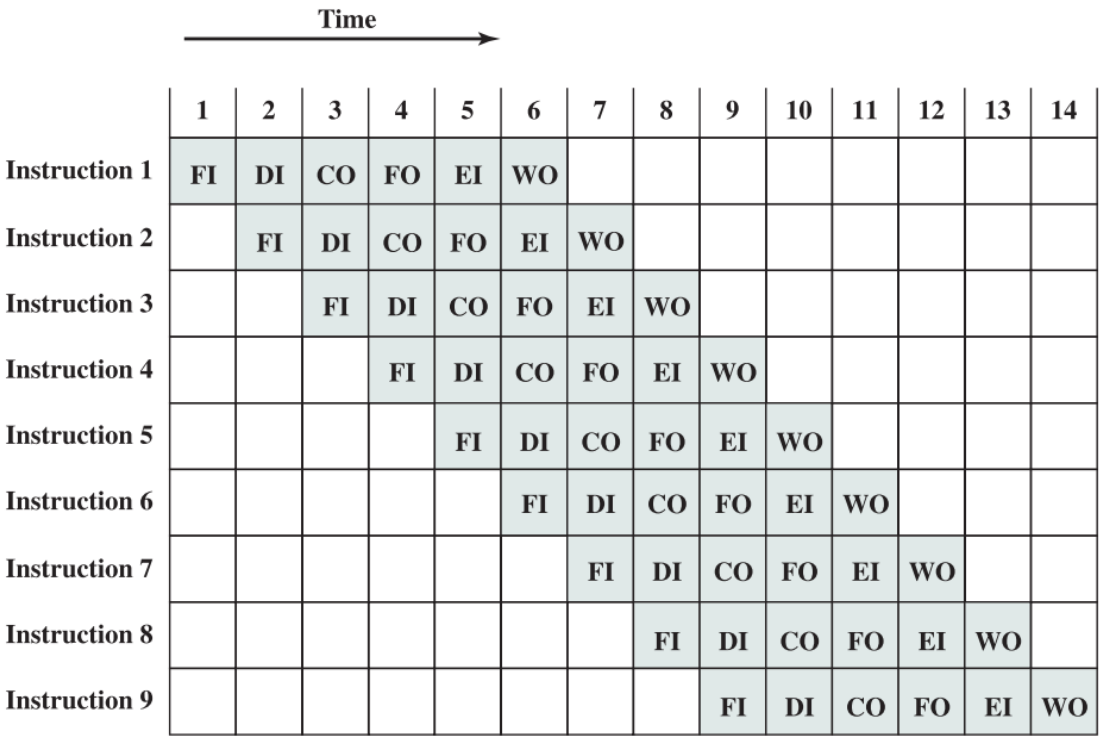


Ilustración 16. Timing Diagram for Insdtruction Pipeline Operation. En Computer Organization and Architecture (p.502), por Stallings, W, 2010, Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

De esta forma, existen lapsos de tiempo en los que la memoria principal no está siendo accesada. Este tiempo podría utilizarse para captar una nueva instrucción, paralelamente a la ejecución de la primera. Partiendo de este análisis se puede deducir que al tener el proceso de captación y de ejecución la misma duración, este proceso disminuiría el tiempo de ejecución de instrucciones a la mitad. Sin embargo, esta condición no se cumple por los siguientes motivos:

a. El tiempo de ejecución suele ser mayor que el tiempo de captación, ya que involucra lecturas y escrituras de operandos y la realización de alguna operación.

b. Los saltos condicionales hacen que la dirección de la siguiente instrucción leída en memoria sea desconocida. Por esto, el ciclo de captación tendrá que esperar hasta recibir la dirección de la siguiente instrucción desde la etapa de ejecución. Una medida útil para solventar este problema es que, en el momento de ejecución de la instrucción, el ciclo de captación obtenga la dirección posterior al salto condicional, de esta forma, si el ciclo no se repite no se pierde tiempo en la

captación de la instrucción, pero si el ciclo se repite, la dirección debe ser descartada y se hace necesaria la captación de otra instrucción.

De estos acontecimientos surge la necesidad de que la segmentación sea mayor. Las siguientes líneas exponen un mecanismo común de segmentación:

a. Captación de instrucción: Lee la siguiente instrucción que espera en un búfer.

b. Decodificación de instrucción: Determina el o*pcode* y el especificador de operando.

c. Cálculo de operandos: cálculo de la dirección efectiva de la dirección en memoria cada operando.

d. Captación de operandos: carga operandos desde memoria.

e. Ejecución de instrucción: realización de la operación indicada y almacenamiento de resultado en la ubicación especificada.

f. Escritura de operando: Almacenamiento de resultado en memoria.

En un caso ideal, una segmentación de seis estados, el proceso de ejecución de instrucciones se reduciría en 14 unidades de tiempo. Sin embargo, no todas las instrucciones pasan por los seis estados ya que la duración de las etapas es variable, hay instrucciones de salto, conflictos de memoria y no todas las instrucciones se pueden ejecutar paralelamente, generando penalidades y dando como resultado un escenario más similar a la siguiente ilustración.

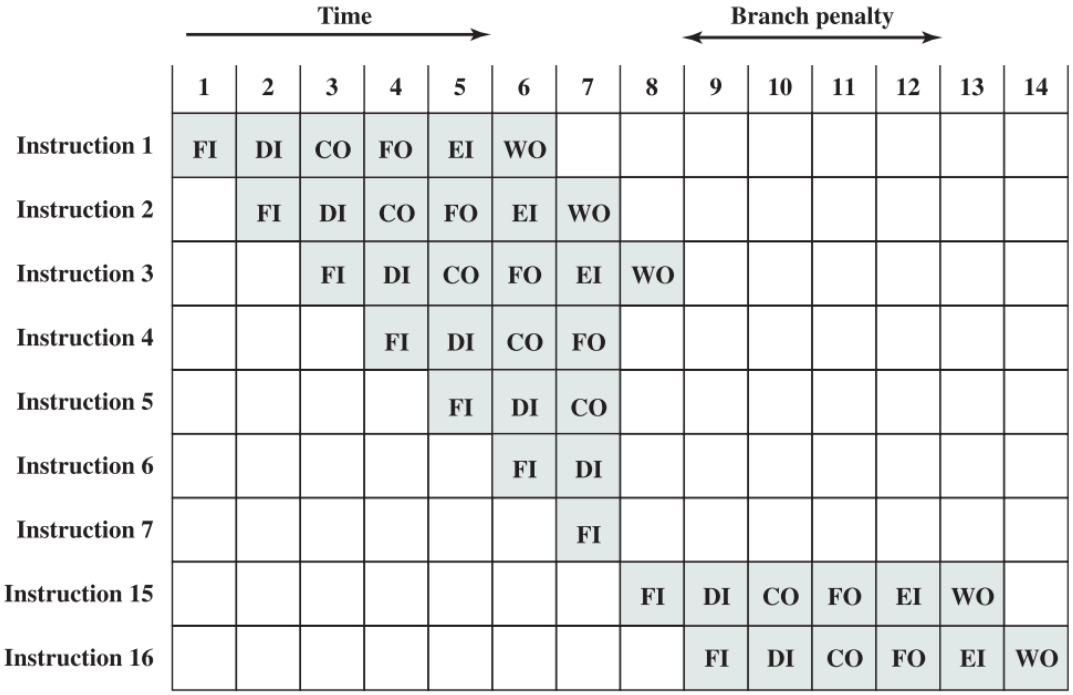


Ilustración 17. The Effect of a Conditional Branch on Instruction Pipeline Operation. En Computer Organization and Architecture (p.503), por Stallings, W, 2010, Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

2.4.3.1. Riesgos en la segmentación

De hecho, según Stallings, W. (2010), los bloqueos en las etapas de la segmentación han sido ampliamente estudiados y se conocen como riesgos de segmentación. En los párrafos que siguen se describen los riesgos que pueden llevar al bloqueo de alguna de las etapas durante la segmentación.

a. Riesgo de recursos: ocurre cuando dos o más instrucciones que están recorriendo las etapas de la segmentación necesitan el mismo recurso para continuar, provocando que las instrucciones deban ser ejecutadas de forma serial y no en paralelo.

b. Riesgo de datos: Ocurre cuando dos instrucciones necesitan accesar al mismo operando. Este es el caso por lo que ambas instrucciones deben ejecutarse de forma serial en vez de en paralelo, ya que, de ser ejecutadas en paralelo, el valor del operando puede no haber sido actualizado a tiempo y con ello provocar un resultado incorrecto.

c. Riesgo de control: también es conocido como riesgo de salto y ocurre cuando la segmentación toma la decisión incorrecta al predecir el resultado de un salto e introduce una instrucción en la segmentación, instrucción que posteriormente deberá ser descartada.

2.4.3.2. Segmentación y predicción de saltos.

Tal y como fue señalado en los apartados anteriores y lo menciona Parhami, B. (2007), la gran dificultad en relación con los saltos es que hasta el momento en que la instrucción es ejecutada, no puede saberse a ciencia cierta si el ciclo será repetido o no. Para lidiar con esta disyuntiva, se ha recurrido han implementado las siguientes alternativas:

a. Secuencias múltiples: Replicar el segmento de forma tal que se contemplen tanto la instrucción siguiente en caso de que el ciclo se repita, como, en el otro segmento, la instrucción que sigue en el caso de que el ciclo no se repita. Este tipo de implementación tiene la desventaja por un lado, de que implica retrasos en los accesos a memoria y registros; y por otra parte, la introducción de más instrucciones en las líneas de segmentación, implica a su vez que cada decisión implica una línea de secuencia adicional.

b. Carga anticipada de instrucción siguiente al bucle: la instrucción que el ciclo persigue para su ruptura es captada antes de que el ciclo sea ejecutado y se guarda hasta que la condición se cumpla. Cuando sucede, la instrucción ya no debe pasar por el ciclo de captación.

c. Búfer de bucle: un búfer de bucle es una memoria muy pequeña y de rápido acceso que se ubica en el segmento de captación de la instrucción, este búfer contiene las instrucciones captadas más recientemente en forma secuencial. Después de la salida del bucle, se corrobora que la siguiente instrucción esté en el búfer, en cuyo caso, la instrucción sería captada desde el búfer en vez de desde memoria.

2.5. Análisis de rendimiento.

Según Jain Raj (1991), el análisis de rendimiento es un aspecto vital en la industria de las computadoras, tanto usuarios como administradores y diseñadores de sistemas computacionales ya que su meta principal radica en lograr el mejor rendimiento al menor costo posible. Es por esto que los análisis de rendimiento son requeridos en todo el ciclo de producción de las computadoras. Por esto resulta sumamente importante comprender los pasos que deben seguirse, escoger las herramientas, medidas y técnicas más indicadas de acuerdo al caso en estudio.

En este sentido, es vital tener en claro una serie de principios que guían las evaluaciones de rendimiento, entre ellas:

a. Metas: No existe una herramienta de propósito general, es decir, que cumpla todas las necesidades o metas que se persigan en los casos de análisis de rendimiento particulares. Por ello es necesario comprender el funcionamiento del sistema con el que se trabajará, así como el problema, para establecer posteriormente las herramientas, métricas y pruebas de carga de trabajo a las que se someterá el sistema para cumplir con los objetivos.

b. Imparcialidad: No dar por sentado el resultado del estudio, ya que esto puede parcializar los resultados.

c. Aproximación sistemática: seleccionar parámetros, factores, métricas y cargas de trabajo de forma tal que el estudio, al ser replicado, obtenga los mismos resultados.

d. Comprensión del problema: Usualmente la definición del problema involucra un cuarenta por ciento del tiempo dedicado a un estudio, ya que el experimento diseñado no es un fin en sí mismo, sino un medio para el análisis y la obtención de conclusiones posteriores.

e. Métricas de rendimiento: las métricas, que son el criterio utilizado para cuantificar el rendimiento de un sistema específico, pueden proveer de resultados erróneos o inútiles en el análisis que se desarrolla; por ejemplo, una forma de comparar el rendimiento de dos CPUs es la cantidad de instrucciones que se ejecutan por segundo en millones (MIPS, por sus siglas en inglés). Sin embargo, comparar dos

arquitecturas diferentes, como lo podrían ser una RISC y una CISC, por medio de este parámetro no arrojaría un resultado útil, ya que, a partir del hecho de que las instrucciones en ambos sistemas son muy diferentes.

f. Selección de carga de trabajo: que debe estar acorde con el funcionamiento real del sistema. Por ejemplo, la carga de trabajo utilizada para comparar dos sistemas que transfieren paquetes largos y cortos, la carga de trabajo debe de estar compuesta de paquetes largos y cortos.

g. Técnicas de evaluación: en la actualidad hay tres técnicas básicas de evaluación: medición, simulación y modelos analíticos. La selección de la técnica a ser utilizada debe depender de las necesidades del experimento y la compañía por lo que quien analiza los datos debe tener un conocimiento general de las tres técnicas.

h. Parámetros: la lista de factores que pueden afectar el rendimiento de un sistema se llaman parámetros. Ignorar algún parámetro importante puede invalidar el experimento.

i. Factores: los parámetros que varían durante el estudio son llamados factores. Resulta indispensable llevar cuenta de los factores que tienen un impacto significativo en el rendimiento. Para este efecto, se utilizan análisis de sensibilidad por medio de los cuales puede observarse el efecto que tiene el cambiar los valores por defecto de los distintos parámetros involucrados en el proceso.

j. Diseño del experimento: en este debe definirse tanto el número de medidas o simulaciones a realizarse como los parámetros que se utilizarán en el experimento, que deben ser cambiados uno a uno para conocer el efecto de cada uno de ellos.

k. Nivel de detalle: dependerá del problema a solucionar. Usualmente, lo más adecuado resulta en un punto medio de detalle que no sea demasiado amplio ni demasiado pormenorizado.

l. Análisis: la obtención de datos es la base, pero no es la única parte del proyecto, el análisis de resultados, no debe pasarse por alto, sin importar qué tal exhaustivas han sido las técnicas de los experimentos.

m. Sensibilidad de los resultados: tomar el margen de error presente en las herramientas utilizadas como cargas de trabajo, así como otros parámetros del sistema en sí mismo, pueden llevar a resultados erróneos durante el análisis.

n. Datos atípicos: usualmente los datos atípicos no son causados por un fenómeno del sistema en sí mismo, por lo que deben descartarse. Sin embargo, si pueden ser evidencia de una ocurrencia real en el sistema, deben ser incluidos en el estudio.

o. Modificaciones en el futuro: no puede darse por sentado que los resultados del análisis serán útiles en el futuro. Por esto, es necesario definir el marco espacial en el que el experimento se realizó.

p. Complejidad del análisis: es preferible un análisis simple y fácil de explicar que uno complejo que llega a las mismas conclusiones que el simple.

q. Presentación de los resultados: siendo la finalidad de los resultados ser de utilidad para la toma de decisiones, es indispensable, no sólo que revele resultados útiles para este fin, sino que éstos sean comprensibles para la industria.

r. Aspectos técnicos y sociales: el estudio no sólo debe estar respaldado por bases técnicas sino también por habilidades sociales para la transmisión de ideas, de forma tal que los resultados tengan la mayor repercusión posible en la toma de decisiones.

s. Alcances y limitaciones: hacer explícitos los alcances y las limitaciones en el análisis permite que al replicar el análisis se puedan obtener resultados similares.

Una forma de tomar en cuenta los aspectos anteriores es seguir una serie de pasos que se exponen a continuación y son comunes a todos los proyectos de análisis de rendimiento:

a. Definición del problema y los objetivos: delimitar las fronteras del estudio en términos de hardware, software y objetivos. Si no es posible tener control sobre un aspecto en específico es mejor no tomarlo en cuenta en el estudio para no generar resultados erróneos.

b. Listar servicios y salidas del sistema en estudio: Todos los sistemas proveen una serie de servicios. Los procesadores, por ejemplo, pueden ejecutar una serie de instrucciones que pueden ser catalogadas, así como las salidas que pueden obtenerse de cada una de ellas.

c. Selección de métricas: implica la selección de criterios de rendimiento, relacionados con la velocidad, precisión y disponibilidad de los servicios.

d. Listar los parámetros: Se refiere a listar los parámetros que afectan en el rendimiento del sistema. Estos sistemas serán de dos tipos: parámetros de sistema y parámetros relacionados con la herramienta de carga de trabajo que será utilizada.

e. Selección de factores que serán estudiados: La elaboración de la lista de factores más relevantes se dividirá en dos clases: los parámetros que no van a variar durante el estudio y los que sí, es decir, los factores, estos factores obtendrán diferentes valores que se llamarán niveles. Es recomendable comenzar con un número de factores pequeño con un número limitado de niveles que podrán extenderse de acuerdo a los resultados obtenidos.

f. Selección de técnica de evaluación: ésta dependerá del tiempo y los recursos destinados al estudio para resolver el problema.

g. Selección de carga de trabajo: siendo una carga de trabajo, un número determinado de solicitudes de servicios a un sistema específico, la carga de trabajo se operacionalizará en un número de *scripts* que el sistema deberá ejecutar y que a su vez resulte en una muestra representativa de las tareas que un sistema realiza en la vida real.

h. Diseño de experimento: Una vez que se han definido los factores y sus respectivos niveles, se diseña una serie de experimentos capaces de ofrecer la mayor cantidad de información sobre el problema en la menor cantidad de tiempo posible.

i. Análisis e interpretación de datos: En este momento es vital conocer el grado de variabilidad de los resultados a través de mecanismos estadísticos. El análisis de los resultados debe estar elaborado para permitir que quien analiza los datos y quienes toman las decisiones puedan formular conclusiones con base en ellos.

j. Presentación de resultados: es el momento de la comunicación de los resultados, que se debe realizar de forma gráfica, evitando vocabulario técnico y estadístico que dificulten la comprensión de una audiencia no familiarizada.

2.5.1. Aspectos importantes en la selección de métricas y técnicas.

Desde el punto de vista de Leung, Joseph Y-T. (2011), la escogencia de la técnica de evaluación depende en gran medida del estadio del ciclo de vida en el que se encuentre el sistema a ser analizado. Por ejemplo, las mediciones sólo son posibles en sistemas existentes que ya han sido diseñados y hasta actualizados. Los modelos analíticos y las simulaciones, por su parte, se utilizan en los casos en que las mediciones no son posibles.

2.5.1.1 Variables que tomar en cuenta para la selección de técnicas.

Asimismo, para Jain Raj (1991), la selección de una técnica para un análisis de rendimiento no es antojadiza, sino que está ligada a los recursos con que cuenta la organización, herramientas disponibles, tiempo, credibilidad y comerciabilidad de esta. En las líneas siguientes se detallan estos aspectos:

a. Tiempo: Usualmente el modelado resulta en la técnica de menor duración. Por el contrario, las simulaciones suelen tardar demasiado. Las mediciones suelen encontrarse en el punto medio entre el modelado y la simulación. Sin embargo, vale la pena tomar en consideración que el tiempo de las mediciones puede aumentar o acortarse más frecuentemente que el resto de las técnicas, debido a los hallazgos que pueden arrojar.

b. Herramientas disponibles: es indispensable recabar acerca de las herramientas existentes de modelado, lenguajes de simulación y herramientas de medición disponibles en el mercado y la institución.

c. Nivel de precisión: usualmente los modelos analíticos requieren tantas suposiciones y simplificaciones que su precisión es sumamente baja. Las simulaciones requieren menos suposiciones y simplificaciones por lo que sus resultados son más acercados. Aunque las mediciones pueden considerarse el mecanismo más preciso, esto no es necesariamente cierto, ya que sus resultados dependen de muchos parámetros como la configuración del sistema, el tipo de carga de trabajo, el lapso de la medición, factores que pueden variar de una medición a otra.

d. Costo del proyecto: en este caso, las mediciones son la técnica más costosa de todas, ya que implican equipo especializado, instrumentos y tiempo. De hecho, el costo y la facilidad para la fijación de configuraciones específicas son las razones principales por las que se utilizan simulaciones. Por otra parte, los materiales necesarios para efectuar un modelo analítico se limitan a lápiz y papel, por lo que es considerada la técnica menos costosa.

e. Comerciabilidad: El factor clave para la selección de una técnica sobre las demás es la facilidad de hacer los resultados asequibles a quienes se encargan de la toma de decisiones. En este sentido, las mediciones son la técnica predilecta, ya que están hechas sobre un sistema en funcionamiento. Además, los resultados obtenidos por medio de modelado o simulación provocan desconfianza, ya que los aspectos técnicos que les dan credibilidad son difíciles de comprender para la población no especializada.

En lo que respecta a la verificación de los resultados se recomienda la utilización de dos técnicas de acuerdo a los siguientes principios:

a. No confiar en los resultados arrojados por un modelo de simulación, si estos no han sido validados por un modelo analítico o mediciones.

b. No confiar en un modelo analítico si este no está respaldado por una simulación o mediciones.

c. No confiar en los resultados arrojados por mediciones si no han sido validados por un modelo analítico o una simulación.

2.5.1.2 Selección de métricas

Como se ha hecho notar en los párrafos anteriores, los criterios de rendimiento se escogen de acuerdo a los servicios que brinda el sistema. Estos criterios se clasificarán en tres posibles categorías: ejecución correcta del servicio, ejecución incorrecta del servicio o la ejecución del servicio fue denegada.

Jain Raj (1991), propone la clasificación que se muestra en la ilustración *n.18*.

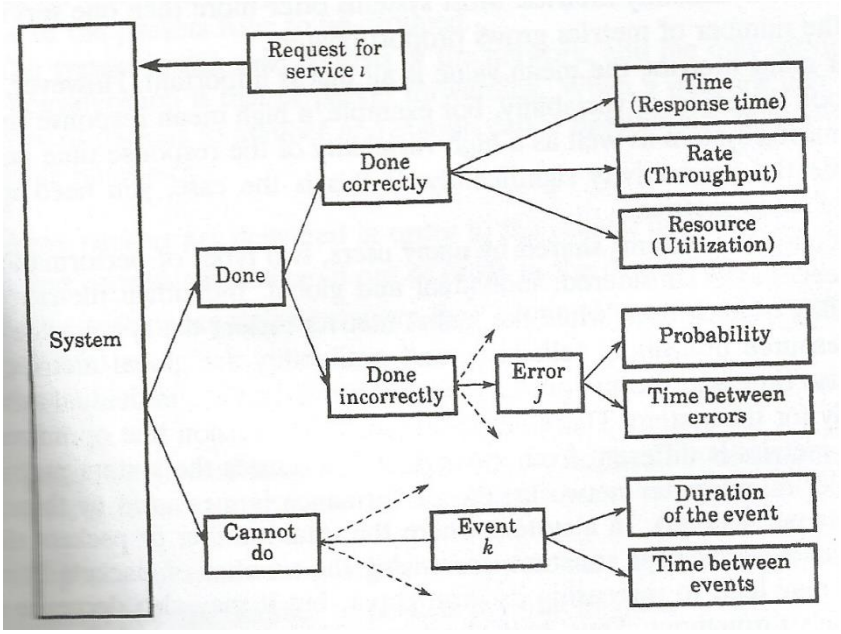


Ilustración 18. The possible outcomes of a service request. En The art of computer systems performance analysis (p.33), por Jain, R., 1991, New York: Willey. Derechos de autor [1991] por WILEY. Reimpresión autorizada.

Siguiendo esta categorización, de efectuarse correctamente el servicio, se podrá medirse según el tiempo en el que se efectuó, la velocidad con que se ejecutó el servicio y los recursos consumidos durante su ejecución. Estas métricas se conocen respectivamente como: capacidad de respuesta, productividad y utilización. La capacidad de respuesta está asociada al tiempo de respuesta, o al intervalo desde que la petición llega, hasta que es completada de forma satisfactoria. La productividad, por su parte, se referirá al número de pedidos que pueden ser ejecutados por unidad de tiempo y finalmente, la utilización se refiere al tiempo en porcentaje, mediante el cual, los recursos fueron utilizados para ejecutar el pedido. Asimismo, el recurso con mayor porcentaje de utilización se llamará cuello de botella. Si el pedido es efectuado de forma incorrecta provocará la aparición de un error. Por otra parte, si el sistema da el servicio del todo, se dice que no está disponible o está caído.

2.5.1.3 Tipos de cargas de trabajo.

Según Jain Raj (1991), el término carga de trabajo de prueba se refiere a cualquier carga de trabajo utilizada para la realización de estudios. Hay dos tipos de cargas de trabajo:

a. Cargas de trabajo reales: se refiere a la carga de trabajo que cualquier sistema tiene en sus operaciones cotidianas de usuario por lo que no puede replicarse

ni ser utilizada como una medida sistemática para ser utilizada como una carga de trabajo de prueba.

b. Carga de trabajo sintética: emulan al usuario real pero pueden ser replicadas en un ambiente controlado y por tanto pueden utilizarse como cargas de trabajo de prueba.

Los apartados que siguen se refieren a las cargas de trabajo que según Jain Raj (1991), se han diseñado y utilizado para comparar sistemas computacionales.

2.5.1.3.1 Sumadores de instrucciones

Para este autor, el rendimiento de las primeras computadoras dependía únicamente del rendimiento del procesador y su operación más frecuente era la suma. Por esto, cuanto mejor fuera el rendimiento de las operaciones de suma, mejor era considerada la computadora. Por tanto, en los sumadores de instrucciones, la suma era la única carga de trabajo utilizada y el tiempo de ejecución de las sumas, era la única medida de rendimiento que se tomaba en cuenta.

2.5.1.3.2 Mix de instrucciones

Con el aumento en la complejidad de los procesadores, se incluyeron otras operaciones por lo que los sumadores quedaron rápidamente en desuso, dando lugar a los mixes de instrucciones que se dan como consecuencia de la relación entre los tipos de instrucciones utilizadas y su frecuencia de uso, de esta forma es factible comparar dos procesadores a través del promedio del uso de instrucciones en el tiempo.

Los mixes encontraron un lugar en el mercado. El *Gibson mix*, por ejemplo, fue elaborada en 1959 por Jack C. Gibson y se utilizó para sistemas *IBM 704*. Gibson dividió las instrucciones en 13 clases a partir de las cuales se calculaba el promedio de uso según la frecuencia de uso de cada una de las categorías.

Tabla 7. Ejemplo de Mix de instrucciones de Gibson (elaboración propia)

Mix de Instrucciones, Gibson		
fijo	1. Carga y guardado	31.2
	2. Suma y resta de punto	6.1
		3.8
	3. Comparaciones	16.6
	4. Bucles	6.9

5. Suma y resta de punto flotante 6. Multiplicación de punto flotante 7. División de punto flotante 8. Multiplicación de punto fijo 9. División de punto fijo 10. Desplazamientos 11. Operaciones lógicas (*And/Or*) 12. Instrucciones sin uso de registros 13. Indexación	3.8 1.5 0.6 0.2 4.4 1.6 5.3 18.0
Total	100.0

En la actualidad las computadoras pueden ejecutar muchas más instrucciones que no están reflejadas en los mixes de instrucciones. Asimismo, el tiempo de ejecución de cualquier instrucción depende de los modos de direccionamiento, *hits* de caché, eficiencia de la segmentación y otros parámetros. Aun así los mixes instrucciones son eficientes al medir la velocidad del procesador en sí mismo, lo que quiere decir que, son útiles sólo si se desea descubrir si el procesador es el cuello de botella en el sistema.

2.5.1.3.3 Kernels

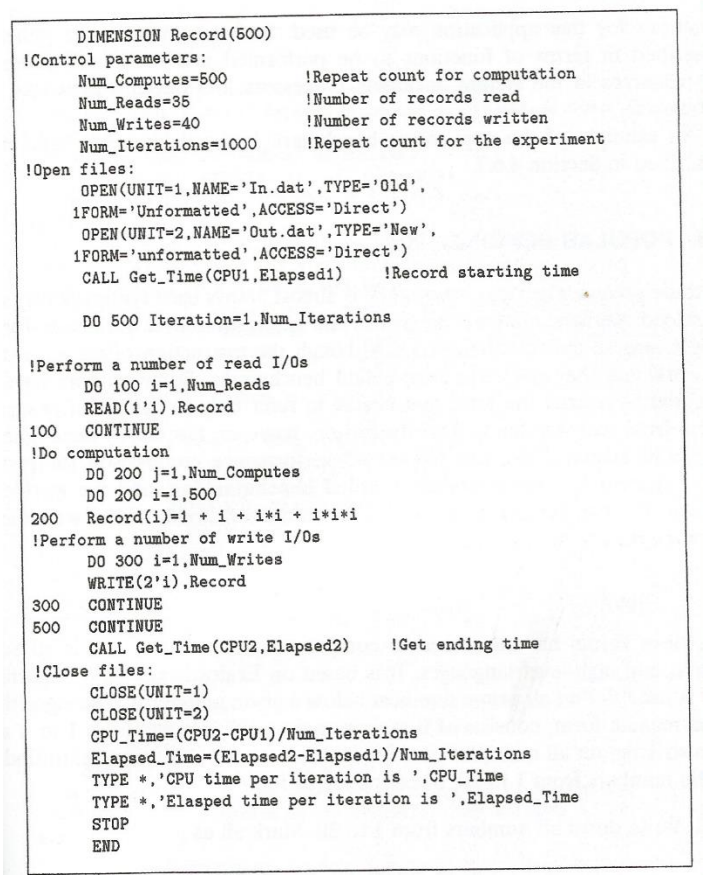
Los *Kernels* son una generalización de los mixes de instrucciones, ya que clasifican las instrucciones por medio de los servicios ofrecidos por el procesador, como lo son la segmentación, la carga de instrucciones y operaciones comunes. Ejemplo de Kernels son:

Sieve, *Puzzle*, *Tree searching*, *Ackermann´s Function*, *Matrix Invertion* y *Sorting*. La desventaja principal de los *Kernels* es que no toman en cuenta dispositivos de E/S y al igual que los mixes de instrucciones, sus resultados no muestran el rendimiento del sistema en su totalidad.

2.5.1.3.4 Programas sintéticos

Los programas sintéticos nacen como una forma de incorporar en las mediciones los servicios ofrecidos por el sistema operativo, y con éstos, incorporar las llamadas a los dispositivos de E/S. Los programas sintéticos utilizan bucles para hacer llamadas a los

dispositivos de E/S y así computar el tiempo promedio que le toma al CPU atender cada uno de los pedidos. Los programas sintéticos tienen la ventaja de que se escriben en lenguajes de alto nivel. De hecho, los primeros de ellos se escribieron el FORTRAN o Pascal, tal y como se puede observar en la siguiente imagen.



```
      DIMENSION Record(500)
!Control parameters:
      Num_Computes=500         !Repeat count for computation
      Num_Reads=35             !Number of records read
      Num_Writes=40            !Number of records written
      Num_Iterations=1000      !Repeat count for the experiment
!Open files:
      OPEN(UNIT=1,NAME='In.dat',TYPE='Old',
     1FORM='Unformatted',ACCESS='Direct')
      OPEN(UNIT=2,NAME='Out.dat',TYPE='New',
     1FORM='unformatted',ACCESS='Direct')
      CALL Get_Time(CPU1,Elapsed1)     !Record starting time

      DO 500 Iteration=1,Num_Iterations

!Perform a number of read I/Os
      DO 100 i=1,Num_Reads
      READ(1'i),Record
100   CONTINUE
!Do computation
      DO 200 j=1,Num_Computes
      DO 200 i=1,500
200   Record(i)=1 + i + i*i + i*i*i
!Perform a number of write I/Os
      DO 300 i=1,Num_Writes
      WRITE(2'i),Record
300   CONTINUE
500   CONTINUE
      CALL Get_Time(CPU2,Elapsed2)     !Get ending time
!Close files:
      CLOSE(UNIT=1)
      CLOSE(UNIT=2)
      CPU_Time=(CPU2-CPU1)/Num_Iterations
      Elapsed_Time=(Elapsed2-Elapsed1)/Num_Iterations
      TYPE *,'CPU time per iteration is ',CPU_Time
      TYPE *,'Elasped time per iteration is ',Elapsed_Time
      STOP
      END
```

Ilustración 19. Synthetic worload generation program. En The art of computer systems performance analysis (p.51), por Jain, R., 1991, New York: Willey. Derechos de autor [1991] por WILEY. Reimpresión autorizada.

Además, tienen la ventaja de que, una vez desarrollados, el proceso de mediciones es automatizado y admiten su portabilidad a otros sistemas operativos, permitiendo caracterizar ganancias o pérdidas.

La desventaja principal de los programas sintéticos resulta de que su alcance es pequeño y no hacen mediciones representativas de accesos a memoria.

2.5.1.3.5 Aplicaciones de Benchmark

Las aplicaciones de *Benchmark* son capaces de mostrar el funcionamiento de un sistema en su totalidad en términos de rendimiento. Por esto, toman en cuenta el uso de procesadores, dispositivos de E/S, dispositivos de red y bases de datos. Algunas aplicaciones de Benchmark existentes en la actualidad son: *Sieve, Ackermann´s Function*, *Whetstone, LINPACK*, *Dhrystone*, *Lawrence Livermore Loops*, *Debit-Credit Benchmark* y *SPEC Benchmark Suite*.

2.5.2. Consideraciones finales en relación con los análisis de rendimiento.

La escogencia de la carga de trabajo es la parte más importante en la ejecución de cualquier análisis de rendimiento. Por ello, según Jain Raj (1991), es necesario tomar en cuenta algunas consideraciones básicas:

2.5.2.1. Los servicios que se medirán durante el ejercicio.

Para hacer cualquier proyecto de análisis es indispensable tener claro el sistema con el que se trabaja. Al set de componentes de un sistema particular se le llama Sistema en Estudio (SUT, por sus siglas en inglés), asimismo, en muchas ocasiones los estudios están diseñados para conocer el comportamiento de un componente particular dentro del sistema, a ese componente se le llama Componente en Estudio (CUS, por sus siglas en inglés). En el caso de un estudio en el que se desee conocer el comportamiento del CPU, sería de utilidad conocer el impacto que tiene la ALU. De esta forma, la CPU sería el SUT y la ALU sería el CUS.

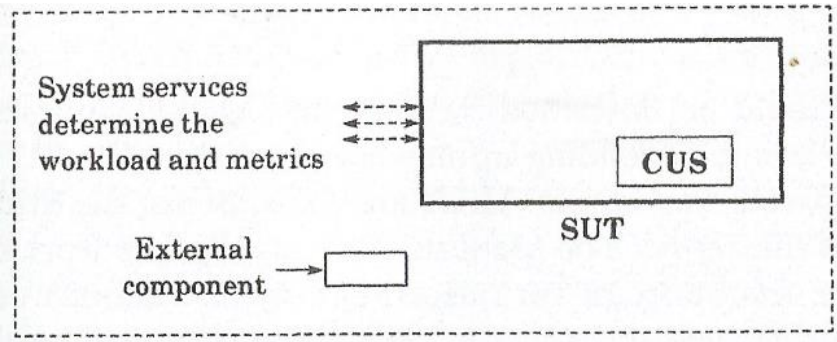


Ilustración 20. The SUT and CUS. En The art of computer systems performance analysis (p.61), por Jain, R., 1991, New York: Willey. Derechos de autor [1991] por WILEY. Reimpresión autorizada.

Siguiendo este mismo ejemplo, las MIPS, serían una métrica justificable para comparar dos CPUs con la misma arquitectura, ya que los servicios proporcionados por el CPU se hacen tangibles por medio de las instrucciones, por lo que el análisis de la frecuencia de estas es un criterio válido.

2.5.2.2. Nivel de detalle.

Una vez que el SUT y el o los CUS se han identificado, es necesario decidir el nivel de detalle que se espera del experimento, para ello se ofrecen tres alternativas:

a. Medir los pedidos que se hacen con más frecuencia: esta alternativa es válida si un tipo particular de servicio es utilizado mucho más frecuentemente que los otros.

b. Listar varios servicios, sus características y frecuencia: Si el rendimiento depende bastante del contexto, una medida útil resulta de medir los servicios que no dependen del contexto.

c. Registro de tiempo (*trace*): significa llevar un registro de pedidos en un sistema en tiempo real y usarlo como una carga de trabajo. Cuando los servicios prestados por el SUT son muy similares, estos pueden ser agrupados en clases.

2.5.2.3. Representatividad.

La carga de trabajo debe de asemejarse a la carga de trabajo del tipo de usuario más frecuente en aspectos como tiempo de llegada, recursos solicitados y perfil de uso (secuencia y cantidad de recursos solicitados).

2.5.2.4. Línea de tiempo.

Con el tiempo la forma en la que los usuarios manipulan las aplicaciones se modifica lo que obliga a la especificación no sólo del tiempo y la metodología seguida para realizar un experimento sino también a actualizarlo de forma que esté acorde con los cambios.

2.5.2.5. Últimas consideraciones.

Hay todavía una serie de detalles que deben tomarse en cuenta además del nivel de detalle y los servicios ofrecidos por el SUT. Algunos de ellos se refieren a:

a. Nivel de carga de trabajo: el experimento puede ser ejercitar el sistema a su capacidad total, por debajo de su capacidad total o un uso similar al usuario promedio, que el caso más típico.

b. Impacto de los elementos externos: el impacto que tienen los componentes externos al SUT puede parcializar los resultados. Por esto, se debe seguir una de las siguientes estrategias: o modificar la carga de trabajo para que no involucre la utilización de aspectos ajenos al SUT, o, la configuración del sistema debe cambiar de forma que no utilice componentes fuera del SUT.

c. Repetición: las cargas de trabajo deben ser replicables por lo que su calidad está relacionada con el grado de variabilidad de los resultados obtenidos.

2.6. Técnicas de caracterización por cargas de trabajo.

Para Jain Raj (1991), el proceso de la elaboración de una carga de trabajo es un proceso complejo que implica el estudio del uso de las aplicaciones en tiempo real con el fin de obtener generalizaciones útiles para definir la carga de trabajo, este proceso suele llamarse caracterización. En el caso de las caracterizaciones, los siguientes son otros elementos que también se encuentran presentes:

a. Usuario: se le llama usuario a la entidad que realiza los pedidos desde el SUT. Sin embargo, es más común en vez de usuario encontrar los términos de componente de carga de trabajo o unidad de carga de trabajo.

b. Aplicaciones: Las aplicaciones serán los programas desde los que se hacen los pedidos, por ejemplo, editores de texto, correo electrónico o navegadores de internet.

c. Sitios: componentes de la carga de trabajo.

d. Sesiones de usuario: las sesiones de usuario (*login* y *logout*), si las hay, deben ser monitoreadas.

Las técnicas más comunes para la realización de caracterizaciones se derivan en gran medida de la estadística. Ejemplos de estas son: promedio, dispersión, histogramas de un parámetro, histogramas de múltiples parámetros, análisis de los principales componentes, modelo de Markov y agrupación.

2.7. Consideraciones sobre el tipo de análisis que se realizará.

Tal y como lo exponen Marusarz & Ryabtsev (2019), el Top-Down Analysis de microarquitectura es una forma útil para conocer la forma en que los recursos de hardware disponibles están siendo aprovechados. Para este fin se utilizan Unidades para Monitorear el Rendimiento (PMUs, por sus siglas en inglés). En otras palabras, los PMUs son piezas lógicas que se encuentran dentro del CPU y que tienen como fin contar los eventos del hardware cada vez que hay una ocurrencia de estos en el sistema. Usualmente, estos eventos se combinan para obtener métricas más complejas, como es el caso de los CPI.

Cada arquitectura cuenta con su propio set de eventos que pueden ser registrados a través de PMUs. A pesar de esto, no todos los eventos son útiles para detectar y solucionar problemas de rendimiento específicos y usualmente, se necesita de un conocimiento muy profundo en microarquitectura para comprender la forma en que los PMUs se relacionan entre sí, cuando sus valores no han sido procesados. El método de *Top-down Analysis*, se utiliza para convertir los datos aportados por los PMUs en información procesable. De ahí, que este método haya sido el elegido para el estudio.

El *Top-down Analysis* parte de un esquema jerárquico de métricas categorizadas de forma tal que permiten la identificación de cuellos de botella de rendimiento en las aplicaciones de usuario, poniendo en evidencia la forma en que se utilizan las unidades de microarquitectura de los procesadores al ejecutar la aplicación.

La implementación de la segmentación de una CPU moderna es compleja. Tal y como se pudo observar en el apartado n. 5.2.5, sobre Detección y selección de unidades de microarquitectura del procesador seleccionado, el proceso de segmentación se divide conceptualmente en dos mitades, la parte frontal y la parte posterior. La interfaz, es la parte responsable de obtener el código del programa representado en las instrucciones de arquitectura y decodificarlo en una o más operaciones de hardware de bajo nivel llamadas micro-ops (uOps). Los uOps se envían al back-end, la parte posterior, por medio de un proceso llamado asignación. Una vez asignado, el back-end es responsable de supervisar cuándo están disponibles los operandos de datos de uOp. Por otra parte, el back-end también es responsable de ejecutar uOp en una unidad de ejecución disponible, la finalización de la ejecución de uOps, y su posterior escritura en memoria. Por lo general, los uOps pasan

completamente a través de la estructura de segmentación y son despachadas, pero a veces los uOps obtenidos por medio de procesos de especulación pueden cancelarse antes de que sean despachados, como sucede cuando las instrucciones no se predicen de forma correcta.

En la actualidad, las microarquitecturas Intel pueden asignar cuatro uOps por ciclo, y retirar desde el Back-end el mismo número de operaciones por ciclo. Para comprender mejor este comportamiento, resulta indispensable introducir el concepto de ranura. Una ranura en la estructura de segmentación representa los recursos de hardware necesarios para procesar un uOp. El *Top-down Analysis* asumen que, para cada núcleo de CPU, en cada ciclo de reloj, hay cuatro ranuras de tubería disponibles. Posteriormente, hace uso de los eventos de PMU especialmente diseñados para medir la utilización de esas ranuras durante la ejecución de aplicaciones.

Durante cualquier ciclo, una ranura de la estructura de segmentación puede estar vacía o llena con un uOp. Si una ranura está vacía durante un ciclo de reloj, esto se atribuye a un bloqueo, por lo que el siguiente paso consiste en detectar si el bloqueo proviene de la interfaz o del back-end. Esto se hace usando los eventos y fórmulas de PMU designados. Si el bloqueo es causado por la incapacidad de la interfaz para llenar la ranura con un uOp, se clasificará como una ranura de límite de interfaz.

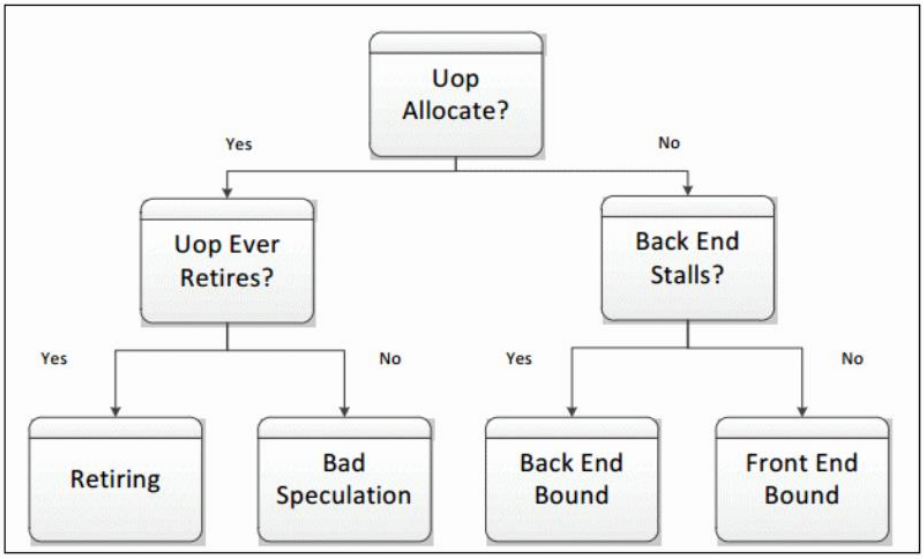


Ilustración 21.pipeline slot categorization. En Top-down Microarchitecture Analysis Method (p.4), por Marusarz, J. & Ryabtsev, D., 2019,EEUU: Intel. Derechos de autor [20109] por Intel.

Tal y como se muestra en la ilustración n.20, en caso de que la interfaz tenga un uOp preparado, pero no pueda entregarlo porque el back-end no está listo para absorberlo, la ranura vacía se clasificará como Límite de Back-end.

Por otra parte, cuando las ranuras no se encuentran bloqueadas, los cuellos de botella también pueden clasificarse de acuerdo a la forma en que los uOps son despachados. En caso de que se despachen correctamente, se clasifica como Retiring. En caso contrario, se clasificarán como Especulación Fallida.

2.8. Notas sobre los monitores de tareas.

Para Shobaki & Lindh (2001), los monitores son herramientas utilizadas para observar las actividades que realiza un sistema en particular. Sus funciones principales radican en la observación del rendimiento del sistema, recolección de estadísticas sobre el sistema, análisis de datos y presentación de resultados, por lo que son una herramienta indispensable en las mediciones de rendimiento.

De hecho, según Jain Raj (1991), algunos términos presentes en las discusiones sobre el tema son:

a. Evento: se refiere a cualquier cambio que ocurra en el sistema.

b. Indicio (*trace*): es un registro de eventos que usualmente incluye el tiempo en el que se dio el evento y el tipo de evento que se dio.

c. Sobrecarga: uno de los retos de los monitores resulta de hacer su trabajo de la forma menos invasiva en el SUT. Sin embargo, necesitan utilizar, de una forma mínima, los recursos del SUT para poder funcionar. Se le llama sobrecarga a la carga de trabajo proveniente de las tareas que realiza el monitor.

d. Dominio: hace referencia al conjunto de actividades que serán observables para el monitor.

e. Tasa de rendimiento de las entradas (*input rate*): se constituye como la frecuencia máxima que el monitor puede observar de forma confiable la velocidad en las entradas.

f. Resolución: es el nivel de detalle con que el monitor es capaz de observar y presentar la información.

g. Ancho de banda de las entradas: el número de bits de información que se guardan de un evento particular.

2.9. Clasificación de los monitores existentes

Tal y como lo menciona Jain Raj (1991), en la actualidad, existen varios tipos de monitores de tareas. Dependiendo de la forma en la que los monitores son implementados se clasifican como monitores de *hardware*, de *software*, de *firmware* o híbridos.

2.9.1. Monitores de software.

Los monitores de software son utilizados para observar las tareas realizadas por el sistema operativo y programas de alto nivel como lo son los programas relacionados con redes y bases de datos.

La primera decisión que cualquier monitor de tareas debe realizar, se refiere a la forma y el momento en el que se desencadenan las recolecciones de datos que hace por rutina. En el caso de los monitores de sotfware se proponen los siguientes mecanismos:

a. Instrucciones de trampa: las instrucciones de trampa son mecanismos de interrupción de software para transferir el control a una rutina de recolección de datos.

b. Modalidad de rastreo (*trace mode*): al utilizar este mecanismo, se cambia la modalidad del procesador a la modalidad de rastreo. Cuando el procesador trabaja bajo esta modalidad, interrumpe sus funciones después de ejecutar cada instrucción, pasando el control a una rutina de recolección de datos. Este método tiene una sobrecarga muy grande por lo que solo se utiliza solo cuando el lapso entre los eventos no se toma en cuenta.

c. Temporizador de interrupciones: En este mecanismo el sistema operativo provee de un servicio de interrupciones temporadas que transfiere el control a una rutina de recolección de datos en intervalos regulares. Este es un mecanismo de muestreo ideal en el caso en el que la tasa de eventos es alta, ya que la sobrecarga producto de la activación de la rutina de recolección, ancho de banda en la entrada de datos, y la tasa de variación de la cantidad de la muestra, definirá la tasa de muestreo deseado. Por ejemplo, si un contador está siendo muestreado, el muestreo debe

realizarse de forma tal que la probabilidad de que el contador se sobrecargue entre muestras se minimice al máximo.

Los monitores de software usualmente se escriben en lenguajes de bajo nivel como ensamblador Bliss o C y están al tanto del tamaño de los búferes en los que se guardan los datos recolectados por las rutinas, que se almacenarán después en discos o cintas magnéticas. Además, el número de búferes se puede definir de forma tal que el proceso de guardado sea inmediatamente seguido por una nueva rutina de monitoreo. Además, el proceso de monitoreo implica la capacidad de sobrescribir búferes o detener las rutinas de monitoreo hasta que un búfer se encuentre disponible de nuevo para recolectar información. Por otra parte, debe ser capaz de procesar información para reducir la cantidad de datos almacenados y un interruptor para activar o desactivar las rutinas de recolección de datos.

2.9.2. Monitores de hardware.

Consiste en una pieza de hardware que se sujeta al sistema por medio de puntas de prueba. En este caso, el monitor de tareas no hace uso de los recursos del sistema y el grado de sobrecarga es mucho menor que el de los monitores de software por lo que la posibilidad de que alteren los resultados del SUT también es menor. Los monitores de hardware tienen en común los siguientes componentes: puntas de prueba, contadores, componentes lógicos, comparadores, dispositivos de mapeo, temporizadores y cinta o disco para el almacenamiento de la información.

2.9.3. Firmware y monitores híbridos.

Los monitores de *firmware* modifican el microcódigo del procesador por lo que son utilizados para aplicaciones cuyas fronteras se encuentran entre el *software* y el *hardware*. Los monitores de *firmware* son muy similares a los monitores de *software*, pero se utilizan en el reducido espacio del microcódigo de los procesadores. Por este motivo se utilizan en el monitoreo de redes en donde las interfaces de red pueden ser microprogramadas.

Por otro lado, un monitor que utiliza una combinación de *software*, *hardware* y *firmware* es considerado un monitor híbrido. Usualmente los monitores híbridos aprovechan las mejores características de los monitores que combinan.

2.10. Ciclo de vida del software y modelos de desarrollo.

Según Z. Cataldi (2000), el desarrollo de software implica necesariamente una serie de actividades que pueden agruparse en cinco procesos primordiales, ocho procesos de soporte y cuatro procesos globales.

En la tabla n.8 muestra el detalle de estos procesos.

Tabla 8. Los procesos del ciclo de vida del software según ISO 12207-1. En Una metodología para el diseño, desarrollo e implementación de software educativo (p.36), por Cataldi, Z., 2000, Uruguay: Universidad Nacional de la Plata. Derechos de autor [2000] por Universidad Nacional de la Plata. Reimpresión autorizada.

PROCESOS PRINCIPALES (Aquellos que resultan útiles a las personas que inician o realizan el desarrollo, explotación o mantenimiento durante el ciclo de vida).	ADQUISICIÓN	Contiene las actividades y tareas que el usuario realiza para comprar un producto
	SUMINISTRO	Contiene las actividades y tareas que el suministrador realiza
	DESARROLLO	Contiene las actividades de análisis de requisitos, diseño, codificación, integración, pruebas, instalación y aceptación.
	EXPLOTACIÓN	También se denomina operación del software.
	MANTENIMIENTO	Tiene como objetivo modificar el software manteniendo su consistencia.
PROCESOS DE SOPORTE (se aplican en cualquier punto del ciclo de vida)	DOCUMENTACIÓN	Registra la información producida en cada proceso o actividad del ciclo de vida
	GESTIÓN DE LA CONFIGURACIÓN	Aplica procedimientos para controlar las modificaciones
	ASEGURAMIENTO DE LA CALIDAD	Para asegurar que todo el software cumple con los requisitos especificados de calidad.
	VERIFICACIÓN	Para determinar si los requisitos están completos y son correctos.
	VALIDACIÓN	Para determinar si cumple con los requisitos previstos para su uso.
	REVISIÓN	Para evaluar el estado del software en cada etapa del ciclo de vida
	AUDITORÍA	Para determinar si se han cumplido los requisitos, planes y el contrato.
	RESOLUCIÓN DE LOS PROBLEMAS	Para asegurar el análisis y la eliminación de problemas encontrados durante el desarrollo.

Asimismo, existen diversos modelos que pueden seguirse para el desarrollo de software. En las secciones que siguen, se describe una pincelada de estos métodos.

2.10.1. Modelo en cascada.

Según Hueso & Cascant (2012), en este modelo las etapas están bien delimitadas y se desarrollan de forma lineal sin que esto imposibilite las iteraciones a partes anteriores. Usualmente las etapas definidas corresponden a:

a. Análisis de requerimientos del sistema.
b. Análisis de requerimientos de software.
c. Diseño preliminar.
d. Diseño del software.
e. Codificación y pruebas.
f. Explotación y mantenimiento.

Este modelo es sumamente útil para el control de fechas de entrega y permite a los usuarios llevar un seguimiento del avance del proyecto a través de los entregables definidos en los requerimientos.

Sin embargo, se le critica que en la mayor parte de proyectos, la linealidad de estos es ficticia. Por otra parte, como el funcionamiento del proyecto se observa hasta la última etapa, no permite muchas modificaciones en el caso de que el proyecto no satisfaga las necesidades o expectativas del cliente debido a deficiencias o malentendidos en las etapas anteriores.

2.10.2. Prototipado evolutivo.

Según Z. Cataldi (2000), este modelo se recomienda cuando el usuario está muy seguro de lo que quiere y/o espera del software terminado. Desde este modelo se plantean entregas paulatinas y funcionales del software, comenzando con unidades simples, hasta lograr, a través de las iteraciones, la complejidad deseada.

Las etapas de este modelo son las siguientes:

a. Análisis de requerimientos del sistema.
b. Análisis de requerimientos de software.
c. Diseño, desarrollo e implementación de un prototipo.
d. Refinamiento iterativo del prototipo.
e. Refinamiento de especificaciones del prototipo.
f. Diseño e implementación final.
g. Operación y mantenimiento.

2.10.3. Modelo en espiral.

Para Fariño (2011), se realiza con el fin de solventar las deficiencias del modelo en cascada. Cada ciclo comienza con la identificación de objetivos, alternativas y restricciones de dichas alternativas. Después de evaluar las alternativas, se selecciona una y se pasa al siguiente ciclo. Este modelo cuenta con tres etapas básicas:

a. Explicitación de las alternativas posibles.
b. Identificación de riesgos de las alternativas y su respectiva mitigación.
c. Desarrollo y mantenimiento de software.

Este modelo es una opción viable para todo tipo de proyectos, pero necesita de la presencia de un experto en riesgos para la identificación de estos.

2.10.4. Modelo orientado a objetos.

Para Z. Cataldi (2000), los modelos orientados a objetos sostienen la descomposición del problema en componentes más pequeños que pueden ser reutilizados o bien asignados a distintas personas para su desarrollo. Este modelo plantea un desarrollo interactivo e incremental que se lleva a cabo a través de la iteración.

2.11. Relación entre microarquitectura, análisis de rendimiento, monitoreo y software.

La realización de una pieza de *software* para el monitoreo de un procesador implica un conocimiento básico de la microarquitectura de las computadoras en general con el fin de situar el fenómeno que se desea observar. Asimismo, hace necesaria la compresión de la lógica que se encuentra en el funcionamiento de los microprocesadores y sus componentes, problemáticas y servicios que prestan con el fin de lidiar con estos. Además, implica la familiarización con el contexto que rodea a cada microprocesador y sus funciones dentro del ecosistema de la computadora.

Además, implica necesariamente conocimiento acerca de análisis de rendimiento y monitoreo con el fin de tener las bases para la escogencia de procesos, herramientas y seguimiento de estándares.

Finalmente, el desarrollo y mantenimiento de este implica la consecución de un marco referencial, un modelo para sistematizar las etapas que se seguirán al construirlo.

El apartado del Marco Teórico cumple con esta función. Haciendo un recorrido general por cada uno de estos aspectos.

CAPÍTULO III: MARCO METODOLÓGICO.

Todo proceso investigativo se guía por paradigmas, es decir por un sistema de creencias, visión de la realidad y del individuo en su relación con el mundo; limitando de esta forma las problemáticas que pueden resolverse y tratarse a través de un abordaje metodológico. El intento por sistematizar la realidad y el conocimiento que puede o no ser asequible llevó a la comunidad griega a interpretar al mundo desde una perspectiva dual que se manifiesta en la separación heredada por el mundo occidental entre mente y cuerpo, esferas física y espiritual, así como la diferenciación entre lo etéreo y lo infinito. En la investigación científica, esa herencia se proyecta como la división entre los métodos cualitativos y cuantitativos. Estos métodos se sustentan en paradigmas como lo son el positivismo, el post-positivismo, la teoría crítica y el constructivismo. (Ramos, 2015)

En este apartado se describen los paradigmas que funcionan como cimientos de este proyecto, que derivarán en métodos y técnicas de investigación acordes con dicho abordaje de la investigación.

3.1. Tipo de investigación.

Este proyecto forma parte del cúmulo de investigaciones aplicadas. Según Vargas Cordero (2009), la investigación aplicada nace en el siglo XX para referirse a los estudios científicos que se proponen solucionar problemas prácticos o de la vida cotidiana, ya fuera para solucionar dichos problemas, como es el caso de invenciones técnicas, artesanales o industriales; o más bien la puesta en práctica de teorías científicas para la solución de problemas cotidianos.

La investigación aplicada, por tanto, responde a una realidad histórica y social específica e implica la relación intrínseca entre *el saber y el hacer*, al asirse del saber teórico para enfocarse en sus aplicaciones por medio de la triangulación de la información, sin que esta bifurcación implique un empobrecimiento del abordaje teórico-metodológico.

Teniendo la investigación aplicada como fin el aumento del caudal cultural, científico y productivo, se proponen los siguientes pasos para su ejecución:

a. Partir de una situación problemática que necesita ser resuelta o mejorada.

b. Seleccionar un punto de vista teórico sobre el qué soportar la propuesta para el tratamiento del problema.

c. Examinar la situación a través del sustento teórico seleccionado.

d. Probar y ensayar un prototipo acorde con el análisis del paso anterior.

e. Documentar y presentar los resultados del trabajo realizado.

De hecho, el presente proyecto se incorpora al conjunto de investigaciones aplicadas que pretenden tener un impacto en la producción, que tal y como lo menciona Lozada (2014), tiene como finalidad principal la generación de conocimiento con aplicación directa a mediano plazo en la sociedad, aumentando el valor agregado de la actividad investigativa como tal.

En la ilustración n.21 se muestra un diagrama sobre los roles de las universidades, la industria y la investigación aplicada.

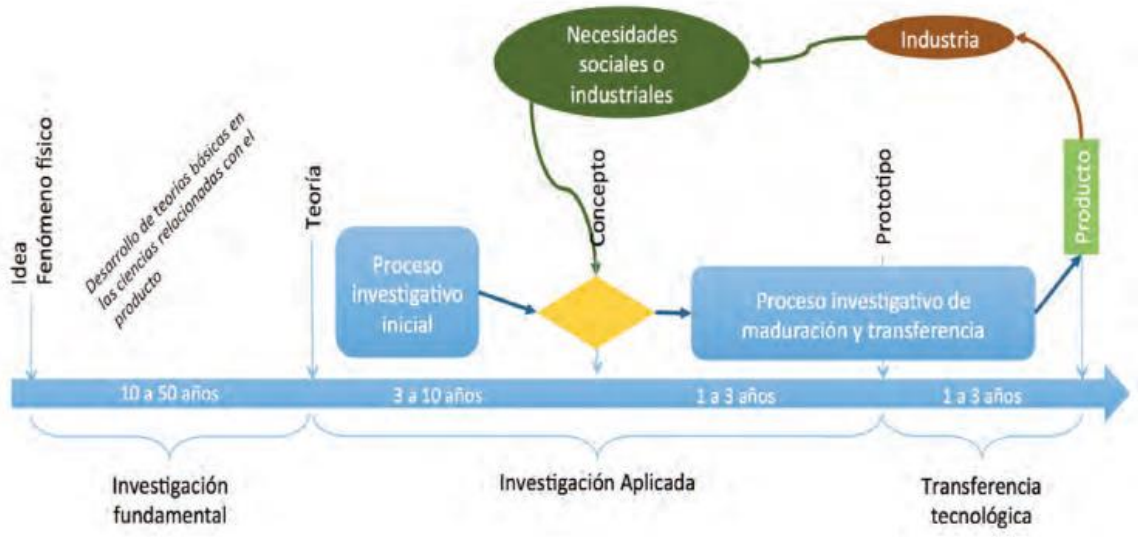


Ilustración 22. Proceso de producción de conocimiento. En Investigación Aplicada. (p.35), por Lozada,J., 2014, Ecuador: Centro de Investigación en Mecatrónica. Derechos de autor [2014] por Universidad Tecnológica Indoamérica. Reimpresión autorizada.

Durante un proceso de investigación aplicada cuyo impacto se proyecta como mejoras en la producción, como es el caso del proyecto que se plantea en este texto, se proponen tres etapas fundamentales:

a. Un proceso de investigación inicial que involucra la búsqueda de referencias bibliográficas, aplicaciones que se hayan hecho y resultados de dichas aplicaciones.

b. El planteamiento de una aplicación de las fuentes estudiadas a la problemática que desea solucionarse o mejorarse.

c. Un proceso en el que se pone en práctica la o las alternativas planteadas. Este proceso tendrá como resultado una transferencia de tecnología que será materializada en un prototipo, que, de ser exitoso, podrá llevarse a la esfera de producción industrial.

3.2. Enfoque de investigación.

El proyecto propuesto comparte algunas preocupaciones afines con los aportes del paradigma constructivista, al reconocer en el pensamiento científico una forma de conocer el mundo entre muchas otras, que se ha caracterizado, entre otras cosas por esfuerzos de coherencia y verificación a través de teorías y leyes, en el que la experimentación, la observación y la sistematización tienen un papel fundamental para la explicación u obtención de conocimientos socialmente valorados acerca de fenómenos percibidos a través de los sentidos. Asimismo, comparte la calidad y cualidad de un saber científico como dependiente de una escuela, academia o revista que lo acuñe como tal. Dicho en otras palabras, la existencia entre un ligamen intrínseco entre los saberes científicos y las esferas de poder (lógicos, científicos, políticos, sociales, religiosos, etc.), que los admiten como tales, por lo que todo saber científico tiene la capacidad de favorecer o desfavorecer a conglomerados sociales específicos, lo que a su vez implica que toda investigación debe conllevar un cuestionamiento ético acerca de sus posibles alcances. Además, a partir de este trabajo, se reconoce en quien investiga, un sujeto biológica, psicológica e históricamente construido por lo que no puede abarcar en la investigación todos los puntos de vista posibles sobre un fenómeno, sino sólo parte de ellos. (Daros, 2011)

Sin embargo, se desliga de estas perspectivas al comprender que no basta con abordar la necesidad explícita desde este paradigma constructivista por desnaturalizar la concepción de la ciencia, sino que rescata la necesidad de la rigurosidad en la construcción metodológica

de forma tal que puedan ofrecerse criterios claros de evidencia y validez en la construcción de conocimiento. (Fernández Zubieta, 2009)

Es por esto que se acerca más bien a una perspectiva post-positivista, desde la que la realidad, aunque existe, no puede ser aprehendida en su totalidad ya que el intelecto y la percepción del ser humano son limitados y engañosos. En este sentido, se parte de la conciencia de que quien investiga lleva consigo un conglomerado de valores, conocimientos técnicos o tendencias que pueden influir en su trabajo investigativo, por lo que siempre existirá un margen de error en sus hallazgos. (Ramos, 2015)

Desde este paradigma, la verificación de los resultados y fines del proyecto tendrían como fin predecir el fenómeno estudiado y controlarlo a través de la experimentación y del método cuantitativo.

Las investigaciones de tipo cuantitativas utilizan la recopilación de información para poner a prueba los resultados de los experimentos y así comprobar la validez de los resultados. En este sentido, se utilizan estrategias estadísticas para identificar patrones de comportamiento que son contrastados con los atestados teóricos que podrían explicar los comportamientos identificados.

Esta investigación, sin embargo, también se desliga del post-positivismo al reconocer el valor de las técnicas de recolección cualitativas, ya que las técnicas cuantitativas son útiles para encontrar patrones y tendencias, pero las cualitativas ayudan a comprender la subjetividad y las motivaciones para la realización de cualquier acción y por tanto se inscribe más bien dentro del realismo científico.

El realismo científico comparte los planteamientos rescatados en este texto tanto del paradigma constructivista como del post-positivismo, pero además rescata la utilidad, no solo de las técnicas cuantitativas como lo es la estadística, sino también de técnicas cualitativas como lo son las conversaciones y la observación participante. (Hueso & Cascant, 2012)

3.2. Fuentes y sujetos de información.

En este apartado se detallan las fuentes primarias y secundarias de información que se utilizarán durante el desarrollo del proyecto, así como los sujetos que tienen una relación directa con la funcionalidad de este.

3.2.1. Fuentes de información

En este apartado se describen las fuentes de información, primarias y secundarias, que se utilizarán para el diseño y realización de los experimentos, así como para el análisis de los resultados obtenidos en cada una de las etapas del desarrollo del prototipo planteado.

3.2.1.1. Fuentes de información primarias

Las fuentes de información primaria comprenden las fuentes de información más directa, implican un recurso básico para la comprensión del contexto y la solución de problemas similares al del proyecto presentado en este texto. Asimismo, incluyen los datos obtenidos en las diferentes etapas del desarrollo de este prototipo.

a. Resultados de estudios similares: Se estudiarán casos de análisis de mediciones de rendimiento de procesadores de la esfera comercial, como lo son análisis de rendimiento en procesadores Intel en sus distintas generaciones para portátiles, tanto en prototipos de diseño (*pre-silicon*) como en procesadores que ya fueron implementados (*post-silicon*). Estos estudios pueden encontrarse en formato de artículo científico o informe.

En la *ilustración n.22* se muestra una síntesis de los procesadores que podrán ser tomados en cuenta como elementos de análisis y observación.

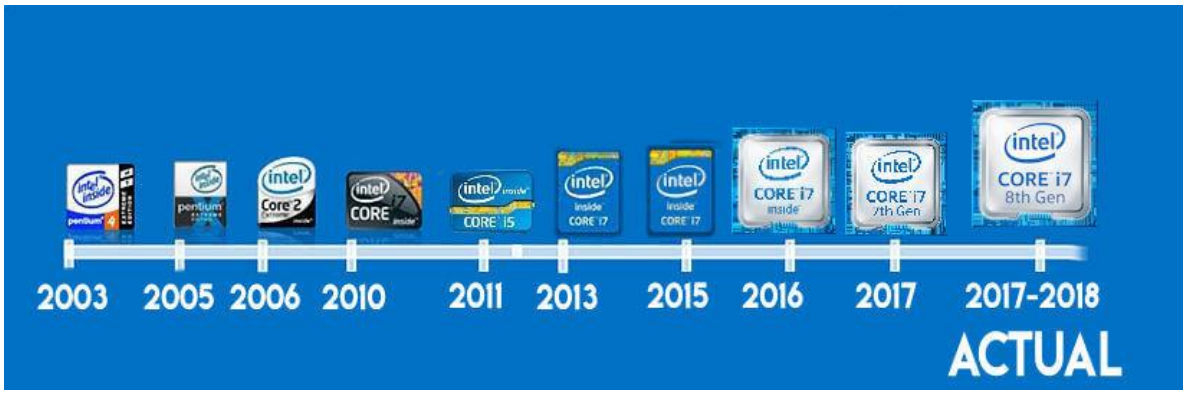


Ilustración 23. etiquetas Intel procesadores generaciones NEOSTUFF. En Diferencias entre generaciones de procesadores Intel (p.01), por Carvajal, H., 2018, México: NEOSTUFF. Derechos de autor [2018] por NEOSTUFF. Reimpresión autorizada.

b. Conversaciones formales: se realizarán reuniones con el cliente del sistema con el fin de contrastar la información recolectada y sistematizarla en el desarrollo del proyecto.

3.2.1.2. Fuentes de información secundaria.

Las fuentes de información secundaria comprenderán todo lo dicho sobre investigaciones publicadas y en sí el corpus teórico-práctico consolidado a través de la experiencia adquirida por la disciplina a través de sus años de existencia.

a. Libros de texto: A pesar de que la informática y en sí las computadoras tienen menos de doscientos años de existir, los tratados de arquitectura, caracterizaciones y monitoreo de tareas se han multiplicado y complejizado cada vez más. (Asensi Artiga, 1993)

Para la realización de este estudio se ha recopilado, clasificado y analizado la información encontrada en libros de referencia sobre arquitectura, organización de computadoras, análisis de rendimiento, monitoreo y caracterizaciones.

b. Sitios web: La visita de sitios web fue imprescindible para la recopilación de artículos y noticias relacionados con la problemática. Asimismo, los sitios web se han convertido en el mecanismo por excelencia para la difusión de herramientas de monitoreo de software tanto gratuitas como licenciadas. Usualmente, fue posible tener acceso a los sitios oficiales de los licenciatarios de los Derechos de Autor para la descarga de versiones gratuitas de herramientas especializadas ofrecidas

para proyectos educativos o investigativos (*community editions*). En estos sitios fue posible encontrar instrucciones de instalación, puesta en marcha y decodificación de instrucciones. Las herramientas encontradas fueron analizadas. Tras el análisis se seleccionó una herramienta que funcionó como materia prima para el desarrollo del proyecto. Vale la pena mencionar que, las herramientas encontradas están orientadas a la optimización de código desde el punto de vista del desarrollador, por lo que uno de los retos es la transformación de las métricas utilizadas para que cumplan con la funcionalidad que se espera del producto del proyecto.

c. Tutoriales de autores independientes: popularizados por medio de plataformas como *Youtube, W3Schools* y foros como *Stackoverflow, Stackexchange* y *lawebdelprogramador* fueron un auxiliar en la depuración del código asociado a la elaboración de las capas de código del prototipo.

3.2.2. Sujetos de información.

En este apartado se describen las personas físicas que de una forma u otra están involucradas en el proyecto. Ya sea porque por medio de ellas se obtuvo información relevante para su realización, o porque tienen alguna relación directa con éste.

Tabla 9. Definición de sujetos de información, elaboración propia.

Puesto laboral o descripción general	Profesión u oficio	Experiencia	Relación con el tema
Gerencia	Ingeniería en electrónica.	Planificar, organizar, dirigir, controlar, coordinar, analizar, calcular y deducir el trabajo del equipo.	Interesado principal en que el proyecto se lleve a cabo.
Analista de sistemas computacionales	Ingeniería en Sistemas.	Análisis del funcionamiento de sistemas computacionales.	Usuario principal del prototipo en desarrollo.
Investigación	Practicante.	Asistencia en el análisis del funcionamiento de sistemas computacionales.	Desarrollo de prototipo planteado.

3.3. Técnicas y herramientas de recolección de datos.

En este apartado se describen de forma general tanto las técnicas como las herramientas para la recolección de los datos necesarios para que el proyecto se lleve a cabo.

3.3.1. Técnicas de recolección de datos.

En la siguiente sección se describen las técnicas se utilizarán en el proceso de investigación y desarrollo del prototipo en cuestión.

3.3.1.1. Caracterización.

Se utilizará el formato de una caracterización por cargas de trabajo.

La *tabla n.10* describe los pasos según la propuesta de Jain, R. (1991), para este tipo de proyecto.

Tabla 10. Solución técnica para técnica seleccionada: Análisis de Rendimiento, elaboración propia.

Pasos a seguir para una caracterización	
Definición del problema	• Definición de metas en relación con los objetivos.
Lista de servicios	• Lista de servicios brindados por el microprocesador en estudio y posibles salidas.
Selección de métricas	• Selección de técnica de recolección de datos. • Selección de herramienta de recolección de datos. • Selección de técnica de procesamiento de información para análisis. • Selección de unidades de hardware que serán tomadas en cuenta. • Selección de criterios de rendimiento.
Lista de parámetros	• Selección de parámetros de sistema. • Selección de parámetros para aplicación piloto.
Selección de factores	• Selección de factores que se mantendrán estables. • Selección de factores que variarán.
Diseño de experimento	• Diseño de cargas de trabajo preliminares. • Puesta en marcha de pruebas preliminares (cargas de trabajo sintéticas).

	• Diseño prototipo para medición de carga de trabajo en tiempo real (carga de trabajo real). • Puesta en marcha de prototipo para medición en tiempo real.
Análisis e interpretación	• Análisis de resultados de las pruebas preliminares. • Análisis de puesta en marcha del prototipo.
Presentación de resultados	• Presentación de resultados de experimento de forma amigable para personas no familiarizadas con la temática.

3.3.1.2. Entrevista semiestructurada.

Las entrevistas son una conversación dirigida hacia una finalidad específica muy útil para recabar datos. Las entrevistas semiestructuradas se basan en una guía básica que aborda los temas en relación con los objetivos planteados para la misma. En una entrevista semiestructurada, el sujeto entrevistado puede hablar de forma libre sobre las temáticas planteadas. En el curso de una entrevista semiestructurada, es papel de quien entrevista reconocer los casos en los que es necesaria mayor profundidad sobre los temas inicialmente propuestos. (Díaz B, Torruco G, Martínez H, & Varela R, 2013a)

El diagrama que se muestra a continuación trata sobre las etapas que se deben seguir al llevar a cabo una entrevista semiestructurada.

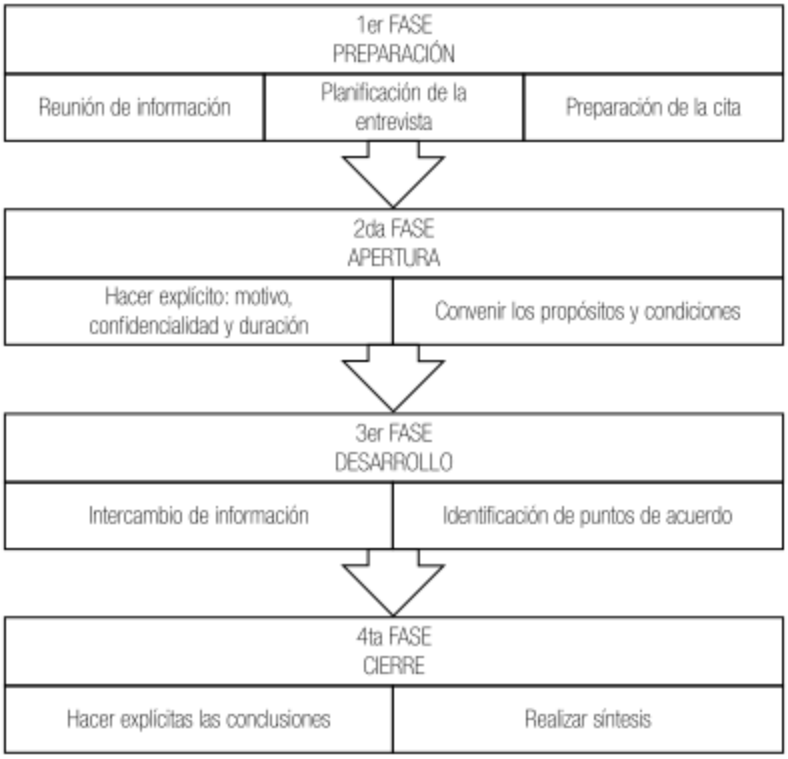


Ilustración 24. Fases de la entrevista. En La entrevista, recurso flexible y dinámico (p.164), por L. Días, B.; Torrunco G.,M.; Martínez H. et al, 2013, México D.F.: Investigación en educación médica. Derechos de autor [2013] por Revista de educación médica. Reimpresión autorizada.

3.3.1.4. Observación participante.

La observación participante se define como la observación activa a través de los sentidos para proporcionar una fotografía escrita con el fin de, a través de la exposición a las experiencias de las personas involucradas con el fenómeno, tener una comprensión más clara del mismo (Kawulich, 2005). La observación participante fue un mecanismo indispensable para conocer la forma en que los distintos actores interactúan con las herramientas de análisis de rendimiento en su trabajo.

3.3.4. Entrevista estructurada.

La entrevista estructurada utiliza procedimientos estandarizados para recolectar y analizar datos por medio de una muestra representativa de la población. Las preguntas y el

formato en esta herramienta se formulan de forma idéntica, lo que permite la tabulación de resultados para la detección de tendencias. (Díaz B, Torruco G, Martínez H, & Varela R, 2013b)

3.3.5. Estadística descriptiva e inferencial.

La estadística descriptiva es una técnica sumamente útil para comprender la estructura de los datos e identificar tendencias y anormalidades en los mismos. Es común la utilización de gráficos fáciles de comprender. La estadística inferencial por su parte, permite analizar los datos para hacer predicciones, estimaciones, así como probar o descartar hipótesis.(Vargas-Sabadias, 1995)

3.3.6. Triangulación de datos.

La triangulación de datos es la posibilidad de contrastar los resultados obtenidos desde fuentes de datos diferentes con el fin de tener varias perspectivas de un mismo fenómeno. La triangulación de datos se realiza para obtener puntos de convergencia, para identificar, corroborar o interpretar un fenómeno específico. (Benavides & Restrepo, 2005)

3.3.1.7. Revisión bibliográfica.

La revisión bibliográfica se refiere a la utilización de diversas fuentes documentales como libros, artículos de revista o periódico, videos, estadísticas existentes, libros y manuales que se encuentran disponibles. (Profesional & Valles, 2011)

3.3.2. Herramientas de recolección de datos.

Las herramientas de recolección de datos son sumamente útiles para operacionalizar las técnicas utilizadas en los proyectos. Seguidamente, se exponen las que se desprenden de las técnicas utilizadas en este trabajo.

3.3.2.1. Reuniones con el cliente.

Las reuniones hechas con el cliente son el escenario ideal para la realización de las entrevistas semiestructuradas, ya que proveen de un espacio neutro para la negociación y la exposición de ideas. Además, su carácter presencial, facilita la utilización de recursos como pizarras, hojas y recursos gestuales para la expresión de ideas.

3.3.2.2. Sistemas computacionales.

Las computadoras serán una herramienta primordial para tener un acercamiento a las actividades de trabajo cotidianas de los usuarios finales, así como para las fases de elaboración del prototipo y de pruebas de este al constituirse en la herramienta principal de trabajo de los usuarios finales así como las plataformas para la realización de pruebas y el análisis de los datos recolectados.

3.3.2.3. Cuestionario.

Los cuestionarios serán utilizados como mecanismo de diagnóstico para conocer la frecuencia y el tipo de herramientas de análisis de rendimiento que utilizan. A continuación, se muestra un ejemplo de cuestionario.

Tabla 11. Muestra de cuestionario, elaboración propia.

La información aportada por usted en este cuestionario es de carácter confidencial y anónima por lo que sólo se utilizará con fines académicos y de investigación. Gracias por su colaboración.			
#	**Pregunta**	**Respuesta (Marque con X)**	
1	En su trabajo actual: ¿Con qué frecuencia analiza métricas de uso de las diferentes unidades de microarquitectura?	Frecuentemente (1 o más veces por semana)	
		Poco frecuente (Al menos 1 vez al mes)	
		Muy rara vez	
		Nunca (*pase a la pregunta 5*)	
2	¿Cuál(es) de las siguientes herramientas es la que usted utiliza más frecuentemente? (*Puede marcar varias opciones*)	EMON	
		VTUNE	
		PERF	
		Otro (Especifique: _____)	
3	¿Cuál(es) de las siguientes opciones describen su uso de las herramientas de perfilado (*profiling*)? (*Puede marcar varias opciones*)	Validar resultados de performance (Scores)	
		Obtener/Calcular métricas secundarias (Ej: IPC, Instruction Count, etc)	
		Analizar el uso de bloques funcionales (Ej: AVX, I/O, etc)	
		Analizar la utilización de recursos del CPU en el tiempo	
		Otro (Especifique: _____)	
4	¿Cuál de las siguientes opciones describe mejor su tipo de uso de las herramientas de perfilado (*profiling*)?	Obtengo datos puntuales para configuraciones específicas que son prioritarias para la finalidad del experimento.	
		Obtengo datos para todos mis casos de prueba y luego los analizo independientemente según la finalidad del experimento que estoy realizando.	
		Solo lo utilizo para obtener datos en casos en los que estoy teniendo algún problema o comportamiento atípico específico que debe ser explicado.	
5	¿Considera que sería de utilidad poder visualizar alguna otra métrica de perfilado (*profiling*)?	Si, Especifique la métrica(s) de su interés: _____	
		No	

3.3.2.4. Herramienta para entrevista semiestructurada.

Las preguntas serán abordadas a partir de una guía de temas similar a la que se muestra a continuación.

Tabla 12. Ejemplo de guía de entrevista, elaboración propia.

Guía de entrevista	
Fecha de la entrevista:	
Lugar de la entrevista:	
Se entrevista a:	
Temas que se tratarán	
Tipo de producto esperado	
Plazo con el que se cuenta	
Funcionalidades esperadas del producto.	
Restricciones de la organización para el desarrollo del producto.	
Usuarios principales.	
Herramientas con las que cuenta el equipo en la actualidad.	

3.3.2.5. Programas para lectura, escritura y procesamiento de datos.

Se utilizará el paquete de Office para procesar y mostrar datos tanto en las pruebas preliminares, como en las pruebas de la puesta en marcha del prototipo. Además, se utilizarán programas para lectura de archivos de texto plano y bibliotecas de programación especializadas en matemáticas, estadística, manejo de archivos *.csv* y vectores.

3.3.2.6. Ambiente y lenguaje de programación.

El lenguaje de programación seleccionado para el desarrollo del prototipo es *Python*, versión 3.0. *Python* es el lenguaje de programación más utilizado por los usuarios finales del proyecto. Además, tal y como lo menciona F. Sanner, M. (1998), al ser un lenguaje de alto nivel, permite la utilización de estructuras de datos como listas, diccionarios y arreglos para facilitar la manipulación de datos. Aparte de esto, soporta la utilización de clases,

excepciones. Es u lenguaje de propósito general con gramática sencilla. Permite la construcción de scripts de forma gratuita, aun cuando sea para uso comercial. Funciona en casi cualquier plataforma moderna (Linux, Windows,IRIX, SunOS, OSF entre otros) y es compilado automáticamente por el intérprete del sistema en el que se instala.

Por otra parte, Python es un lenguaje modular por lo que facilita la programación orientada a objetos y permite el acceso a diversas bibliotecas de extensiones, que pueden estar escitas en Python, C, C++, Perl y Java.

Finalmente, Python cuenta con una red de desarrolladores, lo que permite el acceso a paquetes para el tratamiento de datos y la realización de gráficas como lo son NumPy y Matplotlib, que serán necesarias durante las fases de prueba y visualización de resultados del proyecto.

Tal y como lo mencionan Hu, Q, Ma, Lei & Zhao, Jianjun (2018), el ambiente de programación utilizado será *PyCharm*. *Pycharm*, al igual *Atom*, *Visual Studio Code*, entre otras IDEs, es un ambiente que facilita la programación en Python. Ofrecido desde *JetBrains*, Pycharm se considera en la actualidad uno de los ambientes de programación más completos que ofrece una versión *Community*, para el desarrollo científico. Su instalación es sencilla y su interfaz bastante intuitiva. Para la ejecución del código sólo es necesario pulsar run en el menú. Guarda un log de errores detallado de cada uno de los hilos ejecutados. Pycharm, tiene además un editor de código inteligente, que permite la opción de autocompletar instrucciones y la opción de refactorizar el código, que significa, la modificación del código en tiempo de ejecución.

Finalmente, al igual que Python, Pycharm cuenta con una comunidad comprometida de desarrolladores que elaboran plugins que facilitan la integración con otros lenguajes y frameworks, bases de datos y debugging.

3.3.2.7. Motores de búsqueda, bibliotecas virtuales y revistas especializadas.

Se buscarán artículos académicos, técnicos y libros de texto en motores de búsqueda como *Google Scholar*, bibliotecas virtuales como el *CENIT* de la Univerisidad Hispanoamericana y búsquedas en bases de datos especializadas como la del Institute of Electrical and Electronics Engineers *(IEEE)*, la Red de Revistas Científicas de América

Latina y el Caribe, España y Portugal *(Redalyc)* y la Revista del Colegio Superior de Investigaciones Científicas (*Arbor*).

3.4. Variables de investigación.

En esta sección se muestran las variables que se tomarán en cuenta de acuerdo a cada uno de los objetivos planteados con el fin de profundizar en lo que se espera de cada una de estas.

Tabla 13. Variables asociadas a los objetivos, elaboración propia.

Objetivo específico	Variables asociadas	Descripción
Identificar los requerimientos de software para la elaboración de un prototipo funcional que monitoree la utilización.	Fiabilidad	El grado de fiabilidad de la información recolectada por el software se definirá de acuerdo a las herramientas de análisis de rendimiento disponibles en el mercado.
	Rapidez	Se espera que el software arroje resultados al finalizar cada recolección de datos.
	Robustez	Se espera un software funcional. Las validaciones de formato, tipos de datos, e introducción de datos erróneos no son prioridad.
	Portabilidad	El software se construirá para Linux. No será portable.
	Facilidad de uso	No se espera una interfaz que permita la interacción. No es prioridad que sea atractiva visualmente o amigable desde el punto de vista del usuario.
Analizar la relación entre la microarquitectura del procesador en estudio y las métricas utilizadas para medir su rendimiento.	Componentes de microarquitectura procesador intel core i7.	Cantidad de núcleos, tipo de segmentación, estructura de registros, tipo de ALU, GPU, estructura de la memoria caché, características del CPU.
	Unidades de microarquitectura para monitoreo	Clasificación de las unidades de microarquitectura según su función. El detalle queda a discreción de quien desarrolla el proyecto.
	Mecanismos de monitoreo	Tipo de monitor, precisión, variables de medición posibles, fabricante, velocidad, comandos posibles.
	Métricas de rendimiento	Se utilizarán los máximos teoréticos posibles de las métricas de rendimiento así como los estudios similares recolectados para medir la fiabilidad de los cálculos.
Diseñar una propuesta de software que permita monitorear la forma en que las unidades de los procesadores Intel core i7 están siendo ejercitadas desde el nivel de usuario.	Precisión	Los diagramas realizados deben mostrar la lógica del funcionamiento del prototipo desde la perspectiva de un usuario especializado.

Desarrollar un prototipo de software para mostrar la forma en que las unidades funcionales de un procesador core i7 son utilizadas durante la ejecución de la aplicación de un tercero.	Algoritmo(s) de acoplamiento	Iniciar, solicitar indicadores de rendimiento, recibir informes de rendimiento, detener recolección.
	Algoritmo(s) de actualización	Pedidos automáticos de mediciones cada segundo.
	Algoritmo(s) de visualización	Presentación en pantalla de resultados.

3.5. Diseño de la investigación.

En este apartado se muestra la forma en que se integrarán los objetivos en relación con las etapas y las técnicas utilizadas en cada una de ellas. Como hilo conductor se utilizarán los pasos de una caracterización.

3.5.1. Etapa I: Definición del problema.

En la primera etapa del proyecto se analizarán los requerimientos de software para la elaboración de un prototipo funcional para monitorear la utilización de las unidades funcionales de los microprocesadores Intel core i7. Este análisis se dividirá en requerimientos funcionales y requerimientos no funcionales.

El desarrollo de esta primera etapa implica no sólo conocer la finalidad del software desde el punto de vista del cliente, sino también desde los usuarios del sistema. La definición de las técnicas se basa en la comprensión de que, su funcionalidad debe de estar acorde no sólo con los intereses del cliente, sino que debe de estar elaborado de forma tal que sea comprensible para los usuarios finales del mismo.

Es por esto por lo que las técnicas a utilizarse en esta etapa serán:

a. Entrevistas semiestructuradas al cliente: se harán entrevistas informales de seguimiento con el cliente para establecer los límites del proyecto, plazo, finalidad y seguimiento del proceso.

b. Observación participante: se visitará a algunos miembros del equipo y se les asistirá en sus labores cotidianas con el fin de comprender el trabajo que

realizan, el tipo de herramientas que acostumbran a utilizar y así facilitar el proceso de resistencia al cambio que puede surgir al proponer una herramienta nueva.

c. Entrevista estructurada: se realizará un cuestionario como forma de diagnóstico acerca de las necesidades de los miembros del equipo.

d. Triangulación: La triangulación de datos será una técnica útil al permitir el contraste de los intereses del cliente en conjunto con las necesidades de los usuarios y las posibilidades de la industria en relación con los alcances del proyecto.

3.5.2. Etapa II: Selección de métricas de rendimiento.

La segunda etapa del proyecto implica la selección de las métricas de rendimiento que serán utilizadas para el desarrollo del prototipo y las herramientas y cálculos que las originan.

a. Revisión bibliográfica: se hará la revisión de libros, revistas y sitios web especializados y tutoriales en formato de video y/o pdf con el fin de conocer las herramientas que existen en la actualidad para reconocer métricas varias de rendimiento, así como posibles formas de rastrear unidades de microarquitectura a través de esas métricas. Además, la revisión bibliográfica será útil para conocer los elementos de microarquitectura del procesador en estudio para agruparlos en unidades funcionales.

b. Estadística: gran parte de las métricas existentes en la actualidad están basadas en cálculos estadísticos varios que se utilizarán como mecanismo definitorio para el cálculo de métricas que no están disponibles en las herramientas que se utilizan, pero cuyos valores pueden calcularse a partir de las métricas que proveen dichas herramientas.

c. Triangulación: la triangulación de la información permitirá un acercamiento indispensable, ya que permitirá contrastar la información obtenida desde la revisión bibliográfica, con los datos estadísticos para seleccionar las métricas que serán utilizadas y la forma en que se rastrearán dichas métricas hacia las unidades de microarquitectura.

3.5.3. Etapa III: Discusión sobre parámetros y factores.

En esta etapa se definirán la forma en que los parámetros y los factores pueden influir en que el prototipo se comporte de una o de otra forma. Permitiendo clarificar las variantes que puede haber en la manera en que se comportarán las unidades funcionales cuyo rendimiento es medido por el prototipo.

a. Revisión bibliográfica: la revisión de estudios de caso similares permite encontrar tendencias acerca de los parámetros y factores más significativos.

3.5.4. Etapa IV: Diseño de prototipo.

Esta etapa consiste en la elaboración de diagramas que permitan comprender la lógica del prototipo y por tanto las interacciones entre capas, componentes y entre el prototipo y el usuario.

La técnica necesaria para llevar a cabo esta etapa es:

a. Revisión bibliográfica: revisión de artículos científicos, manuales y libros de teoría para la elaboración de diagramas de UML.

3.5.5. Etapa V: Análisis e interpretación.

En esta etapa se dará la elaboración de pruebas preliminares del funcionamiento del prototipo, por lo que implica pruebas y refinamiento del diseño del prototipo planteado en la etapa IV. Esta etapa culmina con la validación de los resultados que brinda el prototipo desde un ambiente controlado de pruebas (cargas de trabajo sintéticas). Después de que la fiabilidad de los datos ha sido validada, se desarrollará la visualización de los resultados (salida de datos) del prototipo por medio de la consola del ambiente de programación seleccionado. Finalmente, se habilitará una opción para que el usuario pueda ejecutar cargas en tiempo real (cargas de trabajo reales).

a. Estadística descriptiva: se utilizará para la selección de la técnica para sistematización de resultados a través del prototipo en desarrollo.
b. Revisión bibliográfica: libros de texto y artículos especializados para el conocimiento a profundidad de los elementos del procesador en estudio.

c. Triangulación: se utilizará para cotejar los datos aportados por las diferentes fuentes bibliográficas y la estadística inferencial con el fin de detectar los cuellos de botella y potencialidades en uso del procesador en estudio desde una aplicación de usuario.

3.5.6. Etapa VI: Presentación de resultados.

En esta última, se desarrollará la interfaz gráfica del prototipo de forma tal que la relación entre métricas y unidades de microarquitectura sea comprensible para el cliente y el usuario final del prototipo.

Las técnicas que se utilizarán en esta etapa son:

a. Revisión bibliográfica: se revisarán tutoriales en formato de video o pdf con la finalidad de comprender las posibilidades existentes desde las librerías de software existentes para el desarrollo de experiencia de usuario. Por otra parte, se buscarán manuales para la presentación de resultados y libros especializados en la experiencia de usuario.
b. Estadística inferencial: se analizarán las tendencias detectadas para detectar la relación entre éstas y la utilización de los recursos del procesador en estudio.
c. Programación orientada a objetos: se utilizará la programación orientada a objetos para la construcción de una clase de interfaz que permita la visualización de los resultados arrojados por las otras clases.
d. Triangulación: se contrastará la información existente en relación con librerías para programación, así como la información obtenida de manuales y la programación orientada a objetos para desarrollar la interfaz de usuario del prototipo desarrollado

3.6. Matriz de coherencia.

La matriz que se presenta a continuación tiene como finalidad ampliar la relación entre los objetivos, los entregables, las fases, las técnicas, instrumentos utilizados para la recolección, análisis de información y los temas propuestos en el marco teórico.

Con el fin de simplificar el cuadro, las etapas están demarcadas por los números romanos que les corresponden, su descripción detallada se encuentra en el apartado *3.5.*, sobre Diseño de la investigación. Asimismo, el detalle de los entregables se encuentra en el

apartado *1.4,* sobre los Alcances, y los temas mencionados por medio de números se encuentran desarrollados con detalle en el Marco Teórico presente en el capítulo II del proyecto.

Tabla 14. Relación entre objetivos, entregables, etapas, técnicas, herramientas y temas desarrollados en el Marco Teórico.

Objetivos	Entregables	Etapas	Técnicas	Herramientas	Temas del marco teórico
Identificar los requerimientos de software para la elaboración de un prototipo funcional que monitoree la utilización de las unidades funcionales de los microprocesadores Intel core i7.	Expuestos en el apartado 1.4.1.1	I	Entrevista estructurada.	Cuestionario.	*2.6, 2.9*
			Entrevista semiestructurada.	Guía de entrevista.	
			Observación participante.	Sistemas computacionales.	
			Triangulación.	Resultados de cuestionarios, guía de entrevista y sistemas computacionales.	
Analizar la relación entre la arquitectura del procesador, sus unidades funcionales y el ejercicio de	Expuestos en el apartado 1.4.1.2	II	Revisión bibliográfica.	Motores de búsqueda, bibliotecas virtuales y revistas especializadas.	*2.1,2.2,2.3,2.4,2.5,2.6, 2.7,2.8*
		III			

éstas unidades al ejecutarse la aplicación de un tercero.					
Diseñar una propuesta de software que permita monitorear la forma en que las unidades de los procesadores Intel core i7 están siendo ejercitadas desde el nivel de usuario.	Expuestos en el apartado 1.4.1.3	IV	Triangulación de datos	Bibliotecas de software.	*2.6, 2.9, 2.10*
				Motores de búsqueda, bibliotecas virtuales y revistas especializadas.	
			Revisión bibliográfica.	Herramientas de monitoreo de software.	
Desarrollar un prototipo de software para mostrar la forma en que las unidades funcionales de un procesador core i7 son utilizadas durante la ejecución de la aplicación de un tercero.	Expuestos en el apartado 1.4.1.4	V	Estadística descriptiva.	Paquete de LibreOffice.	*2.6,2.9,2.10*
			Estadística inferencial.		
			Triangulación	Motores de búsqueda, bibliotecas virtuales y revistas especializadas.	
		VI	Programación orientada a objetos	Python	
				PyCharm	

			Triangulación	Motores de búsqueda, bibliotecas virtuales y revistas especializadas.	
			Estadística descriptiva.	Bibliotecas de software.	

CAPÍTULO IV: DIAGNÓSTICO DE LA SITUACIÓN ACTUAL.

El siguiente apartado describe la visión, misión, valores y estrategias empresariales para después comprender de qué forma estos valores se traducen en políticas y estrategias empresariales para comprender la forma en que éstas se articulan con el proyecto en desarrollo.

4.1. Visión, misión, valores y estrategias empresariales.

Desde un punto de vista general, la visión de una empresa está relacionada con lo que planea ser en el futuro y la misión, está más relacionada con las estrategias planteadas para llegar dichos objetivos. (Fernández-Montesinos, 2017)

Según Rowland, 2017, en el caso de Intel, la visión, **"Si es inteligente y está conectado, es mejor con Intel"**, muestra una descripción de lo que la empresa realiza como tal, como lo son la producción de dispositivos conectados e inteligentes a través del uso de los microprocesadores fabricados por la organización, definiendo de esta forma los productos que desea proveer a su público meta. Además, puntúa el hecho de que la compañía se ve a sí misma como la opción más calificada en el mercado para proveer de este tipo de dispositivos que la competencia.

En el caso de la misión, **"Utilizar el poder de la ley de Moore para brindar dispositivos inteligentes que permitan la conexión entre todas las personas de la Tierra"**, Esta se centra en los conceptos de la Ley de Moore, los dispositivos conectados inteligentes y en todas las personas sobre la Tierra. Dicho en otras palabras, la empresa tiene una estrategia basada en resultados en relación con el aumento en la complejidad y el rendimiento de los circuitos integrados que se ve reflejada en la introducción de la Ley de Moore. Además, la misión de Intel sigue la tendencia organizacional plasmada en la producción y comercialización de dispositivos inteligentes y pone en evidencia como público meta al mercado global de semiconductores, procesadores y tecnologías afines al alcance de todas las personas del planeta Tierra.

Con el fin de alcanzar los objetivos planteados, Intel promueve los siguientes valores empresariales: **calidad, toma de riesgos**, **un lugar inclusivo** y **genial para trabajar**, **disciplina**, **orientación hacia el cliente** y **orientación a resultados**.

4.2. Diagnóstico administrativo.

El comportamiento del conglomerado de empleados de Intel, se rige por el **Código de Conducta de Intel**, que enuncia las expectativas de ética e integridad que deben seguir tanto los empleados como los proveedores y contratistas que trabajan con la empresa.

En este código, se desglosan los cinco principios básicos que deben seguir las personas que tienen algún rol dentro de la empresa.

La tabla *n.15* se elaboró a partir de información obtenida del Código de Conducta de Intel (2019).

Tabla 15. Código de conducta de Intel, elaboración propia.

<table>
<tr><td colspan="3">Código de conducta Intel 2019</td></tr>
<tr><td>Valor</td><td>Descripción</td><td>Instrumentos relacionados</td></tr>
<tr><td rowspan="3">Hacer negocios con honestidad</td><td>Comunicación respetuosa, clara y profesional en la actividad comercial</td><td>N/A</td></tr>
<tr><td rowspan="2">Trato equitativo a clientes, proveedores, distribuidores y otros</td><td>Código de conducta de la Coalición ciudadana de la industria electrónica (EICC)</td></tr>
<tr><td>los principios establecidos en la Corporate Policy Statement: U.S. Government Business (CPS)</td></tr>
</table>

	Ser un ciudadano corporativo responsable	Principios sobre Derechos Humanos y en nuestra Política de Ambiente, Salud y Seguridad promulgados por la Organización de las Naciones Unidas (ONU) en el 2018
	Mantenimiento preciso de la contabilidad financiera, demás libros y registros	N/A
Cumplimiento tanto de la letra como del espíritu de la ley.	El antimonopolio	Intel Corporation's Antitrust and Competition Law Worldwide Policy and Standards
	Soborno y anticorrupción	Supplier Anti-Corruption Policy y Third Party Gifts, Meals, Entertainment and Travel (GMET) Policy
	Gestión y cumplimiento de estándares medioambientales, de salud y de seguridad	N/A
	Cumplimiento de importaciones y exportaciones	N/A
	Abuso de información privilegiada	N/A
	Propiedad intelectual	N/A
	Privacidad	los Principios de privacidad y las Reglas de privacidad

		corporativa de Intel
	Comunicaciones públicas	Documentación existente acerca de U.S. Securities and Exchange Commission (Comisión de Valores de EE.UU.).
Tratar a los demás de manera equitativa.	Comunicación abierta y sincera	N/A
	Igualdad de oportunidades laborales y discriminación	N/A
	Anti-acoso	N/A
	Tráfico de personas, trabajo infantil y forzado	N/A
	Seguridad	N/A
	Violencia en el lugar de trabajo	Workplace Threats and Violence guideline de Intel
Actuar en beneficio de Intel y evitar conflictos de intereses	Obsequios, comidas, entretenimiento y viajes (GMET, por sus siglas en inglés)	Política Global de Obsequios, Comidas, Entretenimiento y Viajes de Negocios (Política de GMET).
Proteger los activos físicos y la reputación de la empresa	Protección de los activos físicos	N/A
	Mantenimiento de la seguridad de la información	N/A
	Protección de marcas registradas y marcas comerciales	N/A

4.2. Diagnóstico técnico.

Intel se encuentra en un proceso de transformación que involucra pasar de ser una compañía enfocada en el desarrollo de PCs a estar centrada en datos, lo que le permite entrar al mercado de Internet de las Cosas (IoT, por sus siglas en inglés), potenciar el trabajo con soluciones en la nube, avances en sistemas masivos de memoria y soluciones programables de vanguardia. Sin embargo, la explotación de estas potencialidades solo es posible al aplicar las mejores prácticas en tecnología, y el trabajo en conjunto con aliados estratégicos en procesos de innovación y colaboración en avances tecnológicos.

Según el informe *Driving the digital Enterprise transformation* de Intel (2018-2019) y tal y como se muestra en la imagen *n.24*, este cambio implica una transformación tecnológica continua a partir de los principios de la creación de estaciones de trabajo modernas con empleados letrados digitalmente, la simplificación de los procesos del negocio, aumentando la ventaja competitiva a través de un mejor tratamiento de los datos y la modernización de las plataformas de arquitectura y seguridad en la nube.

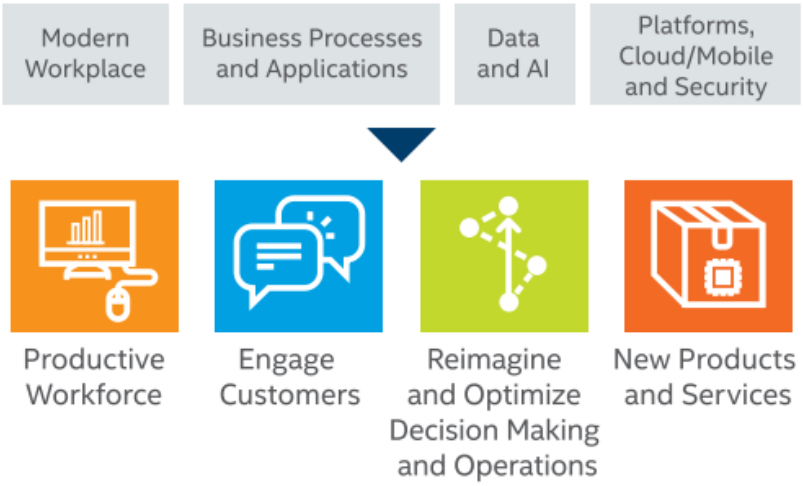


Ilustración 25. Intel Enterprise Digital Transformation Underway. En Driving the digital enterprise transformation (p.4), por Intel., 2019, EEUU: Intel. Derechos de autor [2019] por Intel. Reimpresión autorizada.

Por otra parte, según este mismo informe, este cambio ha significado un enfoque que aumente la velocidad, agilidad y eficiencia de las Tecnologías de la Información (IT, por sus

siglas en inglés), a través de un programa de integración que permita eliminar barreras estructurales y culturales.

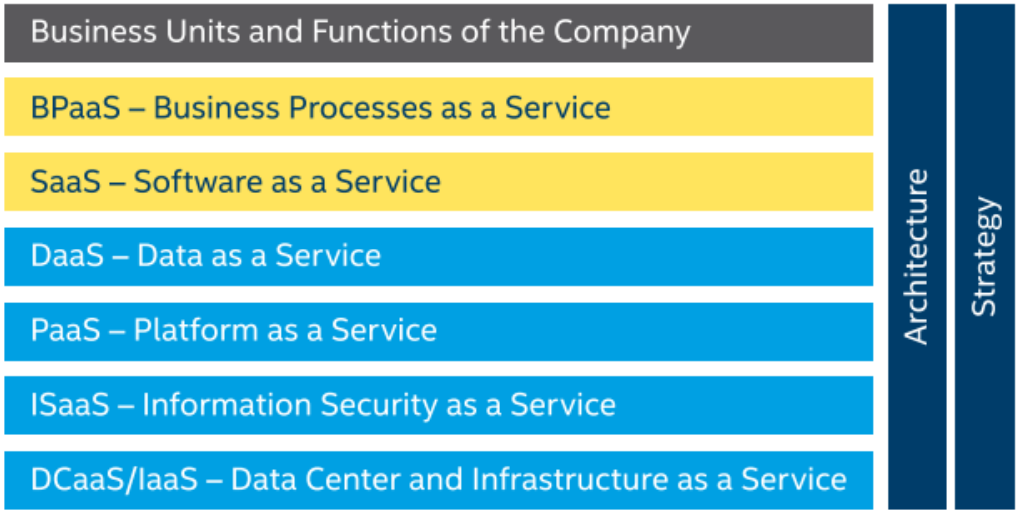


Ilustración 26. Operating model. En Driving the digital enterprise transformation (p.5), por Intel., 2019, EEUU: Intel. Derechos de autor [2019] por Intel. Reimpresión autorizada.

Tal y como se muestra en la ilustración *n.25*, este cambio significa generar flujos de valor de extremo a extremo para aumentar el conocimiento y compromiso empresarial, optimizar y conectar soluciones comerciales y datos, dar soporte continuo y facilitar las transferencias de información y responsabilidades entre los miembros de la organización.

Para lograrlo, Intel trabaja en la actualidad en reducir la deuda técnica con el fin de lograr centrarse en procesos de innovación. Esta mejora se ha enfocado en tres puntos básicos que pueden observarse en la imagen *n.26*.

Ilustración 27. Identify and manage technical debt. En Driving the digital enterprise transformation (p.6), por Intel., 2019, EEUU: Intel. Derechos de autor [2019] por Intel. Reimpresión autorizada.

En donde la identificación y valoración (*identify and Assess*), se refiere al uso del modelo Gartner's TIME: tolerar, invertir, migrar y eliminar; La reducción de la deuda (*Reduce debt*), que desde que se comenzó a permitido la eliminación de cinco mil seiscientas aplicaciones; y la Prevención de nuevas deudas (*Prevent new debt*), que significa la estructuración de un modelo de gobierno de IT lo suficientemente robusto como para prevenir la deuda técnica que podría llegar a surgir.

4.4. Diagnóstico de percepción.

En este apartado se describe la percepción que tienen los empleados acerca de la utilización de herramientas de análisis de rendimiento en sus labores cotidianas. Además, pueden observarse métricas de interés y la relevancia de una herramienta capaz de mostrar la forma en que los recursos de los procesadores Intel de propósito general, de la generación i7, están siendo utilizados.

De los doce miembros del equipo, los doce participaron en el cuestionario. A partir de los datos recolectados se obtuvieron los siguientes resultados:

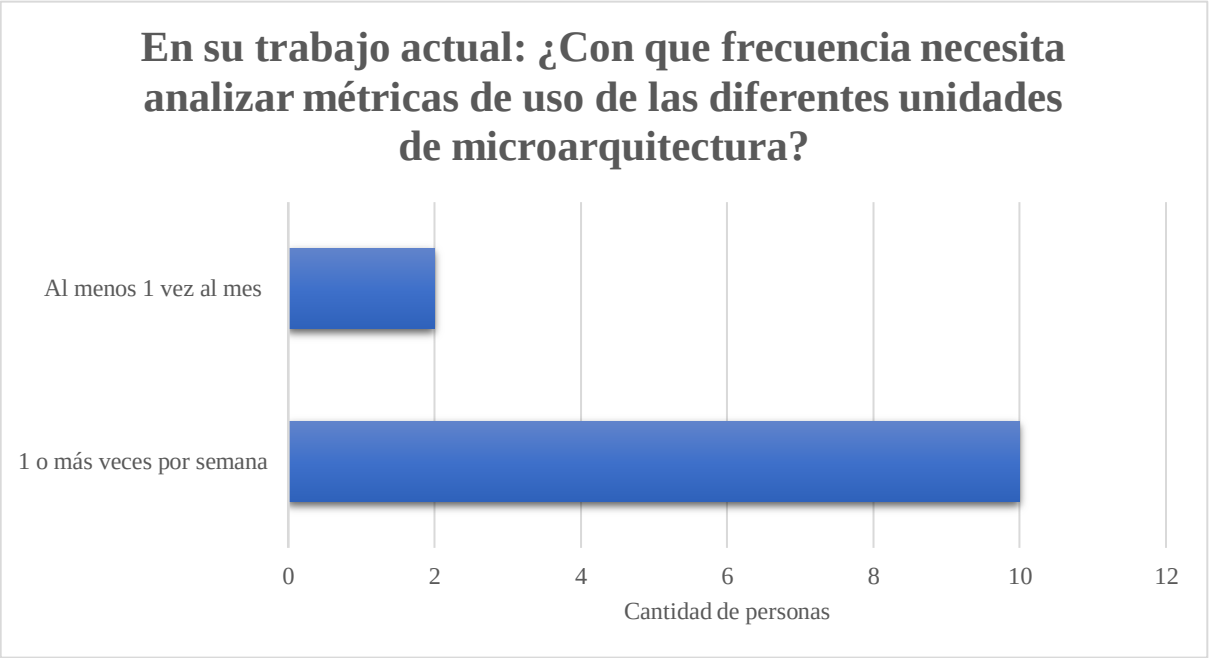


Ilustración 28. Frecuencia de utilización de métricas de rendimiento, elaboración propia.

Como se muestra en la ilustración *n.27*, todos los integrantes del equipo manifestaron la necesidad de utilizar métricas de rendimiento en sus labores cotidianas. Además, mientras 10 de ellos manifestaron hacer uso de métricas de rendimiento una vez por semana, 2 de ellos anotaron que su frecuencia de uso para este tipo de métricas es de al menos una vez al mes.

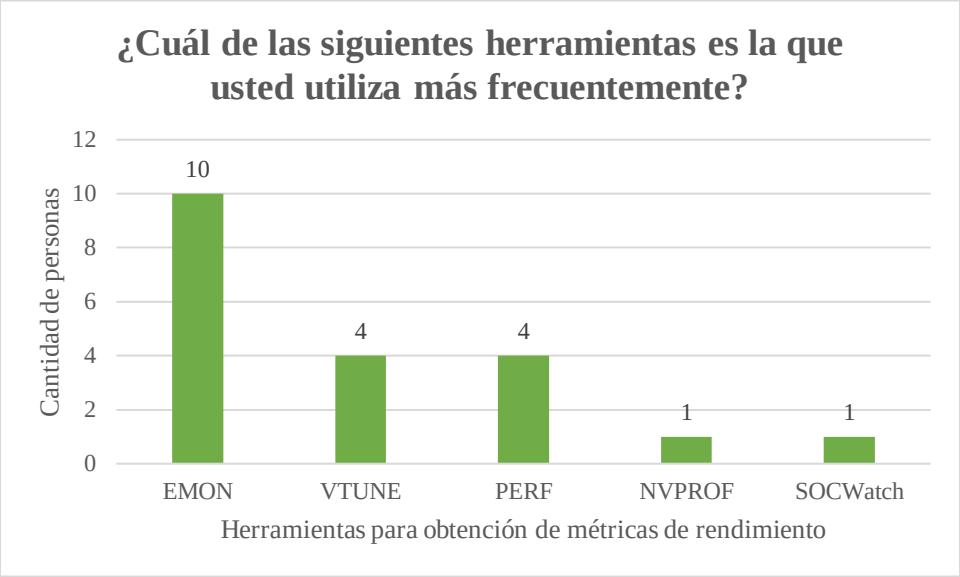


Ilustración 29. Herramientas más utilizadas para obtención de métricas de rendimiento, elaboración propia.

En relación con el conocimiento y la utilización de herramientas para la obtención de métricas de rendimiento, 10 de ellos manifestaron utilizar EMON, 4 de ellos VTUNE y 4 de ellos PERF. Además, tan solo 1 de ellos utiliza NVPROF y 1 SOCWatch.

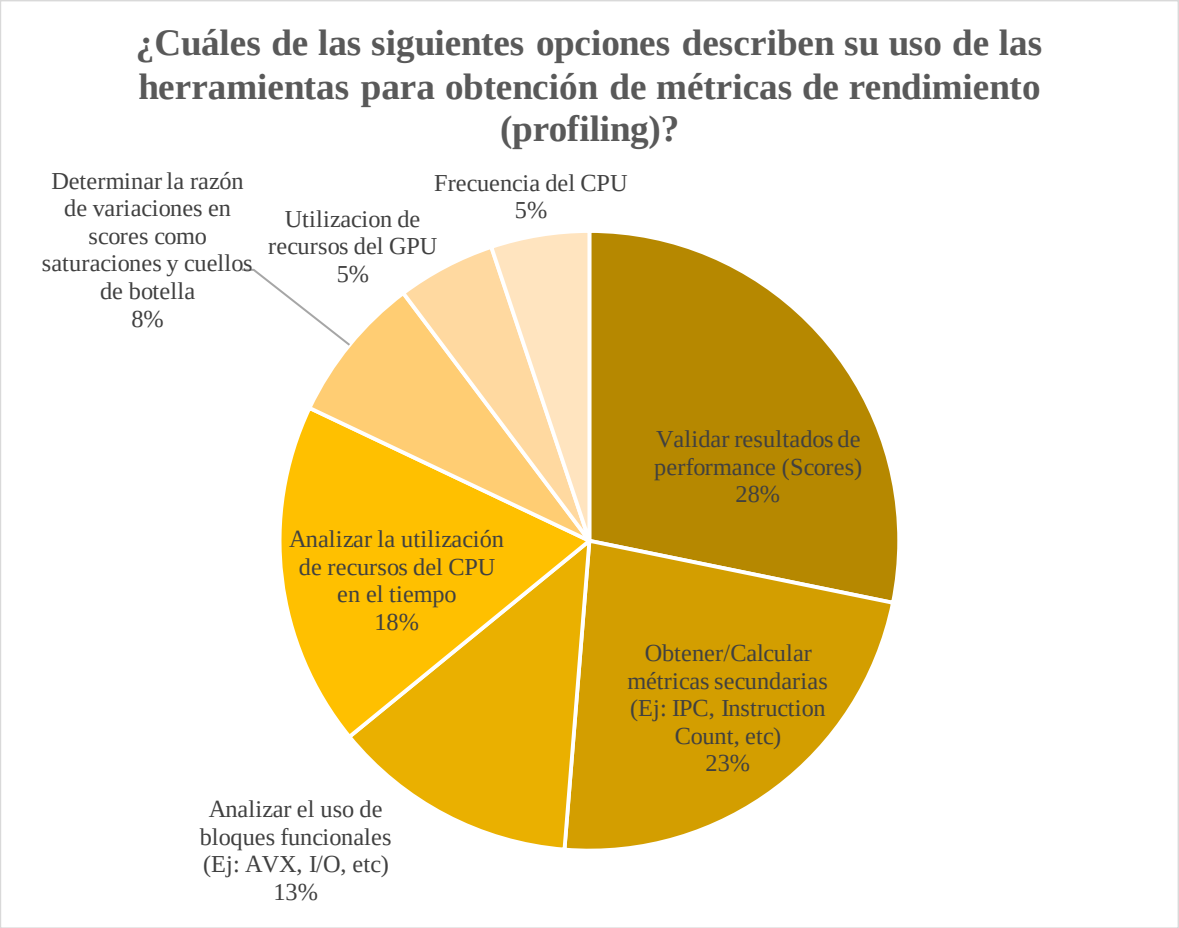


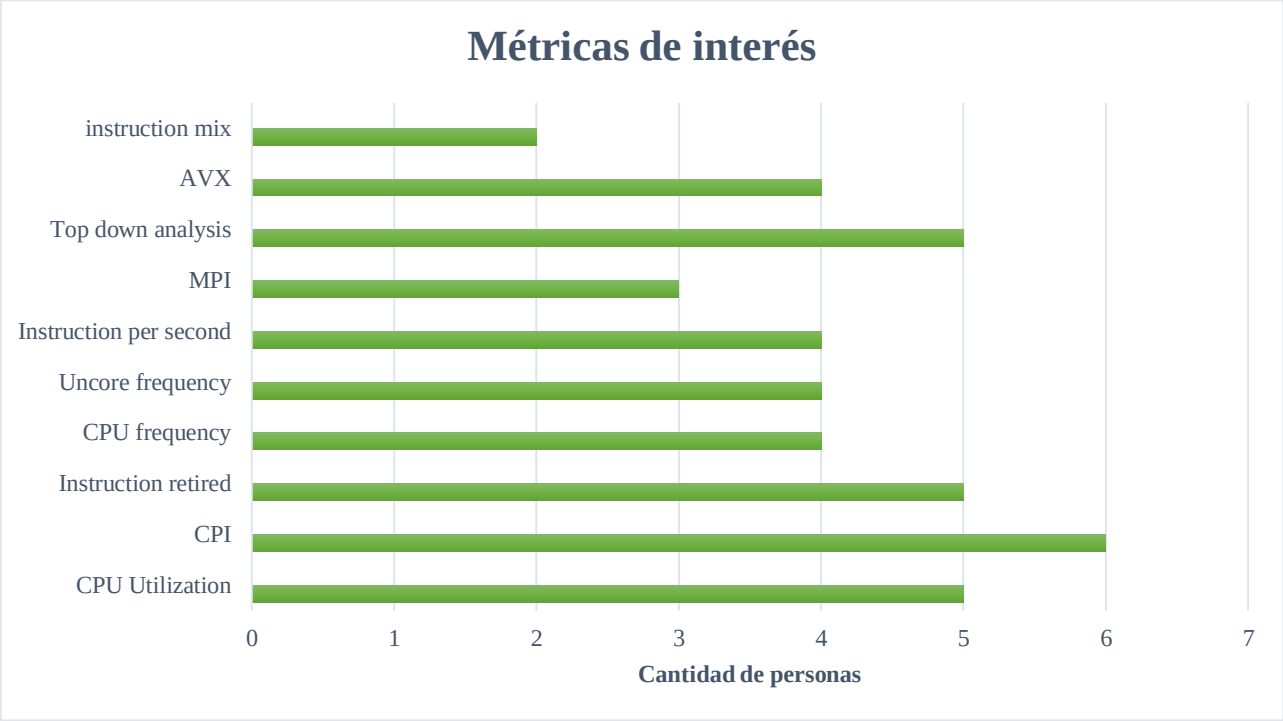
Ilustración 30. Funcionalidad de las herramientas para obtención de métricas de rendimiento, elaboración propia.

El uso que se le da a las herramientas para la obtención de métricas de rendimiento varió en todos los casos. Mientras un grupo mayoritario de miembros correspondiente al 28% indicó utilizar este tipo de herramientas para validar resultados de rendimiento (*scores*), el 23% de los participantes del cuestionario señalaron utilizarlas para obtener métricas específicas como lo son: ciclos por instrucción (IPC, por sus siglas en inglés) y ancho de banda de memoria (*memory bandwidth*). Asimismo, un 13% enunció utilizarlas para analizar la utilización de los recursos del CPU en el tiempo y un 8% denota recurrir a ellas para determinar las razones de variación en s*cores*, como lo son saturaciones o cuellos de botella en el procesador. Finalmente, un 5% de ellos declararon emplearlas para medir la utilización de recursos del CPU y 5% para medir la frecuencia del CPU.

Ilustración 31. Tipo de utilización de métricas de rendimiento, elaboración propia.

En relación con el tipo de validación que se busca averiguar a través de este tipo de herramientas, 7 personas indicación que las usaban solo en configuraciones específicas, 4 personas que las usaban en todos los casos y 3, que las utilizaban en casos atípicos, para obtener indicios y explicar un comportamiento particular.

Ilustración 32. Métricas de interés en tiempo real, elaboración propia.



Finalmente, en relación con las métricas de interés para ser mostradas, se obtuvo el siguiente resultado. 6 personas indicaron que CPI sería una métrica prioritaria, 5 personas consideraron fundamental conocer el desglose de las métricas por porcentaje (*top down analysis*) y 5, que saber el porcentaje de utilización del CPU (CPU *utilization*) podría convertirse en un factor decisivo en el planteamiento de sus análisis. Además, 4 personas indicaron que saber el número de instrucciones por segundo sería crucial, 4 personas indicaron que conocer a frecuencia de *uncore* sería conveniente. Asimismo, 4 personas consideraron que conocer las MPI sería valioso para su trabajo. Finalmente, 4 personas consideraron que conocer las instrucciones AVX aportaría sustancialmente a los resultados obtenidos durante su trabajo.

4.3. Consideraciones finales del diagnóstico.

En esta sección se discute la forma en que el hilo argumentativo, correspondiente al diagnóstico administrativo, el diagnóstico técnico y el diagnóstico de percepción, atraviesa transversalmente al proyecto en desarrollo.

4.3.1. Proyecto en relación con principios empresariales.

La propuesta de proyecto que se define en estas páginas está acorde con el principio de hacer negocios con honestidad al mostrar resultados de forma digital para así no contribuir con el desperdicio de papel y con ello mostrar respeto por los recursos naturales del planeta. Además, se encuentra conforme al principio de cumplimiento tanto de la letra como del espíritu de la ley, al respetar la propiedad intelectual en la citación de autores consultados y los usos posibles según las aplicaciones de software que se encuentran licenciadas. Por otra parte, es coherente con el principio de tratar a los demás de manera equitativa, ya que, tanto para las etapas de diagnóstico como de desarrollo, se propone mantener una comunicación abierta, tanto con el cliente del proyecto como con sus usuarios principales, para así, construir un prototipo que no solo cumpla con las expectativas del cliente, sino con las necesidades de los usuarios principales. Finalmente, el proyecto es consecuente con el principio de protección de activos de Intel, al no compartir información que se considera secreta o confodencial acerca de marcas, sistemas informáticos y convenios a los que se pudo tener acceso durante el proceso de investigación y desarrollo.

4.3.2. La propuesta técnica de Intel en relación con la propuesta de proyecto.

El proyecto a desarrollarse, se enmarca dentro de la movilización de la compañía hacia los centros de datos e información al disminuir la duplicidad de herramientas con fines similares. Además, es compatible con la estrategia de Intel de tener estaciones de trabajo modernas con empleados digitalmente letrados, al favorecer el aprendizaje de una herramienta que cumple de forma menos laboriosa con mayor cantidad de finalidades específicas.

Por otra parte, es coherente con el interés de la empresa por alcanzar cada vez más agilidad, velocidad y eficiencia de la información, al implicar una menor inversión de tiempo laboral para obtener los resultados. Asimismo, se enmarca dentro de las iniciativas puestas en marcha para aumentar el valor extremo a extremo, al aportar mayor cantidad de métricas

útiles para los análisis de rendimiento realizados de forma cotidiana en el equipo. Además, facilita las transferencias de información entre los miembros del equipo, al minimizar el tiempo de capacitación, al aportar el prototipo de una herramienta de la que pueden obtenerse las métricas de mayor relevancia para los miembros.

Finalmente, es coherente con la estrategia de Intel acerca de la reducción de la deuda técnica, al invertir en una solución a la que se podrá migrar después de haber sido puesta en producción y que a su vez previene la existencia de nuevas deudas de naturaleza similares en el futuro al contar con una documentación transparente acerca de los pasos que dieron origen a la idea y otra documentación en relación con el diseño y funcionamiento del prototipo como tal.

4.3.3. Recomendaciones a partir de diagnóstico de percepción.

Los resultados arrojados por el cuestionario aplicado, hicieron constatar que todos los miembros del equipo utilizan métricas de rendimiento para el desempeño de sus labores. Los datos aportados, también permitieron observar que las herramientas utilizadas por el equipo por orden de prioridad son: EMON, VTUNE, PERF, NVPROF y SoCWatch.

Igualmente, permitieron notar que las métricas más utilizadas por los miembros del equipo son: la utilización de recursos del CPU en el tiempo, la utilización de bloques funcionales como AVX, IPC, utilización de la GPU integrada y frecuencia del CPU.

Aparte, fue posible detectar casos de utilidad como lo son la detección de saturaciones o cuellos de botella entre los componentes, validaciones en configuraciones específicas, investigación de comportamientos anómalos y validación de resultados de mediciones en todas las configuraciones de los experimentos.

En conclusión, la mayor parte de los miembros del equipo consideraron de utilidad para el desempeño de sus labores el contar con una herramienta que arroje los resultados de las métricas de interés. Asimismo, anotaron que otros aspectos de interés son: análisis *top-down*, *mixes* de instrucciones y cantidad de instrucciones por segundo/milisegundo.

CAPÍTULO V: PROPUESTA DEL PROYECTO.

En el presente apartado se desarrollan las etapas expuestas en el apartado *3.5.*, sobre el Diseño de la investigación. Asimismo, se incluyen los entregables expuestos en los alcances descritos en el punto *1.4.1.* del proyecto, en el orden en que aparecen en la matriz de coherencia presentada en el punto *3.6.* de este escrito. Para ello, se ha desglosado la matriz de coherencia propuesta en cada una de las etapas, para que pueda observarse la lógica entre los objetivos y los esfuerzos concretados en cada una de las etapas.

5.1. Etapa I: Definición del problema.

Tal y como se observa en la *tabla n.16,* esta primera etapa del proyecto se exponen los requerimientos funcionales y no funcionales del prototipo que se realizará. Los requerimientos se han definido y redefinido de acuerdo a las necesidades detectadas tanto del cliente como de los usuarios principales del sistema.

Tabla 16. Desglose de la primera etapa del proyecto, elaboración propia.

Objetivos	Entregables	Etapas	Técnicas	Herramientas	Temas del marco teórico
Identificar los requerimientos de software para la elaboración de un prototipo funcional que monitoree la utilización de las unidades funcionales de los microprocesadores Intel core i7.	Expuestos en el apartado 1.4.1.1	I	Entrevista estructurada.	Cuestionario.	*2.6, 2.9*
			Entrevista semiestructurada.	Guía de entrevista.	
			Observación participante.	Sistemas computacionales.	
			Triangulación.	Resultados de cuestionarios, guía de entrevista y sistemas computacionales.	

5.1.1. Requerimientos funcionales.

En este apartado se describen la funcionalidad que se espera del prototipo, por tanto, se especifica el sistema en el que se espera el prototipo sea desarrollado, además de los requisitos mínimos para que cumpla con las expectativas tanto del cliente como de los usuarios principales.

5.1.1.1. Especificaciones sobre el procesador y el sistema.

Las especificaciones siguientes muestran en detalle las características del sistema en el que se realizará el prototipo y la forma en que deben ser desplegados. Además, se expone la relación entre el prototipo, la herramienta de monitoreo y las métricas seleccionadas para el proyecto.

Tabla 17. R1: Mostrar Especificaciones del sistema, elaboración propia.

ID del Requerimiento:	**R1**
Nombre	Mostrar especificaciones del sistema.
Módulo:	Sistema
Objetivo:	Mostrar las especificaciones del sistema que se monitorea.
Descripción:	El prototipo deberá ser capaz de obtener la información de un procesador con las siguientes características de hardware.
Importancia/Prioridad:	**Media**
Elementos de entrada de datos:	N/A
Elementos de resultados de datos:	El sistema debe desplegar la siguiente información: a. Fabricante Intel. b. Familia Core: i7. c. Serie: i7-6000. d. Arquitectura: Skylake. e. GPU incorporada. f. Arquitectura industrial estándar (ISA): x86-64bits (x86).
Restricciones:	N/A

Tabla 18. R2: Recolección de métricas, elaboración propia.

ID del Requerimiento:	**R2**
Nombre	**Recolección de métricas.**
Módulo:	Monitoreo.
Objetivo:	Monitorear eventos de hardware de procesador.
Descripción:	El prototipo deberá ser capaz de monitorear los eventos de hardware de un procesador con las siguientes características: a. Fabricante Intel. b. Familia Core: i7. c. Serie: i7-6000. d. Arquitectura: Skylake. e. GPU incorporada. f. Arquitectura industrial estándar (ISA): x86-64bits (x86).
Importancia/Prioridad:	**Alta**
Elementos de entrada de datos:	N/A
Elementos de resultados de datos:	El prototipo debe recolectar eventos de hardware y transformarlos en métricas de rendimiento cuantificables.
Restricciones:	Se recolectarán sólo las métricas de hardware soportadas por la herramienta de monitoreo utilizada.

Tabla 19. R3: Sistema destino, elaboración propia.

ID del Requerimiento:	**R3**
Nombre	**Sistema destino.**
Módulo:	Monitoreo.
Objetivo:	Muestrear en sistema destino.
Descripción:	El prototipo deberá poder ser ejecutado en un sistema con las siguientes características de software: a. Sistema operativo: Ubuntu 18.04.2 LS b. Kernel: Linux 4.15.0.

	c. Controlador de Bios (SMBIOS): 2.8. d. Bios: tiempo de ejecución 64KB y tamaño ROM 16MB.
Importancia/Prioridad:	**Alta**
Elementos de entrada de datos:	N/A
Elementos de resultados de datos:	El sistema destino asigna recursos al prototipo para ser ejecutado.
Restricciones:	La utilización de otras versiones de Linux puede limitar o no soportar el prototipo.

5.1.1.2. Otros requerimientos funcionales.

Los requerimientos que se exponen a continuación se refieren al manejo de la reportería que hará el prototipo.

Tabla 20. R4: Registrar mediciones realizadas, elaboración propia.

ID del Requerimiento:	**R4**
Nombre	**Registrar mediciones realizadas.**
Módulo:	Reportería
Objetivo:	Mantener un registro de los eventos monitoreados en el sistema destino.
Descripción:	El prototipo deberá clasificar las métricas según las unidades funcionales de arquitectura seleccionadas para el desarrollo del proyecto y mostrar los porcentajes de uso de cada una de las unidades durante la recolección.
Importancia/Prioridad:	Alta
Elementos de entrada de datos:	El usuario deberá seleccionar de una lista o menú, la aplicación que desea sea monitoreada.

Elementos de resultados de datos:	El sistema deberá guardar un archivo con los resultados de las métricas monitoreadas.
Restricciones:	El sistema sólo mostrará las métricas soportadas por la herramienta de monitoreo seleccionada.

Tabla 21. R5: Inicio y finalización de recolección de datos, elaboración propia.

ID del Requerimiento:	**R5**
Nombre	**Inicio y finalización de recolección de datos.**
Módulo:	Pantalla principal
Objetivo:	Generar un mecanismo que permita a los usuarios iniciar y culminar la recolección.
Descripción:	El prototipo deberá proveer de un mecanismo para el inicio de recolección de las métricas (ej. botón de inicio o comando específico de teclado) y un mecanismo para detener la recolección (ej. botón de finalizar o comando específico de teclado).
Importancia/Prioridad:	**Alta**
Elementos de entrada de datos:	El usuario debe desencadenar el mecanismo de inicio o fin de recolección.
Elementos de resultados de datos:	El sistema debe comenzar o finalizar el proceso de recolección de métricas.
Restricciones:	N/A

5.1.1.3. Especificaciones de presentación.

Los requerimientos que siguen están relacionados con la presentación de los resultados en pantalla.

Tabla 22. R6: Síntesis de resultados, elaboración propia.

ID del Requerimiento:	**R6**
Nombre	**Mostrar síntesis de resultados.**
Módulo:	Reportería.
Objetivo:	Proveer de un mecanismo de síntesis para mostrar los resultados de la medición realizada.
Descripción:	El prototipo debe proveer de un mecanismo visual para mostrar una síntesis por porcentajes sobre el uso de las unidades de arquitectura seleccionadas.
Importancia/Prioridad:	Media
Elementos de entrada de datos:	El usuario selecciona desde un menú, el tipo de síntesis que desea observar.
Elementos de resultados de datos:	El sistema debe desplegar una pestaña o pantalla con la síntesis seleccionada
Restricciones:	N/A

Tabla 23. R7: Guardar síntesis de recolección, elaboración propia.

ID del Requerimiento:	**R7**
Nombre	**Guardar síntesis de recolección.**
Módulo:	Reportería.
Objetivo:	El sistema deberá proveer de un mecanismo para guardar las síntesis expuestas en el requerimiento D6.

Descripción:	El sistema debe permitir al usuario de un mecanismo visual para guardar las síntesis generadas por el sistema.
Importancia/Prioridad:	**Baja**
Elementos de entrada de datos:	El usuario debe activar el mecanismo para guardar la síntesis generada por el sistema.
Elementos de resultados de datos:	El nuevo archivo se debe ver reflejado en el sistema destino.
Restricciones:	N/A

5.1.2. Requerimientos no funcionales.

Los requerimientos funcionales definen las especificaciones relacionadas con las formas de almacenamiento y formato para salvar la información, entre otras convenciones e establecen para que se asemejen a las aplicaciones que el equipo utiliza en la actualidad y así disminuir las barreras correspondientes a la resistencia que podrían experimentar algunos miembros del equipo para utilizar una herramienta nueva.

Tabla 24. R8: Paradigma de programación, elaboración propia.

ID del Requerimiento:	**R8**
Nombre	**Paradigma de programación.**
Módulo:	Escalabilidad y mantenimiento del sistema
Objetivo:	Realizar un prototipo escalable tanto en funcionalidades como en portabilidad.
Descripción:	El prototipo deberá ser desarrollado por medio de elementos propios de la programación orientada a objetos con el fin de facilitar su escalabilidad, mantenimiento y portabilidad posteriores.

Importancia/Prioridad:	**Alta**
Elementos de entrada de datos:	N/A
Elementos de resultados de datos:	N/A
Restricciones:	N/A

Tabla 25. R9: Guía y documentación, elaboración propia.

ID del Requerimiento:	**R9**
Nombre	**Guía y documentación.**
Módulo:	Escalabilidad y mantenimiento del sistema
Objetivo:	Aportar una documentación que facilite la escalabilidad y mantenimiento del sistema.
Descripción:	La entrega del prototipo deberá estar acompañada de una guía rápida de uso, así como de diagramas y documentos que faciliten su compresión.
Importancia/Prioridad:	**Alta**
Elementos de entrada de datos:	N/A
Elementos de resultados de datos:	N/A
Restricciones:	N/A

Tabla 26. R10: Ambiente y lenguaje de programación, elaboración propia.

ID del Requerimiento:	**R10**

Nombre	**Ambiente y lenguaje de programación.**
Módulo:	Escalabilidad y mantenimiento del sistema
Objetivo:	Utilizar herramientas con las que el equipo se encuentra familiarizado para el desarrollo del prototipo.
Descripción:	El prototipo se desarrollará en el lenguaje de programación Python y con el ambiente de programación Pycharm.
Importancia/Prioridad:	**Alta**
Elementos de entrada de datos:	N/A
Elementos de resultados de datos:	N/A
Restricciones:	Se utilizó el emulador de Python 3.0, la utilización de otra versión del emulador puede limitar la funcionalidad del prototipo de forma parcial o total.

Tabla 27. R12: Formato de reportes, elaboración propia.

ID del Requerimiento:	**R11**
Nombre	**Formato de reportes.**
Módulo:	Reportería
Objetivo:	Estandarizar la reportería.
Descripción:	Los registros mencionados en R4 deberán guardarse en formato *csv*. Por otra parte, las síntesis mencionadas en R7 deberán guardarse en un cualquiera de los siguientes formatos: *jpg*, *png* o *jpeg*.
Importancia/Prioridad:	Baja

Elementos de entrada de datos:	N/A
Elementos de resultados de datos:	N/A
Restricciones:	N/A

5.1.3. Logros obtenidos durante la Etapa I del proyecto.

En la primera parte del proyecto se logra delimitar las expectativas tanto del cliente como del equipo que utilizará el prototipo. Estas expectativas se sistematizaron por medio de requerimientos funcionales y no funcionales que guiarán el desarrollo del resto de las etapas.

5.2. Etapa II: Selección de métricas.

Tal y como se muestra en la *tabla n.28*, en este apartado se describe la especificación de las herramientas de monitoreo a bajo nivel que se tomarán en cuenta, unidades de microarquitectura que se utilizarán y su relación con las métricas seleccionadas.

Tabla 28. Desglose de la segunda etapa del proyecto, elaboración propia.

Objetivo	Entregables	Etapas	Técnicas	Herramientas	Temas del marco teórico
Analizar la relación entre la arquitectura del procesador, sus unidades funcionales y el ejercicio de éstas unidades al ejecutarse la aplicación de un tercero.	Expuestos en el apartado 1.4.1.2	II	Revisión bibliográfica.	Motores de búsqueda, bibliotecas virtuales y revistas especializadas.	*2.1,2.2,2.3,2.4,2.5,2.6,2.7,2.8*

5.2.1. Un acercamiento al rendimiento a través de las métricas.

En la actualidad, la mayor parte de los procesadores tienen microarquitecturas complejas y sistemas que permiten ser monitoreados a través de contadores que se encuentran dentro del hardware. (Ammons, Ball, & Larus, 2005)

Según William et al. (2010), usualmente, las herramientas utilizadas para la realización de análisis de rendimiento hacen uso de estos contadores con el fin de determinar el consumo de recursos durante la ejecución de aplicaciones de usuario. Estas herramientas rastrean secuencias de instrucciones, registrando la ocurrencia de ciertos eventos durante los flujos de ejecución del programa y pueden hacer uso de varios acercamientos:

a. Hotspots: es una técnica mediante la cual se identifica la cantidad de tiempo invertida por el procesador para ejecutar las unidades de un programa.

b. Muestreo: significa la interrupción periódica del sistema o el procesamiento de datos a través de intervalos regulares. En cada interrupción, la herramienta guarda el tiempo invertido en el procesamiento de datos, el número de eventos registrados y el lugar en el que ocurrieron.

Comúnmente, estas herramientas hacen uso de otras herramientas para la recolección de los datos y el procesamiento posterior que permitirá extraer la información final.

Por tanto, es importante considerar, que, si bien son herramientas de mucha utilidad para conocer la distribución del tiempo durante el tiempo de ejecución de la aplicación de usuario, no son capaces de diagnosticar el porqué de esta distribución ni la optimización de dichos procesos. (LaFrance-Linden, 2014)

En el apartado que sigue, se describe la herramienta seleccionada y las razones que llevaron a su selección. Para ver el detalle del instrumento utilizado para la elección de la herramienta, ver el anexo 2.

5.2.2. Anotaciones sobre VTUNE Amplifier.

Tal como se indica en Intel® VTune™ Amplifier User Guide (2019), *VTune Amplifier* es una herramienta utilizada para hacer análisis de rendimiento en aplicaciones seriales y de multiproceso. A su vez, permite hacer análisis tanto locales como remotos. A continuación, se detallan las razones por las que se escogió esta herramienta:

a. Razón I. Métricas de rendimiento: *VTune Amplifier* permite la obtención del rango más amplio de métricas de rendimiento. Abarcando las métricas de casi todas las otras herramientas excepto de IMIX.

b. Razón II. Manejo de energía y temperatura: el manejo de energía y en general la temperatura de un procesador permite conocer la medida en que este se encuentra en dificultades para cumplir con los pedidos sobre los servicios que ofrece. Además, permite conocer la capacidad del procesador para distribuir recursos y destinar energía de acuerdo a la utilización que se le da en un momento específico. (Stallings, 2010) VTune no permite la obtención de estas métriacs, sin embargo, puede acoplarse con otras herramientas como *SocWatch* para las etapas de postprocesamiento, permitiendo a su vez que el sistema pueda escalarse al integrar otras herramientas utilizadas dentro del equipo para enriquecer los resultados.

c. Razón III. Tipo de licencia: el tipo de licencia fue un factor determinante en la selección de la herramienta, ya que, las licencias privadas no pueden ser utilizadas de forma externa y las versiones técnicas y empresariales implican una inversión económica significativa. Las versiones para educadores o contribuyentes de software libre suelen ser más accesibles económicamente y estar menos restringidas en sus formas de uso. Las versiones de estudiante usualmente incluyen todas las funcionalidades y su uso se limita a fines académicos y de investigación. Las herramientas de software libre, permiten por lo general una funcionalidad completa de forma gratuita. En el caso de *PERF*, por ejemplo, esta funcionalidad se extiende a la modificación del código existente. La existencia de una licencia de estudiante para el uso de *VTune Amplifier*, resulta ideal para el proyecto, ya que permite obtener todas funcionalidades de la aplicación y descarga gratuita.

d. Razón IV. Sistema operativo: Todas las herramientas analizadas permiten la obtención de las métricas en Linux, *VTune Amplifier* entre ellas.

e. Razón V. Tipo de interfaz: El acoplamiento del software con la herramienta de obtención de métricas de rendimiento se hará por medio de comandos de sistema. *VTune Amplifier* tiene una biblioteca con una larga lista de comandos para la recolección de las métricas seleccionadas, especificadas por tipo de análisis (permitiendo llamar a colección de varios grupos de métricas con pocos comandos) o por métrica de interés.

f. Razón VI. Capacidad de recolectar resultados en tiempo real: el detalle que permiten las herramientas de monitoreo es importante. De igual forma, es

indispensable que la herramienta utilizada para la obtención de las distintas métricas seleccionadas, también sea capaz de recolectar los resultados en tiempo real para garantizar una mayor fiabilidad de los datos a través de la muestra. *VTune Amplifier* también comparte esta característica.

A partir del estudio de las posibles herramientas seleccionadas para la obtención de métricas de rendimiento, se concluye que *VTune Amplifier* es la herramienta de monitoreo más adecuada para las necesidades del proyecto en desarrollo, ya que permite la recolección de la mayor cantidad de métricas de rendimiento, permite análisis de energía y temperatura, permite la obtención de una licencia de estudiante para uso académico y de investigación. Además, soporta Linux, se puede manipular completamente desde línea de comandos y tiene la capacidad de arrojar resultados en tiempo real.

5.2.3. Instalación y ejecución de la herramienta seleccionada.

VTune Amplifier es una herramienta producida por Intel, se puede acceder a ella descargando la suite Intel® Parallel Studio XE[1].

En las siguientes líneas se exponen los pasos que según Intel VTune Amplifier (2019), son necesarios para la instalación de *VTune Amplifier,* así como los comandos necesarios para hacer la aplicación funcionar en Ubuntu.

Para la instalación de *VTune Amplifier* se deben seguir los siguientes pasos:

a. **tar -xzf vtune_amplifier_<*version*>.tar.gz** *(para descomprimir el archivo).*

b. Ir al directorio en donde se encuentra la carpeta descomprimida.

c. **./install_GUI.sh**

Establecer de las variables de ambiente:

a. **directorio/de/instalación/amplxe-vars.sh**

[1] Para hacerlo, se debe accederse a: https://software.intel.com/en-us/vtune/choose-download#parallel-technical y seleccionar la opción que corresponda.

Para comprobar que *VTune Amplifier* se encuentra instalado de forma correcta se puede hacer funcionar un script de prueba ubicado en directorio/de/instalación/bin64/amplxe-self-checker.sh

a. **directorio/de/instalación/amplxe-cl**

5.2.4. Detección y selección de unidades de microarquitectura del procesador seleccionado.

Además de la herramienta de monitoreo, es necesario definir las unidades de microarquitectura que serán monitoreadas, de acuerdo a la arquitectura del procesador con el que se trabajará, las posibilidades de la herramienta seleccionada, los puntos expuestos en los alcances del proyecto y los resultados del diagnóstico realizado.

En este apartado se documenta el proceso seguido para la selección de las unidades funcionales del procesador monitoreadas por el prototipo. Para lograrlo, se siguió una estrategia acumulativa (Stallings, 2010), por lo que se partió de los diagramas más generales de un procesador, hasta diagramas más detallados específicos del procesador seleccionado para la realización de los experimentos.

Para comprender los componentes básicos del procesador se buscó un diagrama de bloques con una arquitectura similar al procesador del estudio. Como se ve en la imagen *n.32*, el primer diagrama de referencia tiene 2 *cores* físicos, gráficos integrados y un sistema agente.

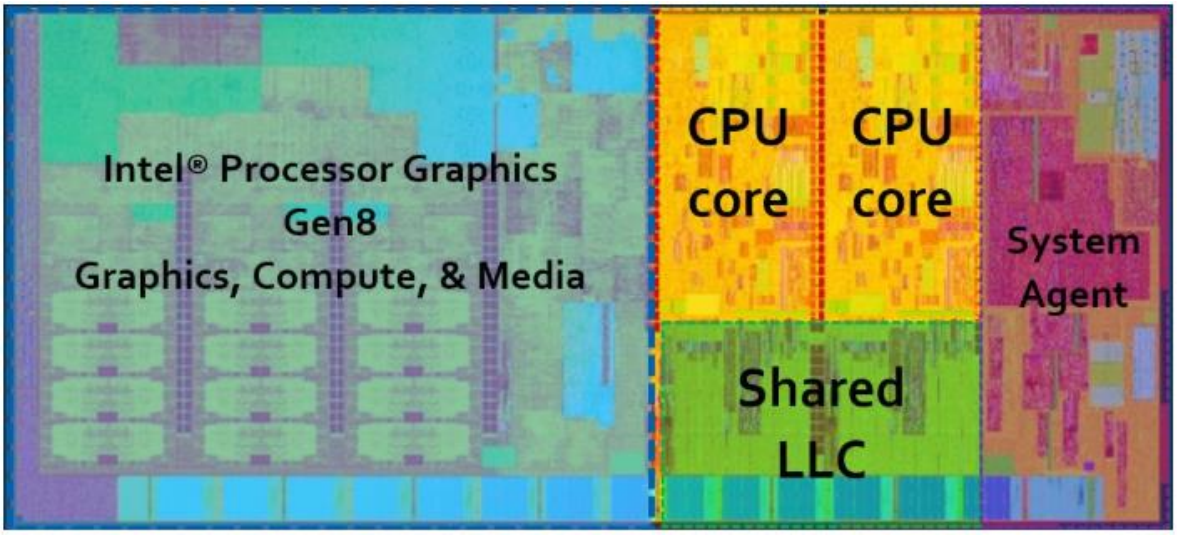


Ilustración 33.Silicon die layout for an Intel® Core™ M Processor. En for an Intel® Core™ M Processor (p.2), por Intel Corporation, 2014, EEUU: Intel Corporation. Derechos de autor [2010] por Intel Corporation. Reimpresión autorizada.

De esta forma, las Unidades Centrales de Procesamiento del core (CPUs) comparten un último nivel de caché (LLC). La imagen además muestra un Sistema Agente y una Unidad de Gráficos Integrados.

Posteriormente, se busca un diagrama más específico, detallando las partes del core i7 6700U.

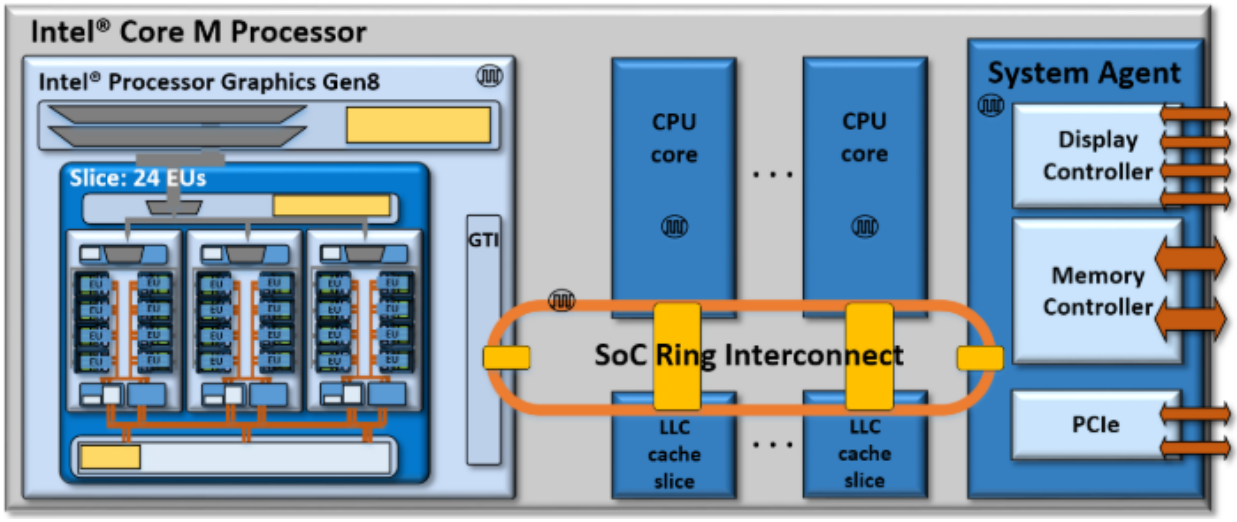


Ilustración 34. An Intel® Core™ M Processor SoC and its ring interconnect architecture. En for an Intel® Core™ M Processor (p.3), por Intel Corporation, 2014, EEUU: Intel Corporation. Derechos de autor [2010] por Intel Corporation. Reimpresión autorizada.

Tal y como puede observarse en la imagen *n.33*, el segundo diagrama analizado es más específico, por lo que cuenta con indicadores para los dominios de reloj de cada uno de los componentes y la forma en que interactúan entre sí. Asimismo, describe otros

componentes en el interior del sistema agente como lo son el controlador de visualización, controlador de memora y PCIe, relacionado con la paginación y la memoria virtual.

Estas primeras impresiones dieron como origen el siguiente diagrama:

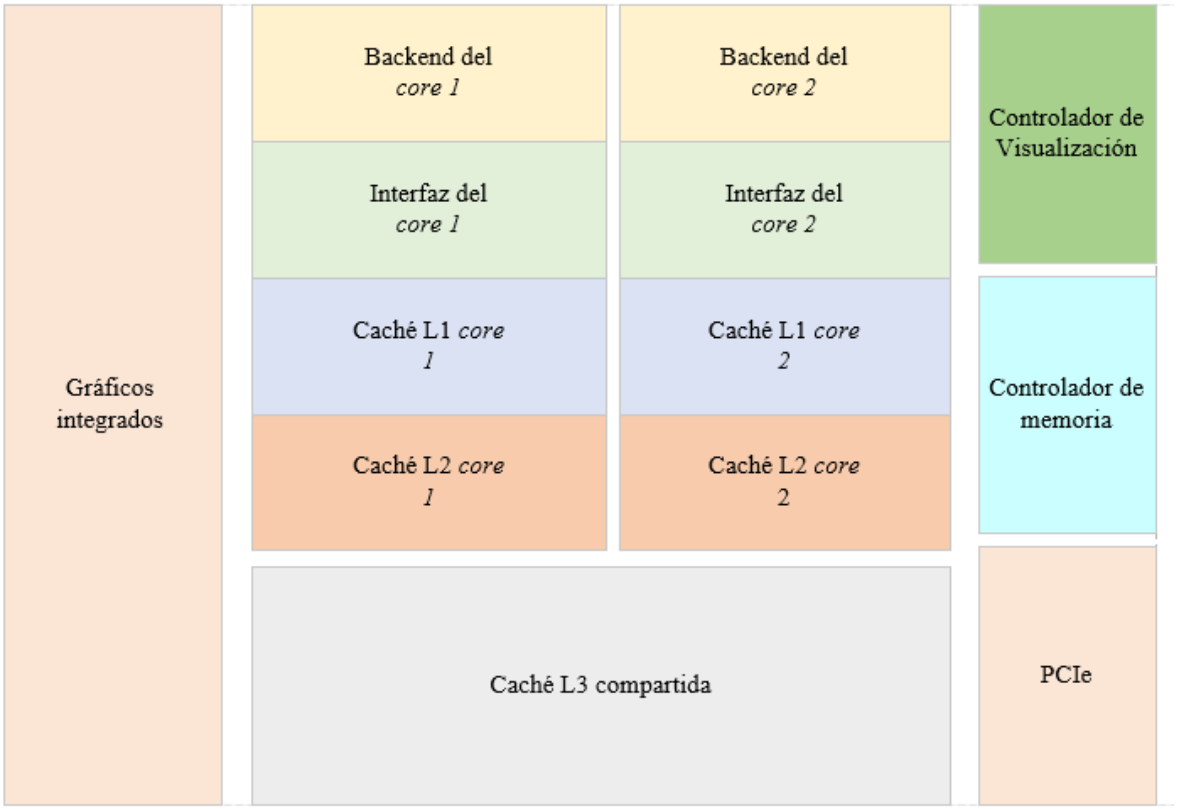


Ilustración 35. Primer diseño preliminar de procesador, elaboración propia.

El diagrama preliminar presente en la imagen *n.34*, muestra las unidades de backend, interfaz, L1 y L2 de cada uno de los *cores*. Aparte de esto, rescata la unidad de gráficos integrados y algunas unidades comunes de los sistemas agentes: controlador de visualización, de memoria y PCIe.

Posteriormente se hizo un análisis del siguiente diagrama, que muestra las unidades funcionales en el interior de un *core*:

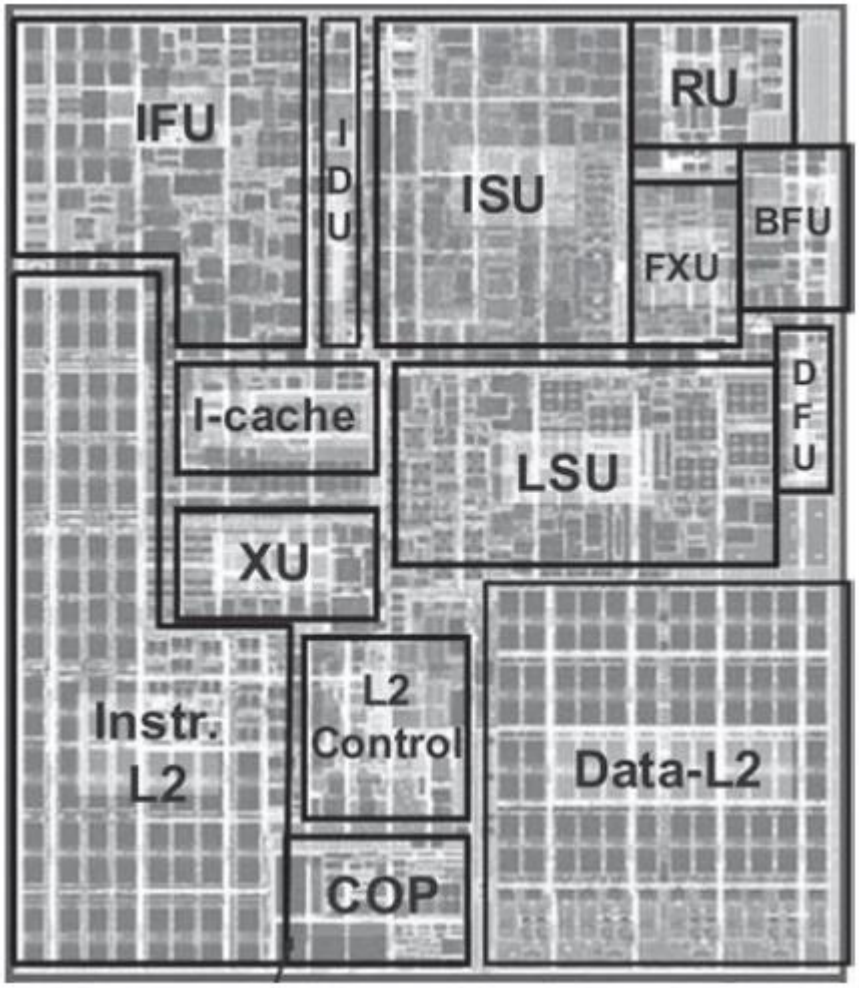


Ilustración 36. Computer. Top level Structure. En Computer Organization and Architecture (p.10), por Stallings, W, 2010,Connecticut, New England: Pearson Education. Derechos de autor [2010] por Pearson. Reimpresión autorizada.

Seguidamente se detalla, desde el punto de vista de Stallings, W. (2010), cada una de las unidades presentes en la ilustración *n.35*:

a. Secuenciador de instrucciones (ISU): se encuentra en las arquitecturas superescalares, es la unidad encargada de determinar la secuencia en la que las instrucciones son ejecutadas.

b. Unidad de Carga de Instrucciones (IFU): unidad que se encarga de la lógica que permite la carga de datos desde memoria.

c. Unidad de decodificación de instrucciones (IDU): esta unidad se alimenta de la Unidad de Carga de Instrucciones, por lo que se encarga de analizar y decodificar los códigos de operación.

d. Unidad de carga y almacenamiento (LSU): esta unidad contiene 96KB de cache de datos de tipo L1. Asimismo, esta unidad se encarga del tráfico de datos entre la cache de datos de tipo L2 y las unidades funcionales del procesador. A su vez, esta unidad maneja los accesos a operandos de todos los tamaños, modos y formatos que se han definido de acuerdo a la arquitectura específica de cada procesador.

e. Unidad de traducción (XU): esta unidad traduce las direcciones lógicas de las instrucciones en direcciones físicas que pueden ser rastreados en la memoria principal. Esta unidad también contiene un Lookaside Buffer (TLB) utilizado para aumentar la velocidad de los accesos a memoria.

f. Unidad de punto fijo (FXU): ejecuta operaciones aritméticas de punto fijo.

g. Unidad decimal de punto flotante (DFU): soporta tanto operaciones de punto fijo como operaciones de punto flotante en números en formato decimal.

h. Unidad de recuperación (RU): mantiene una copia completa del estado del sistema. Esta copia incluye todos los registros y las señales de fallo del *hadware*. Esta unidad también realiza acciones de recuperación en el *hardware* cuando se necesita.

i. Coprocesador (COP): es la unidad responsable de la encriptación de las funciones en cada uno de los *cores*.

j. I-cache: Es una memoria de instrucciones de tipo L1. Permite a la IFU, la precarga de instrucciones antes de que se necesiten.

k. control de L2: controla la lógica del tráfico entre las caches de tipo L2.

l. Data-L2: es una memoria de datos de 1MB de tipo L2 utilizada para todo el tráfico que no está relacionado con instrucciones.

De esta exploración de las unidades en el interior del *core* que se muestran en la ilustración *n.35*, y las unidades funcionales del procesador observadas en las ilustraciones n.32-34 se desprende la siguiente propuesta de unidades para monitoreo:

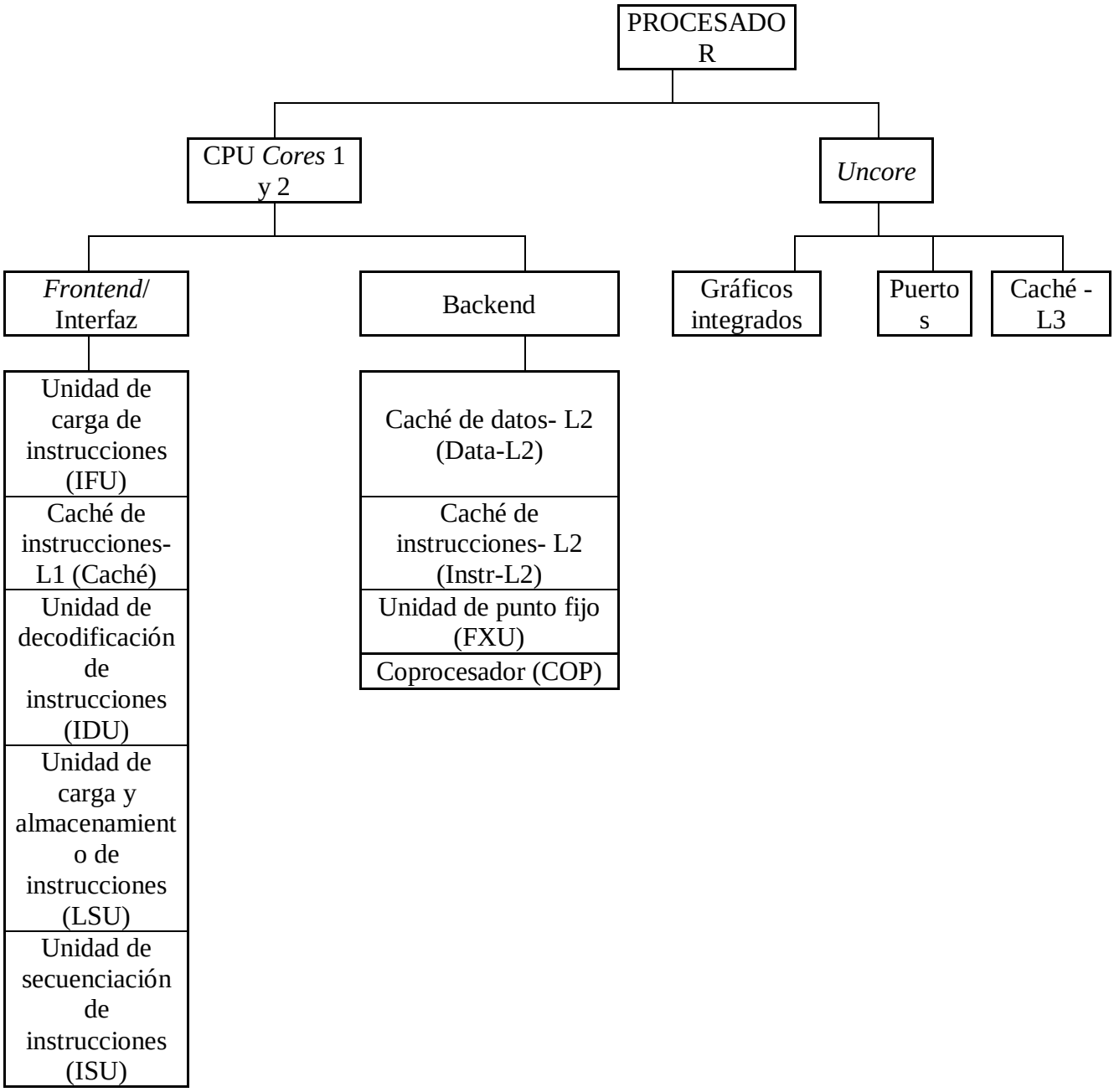


Ilustración 37. Unidades funcionales procesador Intel core i7, elaboración propia.

Finalmente, se hace una triangulación entre el diagnóstico realizado, los alcances y limitaciones del proyecto, la herramienta seleccionada y las unidades del procesador anotadas

en el esquema de la imagen *n.37,* para definir las unidades que se monitorearán y las métricas que se asocian a cada una de esas unidades.

El detalle de la relación entre las métricas y las unidades funcionales del procesador seleccionadas se encuentra en el anexo 3. Asimismo, se aporta el significado de las métricas seleccionadas para la elaboración del prototipo en el anexo 4.

5.2.5. Logros obtenidos durante la etapa II del proyecto.

Durante esta segunda etapa del proyecto se logra cumplir el segundo objetivo del proyecto, al a través del análisis, seleccionar el muestreo como técnica de recolección para las métricas y la elección de VTune Amplifier como herramienta de monitoreo. Además, se utiliza la bibliografía consultada para detectar y seleccionar unidades de microarquitectura y relacionarlas con las métricas de rendimiento tomadas en cuenta para el proyecto.

5.3. Etapa III: Una discusión sobre parámetros y factores.

La etapa III del proyecto, es un complemento de la etapa II, al aportar la existencia de parámetros, su utilidad y la forma en que pueden o no modificar los resultados de las pruebas preliminares. La *tabla n.29* se ha colocado para una mejor comprensión entre la relación que guarda esta etapa con las técnicas, herramientas y elementos del marco teórico.

Tabla 29. Desglose de la tercera etapa, elaboración propia.

Objetivos	Entregables	Etapas	Técnicas	Herramientas	Temas del marco teórico
Diseñar una propuesta de software que permita monitorear la forma en que las unidades de los procesadores Intel core i7 están siendo ejercitadas desde el nivel de usuario	Expuestos en el apartado 1.4.1.2	III	Revisión bibliográfica.	Motores de búsqueda, bibliotecas virtuales y revistas especializadas.	*2.1,2.2,2.3,2.4,2.5,2.6,2.7,2.8*

5.3.1. Relación entre parámetros, factores y métricas.

Existe una serie de parámetros, es decir, elementos que pueden afectar los resultados de los experimentos. Éstos pueden ser inherentes al sistema como lo puede ser el *hardware* con el que se construyó la plataforma sobre la que se ejecutan los experimentos: el tipo y tamaño de los niveles de memoria caché y el número de unidades de procesamiento. Otros aspectos están ligados a la implementación del hardware, como lo son la implementación por medio de hilos y la distribución de recursos de acuerdo al sistema operativo.

Otros parámetros son de carácter externo al procesador, como lo son la temperatura del ambiente y el suministro de energía. Cuando se realizan experimentos, este proyecto incluido, se procura que las condiciones tanto externas como internas sean estables.

Las métricas de rendimiento se utilizan como una forma de detectar cambios o anormalidades en los parámetros. En algunos casos, los comportamientos que originan anormalidades o tendencias en las métricas evidencian el parámetro que provoca la anormalidad permitiendo su corregimiento. Otras veces, el diagnóstico se da como resultado de la triangulación de los resultados de dos o más métricas. En esos casos, se trabaja por medio de hipótesis que deben ser corroboradas a través de la selección de factores. Los factores en estos casos, son parámetros que se alteran sistemáticamente durante las pruebas para detectar la forma en que modifican el comportamiento del sistema.

Sin embargo, este tipo de validaciones no podrían realizarse si no se tiene conocimiento de la forma en que, no sólo el procesador ha sido construido sino los componentes que lo rodean, de esta preocupación se origina la importancia del conocimiento del sistema destino que sirve como ambiente de desarrollo y pruebas. Dicho en otras palabras, cuando se habla de un sistema de destino, *target system*, se refiere al sistema sobre el que se realizarán las pruebas, mediciones y o experimentos, por lo que resulta indispensable conocer las especificaciones de este y mantener configuraciones estables, con el fin de que los experimentos puedan ser replicados. (Barnstijn et al.,1997)

La descripción del sistema destino para este proyecto se encuentra en el anexo 5. En este punto, resulta indispensable mencionar que los experimentos realizados se hicieron sobre la base de un *Out of Box*, es decir, desde la configuración de fábrica del sistema utilizado.

5.3.2. Logros obtenidos en la etapa III del proyecto.

La etapa III del proyecto se constituye como un factor determinante para la concreción del objetivo 2 del proyecto, al aportar una perspectiva más profunda acerca de los cuidados en el establecimiento de parámetros y factores, que serán incorporados en las etapas de diseño y desarrollo del prototipo.

5.4. Etapa IV: Diseño del prototipo.

Las etapas I, II y III de este escrito estuvieron dedicadas a la definición del problema, la selección de unidades, métricas, las especificaciones del sistema destino y otras consideraciones relevantes en el proceso de recolección y análisis. A partir de ésta, la etapa IV, el enfoque se vuelca hacia el prototipo desde la perspectiva del desarrollo de software.

Tal y como se observa en la *tabla n.30*, la etapa IV abarca los alcances propuestos para el tercer objetivo del proyecto.

Tabla 30. Desglose de la cuarta etapa del proyecto, elaboración propia.

<table>
<tr><th>Objetivos</th><th>Entregables</th><th>Etapas</th><th>Técnicas</th><th>Herramientas</th><th>Temas del marco teórico</th></tr>
<tr><td rowspan="3">Diseñar una propuesta de software que permita monitorear la forma en que las unidades de los procesadores Intel core i7 están siendo ejercitadas desde el nivel de usuario.</td><td rowspan="3">Expuestos en el apartado 1.4.1.3</td><td rowspan="3">IV</td><td rowspan="2">Triangulación de datos</td><td>Bibliotecas de software.</td><td rowspan="3">2.6, 2.9, 2.10</td></tr>
<tr><td>Motores de búsqueda, bibliotecas virtuales y revistas especializadas.</td></tr>
<tr><td>Revisión bibliográfica.</td><td>Herramientas de monitoreo de software.</td></tr>
</table>

Los diagramas que se exponen seguidamente se muestran para clarificar el funcionamiento del prototipo, las interacciones con el usuario y con sus propios componentes.

5.4.1. Casos de uso.

En las páginas siguientes se muestra el diagrama y las descripciones de los casos de uso.

5.4.1.1. Diagrama de casos de uso.

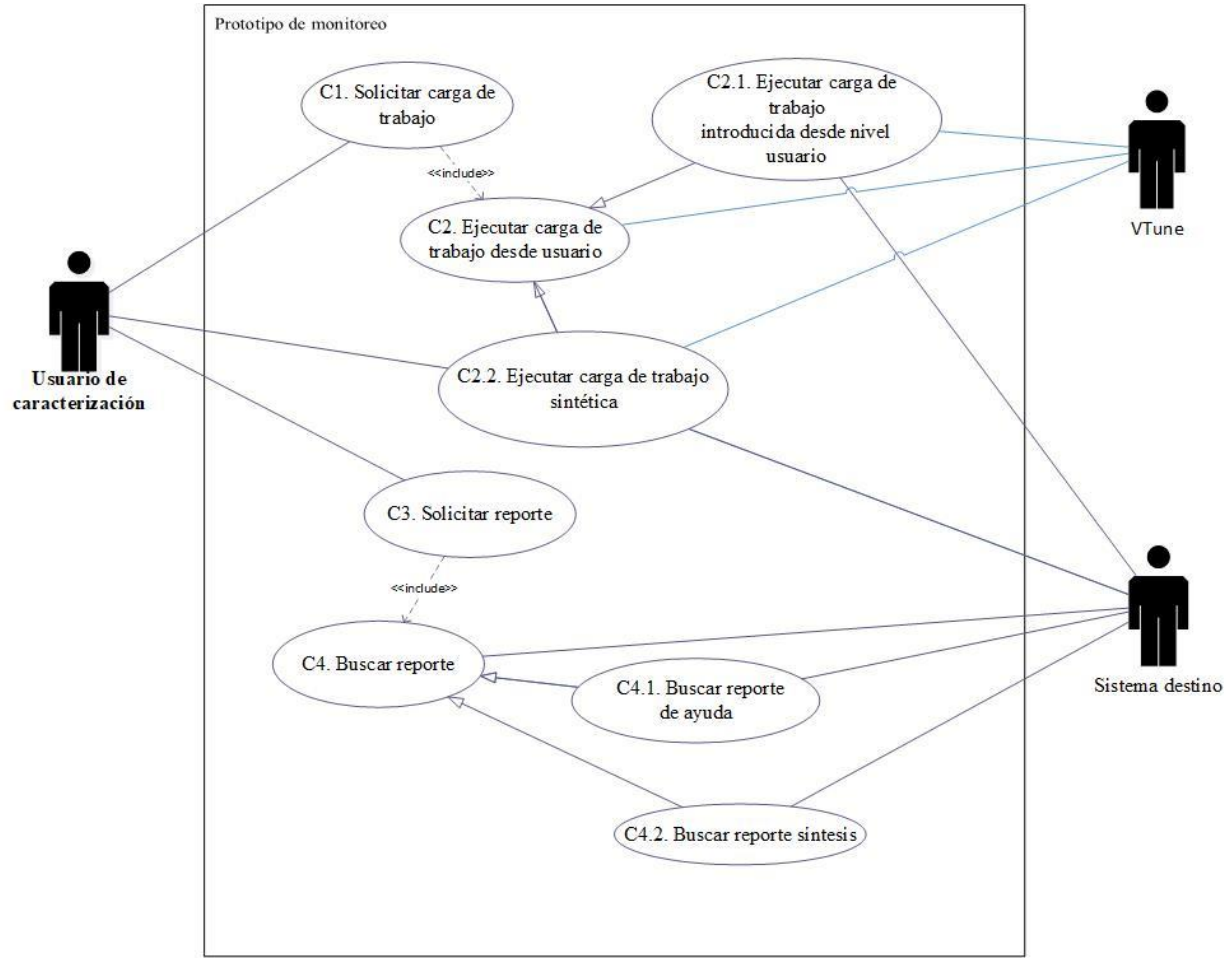


Ilustración 38. Diagrama de casos de uso, elaboración propia.

5.4.1.2. Descripción de casos de uso.

Tabla 31. C1: Solicitar carga de trabajo, elaboración propia.

ID del Caso de Uso:	**C1**
Nombre	**Solicitar carga de trabajo**
Actores	Usuario de caracterización, sistema destino
Objetivo:	Pedir al sistema que ejecute una carga de trabajo
Referencia	R8, R10
Pre-Condiciones	El ambiente de programación, el emulador de Python y las librerías deben estar instaladas. Los directorios para accesar a resultados, demos y VTune deben estar actualizados. La pantalla del prototipo se encuentra en abierta, en ella se observa un menú de cabecera con la descripción: “Demos” y debajo, una lista desplegable con el encabezado: “Seleccionar carga de trabajo”
Post-Condiciones	El prototipo ha iniciado la recolección de métricas.
Flujo Principal	**Descripción**
1.1. Este caso de uso inicia cuando el usuario de caracterización pulsa sobre el menú de demo o la lista para seleccionar el tipo de carga de trabajo que desea ejecutar.	El prototipo despliega las opciones de carga de trabajo que se pueden monitorear: a. De ser la lista, despliega: OpenOffice Writer, OpenOffice Draw, OpenOffice Calc y OpenOffice Scan. b. De ser el menú de Demos, el prototipo despliega los nombres de al menos un demo disponible para pruebas.
1.2. El usuario selecciona el programa que desea monitorear.	N/A
1.3. Fin del caso de uso	N/A

Tabla 32. C2.1: Ejecutar carga de trabajo desde usuario, elaboración propia.

ID del Caso de Uso:	**C2.1**

Nombre	**Ejecutar carga de trabajo desde usuario**
Actores	Sistema destino, usuario caracterizador, VTune.
Objetivo:	Realizar la recolección de las métricas a partir de la interacción directa del usuario caracterizador.
Referencia	R2, R3, R4, R5, R8, R10, R11
Pre-Condiciones	El usuario caracterizador ha solicitado la recolección de métricas desde la lista de la pantalla principal.
Post-Condiciones	VTune ha generado un reporte con los valores porcentuales de las métricas y lo ha guardado en el sistema destino.
Flujo Principal	**Descripción**
C2.1.1. Este caso de uso inicia cuando el sistema destino inicia una sesión de usuario de la aplicación seleccionada.	El sistema destino despliega una pestaña de la aplicación seleccionada desde la que el usuario caracterizador puede ejecutar las tareas que permite la aplicación: escribir, insertar imágenes, gráficos, realizar cálculos matemáticos, entre otros.
C2.1.2. VTune inicia la recolección de los datos sobre las métricas.	El prototipo muestra el mensaje: “Carga de trabajo iniciada”.
C2.1.3.El usuario caracterizador utiliza la sesión abierta de la aplicación.	N/A
C2.1.4. El usuario caracterizador cierra la sesión de la aplicación.	El usuario puede hacer el cierre de la aplicación ya sea pulsando la “X” en el menú de cabecera, la opción de salir en el menú de Archivo de la aplicación o la finalización de los procesos de la aplicación a través de consola con el comando: “Fillall + nombreDeApicación”
C2.1.5. VTune detiene la recolección de datos sobre las métricas.	El sistema muestra el mensaje: “Carga de trabajo finalizada”.
C2.1.6. Fin del caso de uso.	N/A

Tabla 33. C2.2: Ejecutar carga de trabajo sintética, elaboración propia.

ID del Caso de Uso:	**C2.2**

Nombre	**Ejecutar carga de trabajo sintética.**
Actores	Sistema destino, usuario caracterizador, VTune.
Objetivo:	Realizar la recolección de las métricas desde una carga de trabajo predefinida.
Referencia	R2, R3, R4, R5, R8, R10, R11
Pre-Condiciones	El usuario caracterizador ha solicitado la recolección de métricas para alguna de las opciones de carga de trabajo disponibles.
Post-Condiciones	El prototipo ha generado un reporte con los valores porcentuales de las métricas.
Flujo Principal	**Descripción**
C2.2.1. Este caso de uso inicia cuando el sistema destino inicia una sesión de usuario de la aplicación seleccionada.	N/A
C.2.2.2. VTune inicia recolección de datos para las métricas.	El prototipo muestra el mensaje: “Carga de trabajo iniciada iniciada”.
C2.2.3. El sistema realiza las operaciones predefinidas en el demo.	Al haber seleccionado una opción del menú de Demos, el demo seleccionado se ejecutará automáticamente sin intervención del usuario caracterizador.
C2.2.4. VTune detiene la recolección de los valores para las métircas.	Al terminar de ejecutar el demo, el sistema destino cierra automáticamente la sesión abierta para la ejecución de la carga predefinida y el prototipo muestra el mensaje: “Carga de trabajo finalizada”.
C2.2.5. Fin del caso de uso.	N/A

Tabla 34. C3: Solicitar reporte, elaboración propia.

ID del Caso de Uso:	**C3**
Nombre	**Solicitar reporte**
Actores	Sistema destino, usuario caracterizador.
Objetivo:	Dar inicio a la búsqueda del reporte seleccionado.
Referencia	R8, R10, R11

Pre-Condiciones	Hay en el sistema destino al menos un reporte guardado. El sistema muestra en el menú de cabecera las opciones de: Reportes y Ayuda.
Post-Condiciones	N/A
Flujo Principal	**Descripción**
C3.1. Este caso de uso inicia cuando el usuario caracterizador despliega el menú para la selección del reporte.	El usuario caracterizador puede desplegar cualquiera de los siguientes menús: Reportes y Ayuda.
C3.2. El usuario caracterizador selecciona el tipo de reporte que desea obtener.	N/A
C3.3. Fin del caso de uso.	N/A

Tabla 35. C4.1: Buscar reporte del menú de ayuda, elaboración propia.

ID del Caso de Uso:	**C4.1**
Nombre	**Buscar reporte del menú de ayuda**
Actores	Sistema destino.
Objetivo:	Desplegar reporte del menú de ayuda.
Referencia	R1, R8, R9, R10,
Pre-Condiciones	El prototipo muestra en el menú de cabecera la opción del menú de Ayuda. El usuario caracterizador ha seleccionado una opción del menú de Ayuda.
Post-Condiciones	N/A
Flujo Principal	**Descripción**
C4.1.1. Este caso de uso inicia cuando el sistema destino recorre los archivos hasta encontrar el reporte seleccionado.	Este informe puede ser, una descripción del sistema destino o una guía rápida de uso.
C4.1.2. El prototipo despliega el tipo de reporte seleccionado.	El prototipo desplegará un archivo en formato pdf con el reporte que se le solicitó.
C4.1.3. Fin del caso de uso.	N/A
Flujo Alternativo	**Descripción**
En el punto 2.1., el prototipo no encuentra el reporte solicitado.	El prototipo muestra el mensaje: “No se encontró la información solicitada”

Tabla 36. C4.2: Buscar reporte de carga de trabajo definida por el usuario, elaboración propia.

ID del Caso de Uso:	**C4.2**
Nombre	**Buscar reporte de carga de trabajo definida por el usuario.**
Actores	Sistema destino.
Objetivo:	Desplegar tipo de reporte solicitado sobre experimento más reciente.
Referencia	R6, R7, R8, R9, R10
Pre-Condiciones	Hay en el sistema destino al menos un reporte guardado. El prototipo muestra en el menú de cabecera la opción de Ayuda.
Post-Condiciones	N/A
Flujo Principal	**Descripción**
C4.2.1. Este caso de uso inicia cuando el sistema destino busca entre los registros el reporte generado más recientemente.	Los reportes se han guardado con un formato estándar de forma tal que puedan ser recorridos y encontrados.
C4.2.2. El prototipo despliega el reporte más reciente del tipo seleccionado.	El prototipo desplegará un gráfico con la distribución de los porcentajes correspondientes al experimento más reciente. La pantalla desplegada ofrece la opción de guardar la síntesis en formato de imagen.
C4.2.3. Fin del caso de uso.	N/A
Flujo Alternativo	**Descripción**
En el punto 2.1., el prototipo no encuentra ningún reporte o el nombre de los reportes existentes no coincide con las convenciones establecidas para los reportes.	El prototipo muestra el mensaje: Todavía no se han generado reportes”

5.4.2. Diagramas de secuencia.

Ilustración 39. D1, ID de caso de uso: C1, elaboración propia.

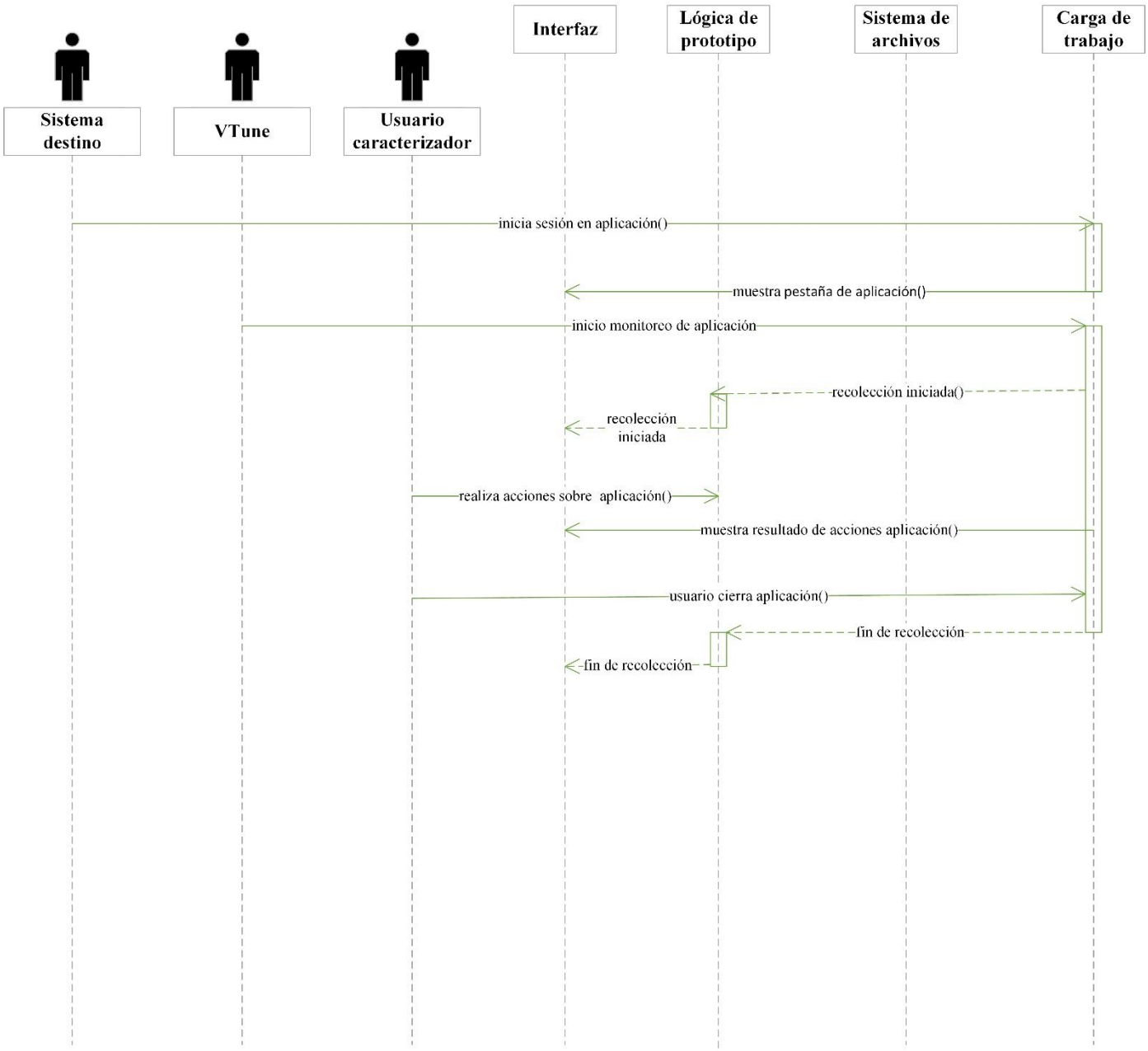


Ilustración 40. D2, ID caso de uso: C2.1, elaboración propia.

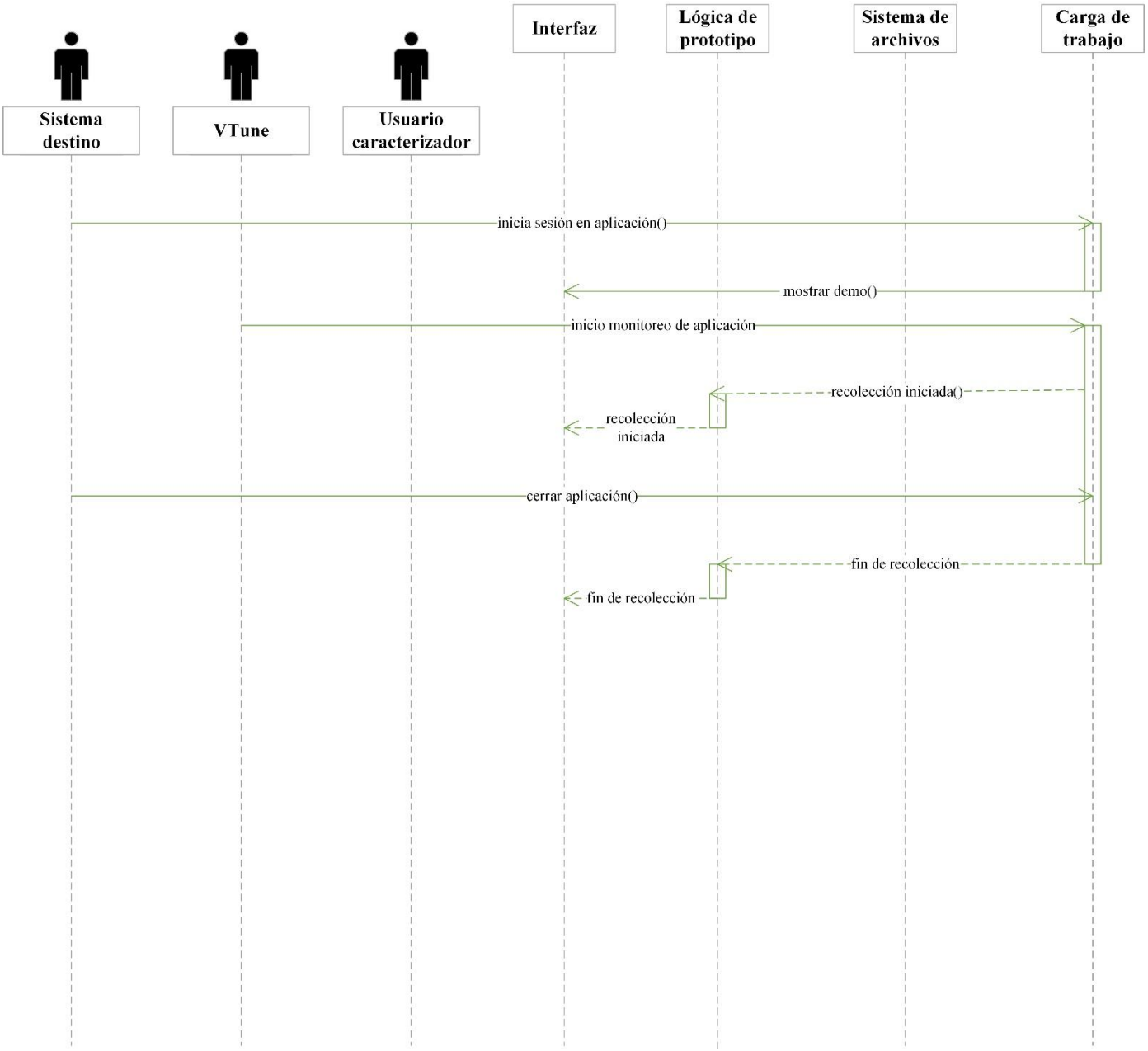


Ilustración 41. D3, ID caso de uso: C2.2, elaboración propia.

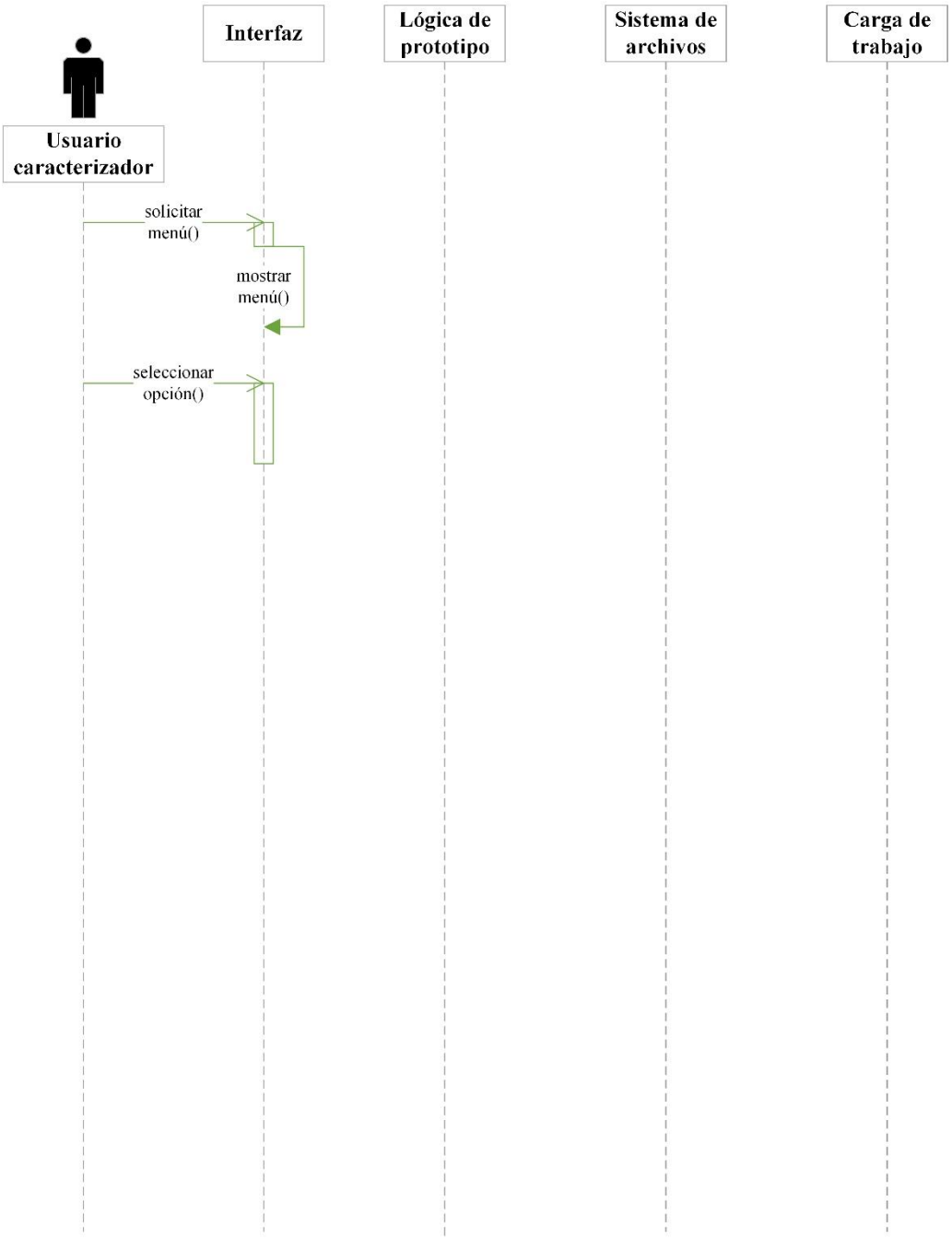


Ilustración 42. D4, ID caso de uso: C3, elaboración propia.

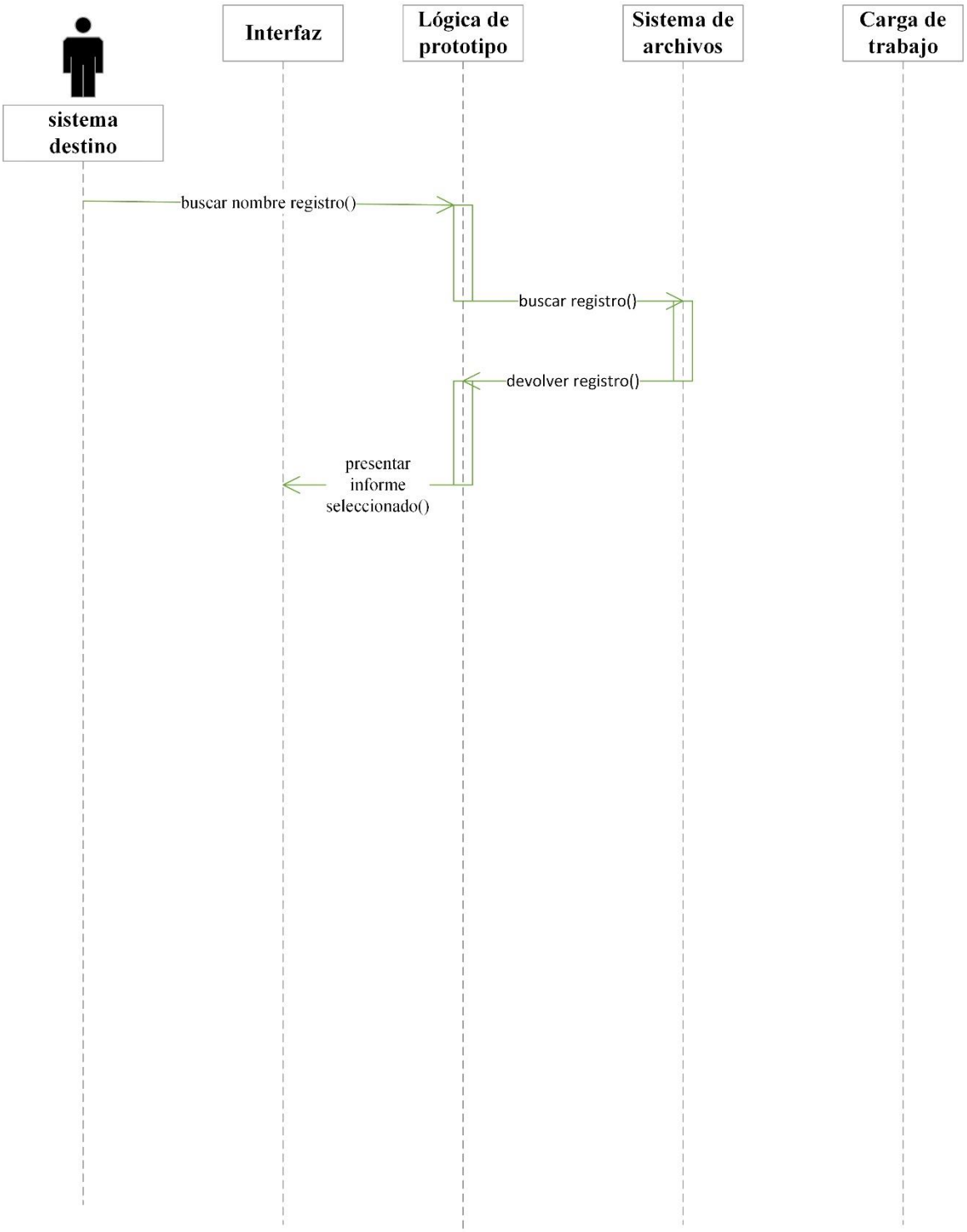


Ilustración 43. D5, ID caso de uso: C4.2, elaboración propia.

5.4.3. Diagrama de clases.

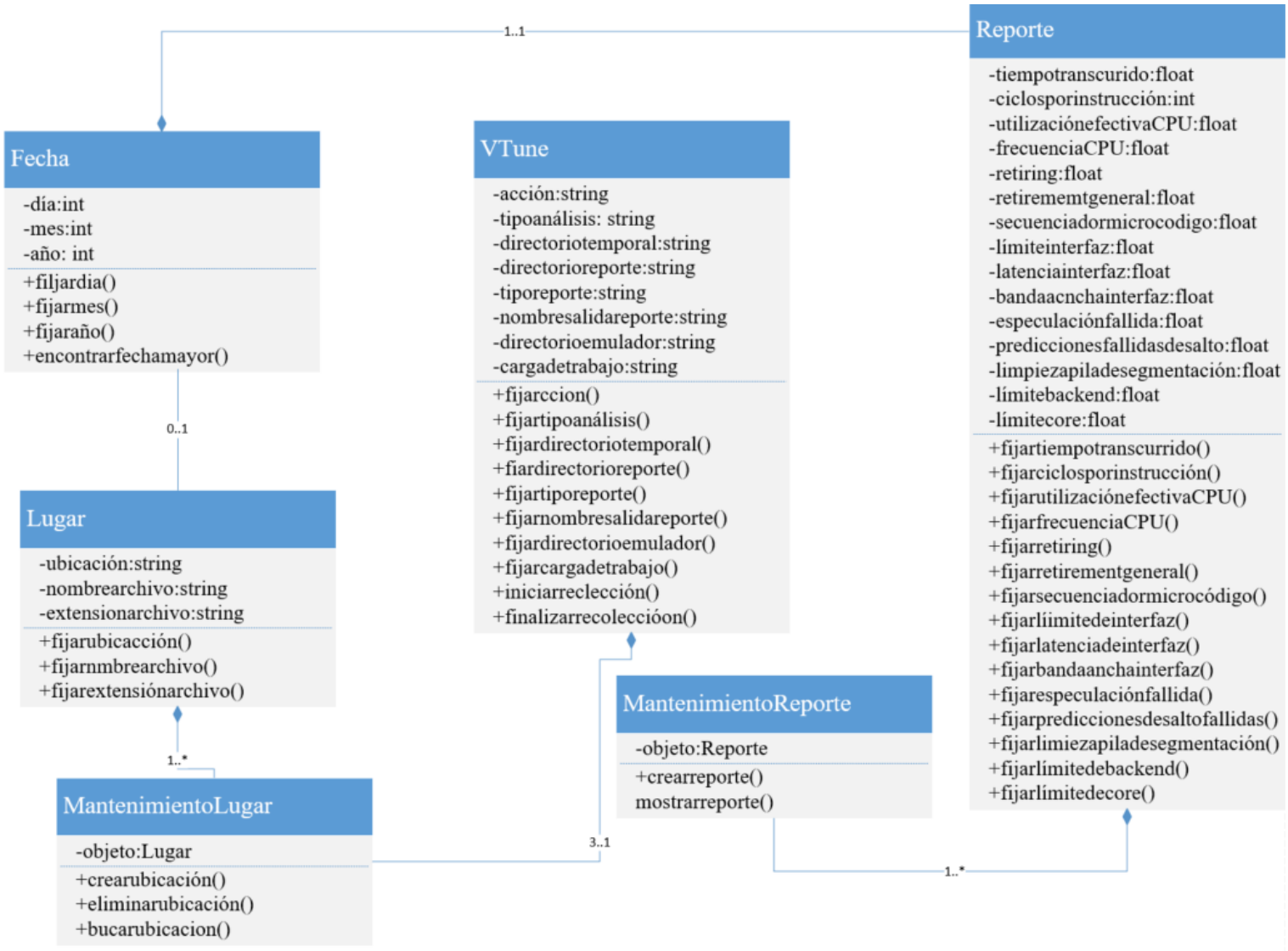


Ilustración 44. Diagrama de clases, elaboración propia.

5.4.4. Diagrama de colaboración.

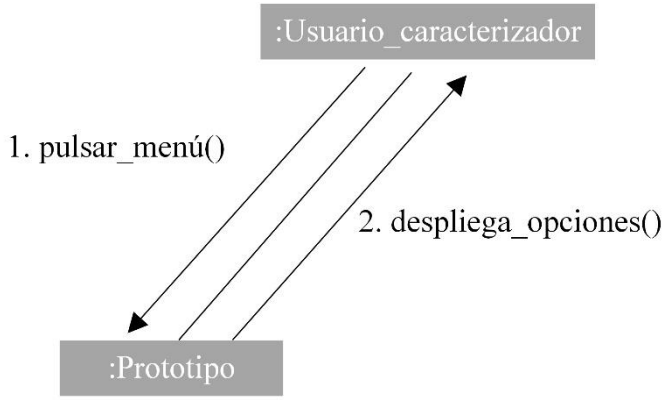


Ilustración 45. Diagrama de colaboración 1, elaboración propia.

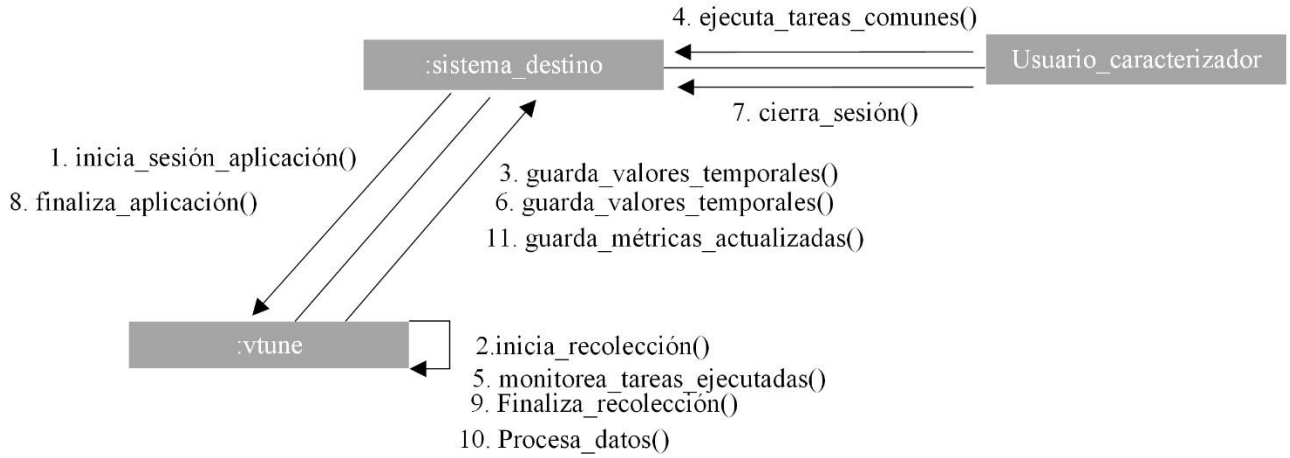


Ilustración 46. Diagrama de colaboración 2, elaboración propia.

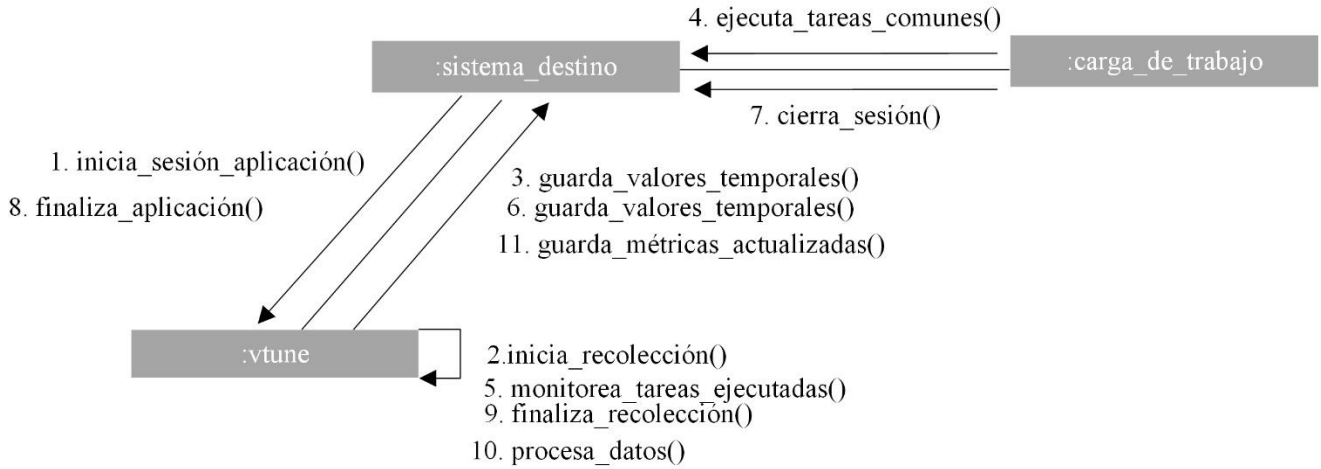


Ilustración 48. Diagrama de colaboración 3, elaboración propia.

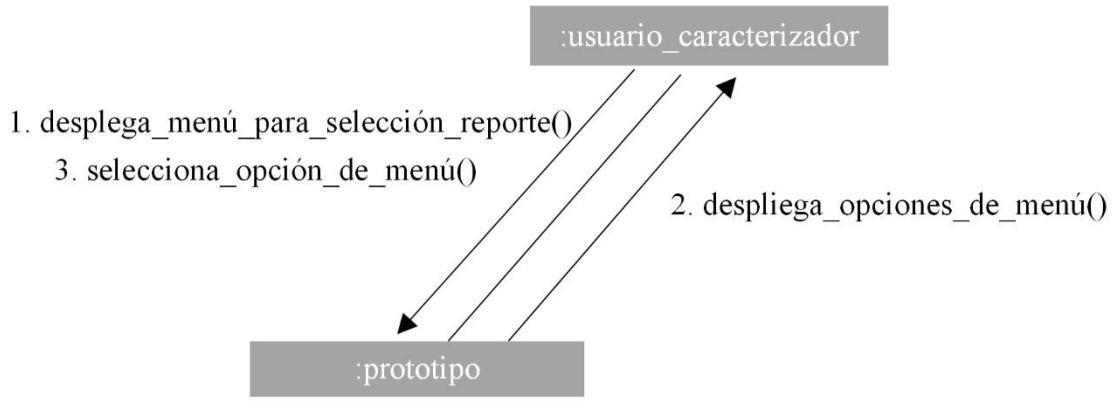


Ilustración 47. Diagrama de colaboración 4, elaboración propia.

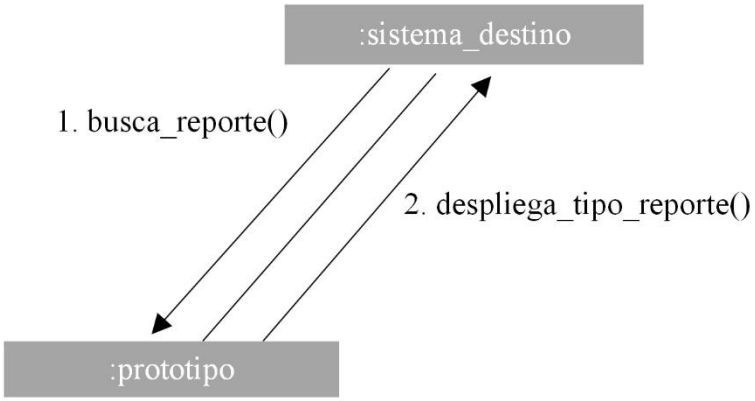


Ilustración 49. Diagrama de colaboración 5, elaboración propia.

5.4.5. Diagramas de actividad.

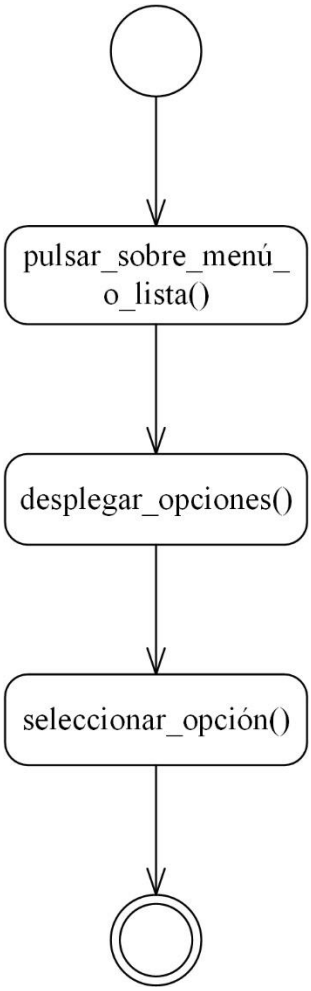


Ilustración 50. Diagrama de actividad 1, ID caso de uso: C1, elaboración propia.

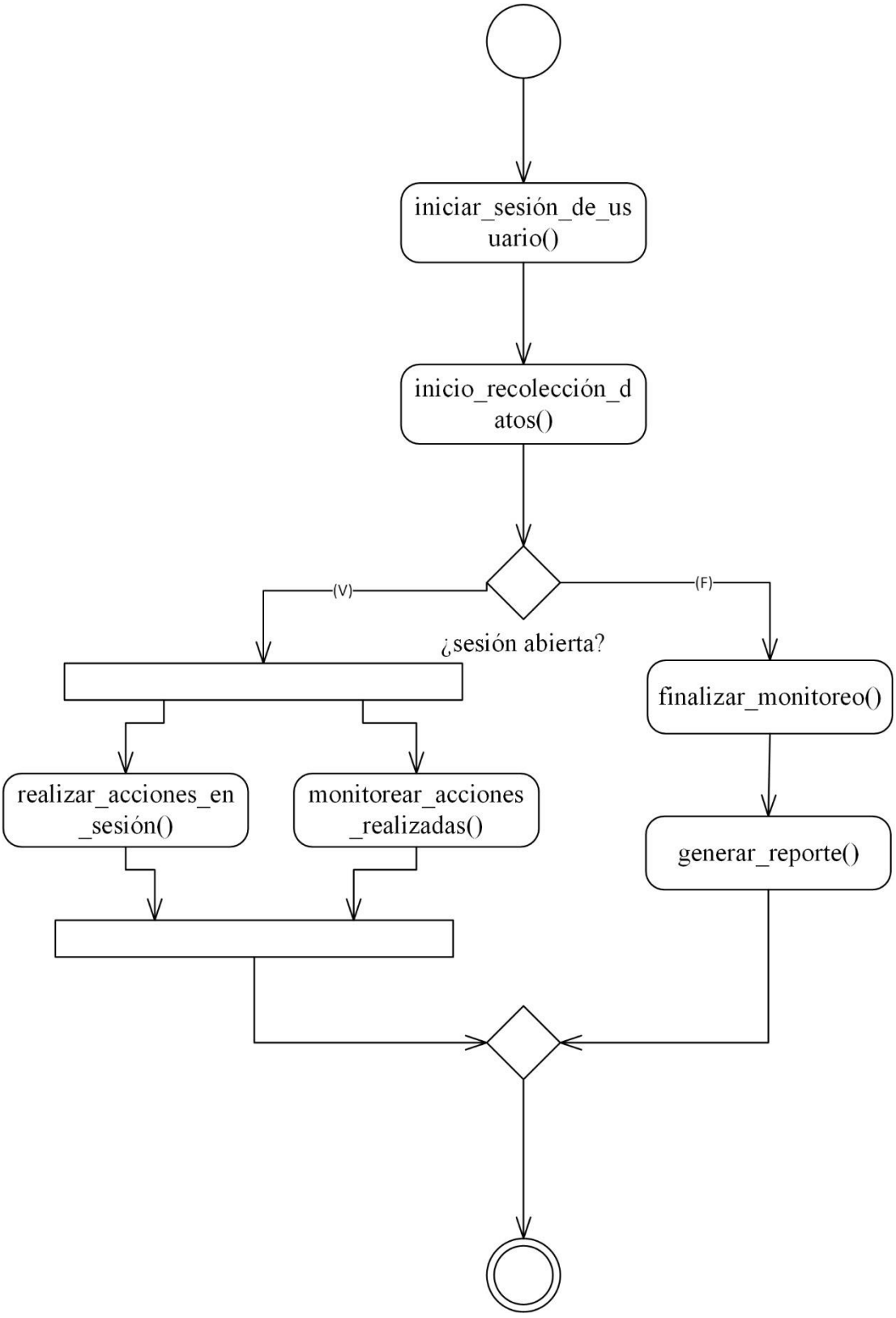


Ilustración 51. Diagrama de actividad 2, ID caso de uso: C2, elaboración propia.

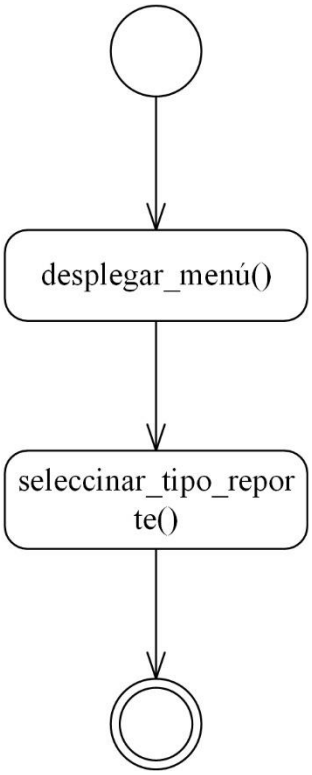


Ilustración 52. Diagrama de actividad 3, ID caso de uso: C3, elaboración propia.

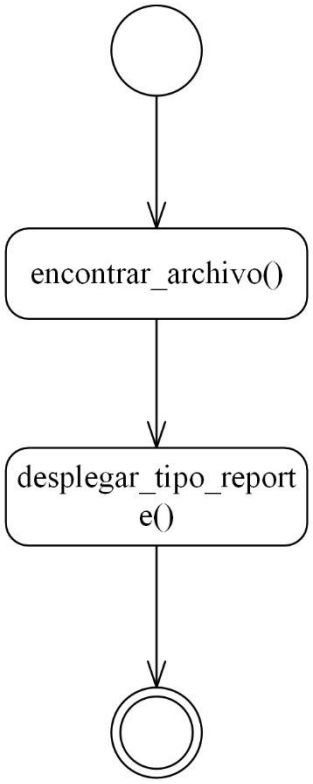


Ilustración 53. Diagrama de actividad 4, ID caso de uso: C4, elaboración propia

5.4.6. Plan de pruebas.

En este apartado del documento se especifica la relación entre los requerimientos planteados en la primera entrega del proyecto y la funcionalidad de este, planteando indicadores puntuales para verificar el cumplimiento de los requerimientos del sistema a desarrollarse.

5.4.6.1. Definición de los casos de prueba.

En este apartado se describen los casos de prueba referenciados con anterioridad, con sus respectivos mecanismos de corroboración.

Tabla 37. P1: Presentar especificaciones del sistema, elaboración propia.

Código de prueba:	P1
Nombre de prueba:	Presentar especificaciones del sistema
Referencia:	R1, R8, R9, R10
Prerrequisitos para la prueba:	N/A
Pasos:	P1.1. El usuario pulsa sobre el menú de Ayuda. P1.2. El prototipo muestra la opción: Información del Sistema. P1.3. El usuario pulsa sobre la opción de Información del Sistema.
Resultado esperado:	El prototipo despliega un pdf con la información del sistema destino.

Tabla 38. P2: Procesar información VTune 1, elaboración propia.

Código de prueba:	P2
Nombre de prueba:	Procesar información VTune 1
Referencia:	R2, R3, R4, R5, R8, R10, R11
Prerrequisitos para la prueba:	Una versión de VTune Amplifier debe de haber sido instalada en el sistema destino.
Pasos:	P2.1. En la pestaña principal, el usuario pulsa sobre la opción de Demos. P2.2. El prototipo despliega las opciones de Demo1 y Demo2. P2.3. El usuario pulsa sobre la opción de Demo 1. P2.4. El prototipo muestra el mensaje: “carga de trabajo iniciada”. P2.5. Un video se ejecuta de forma visible en pantalla.

	P2.6. Cuando el video termina de ejecutarse, se cierra. P2.7. El prototipo muestra el mensaje: "Carga de trabajo terminada"
Resultado esperado:	En la carpeta "Resultados" del sistema destino, hay un nuevo archivo con la fecha completa en el siguiente formato: "dia-mes-año—hora:minutos:segundos" en formato *csv*

Tabla 39. P3: Procesar información 2, elaboración propia.

Código de prueba:	P3
Nombre de prueba:	Procesar información VTune 2
Referencia:	R2, R3, R4, R5, R8, R10, R11
Prerrequisitos para la prueba:	Una versión de VTune Amplifier debe de haber sido instalada en el sistema destino.
Pasos:	P3.1. En la pestaña principal, el usuario pulsa sobre la opción de Demos. P3.2. El prototipo despliega las opciones de Demo1 y Demo2. P3.3. El usuario pulsa sobre la opción de Demo 2. P3.4. El prototipo muestra el mensaje: "carga de trabajo iniciada". P3.5. Un video se ejecuta de forma visible en pantalla. P2.6. Cuando el video termina de ejecutarse, se cierra. P3.7. El prototipo muestra el mensaje: "Carga de trabajo terminada"
Resultado esperado:	En la carpeta "Resultados" del sistema destino, hay un nuevo archivo con la fecha completa en el siguiente formato: "día-mes-año—hora:minutos:segundos" en formato *csv*

Tabla 40. P4: Procesar información VTune 3, elaboración propia.

Código de prueba:	P4
Nombre de prueba:	Procesar información VTune 3
Referencia:	R2, R3, R4, R5, R8, R10, R11
Prerrequisitos para la prueba:	Una versión de VTune Amplifier debe de haber sido instalada en el sistema destino.

Pasos:	P4.1. En la pestaña principal, el usuario pulsa sobre la lista. P4.2. El prototipo despliega las opciones de: OpenOffice Write, OpenOffice Calc, OpenOffice Draw y OpenOffice Impress. P4.3. El usuario pulsa sobre la opción de OpenOffice Write. P4.4. Se despliega en pantalla una pestaña de OpenOffice Write. P4.5. El prototipo muestra el mensaje: "carga de trabajo iniciada". P4.5. El usuario inserta una imagen en el documento nuevo que se ha generado automáticamente en el inicio de sesión de OpenOffice Write. P4.6. El usuario guarda el documento generado. P4.7. El usuario cierra la sesión de OpenOffice Write. P4.8. El prototipo muestra el mensaje: "Carga de trabajo terminada"
Resultado esperado:	En la carpeta "Resultados" del sistema destino, hay un nuevo archivo con la fecha completa en el siguiente formato: "día-mes-año—hora:minutos:segundos" en formato *csv*

Tabla 41. P5: Procesar información VTune 4, elaboración propia.

Código de prueba:	P5
Nombre de prueba:	Procesar información VTune 4
Referencia:	R2, R3, R4, R5, R8, R10, R11
Prerrequisitos para la prueba:	Una versión de VTune Amplifier debe de haber sido instalada en el sistema destino.
Pasos:	P5.1. En la pestaña principal, el usuario pulsa sobre la lista. P5.2. El prototipo despliega las opciones de: OpenOffice Write, OpenOffice Calc, OpenOffice Draw y OpenOffice Impress. P5.3. El usuario pulsa sobre la opción de OpenOffice Calc. P5.4. Se despliega en pantalla una pestaña de OpenOffice Calc. P5.5. El prototipo muestra el mensaje: "carga de trabajo iniciada".

	P5.6. El usuario inserta la siguiente información en el documento nuevo que se ha generado automáticamente en el inicio de sesión de OpenOffice Calc: 40 92 34 87 P5.7. El usuario suma las celdas digitadas seleccionando una celda nueva y posicionando el número de celda y el signo + para obtener como resultado: 253. 40 92 34 87 **253** P5.8. El usuario guarda el documento generado. 3.9. El usuario cierra la sesión de OpenOffice Calc. P5.10. El prototipo muestra el mensaje: "Carga de trabajo terminada"
Resultado esperado:	En la carpeta "Resultados" del sistema destino, hay un nuevo archivo con la fecha completa en el siguiente formato: "día-mes-año—hora:minutos:segundos" en formato *csv*

Tabla 42. P6: Procesar información VTune 5, elaboración propia.

Código de prueba:	P6
Nombre de prueba:	Procesar información VTune 5
Referencia:	R2, R3, R4, R5, R8, R10, R11
Prerrequisitos para la prueba:	Una versión de VTune Amplifier debe de haber sido instalada en el sistema destino.
Pasos:	P6.1. En la pestaña principal, el usuario pulsa sobre la lista. P6.2. El prototipo despliega las opciones de: OpenOffice Write, OpenOffice Calc, OpenOffice Draw y OpenOffice Impress.

	P6.3. El usuario pulsa sobre la opción de OpenOffice Impress. P6.4. Se despliega en pantalla una pestaña de OpenOffice Impress. P6.5. En la pestaña de OpenOffice Impress, se despliega una estaña con plantillas para selección. P6.6. El prototipo muestra el mensaje: "carga de trabajo iniciada". P6.7. El usuario pulsa sobre una de las plantillas. P6.8. El usuario selecciona una plantilla en el documento nuevo que se ha generado automáticamente en el inicio de sesión de OpenOffice Impress. P6.9. El usuario digita sobre la plantilla: "Documento de prueba". P6.10. El usuario guarda el documento generado. P6.11. El usuario cierra la sesión de OpenOffice Impress. P6.12. El prototipo muestra el mensaje: "Carga de trabajo terminada"
Resultado esperado:	En la carpeta "Resultados" del sistema destino, hay un nuevo archivo con la fecha completa en el siguiente formato: "día-mes-año—hora:minutos:segundos" en formato *csv*

Tabla 43. P7: Procesar informavión VTune 6, elaboración propia.

Código de prueba:	P7
Nombre de prueba:	Procesar información VTune 6
Referencia:	R2, R3, R4, R5, R8, R10, R11
Prerrequisitos para la prueba:	Una versión de VTune Amplifier debe de haber sido instalada en el sistema destino.
Pasos:	P7.1. En la pestaña principal, el usuario pulsa sobre la lista. P7.2. El prototipo despliega las opciones de: OpenOffice Write, OpenOffice Calc, OpenOffice Draw y OpenOffice Impress. P7.3. El usuario pulsa sobre la opción de OpenOffice Draw. P7.4. Se despliega en pantalla una pestaña de OpenOffice Draw.

	P7.5. El prototipo muestra el mensaje: "carga de trabajo iniciada". P7.6. El usuario inserta un cuadrado en la sesión que se ha iniciado automáticamente durante el inicio de sesión de OpenOffice Draw. P7.7. El usuario inserta una flecha en el documento generado de Open Office Draw. P7.8. El usuario guarda el documento generado. P7.9. El usuario cierra la sesión de OpenOffice Draw. P7.10. El prototipo muestra el mensaje: "Carga de trabajo terminada"
Resultado esperado:	En la carpeta "Resultados" del sistema destino, hay un nuevo archivo con la fecha completa en el siguiente formato: "día-mes-año—hora:minutos:segundos" en formato *csv*

Tabla 44. P8: Mostrar resumen, elaboración propia.

Código de prueba:	P8
Nombre de prueba:	Mostrar resumen.
Referencia:	R6, R8, R10
Prerrequisitos para la prueba:	N/A
Pasos:	P8.1. El usuario pulsa sobre el menú de Reportes. P8.2. El usuario pulsa sobre la opción de Resumen.
Resultado esperado:	El prototipo despliega una pestaña con un resumen que contiene los valores porcentuales de: Retiring, Límite de interfaz, Especulación fallida y Límite de backend de la última prueba realizada.

Tabla 45. P9: Desplegar retiring, elaboración propia.

Código de prueba:	P9
Nombre de prueba:	Desplegar Retiring
Referencia:	R6, R8, R10
Prerrequisitos para la prueba:	N/A

Pasos:	P9.1. El usuario pulsa sobre el menú de Reportes. P9.2. El usuario pulsa sobre la opción de Retiring.
Resultado esperado:	El prototipo despliega una pestaña con los valores porcentuales de Retiring: Retirement General y Secuenciador de Microcódigo.

Tabla 46. P10: Desplegar Límite de interfaz.

Código de prueba:	P10
Nombre de prueba:	Desplegar Límite de interfaz
Referencia:	R6, R8, R10
Prerrequisitos para la prueba:	N/A
Pasos:	P10.1. El usuario pulsa sobre el menú de Reportes. P10.2. El usuario pulsa sobre la opción de Límite de interfaz.
Resultado esperado:	El prototipo despliega una pestaña con los valores porcentuales de Límite de interfaz: Latencia de interfaz y Ancho de banda de interfaz.

Tabla 47. P11: Mostrar especulación fallida, elaboración propia.

Código de prueba:	P11
Nombre de prueba:	Mostrar Especulación fallida.
Referencia:	R6, R8, R10
Prerrequisitos para la prueba:	N/A
Pasos:	P11.1. El usuario pulsa sobre el menú de Reportes. P11.2. El usuario pulsa sobre la opción de Especulación fallida.
Resultado esperado:	El prototipo despliega una pestaña con los valores porcentuales de Especulación fallida: Predicciones de salto fallidas y Limpieza de segmentación.

Tabla 48. P12: Desplegar límite de backend, elaboración propia.

Código de prueba:	P12
Nombre de prueba:	Desplegar Límite de backend.
Referencia:	R6, R8, R10
Prerrequisitos para la prueba:	N/A
Pasos:	P12.1. El usuario pulsa sobre el menú de Reportes. P12.2. El usuario pulsa sobre la opción de Límite de backend.
Resultado esperado:	El prototipo despliega una pestaña con los valores porcentuales de Límite de backend: Límite de memoria y Límite de core.

Tabla 49. P13: Guardar síntesis, elaboración propia.

Código de prueba:	P13
Nombre de prueba:	Guardar síntesis.
Referencia:	R7, R8, R9, R10
Prerrequisitos para la prueba:	Se ha generado al menos un reporte en la carpeta de resultados. Una pestaña con el tipo de reporte elegido se encuentra desplegada.
Pasos:	P13.1. El usuario pulsa sobre la opción de guardar. P13.2. El prototipo despliega la una vista del sistema de archivos del sistema destino. P13.3. El usuario selecciona la ubicación donde desea guardar la síntesis. P13.4. El usuario pulsa sobre la opción guardar.
Resultado esperado:	Un archivo con formato de imagen con la síntesis seleccionada se encuentra guardado en la ubicación del sistema destino deseada.

5.4.6.2. Trazabilidad de los casos de prueba.

A continuación, se expone una matriz en la que se hace explícita la relación entre los requerimientos y los casos de prueba de los requerimientos.

Nomenclatura: R= Requerimiento, P=Prueba, X=Puntos de encuentro entre los Requerimientos y las pruebas que los abarcan.

Tabla 50. Trazabilidad de casos de prueba, elaboración propia.

Cod. Prueba	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
P1								X	X	X	
P2		X	X	X	X			X		X	X
P3		X	X	X	X			X		X	X
P4		X	X	X	X			X		X	X
P5		X	X	X	X			X		X	X
P6		X	X	X	X			X		X	X
P7		X	X	X	X			X	X	X	X
P8						X		X		X	
P9						X		X		X	
P10						X		X		X	
P11						X		X		X	
P12						X		X		X	
P13								X	X	X	

5.4.7. Logros obtenidos durante la etapa IV del proyecto.

Durante la etapa IV de la propuesta se logra el cometido de diseñar un prototipo de software capaz de monitorear la forma en que las unidades de los procesadores Intel core i7 están siendo ejercitadas desde el nivel de usuario. Esto a través de diagramas y descripción de los casos de uso, diagramas de secuencia, de clases, de colaboración y de actividad. Los diagramas cumplen a su vez con el fin de la etapa IV del proyecto al permitir la visualización de la lógica interna del prototipo, la relación entre sus clases, componentes y con el usuario final. Además, aporta un primer grado de validación por medio del plan de pruebas llevado a cabo.

5.5. Etapa V: Análisis e interpretación.

Tal y como se denota en la *tabla n.51*, esta etapa se propone cumplir parcialmente con el cuarto objetivo del proyecto, al aportar el código, es decir, la lógica interna que sostendrá las capas de interfaz que se construirán durante la sexta etapa.

Tabla 51. Desglose de la quinta etapa, elaboración propia.

<table>
<tr><th>Objetivos</th><th>Entregables</th><th>Etapas</th><th>Técnicas</th><th>Herramientas</th><th>Temas del marco teórico</th></tr>
<tr><td rowspan="4">Desarrollar un prototipo de software para mostrar la forma en que las unidades funcionales de un procesador core i7 son utilizadas durante la ejecución de la aplicación de un tercero.</td><td rowspan="4">Expuestos en el apartado 1.4.1.4</td><td rowspan="4">V</td><td>Estadística inferencial.</td><td>Paquete de LibreOffice.</td><td rowspan="4">2.6,2.9,2.10</td></tr>
<tr><td rowspan="2">Estadística inferencial.</td><td>Bibliotecas de software.</td></tr>
<tr><td rowspan="2">Motores de búsqueda, bibliotecas virtuales y revistas especializadas.</td></tr>
<tr><td>Triangulación</td></tr>
</table>

Para lograrlo, se comprende que, además de las pruebas que se describen en el Plan de pruebas del apartado 5.4.6., dirigidas a confirmar la relación entre el usuario, el sistema destino y el prototipo, la naturaleza del desarrollo y su carácter experimental, hicieron necesaria la construcción de otro tipo de pruebas, elaboradas para constatar que los valores obtenidos de la herramienta y el prototipo fueran fiables.

En las líneas siguientes se detalla el tipo de carga de trabajo utilizada para las pruebas y los valores porcentuales obtenidos de éstas. Para las pruebas se siguieron las recomendaciones expuestas en el apartado 2.7, acerca de consideraciones sobre el tipo de análisis que se realizó, es decir, un *Top-down Analysis* en el que las métricas se clasificaron en: Límite de Interfaz, Límite de Back-end, Retiring y Especulación fallida, que constituyen el nivel más general de un *Top-down Analysis*. En el caso de VTune, los resultados de la exploración de la microarquitectura se muestran en columnas jerárquicas para reforzar la naturaleza descendente del análisis, mostrando el porcentaje de ranuras de la estructura de segmentación en cada categoría para el tiempo de ejecución de la aplicación.

5.5.1. Descripción de pruebas preliminares realizadas.

Se realizaron diez pruebas sobre una carga de trabajo sintética. Las especificaciones sobre la carga de trabajo se observan en detalle en el Anexo 8. Además, los valores obtenidos de las diez pruebas preliminares realizadas se muestran en el Anexo 6 de este proyecto.

5.5.2. Análisis de las pruebas preliminares realizadas.

En las líneas que siguen se expone el análisis obtenido de las pruebas realizadas. Los valores demarcados con letra negrita corresponden a las cuatro áreas más generales, cuya suma porcentual debe ser 100% en cada uno de los casos. La última fila de la tabla n.52 sobre Descripción de pruebas preliminares, se agregó para comprobar dicha suma.

El primer aspecto que sale a colación durante la observación de los datos reportados es la preponderancia de un alto nivel de Retiring, que rondó entre un 44% y un 45.6%. Los valores en Retiring indican el tiempo en el que las etapas de la segmentación estuvieron ocupadas en trabajo útil. En este sentido, un Retiring del 100% indica el máximo de instrucciones que pueden ser retirados por una estructura de segmentación. Por este motivo, entre más alto sea este valor, hay un mejor aprovechamiento de las ventajas ofrecidas por la estructura de segmentación del procesador. El porcentaje de Retiring obtenido de las pruebas, evidencia que el software ejecutado durante la prueba es capaz de aprovechar de forma eficiente el recurso de segmentación que el procesador ofrece.

Ahora bien, se puede ahondar más en esta primera observación. Dentro de las categorías de Retiring de los datos obtenidos, el Retirement General fue la categoría más sobresaliente, con un rango de 37% y 38%, del ~45% del Retiring. Este valor indica las instrucciones que fueron retiradas y que a su vez no provienen del secuenciador de microcódigo (ISU). Tal y como se menciona en el apartado *5.2.4.*, sobre la detección y selección de unidades de microarquitectura del procesador, el secuenciador de microcódigo es indispensable para determinar el orden en el que serán ejecutadas las instrucciones de un programa, esta operación, sin embargo, es costosa, ya que implica la carga de dos registros A y B, que pasarán por la ALU, cuyo resultado se guardará en un registro C. por ello, la

disminución de las acciones realizadas por el ISU, disminuyen el tiempo de ejecución del software y con ello, la rapidez de respuesta percibida por el usuario.

A pesar de que el valor más alto es el de Retirement General, todavía existe un significativo ~7% de utilización de ISU. Visto desde una perspectiva diferente, el ISU es una implementación de hardware que funciona como un traductor entre el hardware del CPU y los sets de instrucciones visibles para el usuario. De esta forma, cuando un usuario accede desde la interfaz de usuario a un sistema, por ejemplo, pulsando el botón de aceptar en un cuadro de diálogo, esta instrucción, en principio simple, se descompone en muchas microinstrucciones, implicando a su vez la inversión de una cantidad considerable de ciclos de reloj. Usualmente, los porcentajes elevados de ISU están relacionados con una utilización incorrecta de las banderas utilizadas desde el compilador. Tal y como se menciona en el apartado *2.4.1.* de este proyecto, sobre registros visibles para el usuario, las banderas son bits de estado, valores binarios con un significado predefinido. Lenguajes de programación como C, C++ y C#, permiten la utilización de banderas que se ejecutarán para validar datos de entrada y operaciones. (López T., 2019) En el caso estudiado, la utilización incorrecta de estas banderas repercute necesariamente en un uso mayor del ISU y con ello una ejecución más pobre de la aplicación.

El siguiente valor porcentual que se analizó fue el límite de interfaz, con un promedio de 32.6%. Existen ocasiones en que la interfaz, la parte más externa del procesador, encargada de la carga de operaciones no logra satisfacer las necesidades del backend. Durante este proceso, un predictor de salto predice la siguiente dirección que será cargada de las líneas de caché. En este sentido, el límite de interfaz, señala ranuras de la estructura de segmentación que no se utilizaron debido a bloqueos en la memoria. En el caso analizado, estos bloqueos se pueden rastrear a códigos sumamente largos, sin optimización, que requieren constantes accesos a memoria.

Al descomponer las categorías del límite de interfaz, se hizo visible que la latencia de interfaz obtuvo los valores más significativos dentro de las categorías de límite de interfaz, con un 20.11% en promedio, indicando que la CPU estuvo bloqueada la mayor parte del tiempo debido a misses en la caché de instrucciones, misses en los ITLB o bloqueos en los ITLB. Estos bloqueos denotan predicciones de salto fallidas. Tal y como se explicita en el

apartado *2.1.1*, sobre principios de arquitectura de computadoras, la memoria caché juega un papel vital en el rendimiento de un procesador. Visto de forma detallada, el proceso para la inserción de un dato en la memoria caché es el que se muestra en la ilustración *n.53*.

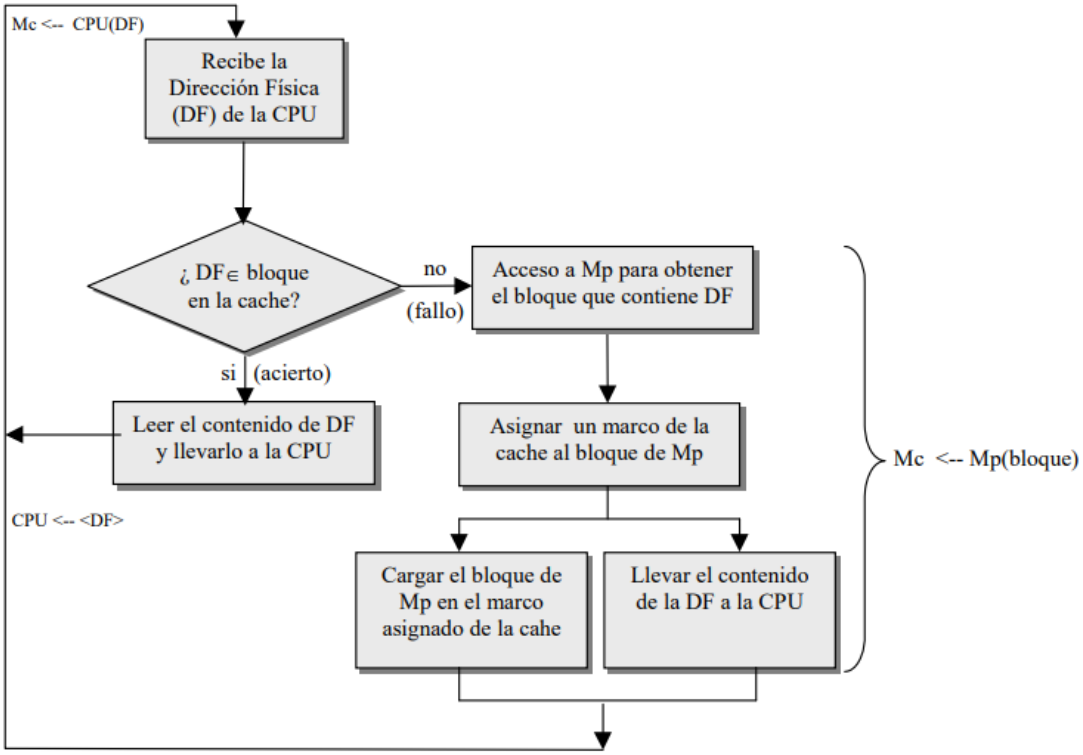


Ilustración 54. Funcionamiento de memoria caché. En Estructura de Computadores, cap.6 (p.3), por Universidad Computense de Madrid, 20919, UCM: Curso 11-12. Derechos de autor [2019] por UCM. Reimpresión autorizada.

Asimismo, los ITLBs expuestos en detalle en el anexo 4, intervienen en las relaciones entre la memoria caché y la memoria principal. Para el caso en estudio, los altos porcentajes de latencia de interfaz evidencian la necesidad de recurrir a viajes a los últimos niveles de memoria caché y la recurrencia en la utilización de los ITLBs. Estos viajes implican una penalidad de tiempo debido, entre otras cosas, al proceso expuesto en la *imagen n.53*, sobre el funcionamiento de la memoria caché. La presencia altos porcentajes de esta latencia puede ser rastreada a las unidades de tiempo invertidas en la interacción entre capas u objetos de programación, por lo que, altos porcentajes de esta latencia suelen ser indicadores de códigos complejos que requieren la interacción de diversos objetos.

Aparte de esto, las especulaciones fallidas alcanzaron alrededor de un 5.6% del tiempo de ejecución. Este valor corresponde, tanto a ciclos de reloj bloqueados por la recuperación de una especulación incorrecta e instrucciones que no fueron retiradas. En otras palabras, esta métrica mide la cantidad de tiempo en que piezas valiosas de información fueron desperdiciadas, ya sea por predicciones de salto fallidas o por instrucciones que nunca fueron retiradas y el tiempo invertido en el reordenamiento de las instrucciones en la estructura de segmentación. En este caso, el valor se mantuvo realmente bajo, lo que podría interpretarse como la capacidad del predictor de saltos para predecir la carga de instrucciones que se ejecutarán.

La última categoría de valores analizada fue el límite de back-end, que alcanzó alrededor de un 19.12%. Tal y como se describe en el anexo 4 de este proyecto, El back-end es la porción del *core* del procesador en donde el programador de instrucciones despacha operaciones de hardware en sus respectivas unidades de ejecución, que serán posteriormente despachadas de acuerdo al orden del programa. El límite de back-end se refiere entonces a los ciclos de reloj en los que las ranuras de segmentación no pudieron aceptar operaciones de hardware debido a falta de recursos por operaciones que toman demasiado tiempo, traduciéndose, a nivel de usuario, en una ejecución más lenta del programa. En el caso en estudio, la preponderancia de los valores en esta categoría puede rastrearse al gran número de divisiones, repetidos accesos a memoria y el direccionamiento de un gran número de operaciones a un mismo puerto debido a la reiteración de un mismo tipo de operación.

5.5.3. Logros obtenidos durante la etapa V del proyecto.

Durante esta quinta etapa del proyecto, no sólo se logró corroborar la validez de las pruebas a partir de la comparación de los resultados obtenidos de las diez pruebas preliminares definidas, sino que también se logró mostrar la relación existente entre las unidades de arquitectura del procesador, como lo son Límite de Interfaz y Backend con la forma en que son construidas (por medio de código), las aplicaciones de usuario. Esta relación fue una oportunidad para precisar la manera en que las aplicaciones de usuario hacen uso de los recursos del procesador.

Finalmente, a través del análisis se logró determinar que el prototipo es capaz de monitorear la forma en que los recursos del procesador están siendo utilizados.

5.6. Etapa VI: Presentación de resultados.

Tal y como se muestra en la tabla *n.53,* sobre el desglose de la sexta etapa del proyecto, esta última etapa se define para la construcción de la interfaz del prototipo. Por tanto, este es la consecución de la quinta etapa del proyecto, ya que la comprensión de la relación existente entre las unidades funcionales del procesador y las aplicaciones de usuario logrado durante el análisis, guiará la forma en que se mostrarán los resultados.

Tabla 53. Desglose de la sexta etapa del proyecto, elaboración propia.

Objetivos	Entregables	Etapas	Técnicas	Herramientas	Temas del marco teórico
Desarrollar un prototipo de software para mostrar la forma en que las unidades funcionales de un procesador core i7 son utilizadas durante la ejecución de la aplicación de un tercero.	Expuestos en el apartado 1.4.1.4	VI	Programación orientada a objetos	Python	*2.6,2.9,2.10*
				PyCharm	
			Triangulación	Motores de búsqueda, bibliotecas virtuales y revistas especializadas.	
			Estadística descriptiva.	Bibliotecas de software.	

5.6.1. Acerca de la sistematización de los reportes para su visualización.

La estadística descriptiva es una herramienta de conocimiento obligatorio en los análisis de rendimiento ya que permite el diseño de estudios, experimentos, el análisis de la variabilidad, relaciones entre variables y asiste en la toma de decisiones. De hecho, para Arteaga, Batanero, Cañadas, & Contreras (2011), permiten sintetizar información y comunicarla a través de la transnumeración, es decir, la capacidad de obtener información útil, de hacer visible información por medio del cambio de representación, haciendo que los conceptos abstractos sean mas sencillos de entender.

A partir de la observación participante se pudo determinar que los principales usuarios del prototipo cuentan con conocimientos acerca de los conceptos básicos en estadística necesarios para encontrar el sentido de los datos, emitir razonamientos y argumentos

estadísticos. Asimismo, permitió determinar que se encuentran en la capacidad de cuestionar argumentos estadísticos que no se encuentren bien fundamentados. Estas observaciones son tomadas en consideración para la selección de gráficos de barras para presentar la información resumida de los informes. Los gráficos de barras, tal y como lo explican Cooper L. & S. Shore (2010), son los más fáciles de interpretar. En ellos, la altura de las barras corresponde a la a la magnitud del valor obtenido tal y como se muestra en la ilustración *n.54*.

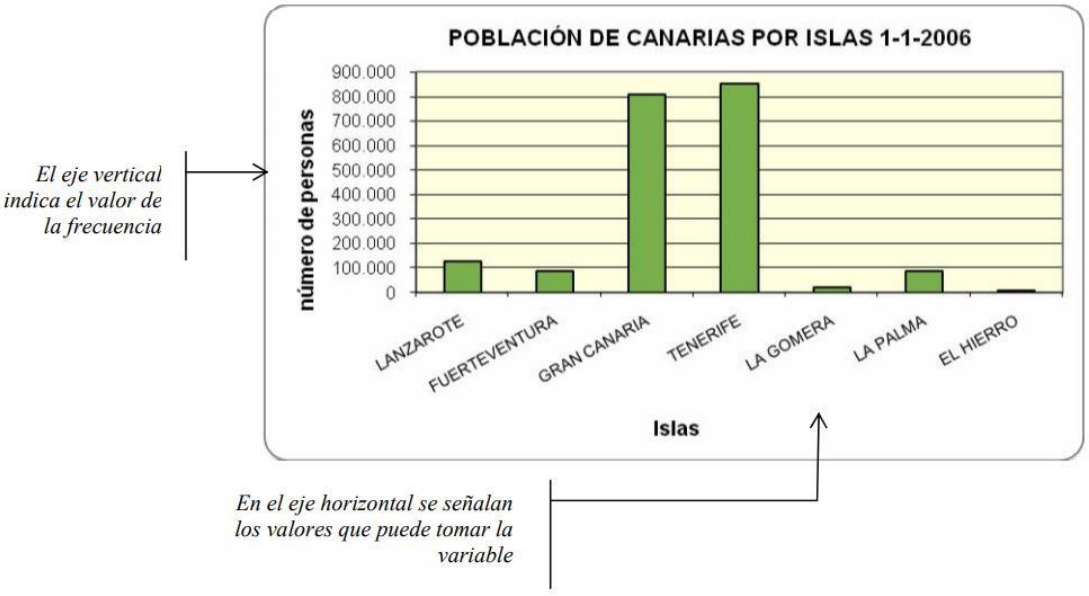


Ilustración 55. Población de Canarias por islas 1-1-2006. En Gráficos estadísticos (p.3), por Gil Armas & Martín González, 2010, Instituto Canario de Estadística. Derechos de autor [2010] por Instituto Canario de Estadística. Reimpresión autorizada.

Además, se decide representar los valores de las categorías descritas en la etapa V, es decir, Retiring, Limite de interfaz y Limite de back-end, de forma general. Asimismo, se decide hacer un informe de cada categoría por aparte con sus respectivas subcategorías.

5.6.2. Resultados del plan de pruebas.

Además de la generación de gráficas para la presentación de resultados, durante esta última etapa se ejecutaron las pruebas descritas en el plan de pruebas para obtener la aceptación final del cliente. El detalle las pruebas realizadas se encuentra en el anexo 10 de este proyecto.

5.6.3. Logros obtenidos durante la etapa VI del proyecto.

En esta etapa se logra definir una forma para sistematizar los datos y representar la información de los datos obtenidos por medio de los reportes. En esta etapa se da la elaboración de la interfaz de usuario de acuerdo a los requerimientos del cliente y las convenciones del equipo. Además, se hicieron pruebas de aceptación del prototipo por parte del cliente. Finalmente, se realizan, un manual de usuario, una guía y una guía rápida de uso, que se encuentran en el anexo 7 del proyecto.

CAPÍTULO VI: CONCLUSIONES Y RECOMENDACIONES DEL PROYECTO.

En las siguientes líneas se describen las conclusiones obtenidas a lo largo del proyecto, así como recomendaciones hacia la empresa en donde se realizó el proyecto en relación con la información obtenida durante el desarrollo del prototipo.

6.1. Conclusiones.

El desarrollo de un prototipo de software capaz de monitorear la forma en que son utilizadas las unidades funcionales del procesador requirió sortear una serie de dificultades. Para enfrentarlas y cumplir con los objetivos planteados, se dividió en seis etapas.

En la primera etapa se logró la definición, redefinición, negociación y renegociación de los requerimientos que demarcaron los alcances y limitaciones del proyecto, acordes estos con las expectativas del cliente y los usuarios del prototipo cumpliendo con ello el objetivo de **Identificar los requerimientos de software para la elaboración de un prototipo funcional que monitoree la utilización de las unidades funcionales de los procesadores Intel core i7.**

El proceso de investigación efectuado durante la segunda etapa puso en evidencia que no sólo existen numerosas métricas para análisis de rendimiento sino también, cuantiosas formas de procesar los datos y emitir informes que bridan perspectivas diferenciadas acerca de las mediciones obtenidas. Como resultado de esta etapa, se seleccionaron técnicas de recolección de métricas, datos y herramientas de monitoreo. Además, se detectaron y seleccionaron unidades de arquitectura que posteriormente se rastrearon en las métricas de rendimiento tomadas en cuenta para el proyecto.

En la tercera etapa se partió de una discusión sobre parámetros y factores para comprender el lugar que tienen los distintos elementos que componen un sistema y la forma en que éstos pueden influir sobre los resultados de los experimentos realizados, sirviendo como guía para la elaboración de estos. La segunda y la tercera etapa, engloban el objetivo de **Analizar la relación entre la arquitectura del procesador, sus unidades funcionales y el ejercicio de éstas unidades al ejecutarse la aplicación de un tercero.**

En la cuarta etapa se elaboraron diagramas UML: diagramas de casos de uso, descripción de casos de uso, diagramas de clases, de colaboración y actividad, que

permitieron imaginar el flujo de información entre las capas del sistema y sus componentes. Asimismo, la definición de la trazabilidad de los requerimientos y la realización de pruebas funcionaron como criterio de aceptación para el cliente, quien tuvo la posibilidad de cotejar los requerimientos con las pruebas realizadas y a su vez, las pruebas con los resultados obtenidos a partir de las mismas. En la cuarta etapa se abarca el objetivo de **Diseñar una propuesta de software que permita monitorear la forma en que las unidades de los procesadores Intel core i7 están siendo ejercitadas desde el nivel de usuario.**

En la quinta etapa se selecciona un *Top-Down Analysis* detallado para la sistematización de los informes de las pruebas preliminares. Esta decisión se vio influenciada primero, por la posibilidad de sistematizar grandes cantidades de información de forma tal que fuera fácilmente comprendida por el usuario, ya que la presentación del informe detallado por niveles propio del *Top-Down Analysis* permite distinguir las métricas implicadas dentro de cada métrica principal y las métricas subyacentes implicadas en cada una de las métricas secundarias. A través del análisis de los datos recolectados se logra determinar que el prototipo es capaz de monitorear la forma en que los recursos del procesador están siendo utilizados.

En la sexta y última etapa del proyecto se definió la visualización de las síntesis de los informes desde el prototipo. En esta etapa se decide que la presentación de los resultados se dará por medio de gráficos de barras que facilitan la detección de tendencias. Las tendencias permitieron redireccionar la mirada hacia las métricas que tuvieron mayor número de eventos y con ello detectar las unidades que se ejecutaron en mayor o menor medida y los cuellos de botella que se dieron en tiempo de ejecución. Con las etapas cinco y seis se abarca el objetivo de **Desarrollar un prototipo de software para mostrar la forma en que las unidades funcionales de un procesador core i7 son utilizadas durante la ejecución de la aplicación de un tercero.**

6.2. Recomendaciones.

La etapa I demostró la importancia que tienen los datos recolectados para el equipo. Por tanto, se recomienda elaborar un plan de respaldo para los datos generados por el prototipo en el que se incorporen los recursos preexistentes para respaldos con las que el equipo cuenta en la actualidad.

La etapa II mostró la utilidad de las métricas de rendimiento para rastrear las unidades funcionales del procesador. Por ello se recomienda la incorporación en el prototipo de métricas que no fueron tomadas en cuenta para el desarrollo del proyecto, como lo son las métricas de *uncore* y de manejo de energía.

Además, otra recomendación que se desprende de esta etapa resulta de la herramienta utilizada. Se recomienda explorar con la manipulación directa de Registros de Estado (MSRs, por sus siglas en ingles), en vez de las herramientas tradicionales de monitoreo, ya que, al no necesitar del tiempo de procesamiento de datos propio de las herramientas auxiliares en los análisis de rendimiento, se facilita la obtención de resultados, que pueden visualizarse casi inmediatamente después de que surgen permitiendo al prototipo mostrar en tiempo real, los resultados obtenidos durante las pruebas.

La etapa III evidenció la relevancia de los conocimientos técnicos. Por esto, se recomienda el refrescamiento de términos, principios de análisis de rendimiento y microarquitectura, esto a través de talleres presenciales o virtuales impartidos por los mismos miembros del equipo.

Por otra parte, se recomienda consultar los diagramas realizados en la etapa IV del proyecto para el escalamiento del prototipo. Acerca de este punto, se recomienda la utilización del prototipo en sistemas con procesadores equivalentes, con distintas configuraciones para la detección de errores que hayan sido transparentes durante la fase de pruebas del proyecto. Se recomienda que estas pruebas contemplen, en principio, sistemas con el mismo procesador, pero diferentes piezas del hardware restante, para así optimizar el código de forma tal que pueda ser utilizado en la mayor cantidad de sistemas posibles.

En otras palabras, se recomienda un escalamiento horizontal previo al escalamiento vertical del prototipo. Para este último, se recomienda la implementación, primero en sistemas con arquitecturas simples, pasando luego a arquitecturas más complejas. Esto para facilitar el proceso de detección de errores en arquitecturas complejas.

De la etapa V se desprende necesidad del equipo por obtener datos cada vez más precisos y generar la mayor cantidad de información de calidad sobre esos datos. Por esto, se le recomienda enfocar los esfuerzos en la exploración de los informes y oportunidades brindados por las herramientas auxiliares existentes para análisis de rendimiento, ya que las

potencialidades de dichas herramientas están siendo desaprovechadas. Algunos análisis que permite *VTune Amplifier*, que se recomiendan explorar y que no fueron contemplados en el proyecto son: *Hotspots Analysis for CPU Usage Issues, Parallelism Analysis group*, *Platform Analysis Group* y *Analyze Energy Usage.* Asimismo, se recomienda ahondar en tipos de reportes como: *hotspots*, *hw-events*, *callstacks, gprof-cc* y *gpu-computing-tasks*.

Finalmente, la etapa VI probó la pertinencia de la presentación de los datos para la transmisión de ideas y el resumen de la información. En este sentido, se recomienda explorar otras formas de presentar los datos para obtener perspectivas diferentes. Entre ellos se recomienda explorar otros tipos de gráficos: gráficos circulares, histogramas, gráficos lineales, de dispersión, gráficos de cajas, de áreas y pictogramas.

REFERENCIAS BIBLIOGRÁFICAS.

A. Mack, C. (2011). FiftyYears of Moore ' s Law. *IEEE Fellow*, *24*(2), 2008.

Abd-El-Barr, M., & El-Rewini, H. (2005). Fundamentals of Computer Organization and Architecture. *Fundamentals of Computer Organization and Architecture*, 1–273. https://doi.org/10.1002/0471478326

Alonso, A. (2018). ¿ Cuál es la estrategia de datacenter de Intel ? *IT Sitio*. Retrieved from https://www.itsitio.com/us/cual-es-la-estrategia-de-datacenter-de-intel/

Ammons, G., Ball, T., & Larus, J. R. (2005). Exploiting hardware performance counters with flow and context sensitive profiling. *ACM SIGPLAN Notices*, *32*(5), 85–96. https://doi.org/10.1145/258916.258924

Arteaga, P., Batanero, C., Cañadas, G., & Contreras, J. (2011). Las tablas y gráficos estadísticos como objetos culturales. *Números*, 55–67. Retrieved from http://www.sinewton.org/numeros/numeros/76/Volumen_76.pdf#page=55

Arquitectura de computadoras. (2005). *Morris Mano, M.* (3rd ed.). Pearson.

Asensi Artiga, V. (1993). Evolución histórica de las tecnologías de la información y su aplicación en el proceso documental. *Revista General de Información y Documentación*, *3*(2), 131–141.

Barnstijn, M. A., Chrurch, M. E., Linkert, B. W., & Lazaridis, M. (1998). United States Patent [191 [11] Patent Number :, (19), 3–6.

Barrachina Mir, S., Castillo Catalán, M., Fabregat Llueca, G., Fernández Fernández, J. C., León Navarro, G., Martí Avilés, J. V., … Montoliu Colás, R. (2018). Introducción a la arquitectura de computadores con QtARMSim y Arduino. *Introducción a La Arquitectura de Computadores Con QtARMSim y Arduino*. https://doi.org/10.6035/sapientia129

Benavides, M., & Restrepo, C. G. (2005a). Métodos en investigación cualitativa: triangulación. *Revista Colombiana de Psiquiatría*, *XXXIV*(1), 118–124. Retrieved from

http://www.raco.cat/index.php/Elies/article/viewArticle/195506/0%5Cnhttp://www.scielo.org.co/scielo.php?pid=S0034-74502005000100008&script=sci_arttext

Benavides, M., & Restrepo, C. G. (2005b). Métodos en investigación cualitativa: triangulación. *Revista Colombiana de Psiquiatría*, *XXXIV*(1), 118–124.

Cataldi, Z. (2000). *Una metodología para el diseño, desarrollo y evaluación de software educativo (Doctoral dissertation, Facultad de Informática).* (1st ed.). Uruguay: Universidad Navional de la Plata.

Daros, W. R. (2011). Introduccion a la epistemologia poperiana, 1–19.

Díaz B, L., Torruco G, U., Martínez H, M., & Varela R, M. (2013a). La entrevista, recurso flexible y dinámico. *Investigación En Educación Médica*, *2*(7), 162–167. https://doi.org/10.1016/S2007-5057(13)72706-6

Díaz B, L., Torruco G, U., Martínez H, M., & Varela R, M. (2013b). La entrevista, recurso flexible y dinámico. *Investigación En Educación Médica*, *2*(7), 162–167. https://doi.org/10.1016/S2007-5057(13)72706-6

Durán, F., Gutiérrez, F., & Pimentel, E. (2007). *Programación orientada a objetos con Java: una introducción prácticaa usando BlueJ*. (D. J. Barnes, M. Kölling, & B. I. Brenta, Eds.) (3rd ed.). Argentina: Pearson, Prentice Hall. Retrieved from https://books.google.es/books?hl=es&lr=&id=3EQdUbkOVGIC&oi=fnd&pg=PP1&dq=lenguaje+de+programacion&ots=04fUAByGLJ&sig=-QJGbMIwSCHdljMfXhdK8IuYn3k#v=nepage&q=lenguaje de programacion&f=false

Fariño, G. (2011). Modelo Espiral de un proyecto de desarrollo de software. *Unemi*, 1–9.

Fernández-Montesinos, F. A. (2017). Repensar el liderazgo estratégico. La visión. *IEEE*, *42*, 182–197.

Fernández Zubieta, A. (2009). El constructivismo social en la ciencia y la tecnología: las consecuencias no previstas de la ambivalencia epistemológica. *Arbor*, *CLXXXV*(738), 689–703. https://doi.org/10.3989/arbor.2009.738n1046

Gil Armas, Antonia R. & Martín González, J. (2010). Graficos estadisticos. *Instituto*

Canario de Estadistica. Canarias. Tomado de: http://www3.gobiernodecanarias.org/istac/webescolar/material_didactico/secundaria/graficos_estadisticos/3Eso_AmpliaGraficos.pdf

Gomez, J. B. V. (2012). *Arquitectura de computadoras I*.

Hueso, A., & Cascant, M. J. (2012). *Metodología y Técnicas Cuantitativas de Investigación Andrés Hueso y Mª Josep Cascant*. *Cuadernos docentes en procesos de desarrollo* (Vol. 1). Valencia, España.

INTEL. (2019). Stock ownership guidelines. *INTEL*, 20.

Isaza-González, J., Serrano-cases, A., Restrepo-calle, F., Cuenca-asensi, S., & Martínez-Álvarez, A. (2017). Evaluación de la fiabilidad de microprocesadores COTS mediante las infraestructuras de depuración On-Chip. *RIDTEC*, *13*(1), 8–17. Retrieved from https://rua.ua.es/dspace/bitstream/10045/66855/1/2017_Isaza_etal_RIDTEC.pdf

John, L. K., Vasudevan, P., & Sabarinathan, J. (2002). Workload Characterization : Motivation , Goals and Methodology The University of Texas at Austin. *Workload Characterization: Methodology and Case Studies. Based on the First Workshop on Workload Characterization*, *9697098*, 3–14. https://doi.org/10.1109/WWC.1998.809354

Kawulich, B. B. (2005a). La observacion participante como metodo de recoleccion de datos. *Forum: Qualitative Social Research*, *6*(2), 1–32. Retrieved from papers2://publication/uuid/025F3C73-8196-44FC-8C16-0414E7049631

L. Cooper, L. & S. Shore, F. (2010). The Effects of Data and Graph Type on Concepts and Visualizations of Variability. *Journal of Statistics Education 18.* Tomado de: http://jse.amstat.org/v18n2/cooper.pdf

LaFrance-Linden, D. (2014). Profiling metrics for computer programs. EEUU.

Leung, Joseph Y-T., J. H. A. (2011). Handbook of Scheduling: Algorithms, Models, and Performance Analysis. *Plastic and Reconstructive Surgery*, *128*(4), 1001. https://doi.org/10.1097/PRS.0b013e31822adba3

Lopez, J. (2018). Intel cambia de estrategia y reducirá los envíos de CPUs de escritorio. *HardZone*. Tomado de https://hardzone.es/2018/11/15/intel-cpus-escritorio/

Lopez Tequeya, B. (2019). Fundamentos de Programación Orientada a Objetos en C# .NET. *AlfaOmega*. Tomado de http://www.itnuevolaredo.edu.mx/takeyas/Apuntes/POO/Apuntes/06b.-CompilacionCondicional.pdf

Lossada, M. A., & Robles, M. (2014). Gestión del mejoramiento continuo como estrategia competitiva de empresas de telecomunicaciones inalámbricas. *Centro DeInvestigación de Ciencias Administrativas y Gerenciales*, (1969), 17–35.

Lozada, J. (2014). Investigación Aplicada: Definición, Propiedad Intelectual e Industria. *Cienciaamérica*, *1*(3), 34–39. Retrieved from http://www.uti.edu.ec/documents/investigacion/volumen3/06Lozada-2014.pdf

Marulanda Grisales, N., Hincapié Pizza, E. A., & Echeverry Correa, J. (2016). Caracterización de la implementación de lean manufacturing vs teoría de restricciones: Estudio de caso colombiano. *Revista Espacios*, *37*(Nº 25), 1–13.

Marusarz, J., & Ryabtsev, D. (2019). Top-down Microarchitecture Analysis Method. *Intel*, 1–13. Retrieved from https://software.intel.com/en-us/vtune-amplifier-cookbook-top-down-microarchitecture-analysis-method

M. F. Sanner. (1998). Python: a programming language for software integration and development. *The scripps Research Institute.* Tomado de https://www.scripps.edu/sanner/html/papers/NewsAndViewsSept99.pdf

Milan Milenkovic. (1996). Sistemas operativos: conceptos y diseño. *Martin Iturbide*. Retrieved from https://books.google.com.mx/books?hl=es&lr=&id=N5yxiwilBhoC&oi=fnd&pg=PR1&dq=sistema+operativo+tipos&ots=02N1CwN5yY&sig=U3q4jbc-JJuFT6OfXeENle4ms0E#v=onepage&q=sistema operativo tipos&f=false

Monge-González, R. (2017). *Ascender en la Cadena Global de Valor : el caso de Intel* (1st ed.). Perú: Organización Mundial del trabajo, sede regional para Latinoamérica y el

Caribe. https://doi.org/10.13140/RG.2.2.18387.48160

Monge-gonzález, R., & Carlos, J. (n.d.). Impacto de los derrames de conocimiento multinacionales en las empresas locales en Costa Rica. *Publicacion Independiente*, 1–24.

Monge-González, R., & González-Alvarado, C. (2007). The role and impact of MNCs in Costa Rica on skills development and training: The case of Intel, Microsoft and Cisco.

Monge-González, R., Leiva Bonilla, J. C., & Rodríguez Álvarez, J. A. (2012). Inversión extranjera directa, movilidad laboral y derrames de conocimiento en Costa Rica. *Tecnología En MArcha*, *25*, 103–115.

Noyce, R. N., & Hoff, M. E. (1981). A History of Microprocessor Development at Intel. *IEEE Micro*, *1*(1), 8–21. https://doi.org/10.1109/MM.1981.290812

Plaza, R. asenjo, Carrasco, E. G., & Cozar, J. R. (2001). *Fundamentos de los computadores*.

Hu, Q., Ma, L. & Zhao, J. (2018). *DeepGraph: A PyCharm Tool for Visualizing and Understanding Deep Learning Models*. IEEE. Tomado de: https://ieeexplore.ieee.org/abstract/document/8719435

Ramos, C. A. (2015). Los Paradgmas de la Investigación Científica. *Av. Psicol*, *23*(1), 9–17.

Rowland, C. (2017). Intel Corporation ' s Vision Statement & Mission Statement. *Panmore Institute*, 1. Retrieved from http://panmore.com/intel-corporation-vision-statement-mission-statement

Shobaki, M. El, & Lindh, L. (2001). A hardware and software monitor for high-level system-on-chi.pdf, 56–61.

Stachniak, Z. (2013). This is not a computer: Negotiating the microprocessor. *IEEE Annals of the History of Computing*, *35*(4), 48–54. https://doi.org/10.1109/MAHC.2013.15

Stallings, W. (2010). *Computer organization and architecture* (10th ed.). Pearson.

UCM. (2019). Estructura de Computadores. *Universidad Computense de Madrid.* Tomado de https://www.fdi.ucm.es/profesor/jjruz/WEB2/Temas/EC6.pdf

Valles, M. S. (2011). *Técnicas cualitativas de investigación social*. Retrieved from papers2://publication/uuid/034684C5-ABB4-4FAB-B9FE-D85967E1DD24

Vargas-Sabadias, A. (1995a). Estadistica descriptiva e inferencial. *Colegio de Bachilleres*, 450. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Estadistica+descriptiva+e+inferencial#2

Vargas-Sabadias, A. (1995b). Estadistica descriptiva e inferencial. *Colegio de Bachilleres*, 450.

Vargas Cordero, Z. R. (2009). La investigación aplicada: Una forma de conocer las realidades con evidencia científica. *Revista Educación*, *33*(1), 155–165. Retrieved from https://revistas.ucr.ac.cr/index.php/educacion/article/viewFile/538/589

William, I., Alexander, P., Us, T. X., Jones, S. T., Us, T. X., Levine, F. E., … Urquhart, R. J. (2010). System and method for collecting a plurality of metrics in a single profiling run of computer code, *2*(12).

GLOSARIO.

1) ALU: Unidad aritmético-lógica del procesador.
2) Arbor: Revista del Colegio Superior de Investigaciones Científicas.
3) CGV: Cadena Global de Valor.
4) CINDE: Agencia de inversión en Costa Rica.
5) COMEX: Ministerio de comercio exterior.
6) Communinty edition: versión gratuita de software licenciado.
7) COP: Coprocesador.
8) Core: núcleo de un procesador.
9) CPU: Unidad Central de Procesamiento.
10) Cuello de botella: factor crítico que empobrece rendimiento de un sistema.
11) CUS: Componente en estudio.
12) DFU: Unidad decimal de punto flotante
13) Firmware: código de bajo nivel que permite controlas las funciones de un dispositivo.
14) FXU: Unidad de punto fijo.
15) Hardware: componentes físicos de un sistema computacional.
16) IA: Inteligencia Artificial.
17) I&D: Investigación y Desarrollo.
18) IED: Inversión extranjera directa.
19) IEEE: Instituto de Ingenieros Eléctricos y Electrónicos.
20) Interfaz: en software, medio que permite al usuario interactuar con el computador sin necesidad de utilizar código.
21) IDU: Unidad de decodificación de instrucciones.
22) IFU: Unidad de Carga de Instrucciones.
23) IPC: Instrucciones por ciclo de máquina.
24) IR: Registro de Instrucciones.
25) ISU: Secuenciador de instrucciones.
26) IT: Tecnologías de la Información.
27) *Lifelong learning process:* aprendizaje no técnico que es útil en la esfera practica de la vida cotidiana o laboral.
28) LSU: Unidad de carga y almacenamiento.

29) MAR: Registro de dirección a memoria.
30) MB: Megabyte.
31) MBR: Registro de buffer de memoria.
32) MICITT: Ministerio de Ciencias, Tecnología y Telecomunicaciones.
33) MIPS: Millones de instrucciones por segundo.
34) MNCS: conjunto de incentivos para empresas transnacionales.
35) Multicore: procesador con más de un núcleo.
36) NTRS: Plan Nacional de Tecnología para Semiconductores.
37) Opcode: código de operación. Especifica la operación que será efectuada con los datos.
38) Out of box: configuración por defecto de un sistema.
39) PC: computadora de escritorio.
40) PC: Program Counter.
41) PCB: circuito impreso.
42) PIB: Producto Interno Bruto.
43) Postsilicon: sistema computacional que se encuentra en el mercado.
44) Presilicon: modelo computacional que se encuentra en etapas de desarrollo.
45) Redalyc: Red de Revistas Científicas de América Latina y el Caribe, España y Portugal.
46) RU: Unidad de recuperación.
47) RZF: Régimen de Zonas Francas.
48) Score: valor obtenido por un sistema después de haber realizado pruebas de rendimiento.
49) SIA: Asociación Industrial de Semiconductores.
50) Single core: sistema computacional de un solo núcleo.
51) Software: lógica interna que hace funcionar una computadora.
52) SSD: Disco duro.
53) SUT: Sistema en Estudio.
54) Tercero: persona física o jurídica ajena a Intel.
55) TLB: Lookaside Buffer. Parte de la memoria caché que interactúa con la memoria principal.
56) TOC: Teoría de las Restricciones.
57) XU: Unidad de traducción.
58) ZF: Zona Franca.

ANEXOS.

Anexo 1: Cronograma detallado.

ID	Task Mode	Task Name	Duration	Start	Finish
1		**Problema del proyecto**	**19 days**	**2/4/19**	**2/28/19**
15		**Marco teórico**	**22 days**	**3/1/19**	**4/1/19**
16		**Generalidades**	**2 days**	**3/1/19**	**3/4/19**
17		Principios de microarquitectura	1 day	3/1/19	3/1/19
18		Un acercamiento a los procesadores Inte	1 day	3/4/19	3/4/19
19		**Los procesadores y la competencia por**	**2 days**	**3/5/19**	**3/6/19**
20		La ley de Moore	1 day	3/5/19	3/5/19
21		Técnicas para mejorar rendimiento en	1 day	3/6/19	3/6/19
22		**Contexto del procesador**	**2 days**	**3/7/19**	**3/8/19**
23		Generalidades sobre los sets de	1 day	3/7/19	3/7/19
24		El dilema entre CISC y RISC.	0 days	3/8/19	3/8/19
25		Una vista al ciclo de una instrucción.	1 day	3/8/19	3/8/19
26		**Estructura y funcionamiento de un**	**3 days**	**3/11/19**	**3/13/19**
27		Registros visibles para el usuario	1 day	3/11/19	3/11/19
28		Registros de estado y control	1 day	3/12/19	3/12/19
29		Una mirada más profunda a los	1 day	3/13/19	3/13/19
30		**Análisis de rendimiento**	**2 days**	**3/14/19**	**3/15/19**
31		Aspectos importantes en la selección de	1 day	3/14/19	3/14/19
32		Consoderaciones finales en relación con	1 day	3/15/19	3/15/19
33		Técnicas de caracterización por cargas de	1 day	3/19/19	3/19/19
34		Notas sobre los monitores de tareas	1 day	3/20/19	3/20/19
35		**Clasificación de los monitores existentes**	**4 days**	**3/21/19**	**3/26/19**
36		Monitores de software	2 days	3/21/19	3/22/19
37		Monitores de hardware	1 day	3/25/19	3/25/19

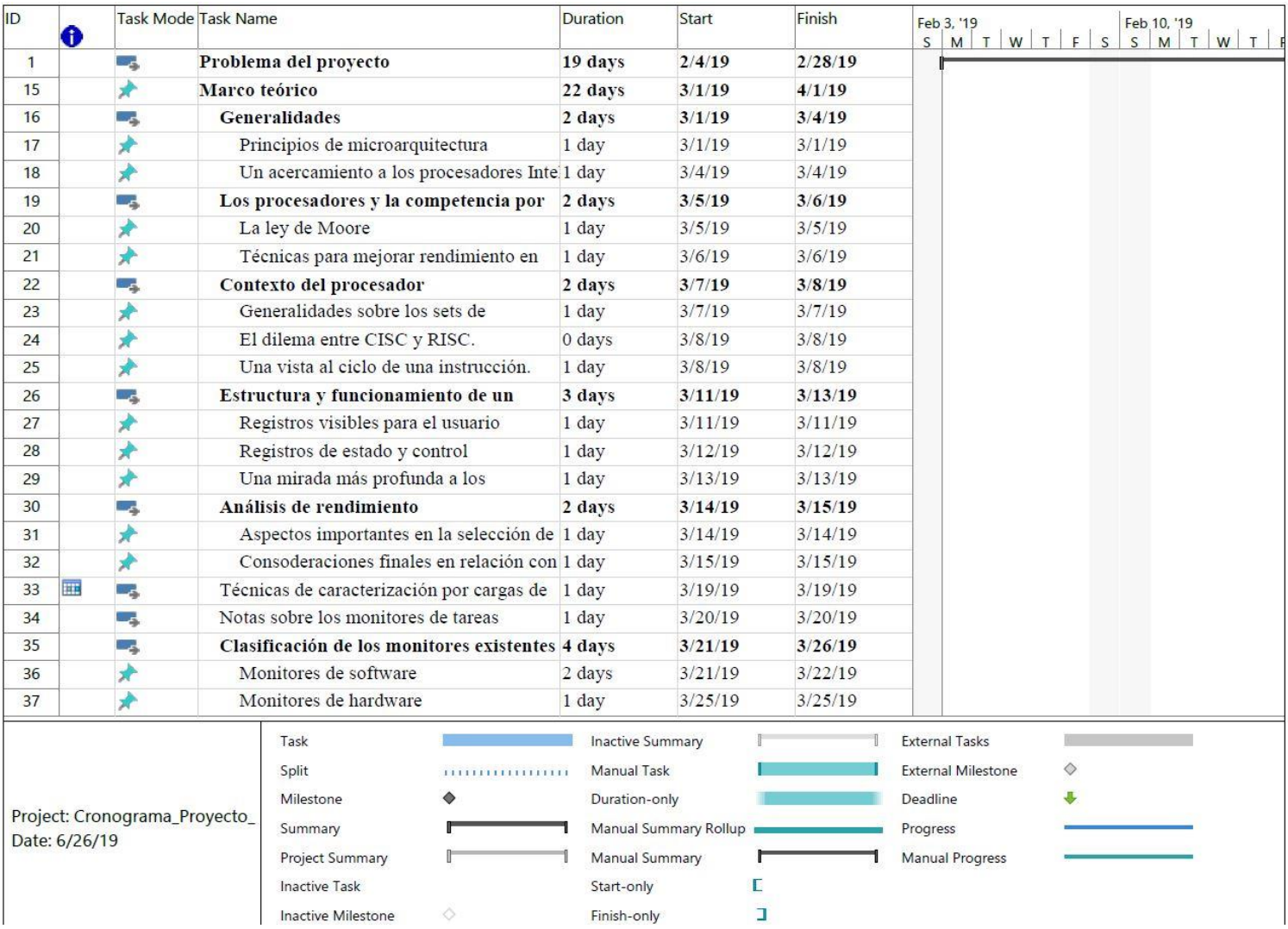


Ilustración 56. Detalle de cronograma, parte 1, elaboración propia.

ID	Task Mode	Task Name	Duration	Start	Finish
38		Firmware y monitores híbridos	1 day	3/26/19	3/26/19
39		**Ciclo de vida del software y modelos de**	**4 days**	**3/27/19**	**4/1/19**
40		Modelo en cascada	1 day	3/27/19	3/27/19
41		Prototipado evolutivo	1 day	3/28/19	3/28/19
42		Modelo en espiral	1 day	3/29/19	3/29/19
43		Modelo orientado a objetos	1 day	4/1/19	4/1/19
44		**Marco metodológico**	**22 days**	**4/1/19**	**4/30/19**
45		Tipo de investigación	1 day	4/1/19	4/1/19
46		Enfoque de investigación	1 day	4/2/19	4/2/19
47		**Fuentes y sujetos de investigación**	**3 days**	**4/3/19**	**4/5/19**
48		Fuentes primarias	1 day	4/3/19	4/3/19
49		Fuentes secundarias	1 day	4/4/19	4/4/19
50		Sujetos de información	1 day	4/5/19	4/5/19
51		**Técnicas y herramientas**	**2 days**	**4/8/19**	**4/9/19**
52		Técnicas de recolección de datos	1 day	4/8/19	4/8/19
53		Herramientas de recolección de datos	1 day	4/9/19	4/9/19
54		Variables	1 day	4/10/19	4/10/19
55		**Diseño de la investigación**	**7 days**	**4/11/19**	**4/19/19**
56		Etapa I: Definición del problema	1 day	4/11/19	4/11/19
57		Etapa II: Selección de métricas de	1 day	4/12/19	4/12/19
58		Etapa III: Lista de parámetros y factores.	1 day	4/15/19	4/15/19
59		Etapa IV: Diseño de experimento.	1 day	4/16/19	4/16/19
60		Etapa V: Análisis e interpretación.	1 day	4/17/19	4/17/19
61		Etapa VI: Presentación de resultados.	1 day	4/18/19	4/18/19

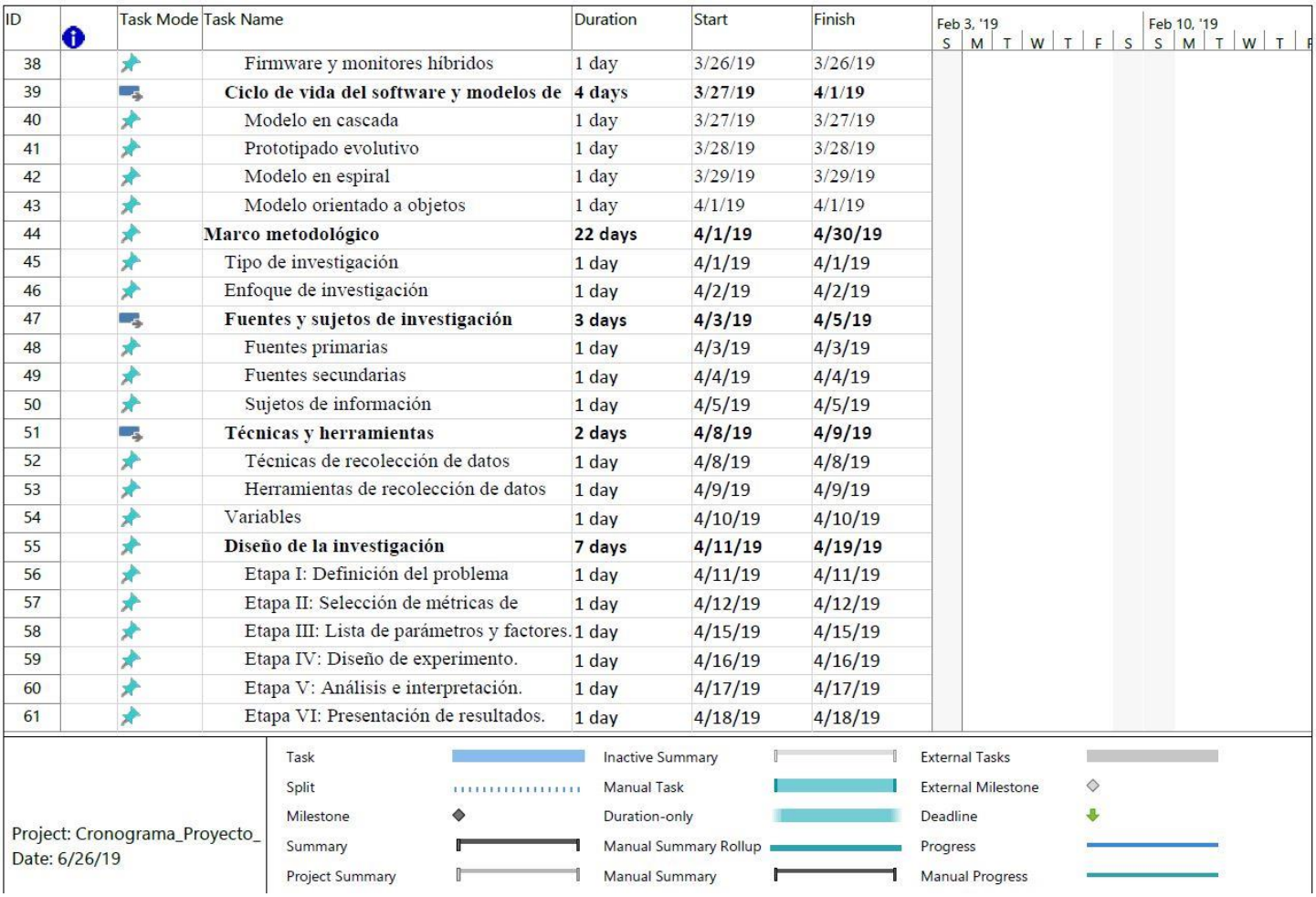


Ilustración 57. Detalle de cronograma, parte 2, elaboración propia.

ID	Task Mode	Task Name	Duration	Start	Finish
62		Matriz de coherencia	1 day	4/22/19	4/22/19
63		**Diagnóstico de empresa**	**8 days**	**4/20/19**	**4/30/19**
64		Visión, misión, valores empresariales y	1 day	4/20/19	4/20/19
65		**Diagnóstico admnistrativo**	**6 days**	**4/22/19**	**4/27/19**
66		Hacer negocios con honestidad	1 day	4/22/19	4/22/19
67		Cumplimiento tanto de la letra como del	1 day	4/23/19	4/23/19
68		Tratar a los demás de manera equitativa.	1 day	4/24/19	4/24/19
69		Actuar en beneficio de Intel y evitar	1 day	4/25/19	4/25/19
70		Proteger los activos físicos y la	1 day	4/26/19	4/26/19
71		Diagnóstico técnico	1 day	4/29/19	4/29/19
72		Diagnóstico de percepción	1 day	4/30/19	4/30/19
73		**Consideraciones finales**	**3 days**	**4/29/19**	**5/1/19**
74		Proyecto en relación con principios	1 day	4/29/19	4/29/19
75		La propuesta técnica de Intel en	1 day	4/30/19	4/30/19
76		Recomendaciones a partir de	1 day	5/1/19	5/1/19
77		**Diseño y desarrollo del proyecto**	**39 days**	**4/30/19**	**6/24/19**
78		**I. Definición del problema**	**8 days**	**5/1/19**	**5/10/19**
79		Requerimientos funcionales	2 days	5/1/19	5/2/19
80		Requerimientos no funcionales	2 days	5/3/19	5/6/19
81		Matriz de riesgo	4 days	5/7/19	5/10/19
82		**II. Selección de métricas**	**4 days**	**5/13/19**	**5/16/19**
83		Un acercamiento al rendimiento a través	1 day	5/13/19	5/13/19
84		Anotaciones sobre Vtune Amplifier.	1 day	5/14/19	5/14/19
85		Instalación y ejecución de la	1 day	5/15/19	5/15/19

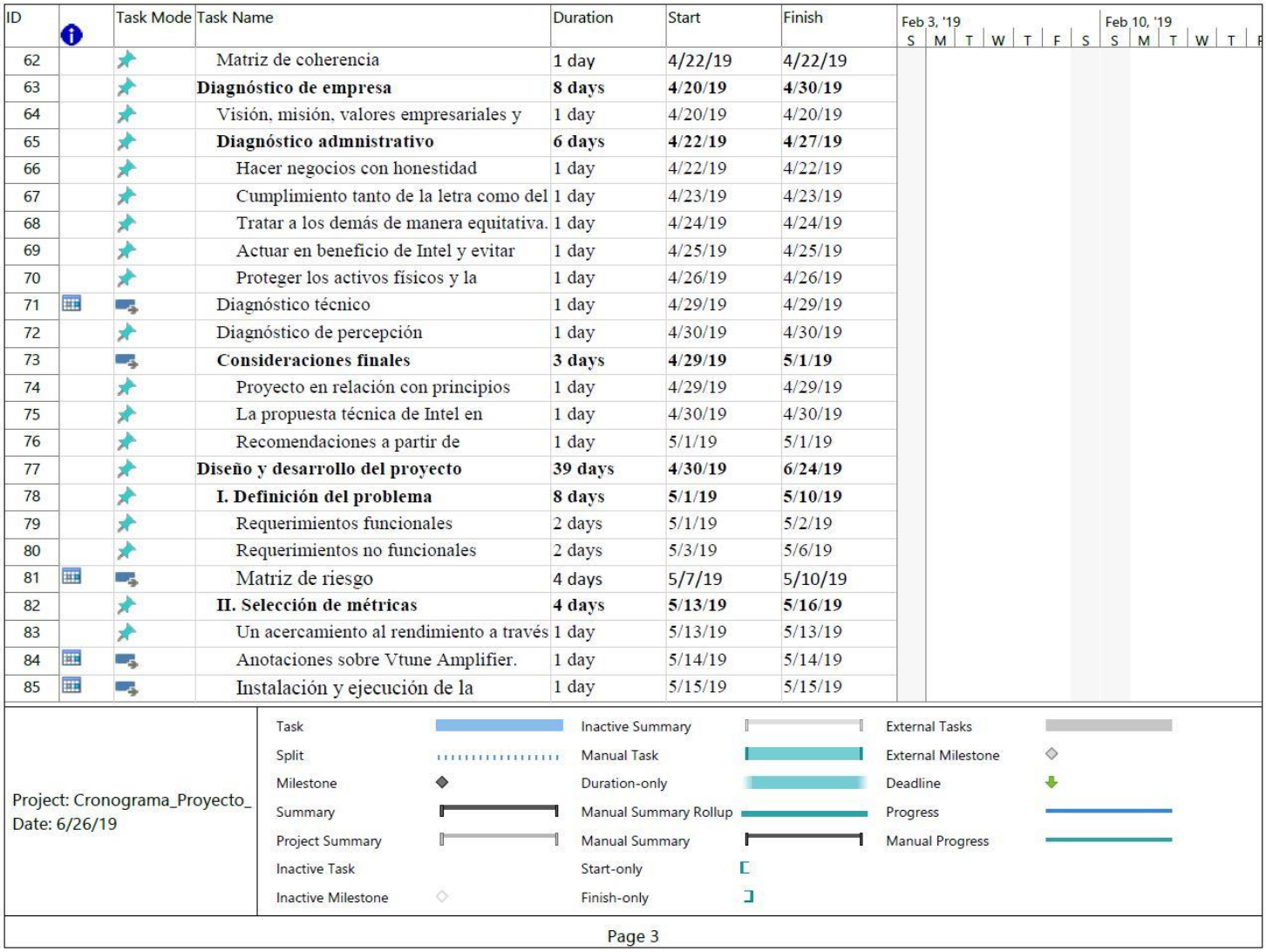


Ilustración 58. Detalle de cronograma, parte 3, elaboración propia.

ID	Task Mode	Task Name	Duration	Start	Finish
86		Detección y selección de unidades de	1 day	5/16/19	5/16/19
87		**III. Una discusión sibre parámetros y**	**1 day**	**5/17/19**	**5/17/19**
88		**IV. Diagramas de UML**	**14 days**	**5/20/19**	**6/6/19**
89		Diagramas de casos de uso.	1 day	5/20/19	5/20/19
90		Descripción de casos de uso.	7 days	5/21/19	5/29/19
91		Diagrmas de secuencia.	1 day	5/30/19	5/30/19
92		Diagramas de clases.	1 day	5/31/19	5/31/19
93		Diagramas de colaboración.	2 days	6/3/19	9/12/19
94		Diagramas de actividad.	1 day	9/13/19	9/13/19
95		Plan de pruebas.	1 day	9/16/19	9/16/19
96		**V. Análisis e interpretación**	**10 days**	**6/6/19**	**6/19/19**
97		Sobre el tipo de análisis seleccionado.	2 days	6/6/19	6/7/19
98		Descripción de pruebas realizadas.	2 days	6/12/19	6/13/19
99		Análisis de las pruebas realizadas.	4 days	6/14/19	6/19/19
100		**VI. Presentación de resultados**	**3 days**	**6/20/19**	**6/24/19**
101		Precondiciones para el uso.	1 day	6/20/19	6/20/19
102		Utilización del prototipo.	1 day	6/21/19	6/21/19
103		Gestión de riesgo.	1 day	6/24/19	6/24/19
104		**Conclusiones y recomendaciones**	**2 days**	**6/25/19**	**6/26/19**
105		Conclusiones.	1 day	6/25/19	6/25/19
106		Recomendaciones.	1 day	6/26/19	6/26/19
107		**Bibliografía consultada**	**1 day**	**6/27/19**	**6/27/19**
108		**Glosario**	**3 days**	**6/28/19**	**7/1/19**
109		**Anexos**	**1 day**	**7/5/19**	**7/5/19**

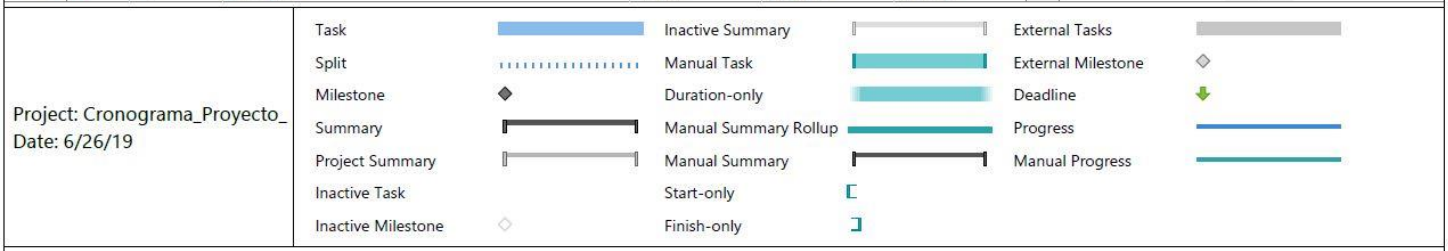


Ilustración 59. Detalle de cronograma, parte 4, elaboración propia.

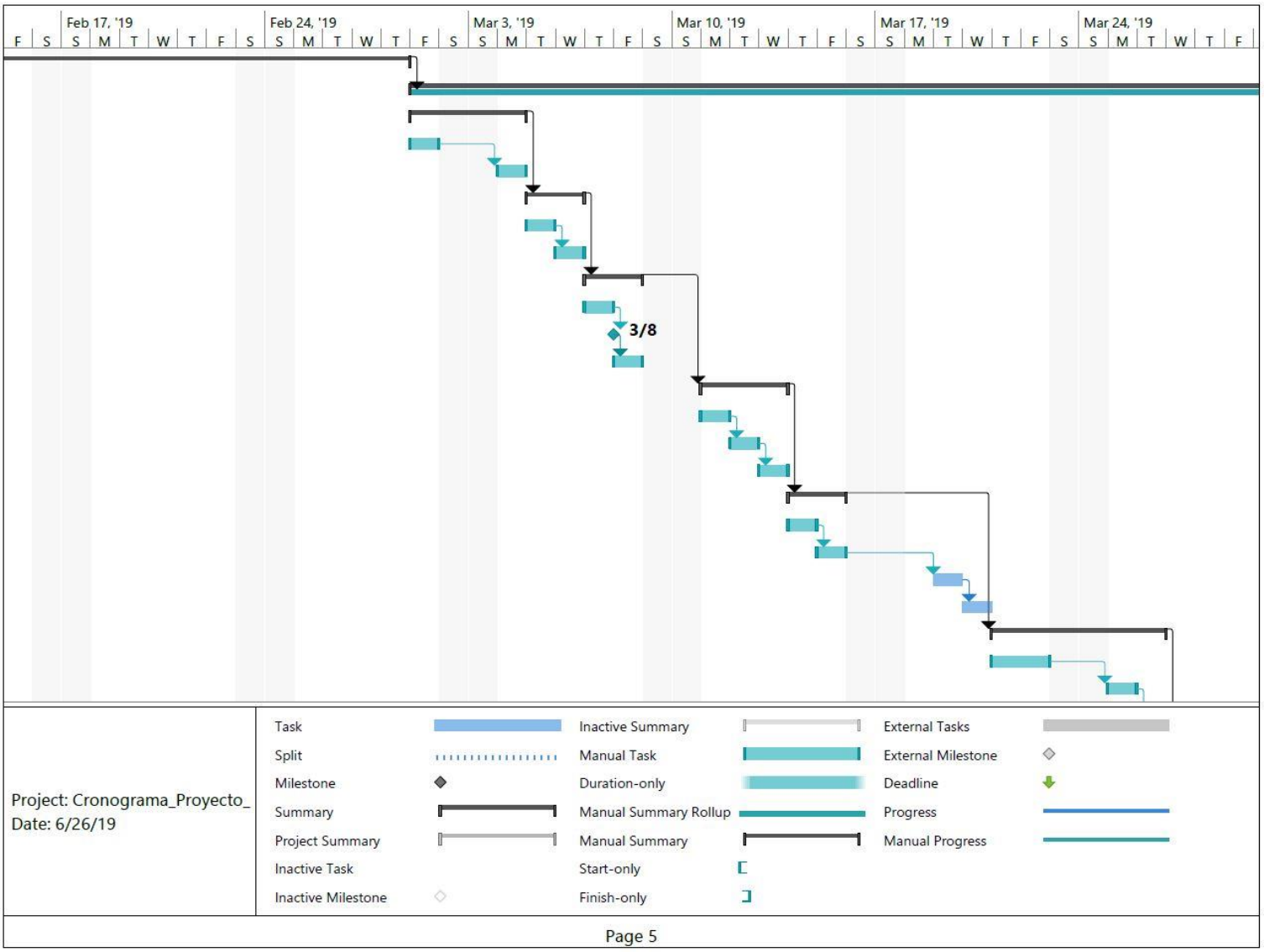


Ilustración 60. Detalle de cronograma, parte 5, elaboración propia.

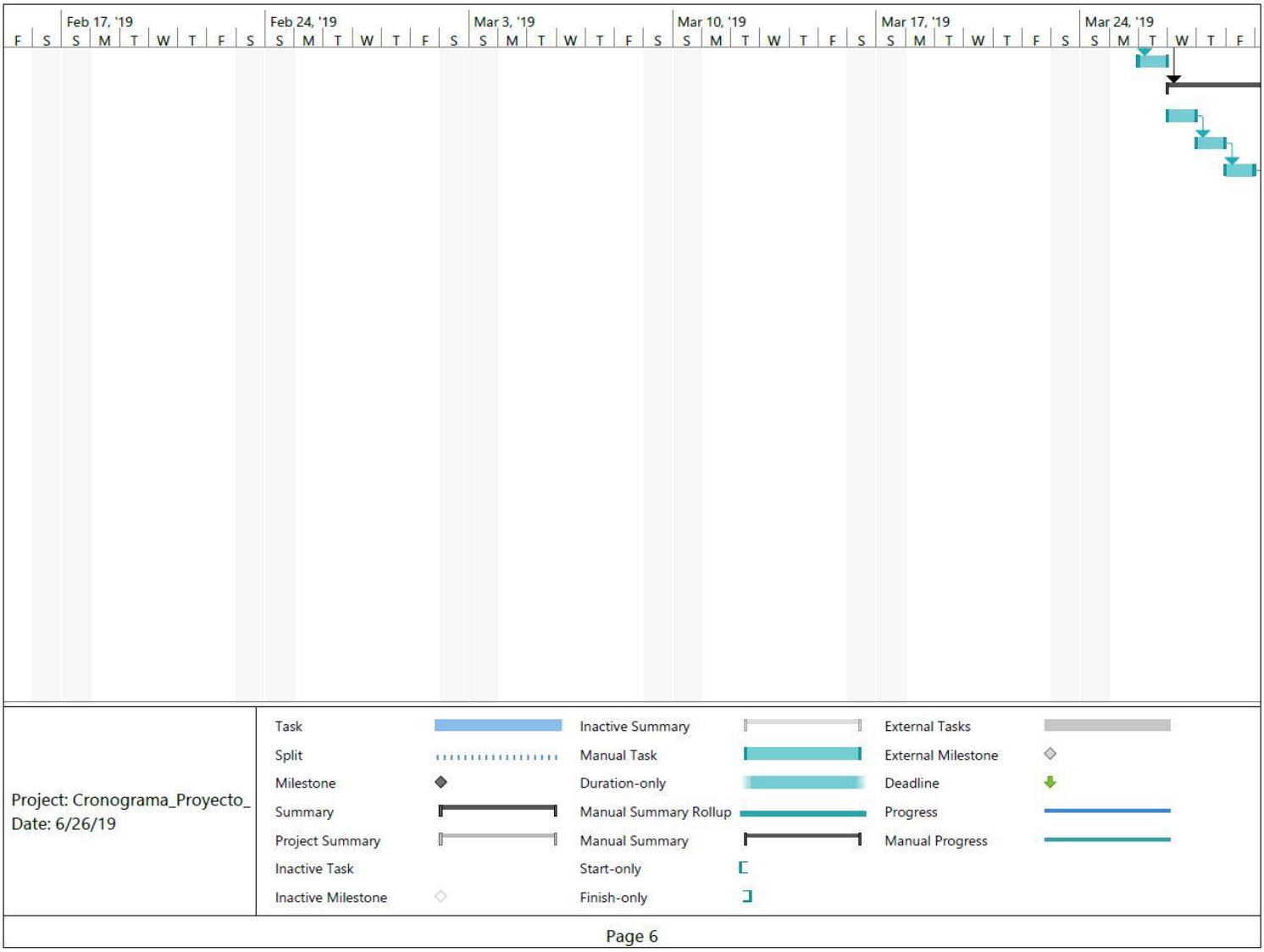


Ilustración 61. Detalle de cronograma, parte 6, elaboración propia.

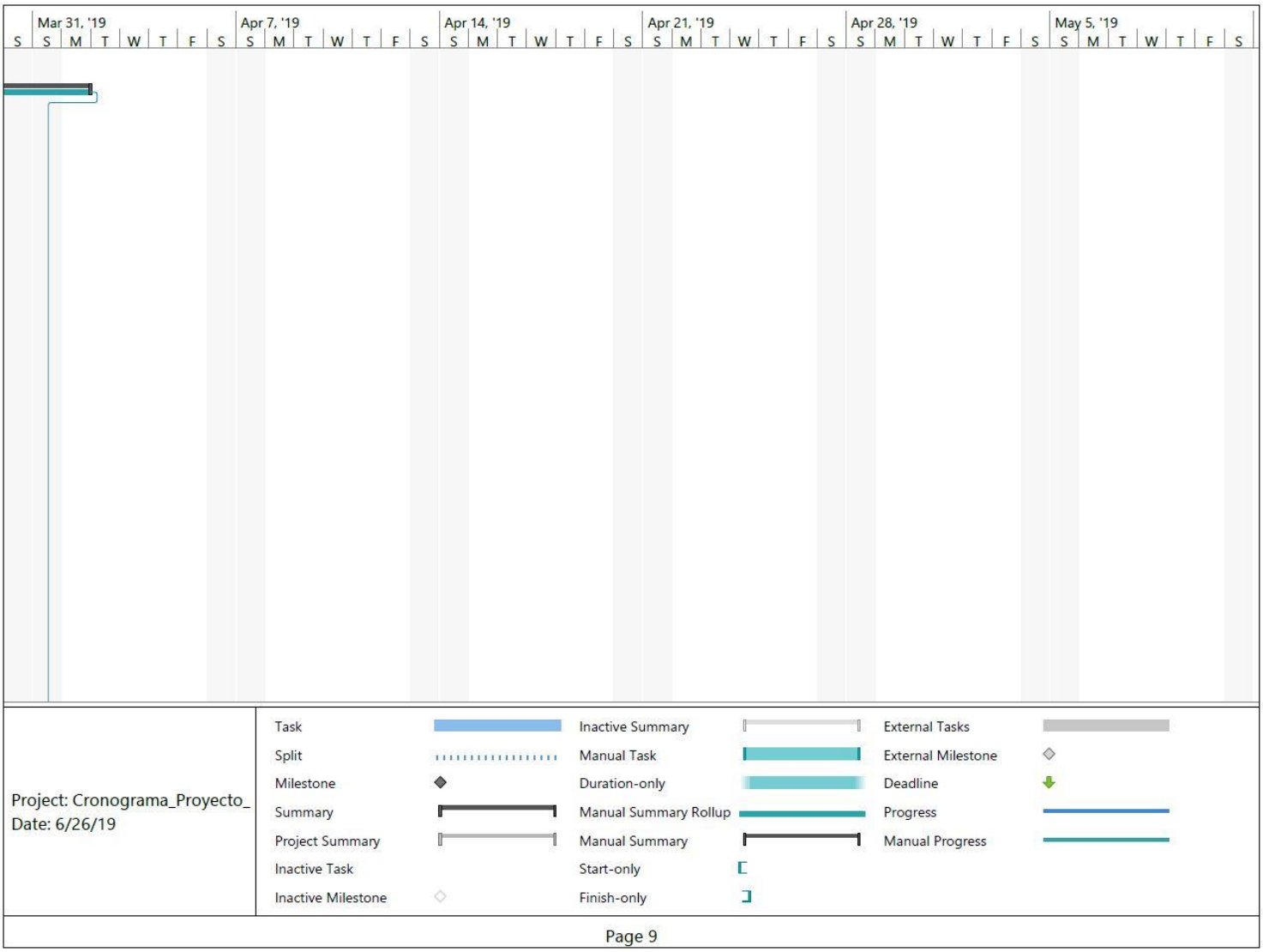


Ilustración 62. Detalle de cronograma, parte 7, elaboración propia.

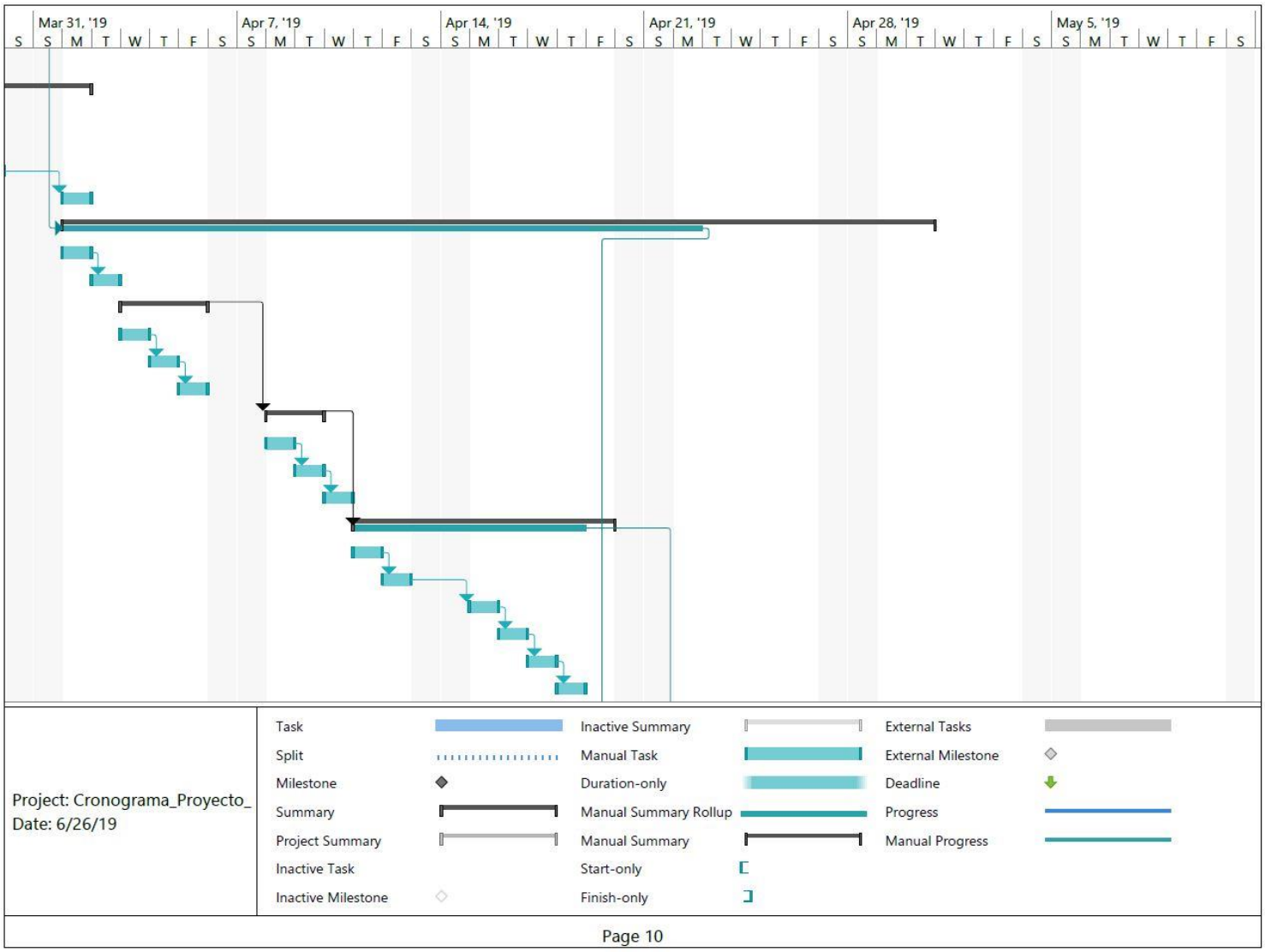


Ilustración 63. Detalle de cronograma, parte 8, elaboración propia.

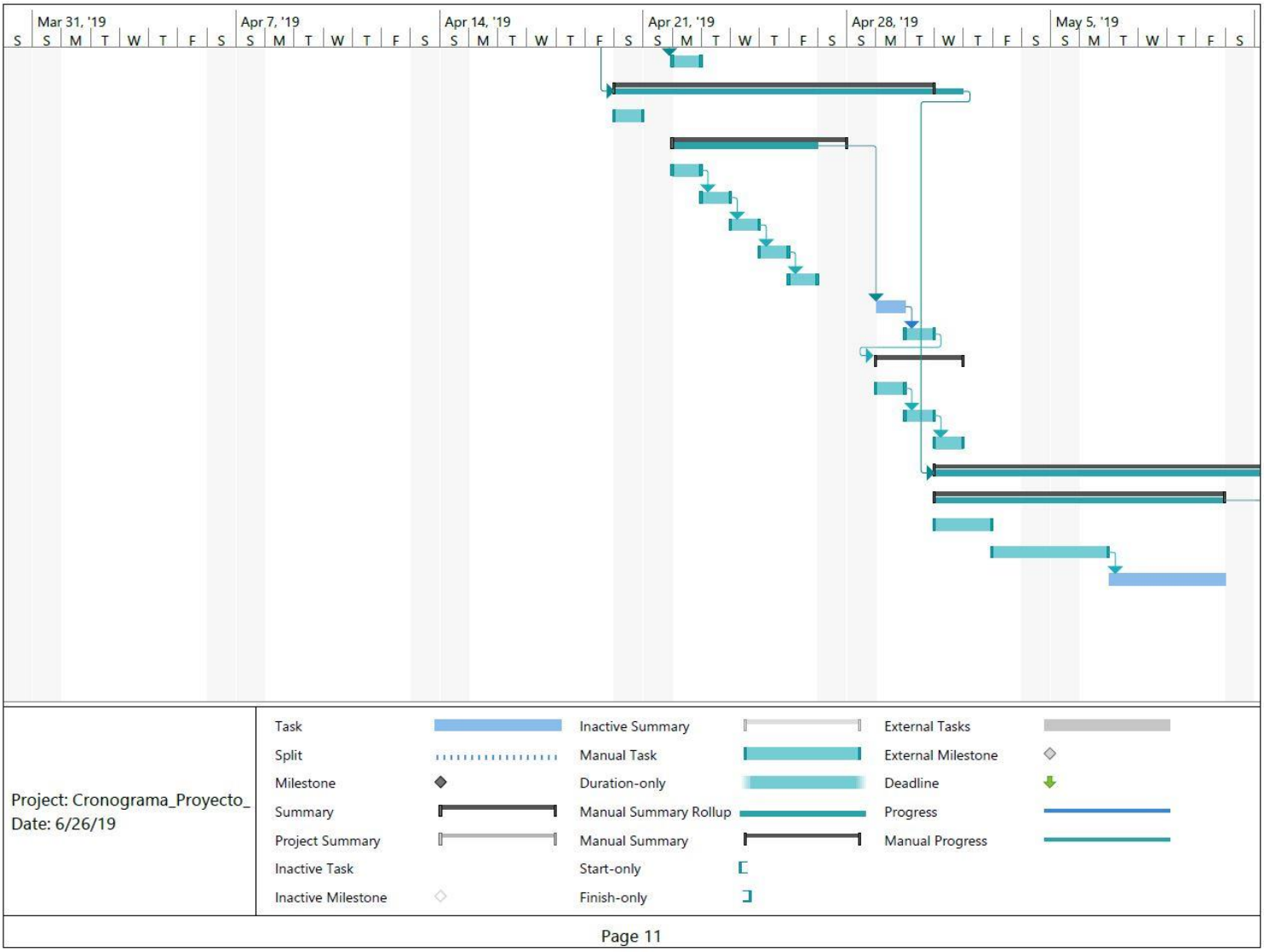


Ilustración 64. Detalle de cronograma, parte 9, elaboración propia.

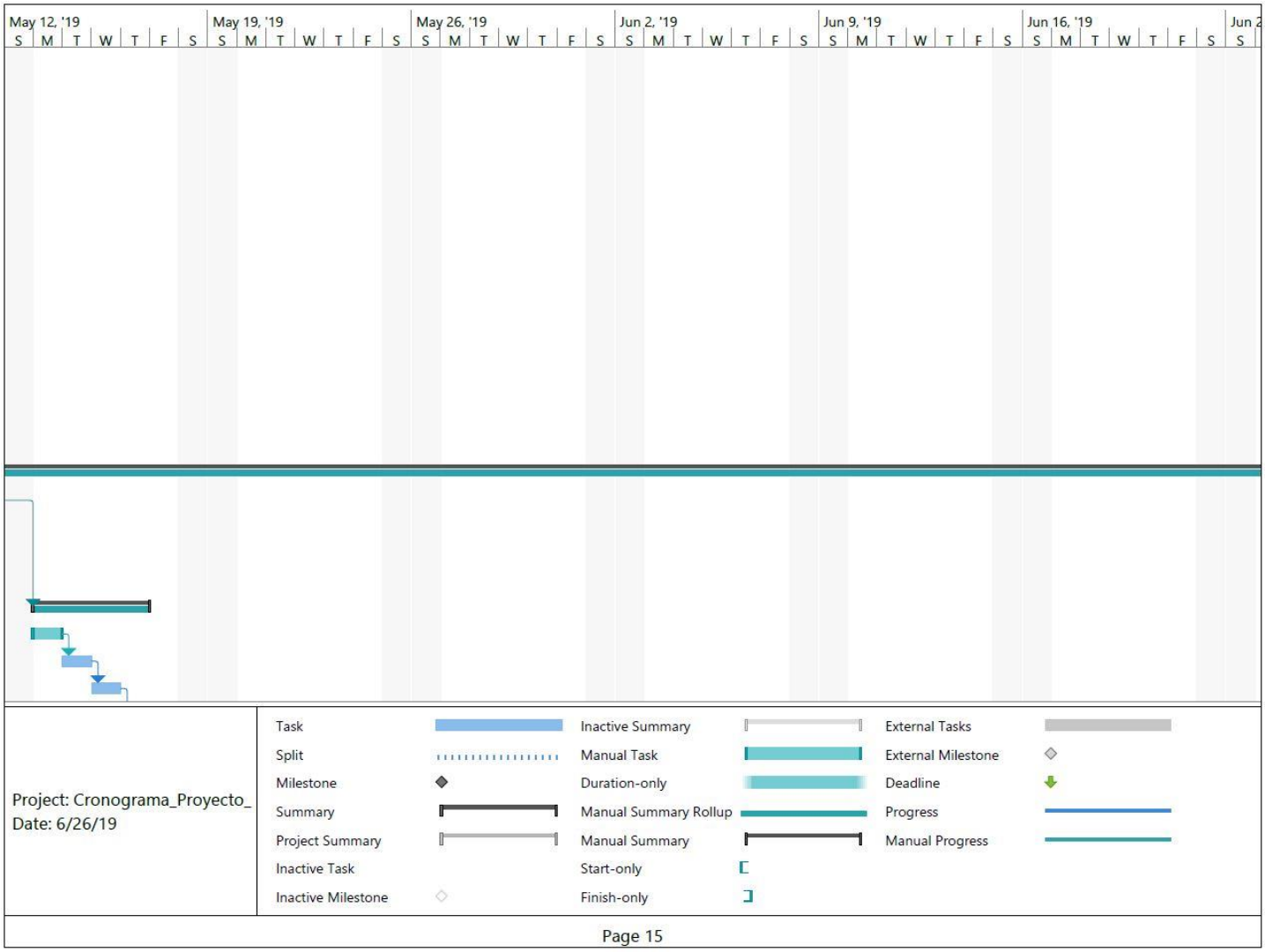


Ilustración 65. Detalle de cronograma, parte 10, elaboración propia.

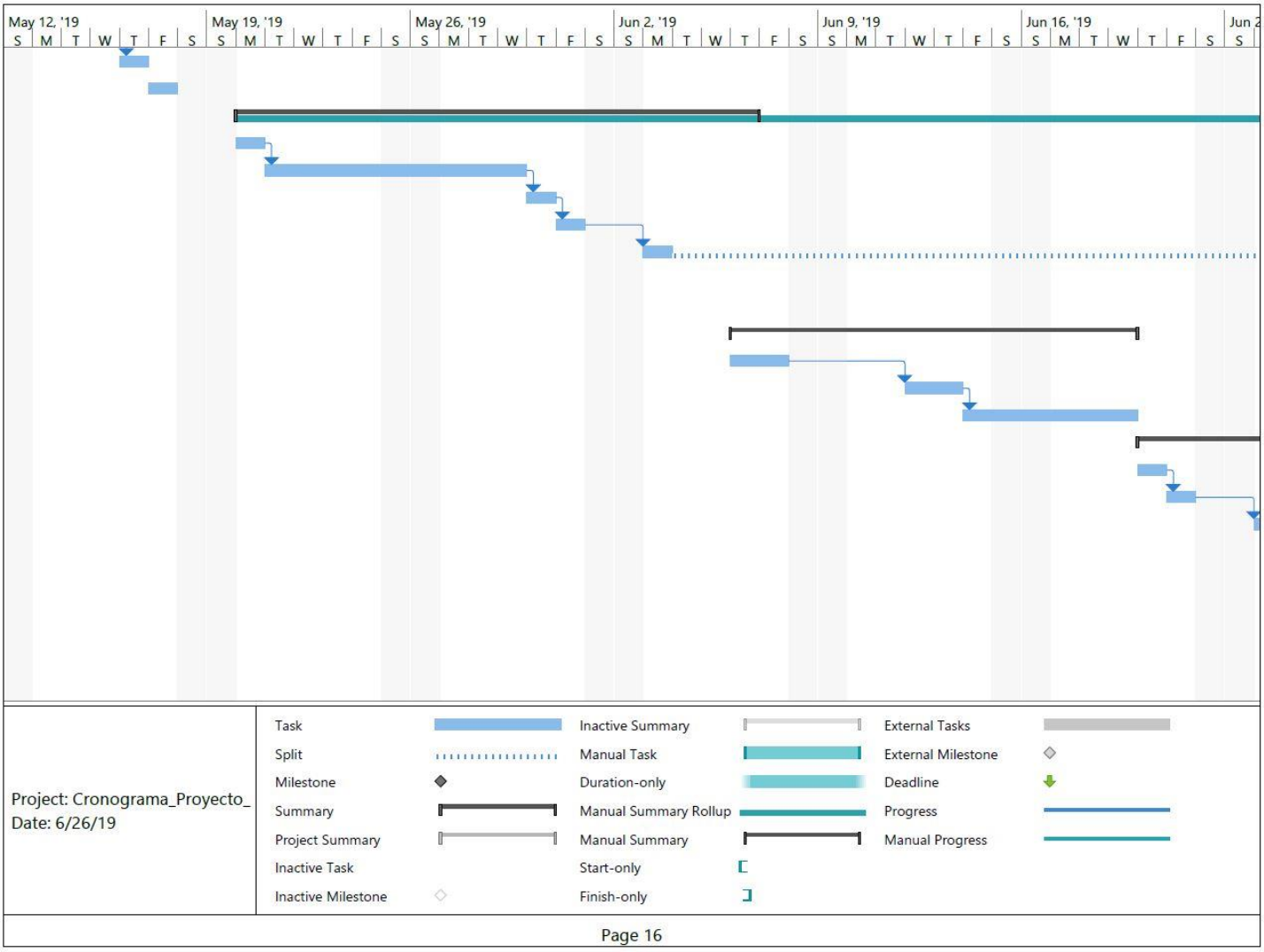


Ilustración 66. Detalle de cronograma, parte 11, elaboración propia.

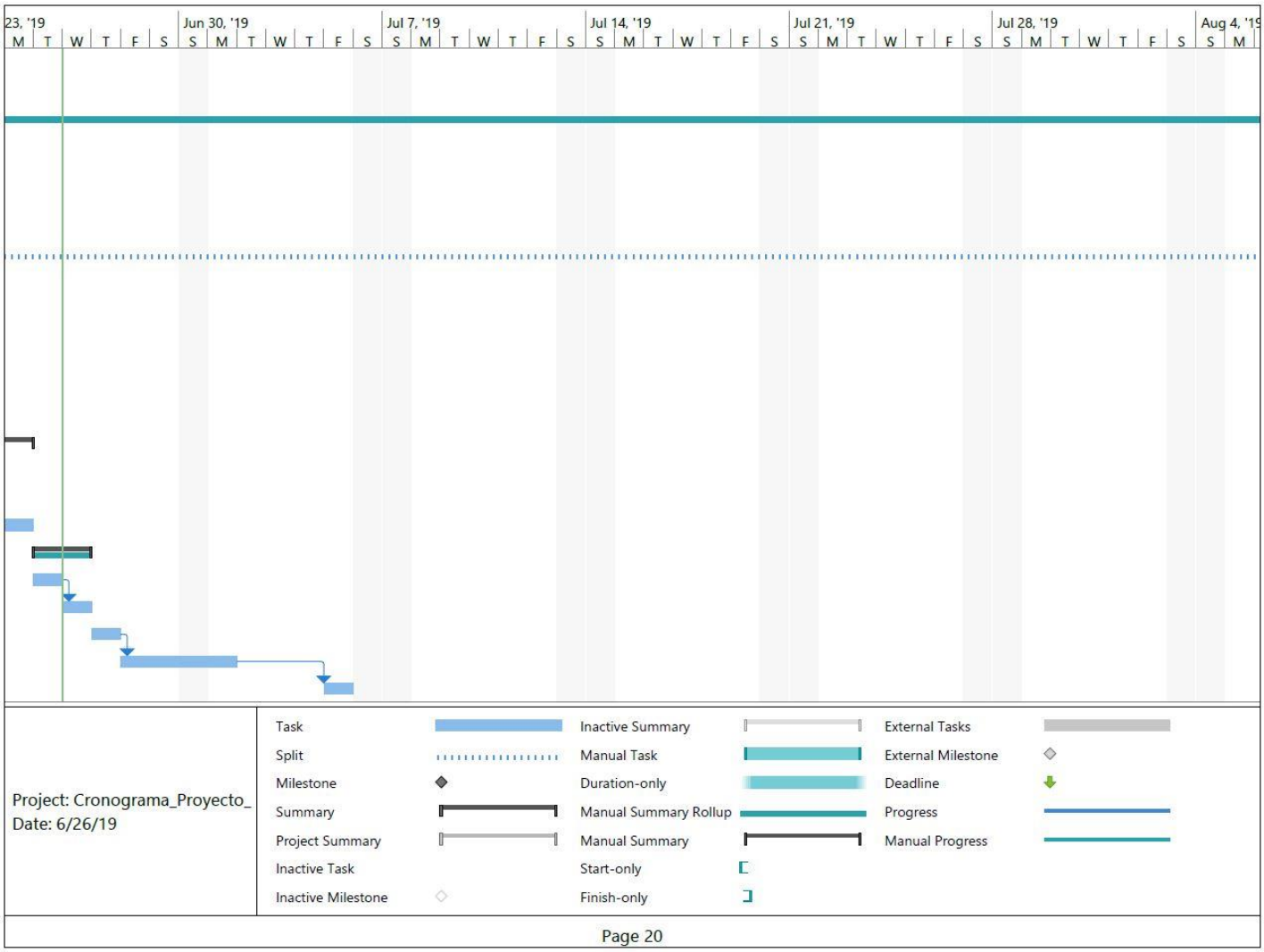


Ilustración 67. Detalle de cronograma, parte 12, elaboración propia.

Anexo 2: Detalle de comparación para métricas de rendimiento.

Tabla 54. Comparación de métricas, elaboración propia.

Métricas posibles según herramienta					
Criterios	EMON	PERF	Soc Watch	VTune	IMIX
Métricas de rendimiento					
Mezclas de instrucciones (ej. AVX)					X
Top-Down Análisis (TMAM)	X			X	
Instrucciones por segundo	X	X		X	
Misses por instrucción	X	X		X	
Instrucciones por segundo	X	X		X	
Frecuencia del CPU	X	X	X	X	
Ciclos por instrucción	X	X		X	
Utilización del CPU (%)	X	X	X	X	
E/S	X	X		X	
Recursos del GPU	X	X	X	X	
Temperatura y energía					
Métricas de Temperatura		X	X	X	
Métricas sobre el manejo de energía			X	X	
Tipo de licencia					
Software libre		X			
Privada	X		X	X	X
Técnica o empresarial	X			X	
Estudiante				X	
Educador				X	
Contribuyentes de software libre				X	
Community (versión simplificada)					X
Sistemas operativos para los que se encuentra disponible					
Windows	X		X	X	X
Linux	X	X	X	X	X

MacOs	X		X		X
Android				X	
Tipo de interfaz					
Gráfica		X		X	
Línea de comando	X	X	X	X	X
Capacidad de arrojar resultados en tiempo real					
Sí	X	X	X	X	
No					X

Anexo 3: Unidades del procesador y su relación con métricas de rendimiento.

Tabla 55. Cruce entre unidades funcionales y métricas asociadas, elaboración propia.

Procesador	
Métricas para diagnóstico general	
IPC, ILP, Ciclos en los que ambos hilos se encuentran activos, *Retiring*, Frecuencia de CPU (tiempo activo e inactivo)	
Frontend	
Unidad funcional	Métricas asociadas
Unidad de carga de instrucciones (IFU)	Especulación equivocada, Predicciones equivocadas,
ICache	Icache *Misses*, ITLB *Misses*
Unidad de decodificación de instrucciones (IDU)	*Resteers* de ramificación, DSB Switches
Unidad de carga y almacenamiento de instrucciones (LSU)	Ancho de banda de interfaz
Unidad de secuenciación de instrucciones (ISU)	MS Switches, Secuenciador de Microcódigo (MS)
Backend	
Caché de datos- L2 (Data-L2)	Límite L2
Caché de instrucciones- L2 (Instr-L2)	Límite de memoria
Cache L1 datos	Límite L1, Almacenamiento de DTLB
Unidad de punto fijo (FXU)	Operaciones aritméticas de punto fijo
COP	Divider
Uncore	
Puertos	Utilización de puertos
Caché - L3	Límite L3, Latencia de L3, Ancho de banda de L3

Anexo 4: Descripción de métricas de rendimiento seleccionadas para la elaboración del prototipo.

Tabla 56. Descripción de métricas de rendimiento para proyecto, elaboración propia.

Métrica	Descripción
Métricas para *cores* 1, 2 y caché L3	
A. Core Instrucciones por ciclo (IPC)	El promedio de instrucciones ejecutadas por ciclo de reloj.
B. Paralelismo a nivel de memoria (ILP)	La habilidad de un procesador de ejecutar más de una instrucción al mismo tiempo.
C. Ciclos en los que ambos hilos se encuentran activos	Ciclos en los que los dos hilos posibles del *core* que estuvieron activos.
D. Límite de interfaz	Fracción de tiempo en la que la interfaz, la parte encargada de la carga de instrucciones y el predictor de saltos, no es capaz de suplir al B*ackend*.
D.1. Latencia de interfaz	Fracción de tiempo en la que el CPU se bloqueó debido a latencias en la Interfaz como *caché misses*, bloqueos en la carga de datos o ITLB *misses*.
D.1.1. Icache *Misses*	La interfaz se bloquea cuando las instrucciones cargadas no están en la caché L1 de instrucciones.
D.1.2. ITLB *Misses*	Se da cuando el dato buscado no se encuentra en el *Instructions Take a Look Buffer* (ITLB).
D.1.3. *Resteers* de ramificación	Se refiere a las fracciones de ciclo en las que la CPU se bloqueó debido a una serie de predicciones de salto equivocadas.
D.1.4. DSB Switches	Es la penalización de tiempo que ocurre cuando un uOp pasa del *Decoded Stream Buffer* (DSB) hacia el *Microinstruction Translation Engine* (MITE).
D.1.5. MS Switches	Fracción de los ciclos en los que la CPU se bloqueó al entregar los uOps al *Microcode Sequencer* (MS).
D.2. Ancho de banda de interfaz	Fracción de los ciclos en los que la CPU se bloqueó por problemas de ancho de banda, como lo podrían ser ineficiencias en la decodificación de instrucciones o restricciones de código.

D.3. Especulación equivocada	Un slot de la segmentación desperdiciado debido a una mala predicción en las predicción de la siguiente instrucción
D.3.1. Predicciones equivocadas	Fracciones de tiempo perdidos debido a malas predicciones de salto.
D.3.2. Despojamiento de máquina.	Fracciones de tiempo que la CPU ha desperdiciado al vaciarse la cola de segmentación debido a la necesidad que tienen algunas instrucciones de que esté vacío para ejecutarse.
E. Límite de *Backend*	*Backend* es una parte del núcleo del procesador donde un *dispatcher* envía uOps listos a sus respectivas unidades de ejecución y, una vez completado, estos uOps se despachan de acuerdo con el orden del programa. Un límite de *Backend* radica en las fracciones de segundo en que no se entregan UOps debido a falta de otros recursos requeridos para hacerlo posible.
E.1. Límite de memoria	Bloqueos en la CPU debido a sobrecargas en los processo de carga de instrucciones.
E.1.1. Límite L1	Latencia para obtener un dato de la caché L1.
E.1.1.1. DTLB Load	Latencia al obtener un dato de DTBL.
E.1.1.2. Latencia por bloqueos	Fracciones de tiempo en que la CPU se bloqueó debido a *misses* de caché debido a operaciones bloqueadas.
E.1.2. Límite L2	Frecuencia en que la maquina estuvo bloqueada debido a L2 caché.
E.1.3. Límite L3	Fracciones de tiempo en las que la CPU estuvo bloqueada en L3 caché.
E.1.3.1. *Contested Accesses*	Ocurre cuando un dato que ha sido escrito en un *core* es necesitado en otro *core*.
E.1.3.3. Latencia de L3	Fracciones de tiempo en las que se dan L3 caché *hits*.
E.1.3.4. Ancho de banda de L3	Fracciones de tiempo en los que se dan latencias en la L3 caché.
E.1.3.5. SQ Full	La *Súper Cola* (SQ) es utilizada para pedir accesos a la L2 caché o *Uncore*. Esta medida indica las fracciones de tiempo en las que esta cola estuvo llena.
E.1.4. Límite de memoria	Latencias por ancho de banda de memoria.

E.1.4.1. Ancho de banda de memoria.	Ciclos en los que una aplicación estuvo detenida debido al casi agotamiento del ancho de banda de la DRAM
E.1.4.2. Latencia de memoria	Ciclos en los que una aplicación estuvo detenida debido al casi agotamiento del ancho de banda de la DRAM
E.1.5. Latencias por almacenamiento.	Limitaciones en el almacenamiento.
E.1.5.1. Almacenamiento de DTLB	Esta métrica representa una fracción de los ciclos gastados manejo de TLB L1 *misses*.
F. *Retiring*	Fracciones de tiempo en que los *uOps* fueron efectivamente retirados de la cola de segmentación.
F.1. Operaciones aritméticas de punto fijo	Tiempo invertido en suma, resta, multiplicación y división.
F.2. Secuenciador de Microcódigo (MS)	Fracción de tiempo en la que la CPU retiro u*Ops* cargados desde el Secuenciador de Microcódigo (MS)
G. Frecuencia de CPU.	Frecuencia del CPU
G.1. Frecuencia de CPU - tiempo activo	Fracciones de tiempo en las que la CPU estuvo activa.
G.2. Frecuencia de la CPU - tiempo inactivo	Fracciones de tiempo en las que la CPU estuvo inactiva.
Métricas para *uncore*	
I. Límite de *core*	Cuellos de botella fuera del *core* que en el c*ore* que no se deben a memoria.
I.1. *Divider*	Fracción de ciclos en que la unidad divisora para operaciones de punto flotante estaba activa.
I.2. Utilización de puertos	Fracción de ciclos durante los cuales una aplicación se detuvo debido a problemas que no se encuentran relacionados con el Divisor para operaciones de punto flotante.

Anexo 5: Información del sistema destino.

Tabla 57. Información sobre sistema destino y procesador, elaboración propia.

Información del sistema y especificaciones del procesador	
Información del sistema	
Sistema operativo	Ubuntu 18.04.2 LTS
Kernel	Linux 4.15.0-47-generic
Información del procesador.	
Arquitectura	x86_64
CPU op-modes	32-bit, 64-bit
Orden de los bits	Little Endian
CPUs	4
Hilos por núcleo	2
Core(s) per socket:	2
Sockets:	1
Nodos NUMA	1
Vendedor ID	GenuineIntel
Familia de CPU	6
Modelo:	78
Nombre del modelo	Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz
CPU MHz	500.018
CPU max MHz	3100
CPU min MHz	400
BogoMIPS	5184.00
Virtualization	VT-x
L1d cache	32K
L1i cache	32K
L2 caché	256K
L3 caché: 4096K	4096K
Información sobre los gráficos integrados	
Modelo	Intel HD Graphics 520
Frecuencia base	300MHz
Max. Frecuencia dinámica	1.05 GHz
Max. Frecuencia de memoria de video	32GB

Anexo 6: Descripción de las pruebas preliminares.

Tabla 58. Descripción de pruebas preliminares, elaboración propia.

Parámetros	**Descripción de pruebas**									
Información general										
Numero de prueba										0
Tamaño final de la prueba (MB):	15	16	16	16	16	17	16	15	18	16
Tiempo transcurrido(seg.):	61,705	62,002	61,512	61,755	61,978	63,519	61,931	61,446	64,167	61,614
Velocidad de ciclos por instrucción:	,121	,128	,124	,125	,124	,139	,126	,117	,154	,124
Pulsos de reloj	63,302,500,000	65,347,500,000	64,337,500,000	64,587,500,000	64,737,500,000	369,772,500,000	65,295,000,000	61,722,500,000	74,995,000,000	63,900,000,000
Numero de instrucciones retiradas	24,112,500,000	23,992,500,000	24,165,000,000	24,072,500,000	24,392,500,000	24,625,000,000	24,367,500,000	23,742,500,000	24,837,500,000	23,862,500,000
Inicio recolección:	5:48:31 23/05/2019 UTC	5:57:06 23/05/2019 UTC	6:04:49 23/05/2019 UTC	6:12:59 23/05/2019 UTC	6:20:13 23/05/2019 UTC	3/05/2019 UTC	6:34:29 23/05/2019 UTC	6:43:00 23/05/2019 UTC	6:50:59 23/05/2019 UTC	7:01:23 23/05/2019 UTC
Fin de recolección:	5:52:53 23/05/2019 UTC	6:01:28 23/05/2019 UTC	6:09:11 23/05/2019 UTC	6:17:21 23/05/2019 UTC	6:24:35 23/05/2019 UTC	6:31:33 23/05/2019 UTC	6:38:51 23/05/2019 UTC	6:47:22 23/05/2019 UTC	6:55:23 23/05/2019 UTC	7:05:44 23/05/2019 UTC
Frecuencia promedio CPU (GHz):	,98	,98	,98	,98	,98	,98	,98	,98	,98	,98
Cuenta total de hilos:	4	4	4	4	4	4	4	4	4	4
Utilización efectiva de	3,0%	3,2%	3,3%	3,2%	3,4%	3,6%	3,4%	3,1%	3,7%	3,2%

los cores físicos:										
Recolección por porcentajes										
Retiring:	**5,5%**	**5,5%**	**4,9%**	**4,9%**	**4,3%**	**3,2%**	**4,0%**	**5,6%**	**4,6%**	**5,0%**
Retirement General	7,8%	8,0%	7,4%	7,0%	7,1%	6,0%	6,6%	7,9%	7,3%	7,8%
FP Aritmético	,0%	,0%	,0%	,0%	,0%	,0%	,1%	,0%	,0%	,0%
FP x87:	,0%	,0%	,0%	,0%	,0%	,0%	,1%	,0%	,0%	,0%
FP Escalar:	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%
FP Vectorial:	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%
Otro	00,0%	00,0%	00,0%	00,0%	00,0%	00,0%	9,9%	00,0%	00,0%	00,0%
Secuenciador de microcódigo:	,7%	,5%	,5%	,9%	,2%	,2%	,4%	,6%	,2%	,2%
Assists:	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%
Límite de interfaz:	**2,5%**	**2,0%**	**1,9%**	**3,1%**	**2,3%**	**2,6%**	**1,4%**	**1,3%**	**6,7%**	**2,0%**
Latencia de interfaz:	0,2%	9,4%	9,3%	0,0%	9,5%	0,7%	9,9%	9,0%	3,9%	9,2%
ICache Misses:	,6%	,1%	,6%	,6%	,4%	,6%	,4%	,0%	,6%	,4%
ITLB Overhead:	,7%	,3%	,8%	,4%	,3%	,0%	,9%	,4%	,5%	,5%
Branch Resteers:	2,6%	2,5%	2,4%	3,5%	3,3%	3,6%	2,9%	2,7%	5,0%	2,5%
Mispredicts Resteers:	,8%	,6%	,7%	,3%	,5%	,9%	,8%	,9%	,9%	,5%
Clears Resteers:	,2%	,2%	,2%	,2%	,1%	,2%	,2%	0,0%	,2%	,1%
Unknown Branches:	,6%	,6%	,6%	,9%	,7%	,6%	,0%	,7%	0,9%	,9%
DSB Switches:	,4%	,4%	,6%	,4%	,4%	,3%	,4%	,0%	,3%	,4%
Length	,3%	,3%	,3%	,3%	,3%	,3%	,3%	,4%	,4%	,3%

Changing Prefixes:										
MS Switches:	,5%	,9%	,9%	,0%	,9%	,4%	,5%	,4%	,6%	,5%
Front-End Bandwidth:	2,3%	2,6%	2,6%	3,1%	2,9%	1,9%	1,5%	2,3%	2,8%	2,8%
Front-End Bandwidth MITE:	2,9%	0,3%	1,5%	2,5%	9,7%	9,2%	0,0%	0,5%	8,5%	0,1%
Front-End Bandwidth DSB:	,1%	,0%	,0%	,9%	,3%	,8%	,4%	,9%	,8%	,1%
Front-End Bandwidth LSD:	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%
(Info) DSB Coverage:	,1%	,0%	,7%	,4%	,4%	,4%	,1%	,7%	,9%	,2%
(Info) LSD Coverage:	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%
Especulación fallida:	**,5%**	**,3%**	**,4%**	**,4%**	**,9%**	**,7%**	**,1%**	**,8%**	**,7%**	**,9%**
Branch Mispredict:	,3%	,1%	,2%	,2%	,8%	,5%	,0%	,7%	,5%	,8%
Machine Clears:	,2%	,2%	,2%	,2%	,1%	,2%	,2%	,1%	,2%	,1%
Límite de Back-End:	**8,5%**	**9,3%**	**9,8%**	**8,6%**	**9,5%**	**0,5%**	**1,5%**	**0,4%**	**4,0%**	**9,1%**
Límite de Memoria:	0,1%	0,5%	0,7%	0,0%	0,5%	1,0%	1,7%	1,1%	,4%	0,5%
L1 Bound:	7,3%	7,1%	6,2%	6,9%	7,1%	6,9%	6,9%	6,9%	5,9%	7,3%
DTLB Overhead:	9,6%	9,8%	8,7%	2,4%	0,3%	8,8%	9,3%	7,8%	0,3%	8,6%
Loads Blocked by Store Forwarding:	,6%	,6%	,5%	,5%	,4%	,5%	,5%	,5%	,4%	,4%
Lock Latency:	,3%	,3%	,3%	,3%	,3%	,3%	,3%	,3%	,2%	,3%

Split Loads:	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%
4K Aliasing: 0.1%	,1%	,1%	,1%	,1%	,0%	,1%	,1%	,1%	,0%	,0%
FB Full:	,0%	,0%	,0%	,6%	,3%	,0%	,0%	,0%	,0%	,0%
Límite de L2:	,0%	,4%	,2%	,0%	,6%	,6%	,8%	,8%	,5%	,0%
Límite de L3:	,8%	,3%	,2%	,7%	,3%	,4%	,4%	,6%	,4%	,7%
Accesos contestados:	2,3%	2,7%	2,6%	2,0%	1,7%	1,2%	1,8%	2,3%	0,0%	,8%
Data Sharing:	,0%	,8%	,1%	,8%	,4%	,6%	,7%	,9%	,1%	1,8%
L3 Latencia:	,6%	,9%	,9%	,5%	,0%	,0%	,0%	,7%	,1%	,9%
SQ Full:	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,9%
DRAM Bound:	,9%	,7%	,8%	,7%	,8%	,9%	,7%	,8%	,6%	,0%
Límite de memoria:	,7%	,7%	,1%	,0%	,3%	,0%	,7%	,0%	,5%	,9%
Latencia de memoria:	7,9%	6,7%	7,8%	7,8%	6,5%	6,1%	5,6%	8,7%	5,3%	9,0%
LLC Miss:	,1%	,2%	,8%	,0%	,1%	,4%	,3%	,2%	,7%	,7%
Límite de almacenamiento:	,1%	,0%	,0%	,9%	,8%	,9%	,0%	,8%	,7%	,8%
Latencia de almacenaiento:	3,3%	3,1%	3,5%	2,9%	2,2%	2,9%	2,4%	1,5%	,9%	1,4%
False Sharing:	,9%	,0%	,9%	,2%	,3%	,8%	,8%	,3%	,2%	,9%
Split Stores:	,1%	,1%	,1%	,1%	,1%	,1%	,1%	,1%	,1%	,1%
DTLB Sobrecarga de almacenamiento:	,1%	,5%	,9%	,2%	,8%	,8%	,3%	,2%	,6%	,8%
Límite de Core:	,4%	,8%	,1%	,6%	,0%	,4%	,8%	,3%	,6%	,6%
Divider:	,3%	,4%	,3%	,5%	,4%	,3%	,4%	,3%	,3%	,5%

Utilización de puertos:	4,3%	3,7%	4,2%	4,0%	3,5%	3,7%	3,3%	3,3%	3,4%	3,1%
Ciclos en los que se utilizaron 0 puertos:	7,3%	6,3%	7,2%	6,4%	5,1%	5,4%	5,8%	5,8%	5,3%	5,8%
Operaciones de serialización:	0,4%	0,4%	0,6%	0,4%	9,9%	0,0%	0,2%	9,9%	8,9%	0,1%
Slow Pause:	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%	,0%
Ciclos en los que se utilizó 1 puerto:	3,7%	4,6%	4,1%	4,0%	4,7%	3,6%	3,9%	3,9%	5,9%	4,8%
Ciclos en los que se utilizaron 2 puertos:	6,5%	7,0%	6,4%	6,8%	6,2%	6,8%	6,5%	6,6%	6,9%	6,7%
Ciclos en los que se utilizaron 3+ Puertos:	2,1%	0,3%	0,6%	0,9%	0,1%	9,5%	0,1%	0,7%	8,4%	0,1%
Puerto 0:	2,7%	2,2%	2,0%	2,3%	0,9%	0,6%	1,2%	2,4%	9,9%	1,4%
Puerto 1:	2,3%	2,6%	2,0%	2,2%	1,4%	0,8%	1,7%	1,9%	0,9%	1,5%
Puerto 2:	0,3%	0,8%	9,9%	0,4%	9,1%	8,2%	8,8%	0,0%	9,1%	8,6%
Puerto 3:	2,3%	1,6%	1,5%	1,9%	0,4%	0,3%	0,4%	1,3%	1,0%	9,9%
Puerto 4:	2,3%	2,2%	1,4%	3,5%	1,4%	0,6%	0,7%	2,1%	1,9%	0,9%
Puerto 5:	2,2%	1,4%	1,6%	1,9%	1,2%	0,7%	0,6%	1,6%	1,7%	1,8%
Port 6:	3,5%	2,4%	2,8%	3,4%	2,2%	1,2%	1,9%	2,5%	3,8%	2,6%
Port 7:	7,0%	6,0%	6,5%	6,8%	5,9%	6,1%	6,1%	6,5%	5,8%	6,0%
Capacidad de utilización de vectores (FPU):	2,5%	2,5%	2,5%	,0%	2,5%	2,5%	2,5%	2,5%	2,5%	,0%
Somatización de totales	**00,0%**	**00,1%**	**00,0%**	**00,0%**	**00,0%**	**00,0%**	**00,0%**	**00,1%**	**00,0%**	**00,0%**

Anexo 7: Precondiciones y utilización del prototipo realizado.

Precondiciones para el uso.

En esta sección se describe las precondiciones y referencias que el usuario debería tomar en cuenta al decidir hacer uso del prototipo. Para utilizar el prototipo usted debe haber instalado en su computadora:

- Ubuntu 18.04.2. De no haberlo hecho, puede encontrar información útil en el siguiente vínculo: https://okdiario.com/howto/como-instalar-ubuntu-1876359
- Python 3.0. De no haberlo hecho, puede encontrar ayuda en el siguiente vínculo: https://www.python.org/downloads/
- Pycharm. Puede instalarlo desde el siguiente vínculo: https://www.jetbrains.com/pycharm/download/#section=linux
- VTune. Para la elaboración del proyecto se trabajó con la versión de *Estudiante.* Sin embargo, puede escoger la versión de su preferencia desde el siguiente vínculo: https://software.intel.com/en-us/vtune/choose-download

Además, debe asegurarse de haber instalado e importado e el proyecto las siguientes referencias a bibliotecas de Python.

- matplotlib
- numpy
- tkinter
- csv
- datetime

Utilización del prototipo.

En las líneas que siguen se muestra la forma en que un usuario realizaría los dos tipos de recolección de datos posibles.

A. Hacer pruebas en tiempo real.

Al inicializar el prototipo, se desplegará la siguiente ventana:

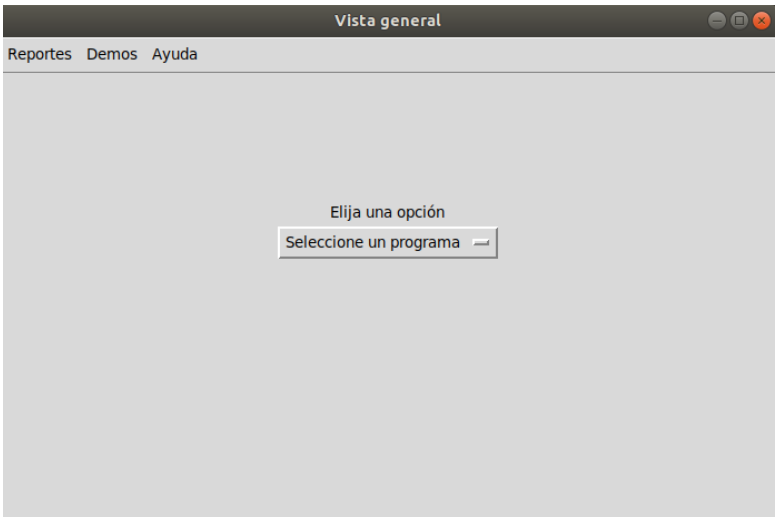


Ilustración 68. Vista general, elaboración propia.

Para ejecutar acciones sobre una aplicación de OpenOffice en tiempo real seleccione la aplicación de la lista que se encuentra en el centro de la pantalla:

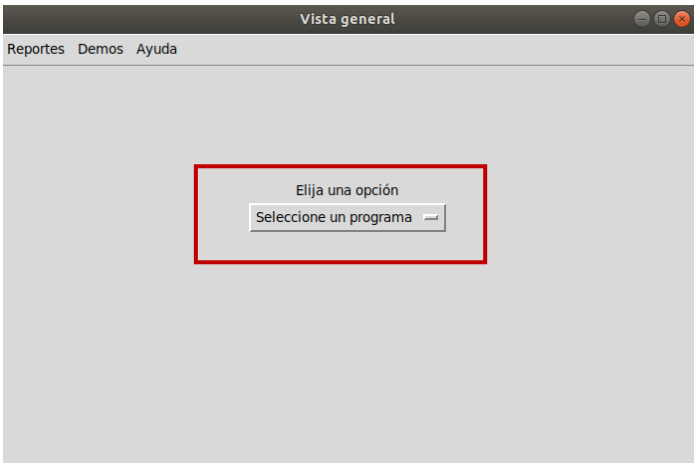


Ilustración 69. Selección de LibreOffice, elaboración propia.

Inmediatamente, el prototipo desplegará un mensaje. Para iniciar recolección pulse sobre el botón *Aceptar*, tal y como se muestra en la imagen. Si no desea que se ejecute la recolección seleccione la opción *Cancelar*:

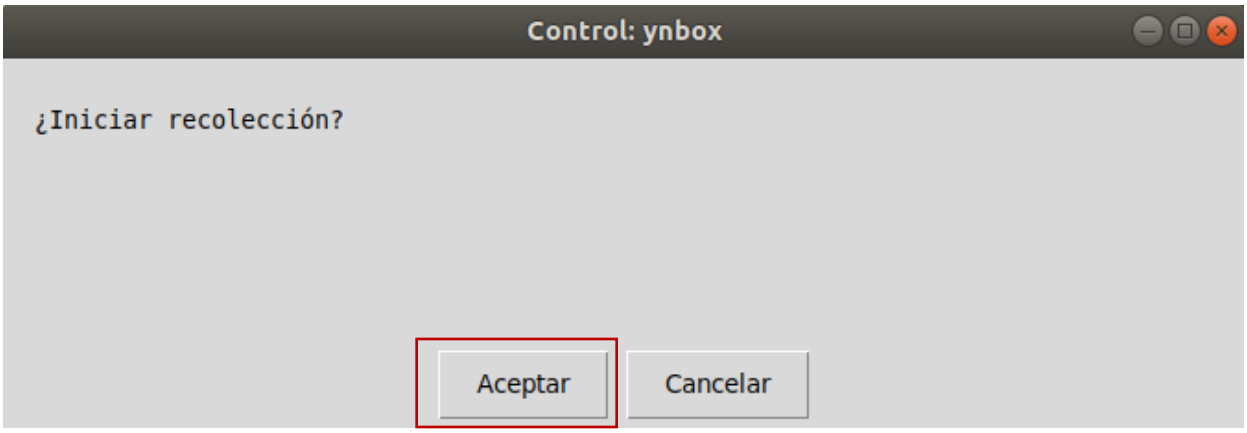


Ilustración 70. Iniciar recolección LibreOffice, elaboración propia.

Si pulsó la opción de *Cancelar*, el mensaje desaparecerá de la pantalla. De haber pulsado sobre la opción de *Aceptar*, el prototipo desplegará el siguiente mensaje:

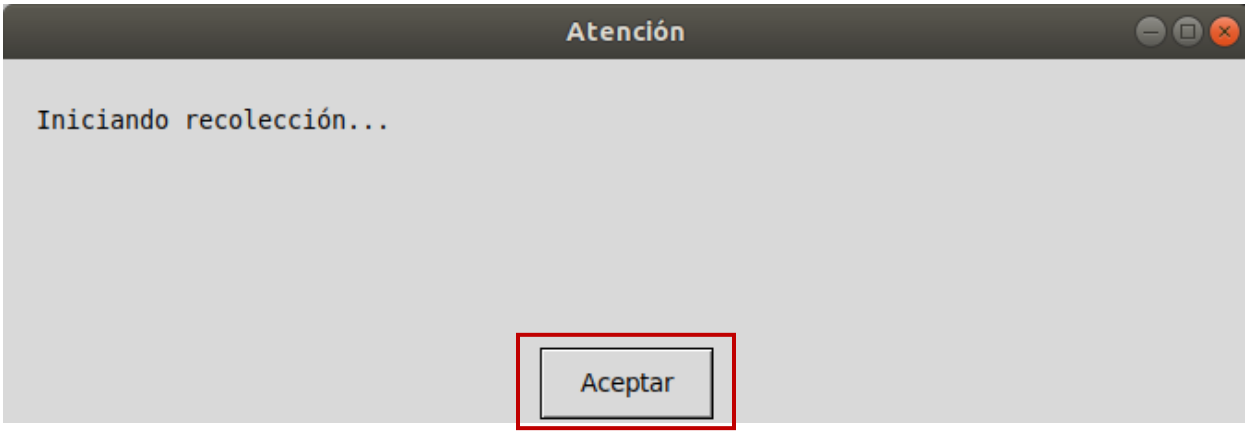


Ilustración 71. Iniciando recolección LibreOffice, elaboración propia.

Pulse sobre el botón de *Aceptar*. La recolección se dará por iniciada y verá desplegarse una pestaña de la aplicación seleccionada.

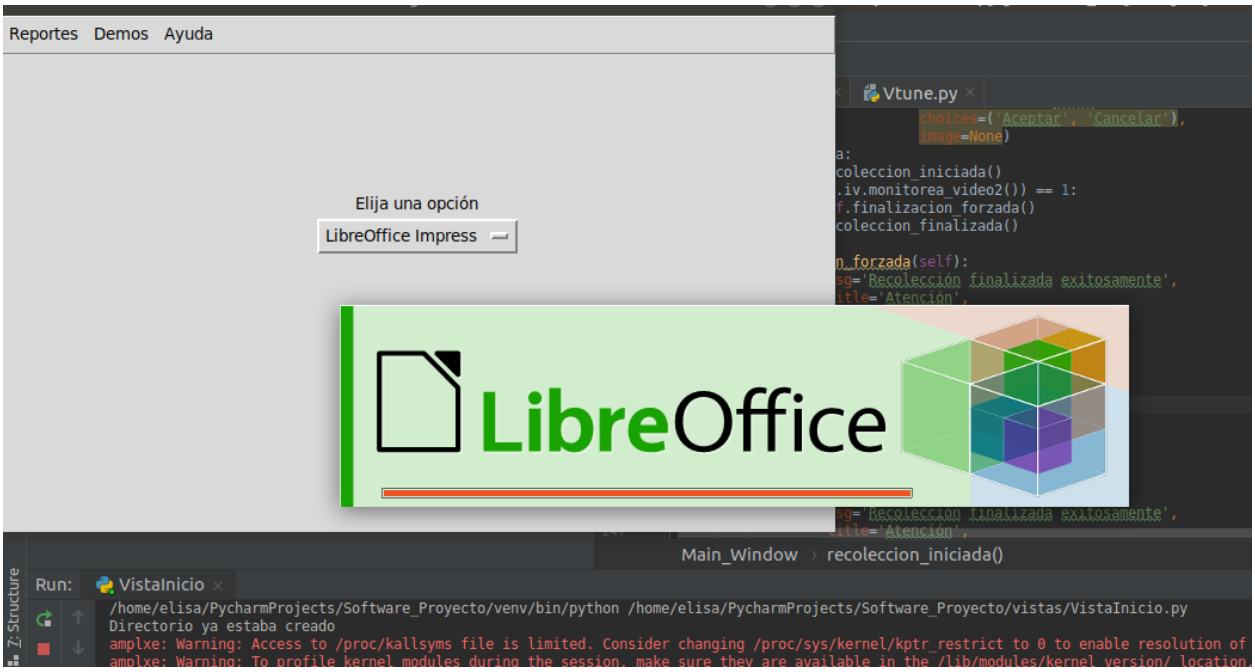


Ilustración 72. Pestaña nueva de LibreOffice, elaboración propia.

Espere a que la pestaña termine de cargar y realice cualquiera de las operaciones permitidas en la aplicación.

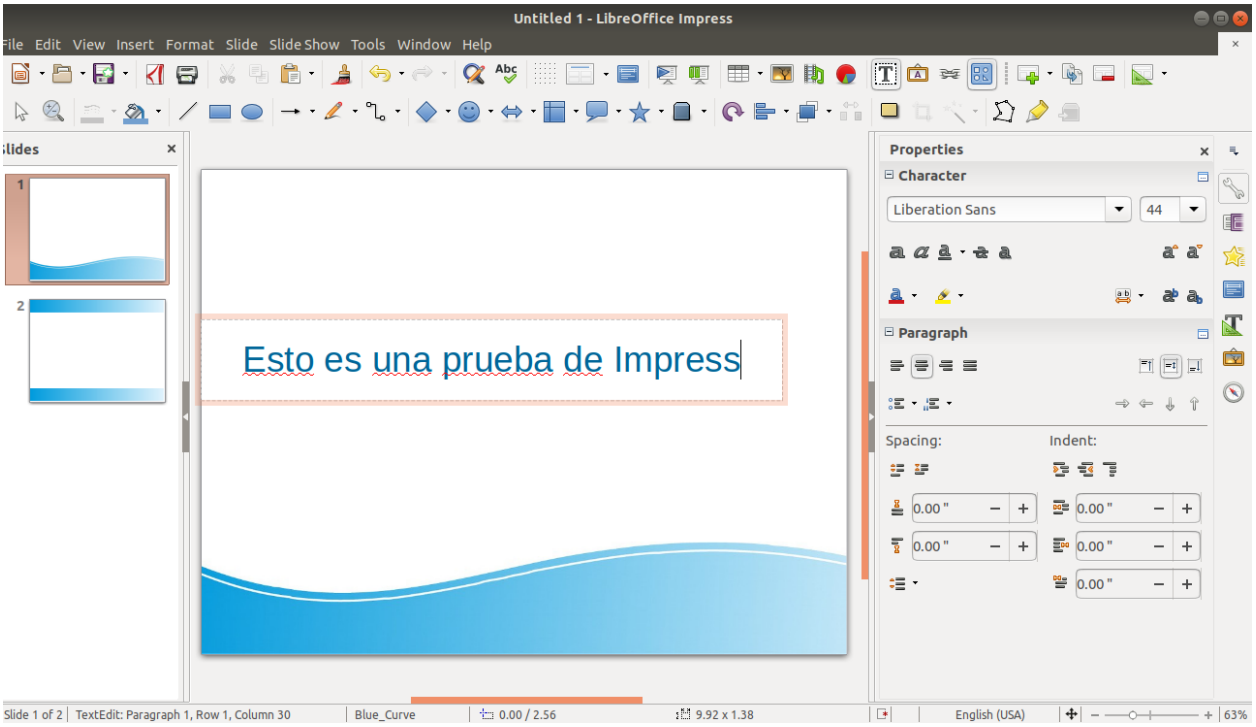


Ilustración 73. Ejemplo de acciones sobre LibreOffice, elaboración propia.

Cuando haya terminado, cierre la aplicación pulsando sobre la *X en la esquina derecha de aplicación.*

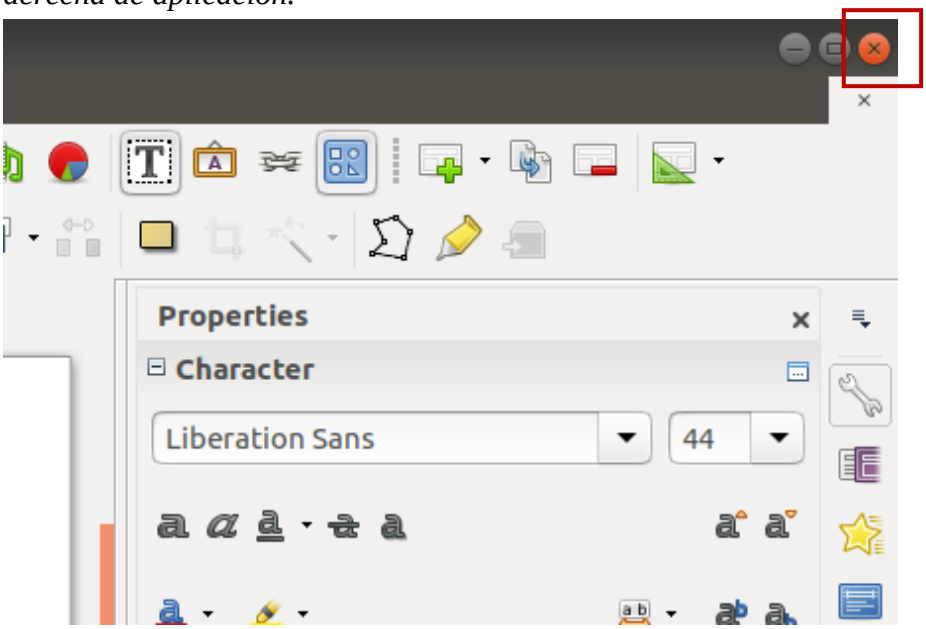


Ilustración 74. Finalización de sesión LibreOffice, elaboración propia.

Espere unos segundos hasta que se despliegue el siguiente mensaje en el prototipo:

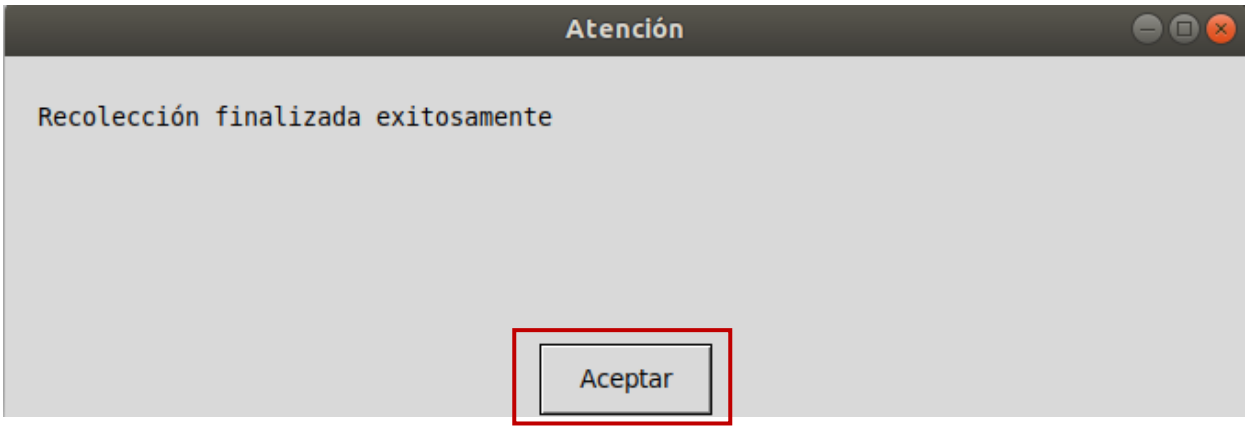


Ilustración 75. Mensaje de finalización de recolección en LibreOffice, elaboración propia.

Pulse sobre el boton de Aceptar.

Puede encontrar el resultado de las mediciones en la carpeta

Resultados.

Hacer recolección desde un demo.

Para hacer funcionar cualquiera de los demos, presione en el menú principal la opción de Demos y seleccione cualquiera de las opciones que se desplieguen.

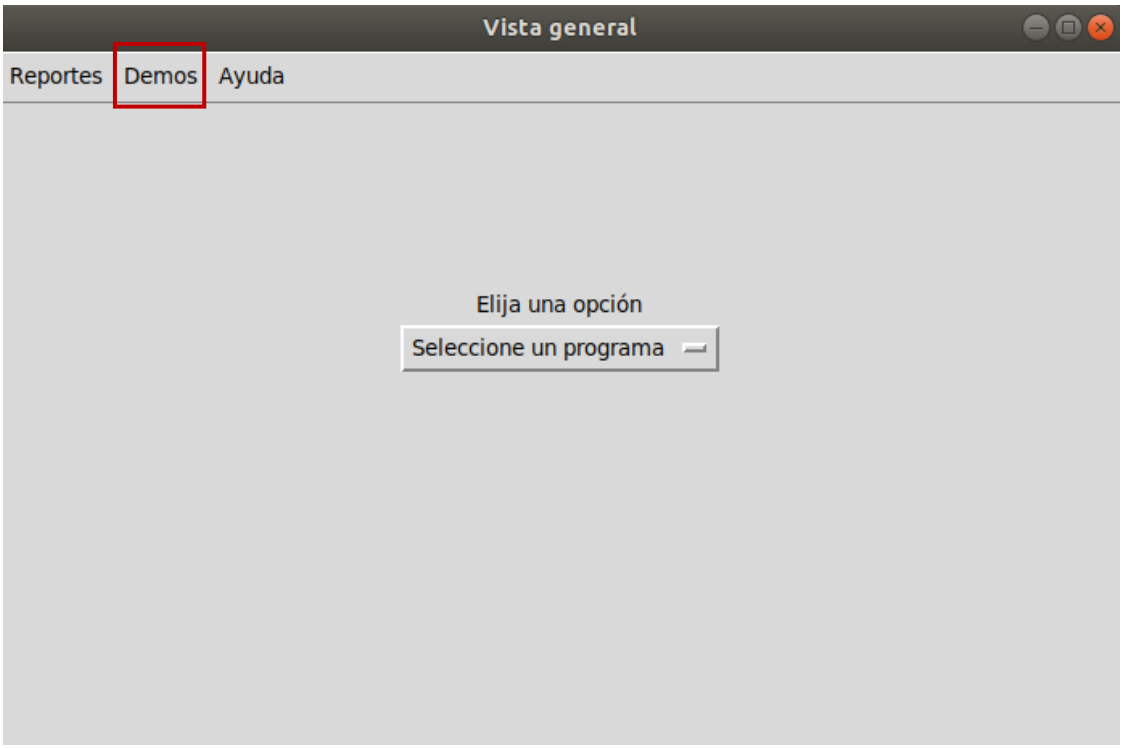


Ilustración 76. Inicio de recolección desde Demo, elaboración propia.

El prototipo desplegará el siguiente mensaje:

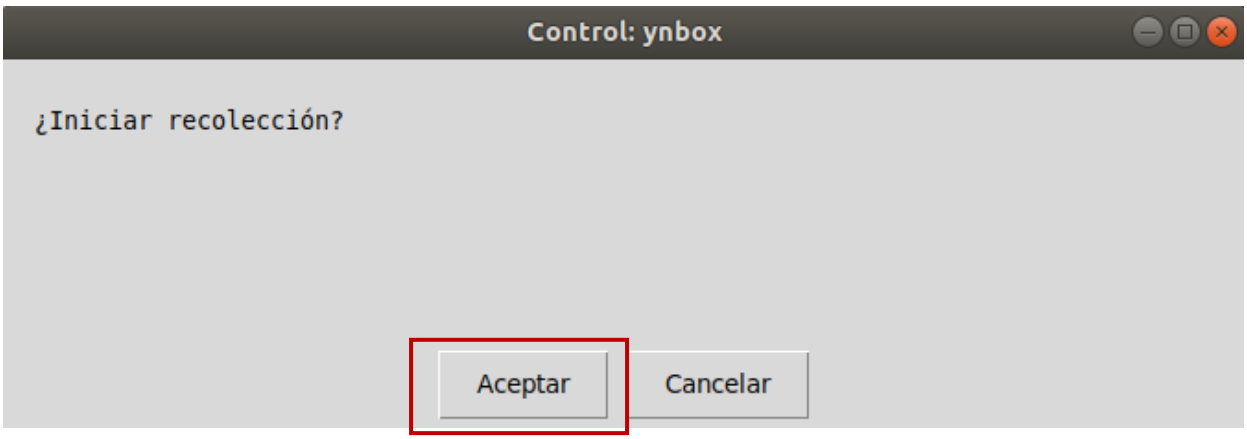


Ilustración 77. Validación de inicio de recolección desde Demo, elaboración propia.

Si pulsó la opción de *Cancelar*, el mensaje desaparecerá de la pantalla. De haber pulsado sobre la opción de *Aceptar*, el prototipo desplegará el siguiente mensaje:

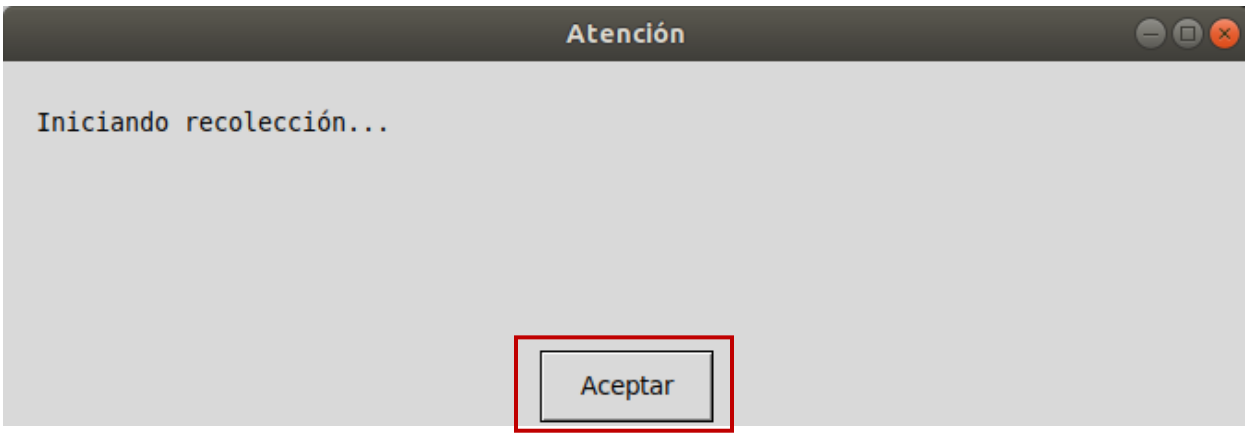


Ilustración 78. Iniciando recolección desde Demo, elaboración propia.

Pulse sobre el botón de *Aceptar*. La recolección se dará por iniciada y verá desplegarse el demo seleccionado. Al finalizar, el demo se cerrará automáticamente. Cuando se cierre, espere unos segundos hasta que aparezca el siguiente mensaje:

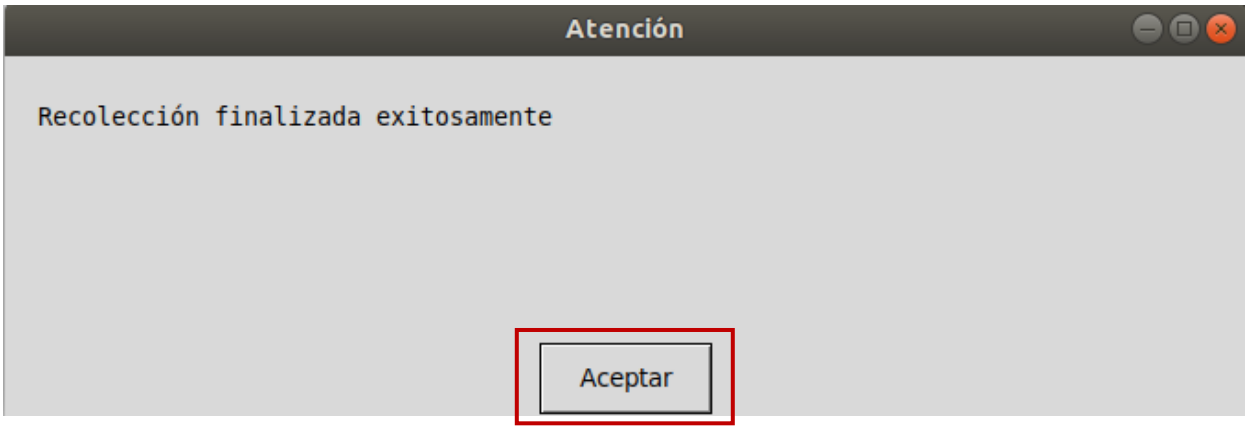


Ilustración 79. Finalización de recolección desde Demo, elaboración propia.

Pulse sobre el boton de Aceptar.

Puede encontrar el resultado de las mediciones en la carpeta

Resultados.

Desplegar las síntesis de los resultados.

Para ver los resultados de la última medición, vaya al menú principal y seleccione la opción de *Reportes*. Se desplegarán las opciones de síntesis.

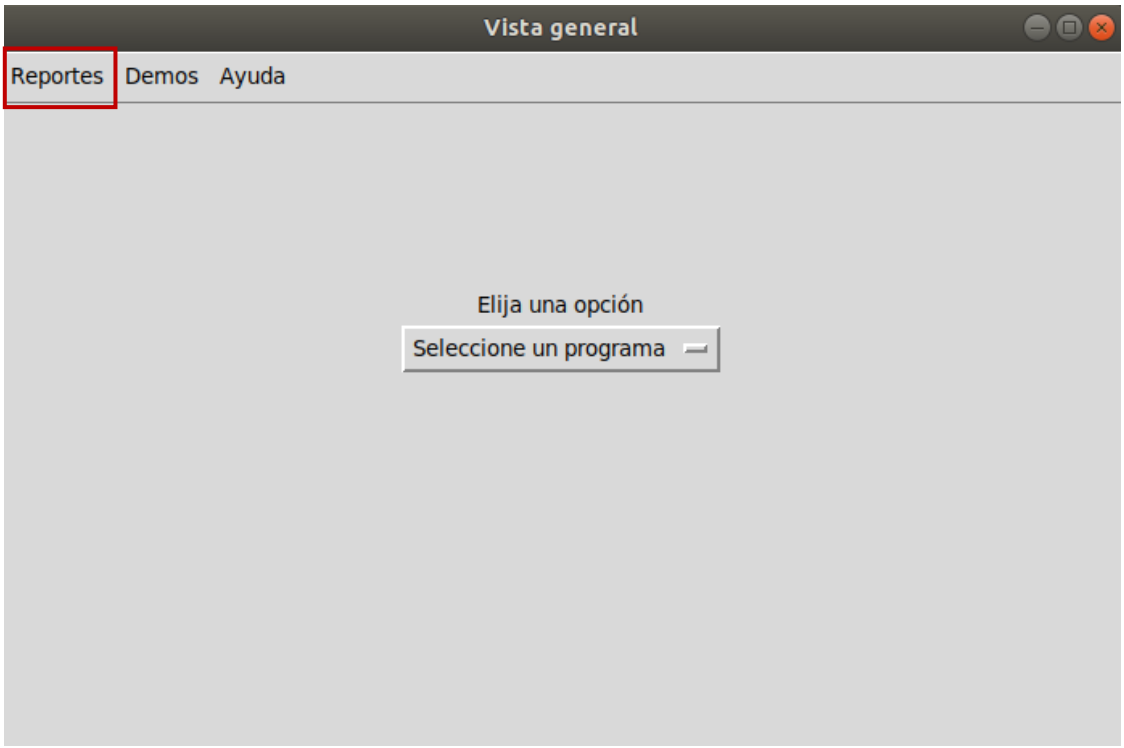


Ilustración 80. Selección de reportes, elaboración propia.

Seleccione la opción de su preferencia. La síntesis se desplegará en forma de grafico de barras.

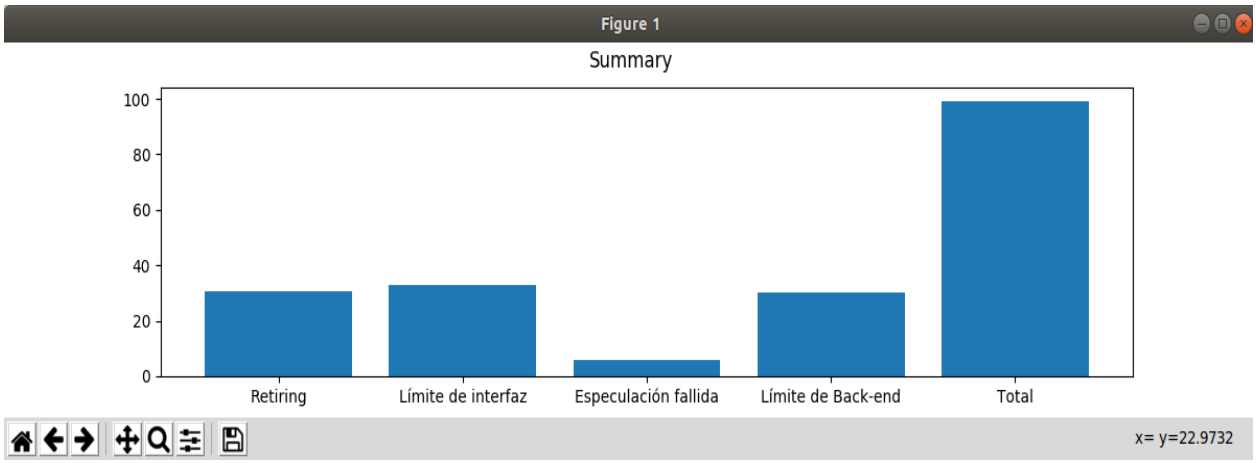


Ilustración 81. Ejemplo de gráfico de barras, elaboración propia.

Si desea guardar los resultados del reporte, pulse sobre el símbolo:  y seleccione el lugar de su preferencia en el sistema:

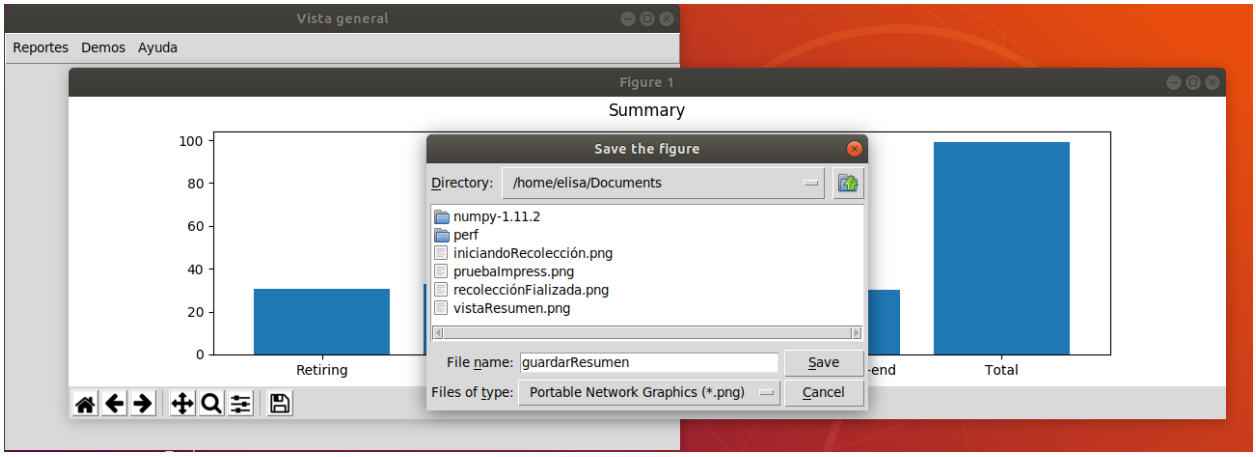


Ilustración 82. Guardar reporte, elaboración propia.

Obtención de ayuda.

Si desea conocer las especificaciones del sistema u obtener una guia rapida de uso. Pulse sobre la opcion de Ayuda y seleccione la opcion de su preferencia. La opcion seleccionada se desplegaraa en formato *PDF*.

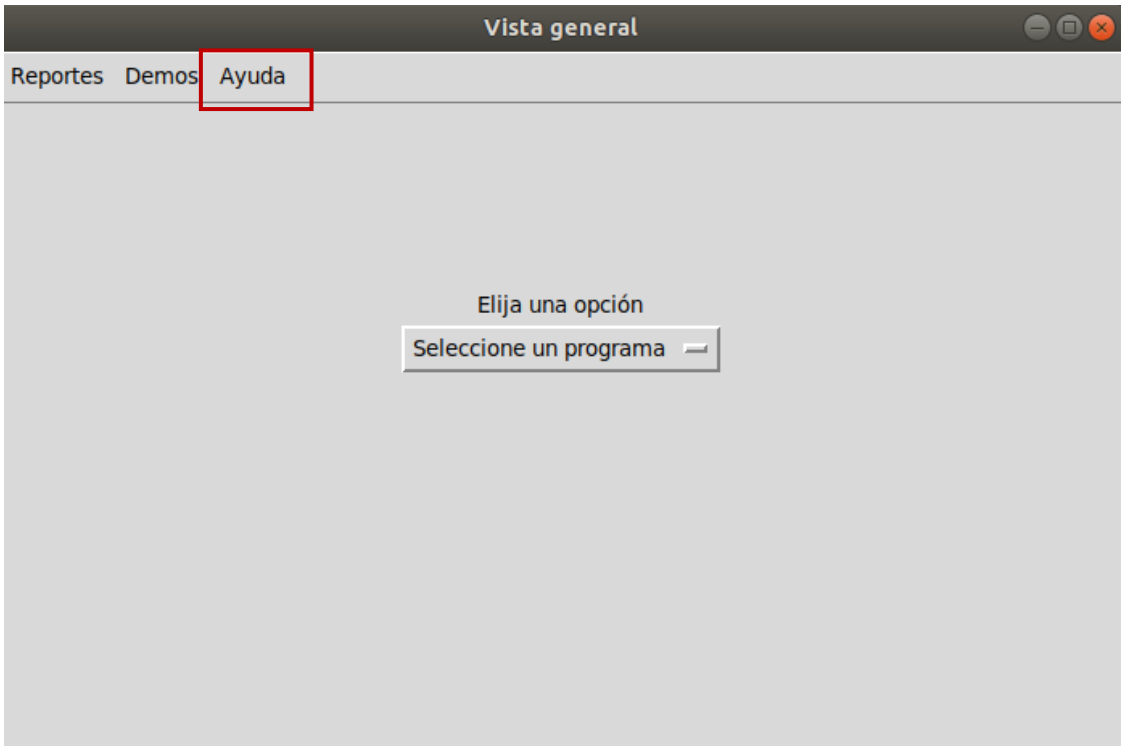


Ilustración 83. Pestaña de ayuda, elaboración propia.

Las siguientes dos capturas de pantalla muestran los dos tipos de informes que se mostrarán:

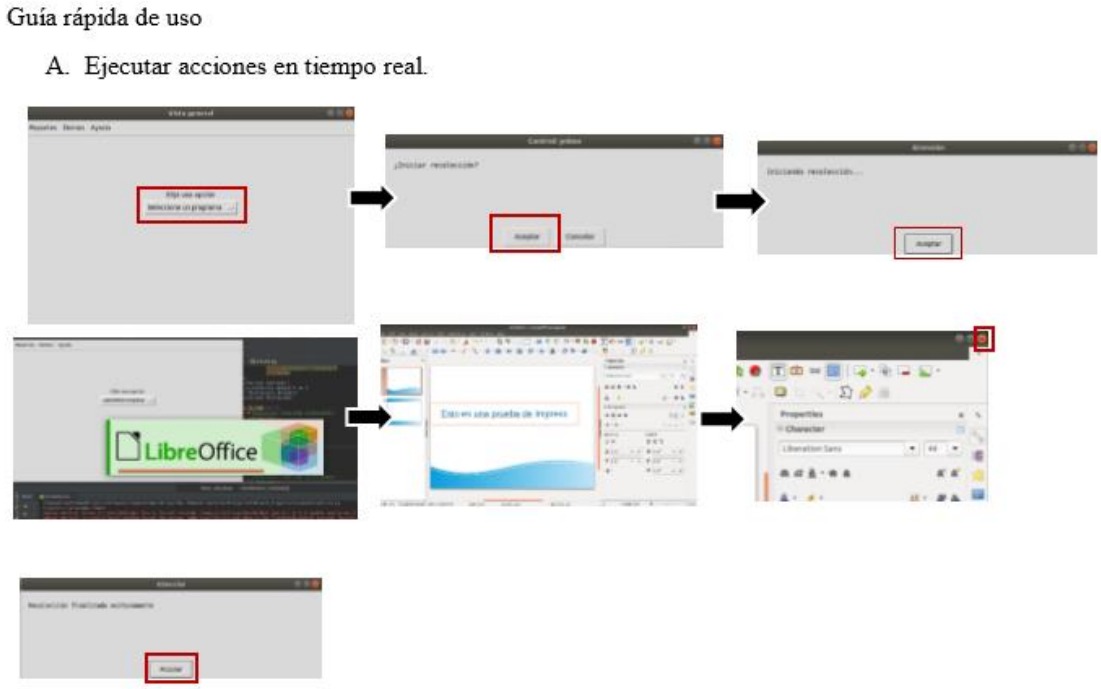


Ilustración 84. Guía rápida recolección en tiempo real, elaboración propia.

Ilustración 85. Guía rápida, recolección desde demo, elaboración propia.

C. Síntesis.

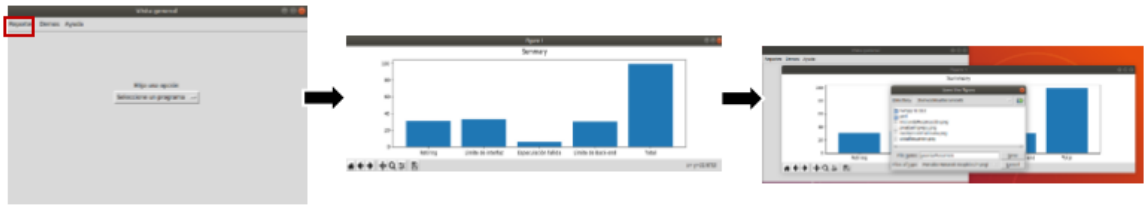
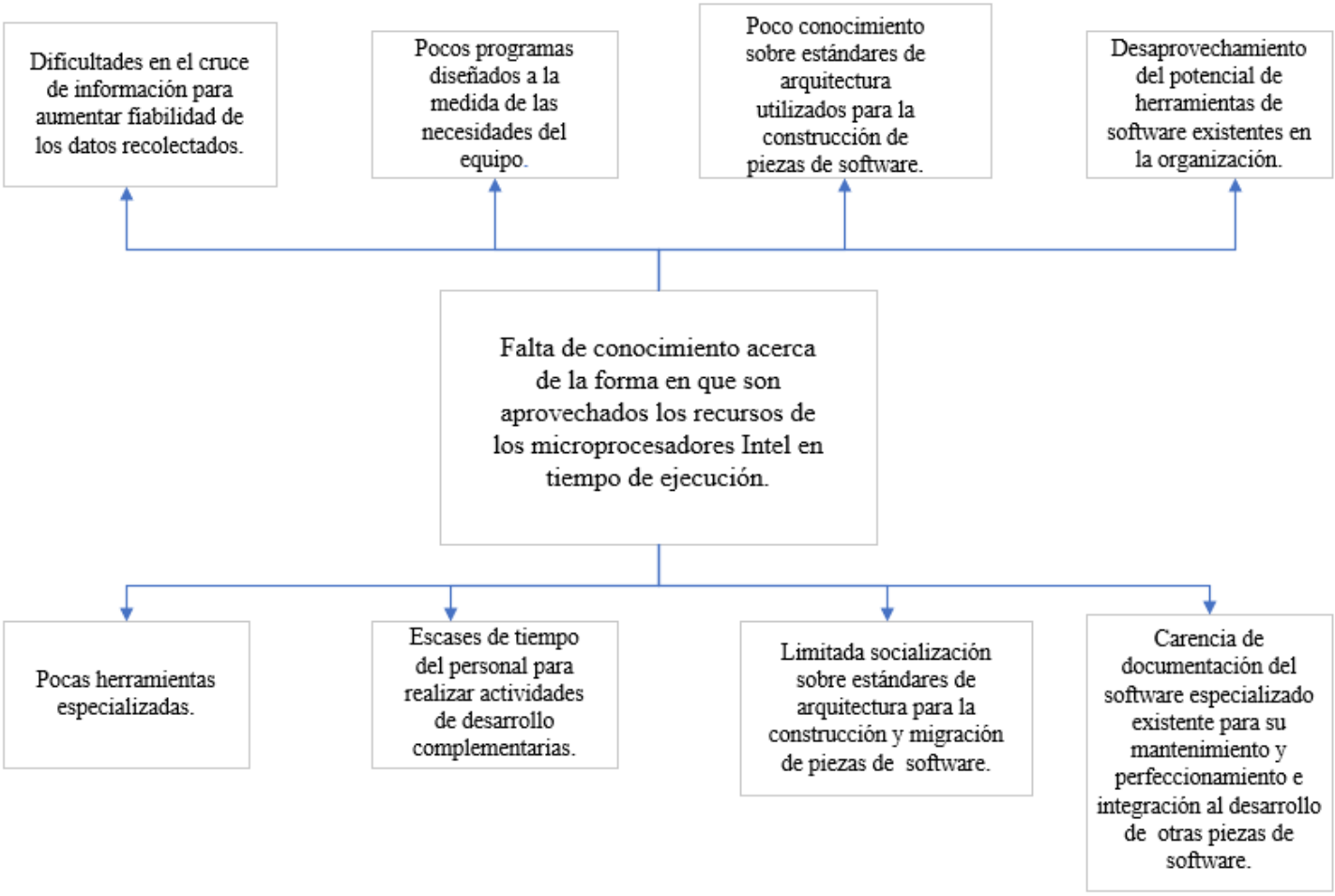


Ilustración 86. Guía rápida, vista de reportes, elaboración propia.

Anexo 8: Diagrama causa-efecto.



Anexo 9: Información sobre sistema destino.

Tabla 59. Información sobre carga de trabajo sintetica, elaboracion propia.

Información sobre carga de trabajo	
Carga de trabajo	Descripción
Video	Datos generales: Tamaño: 11.2 MB Formato de imagen: mp4 Video: Dimensiones: 1280x720 Codec: H:264 Calidad de la imagen: 30 f/s Audio: Codec: MPEG-4 AAC Canales: Estéreo Frecuencia de muestreo: 44100Hz
Operaciones	A continuación, se muestra el algoritmo utilizado para la reproducción de operaciones matemáticas: for x in rango (0, 50000000): self.can += 2+2+2+222+2+2+2- 76643+4576- 34567/6

Anexo 10: Pruebas de aceptación.

Tabla 60. Descripción de prueba realizada 1, elaboración propia.

Código de prueba:	P1
Requerimiento(s):	R1, R8, R9, R10
Objetivo:	Presentar especificaciones del sistema
Resultados esperados:	El prototipo despliega un pdf con la información del sistema destino.
Resultados obtenidos.	Prueba exitosa sí(X) no( )
Aprobado por:	Luis Rosales
Cargo:	Cliente
Fecha:	2-jul-19

Tabla 61. Descripción de prueba realizada 2, elaboración propia.

Código de prueba:	P2
Requerimiento(s):	R2, R3, R4, R5, R8, R10, R11
Objetivo:	Procesar información VTune 1
Resultados esperados:	En la carpeta “Resultados” del sistema destino, hay un nuevo archivo con la fecha completa en el siguiente formato: “dia-mes-año—hora:minutos:segundos” en formato *csv*
Resultados obtenidos.	Prueba exitosa sí(X) no( )
Aprobado por:	Luis Rosales
Cargo:	Cliente
Fecha:	2-jul-19

Tabla 62. Descripción de prueba realizada 3, elaboración propia.

Código de prueba:	P3
Requerimiento(s):	R2, R3, R4, R5, R8, R10, R11
Objetivo:	Procesar información VTune 2
Resultados esperados:	En la carpeta “Resultados” del sistema destino, hay un nuevo archivo con la fecha completa en el siguiente formato: “dia-mes-año—hora:minutos:segundos” en formato *csv*
Resultados obtenidos.	Prueba exitosa sí(X) no( )
Aprobado por:	Luis Rosales
Cargo:	Cliente
Fecha:	2-jul-19

Tabla 63. Descripción de prueba realizada 4, elaboración propia.

Código de prueba:	P4
Requerimiento(s):	R2, R3, R4, R5, R8, R10, R11
Objetivo:	Procesar información VTune 3
Resultados esperados:	En la carpeta “Resultados” del sistema destino, hay un nuevo archivo con la fecha completa en el siguiente formato: “dia-mes-año—hora:minutos:segundos” en formato *csv*
Resultados obtenidos.	Prueba exitosa sí(X) no( )
Aprobado por:	Luis Rosales
Cargo:	Cliente
Fecha:	2-jul-19

Tabla 64. Descripción de prueba realizada 5, elaboración propia.

Código de prueba:	P5
Requerimimento(s):	R2, R3, R4, R5, R8, R10, R11
Objetivo:	Procesar información VTune 4
Resultados esperados:	En la carpeta “Resultados” del sistema destino, hay un nuevo archivo con la fecha completa en el siguiente formato: “dia-mes-año—hora:minutos:segundos” en formato *csv*
Resultados obtenidos.	Prueba exitosa sí(X) no( )
Aprobado por:	Luis Rosales
Cargo:	Cliente
Fecha:	2-jul-19

Tabla 65. Descripción de prueba realizada 6, elaboración propia.

Código de prueba:	P6
Requerimiento(s):	R2, R3, R4, R5, R8, R10, R11
Objetivo:	Procesar información VTune 5
Resultados esperados:	En la carpeta “Resultados” del sistema destino, hay un nuevo archivo con la fecha completa en el siguiente formato: “dia-mes-año—hora:minutos:segundos” en formato *csv*
Resultados obtenidos.	Prueba exitosa sí(X) no( )
Aprobado por:	Luis Rosales
Cargo:	Cliente
Fecha:	2-jul-19

Tabla 66. Descripción de prueba realizada 7, elaboración propia.

Código de prueba:	P7
Requerimiento(s):	R2, R3, R4, R5, R8, R10, R11
Objetivo:	Procesar información VTune 6
Resultados esperados:	En la carpeta "Resultados" del sistema destino, hay un nuevo archivo con la fecha completa en el siguiente formato: "dia-mes-año—hora:minutos:segundos" en formato *csv*
Resultados obtenidos.	Prueba exitosa sí(X) no( )
Aprobado por:	Luis Rosales
Cargo:	Cliente
Fecha:	2-jul-19

Tabla 67. Descripción de prueba realizada 8, elaboración propia.

Código de prueba:	P8
Requerimiento(s):	R6, R8, R10
Objetivo:	Mostrar resumen.
Resultados esperados:	El prototipo despliega una pestaña con un resumen que contiene los valores porcentuales de: Retiring, Límite de interfaz, Especulación fallida y Límite de backend de la última prueba realizada.
Resultados obtenidos.	Prueba exitosa sí(X) no( )
Aprobado por:	Luis Rosales
Cargo:	Cliente
Fecha:	2-jul-19

Tabla 68. Descripción de prueba realizada 9, elaboración propia.

Código de prueba:	P9
Requerimiento(s):	R6, R8, R10
Objetivo:	Desplegar Retiring.
Resultados esperados:	El prototipo despliega una pestaña con los valores porcentuales de Retiring: Retirement General y Secuenciador de Microcódigo.
Resultados obtenidos.	Prueba exitosa sí(X) no( )
Aprobado por:	Luis Rosales
Cargo:	Cliente
Fecha:	2-jul-19

Tabla 69. Descripción de prueba realizada 10, elaboración propia.

Código de prueba:	P10
Requerimiento(s):	R6, R8, R10
Objetivo:	Desplegar Límite de interfaz.
Resultados esperados:	El prototipo despliega una pestaña con los valores porcentuales de Límite de interfaz: Latencia de interfaz y Ancho de banda de interfaz.
Resultados obtenidos.	Prueba exitosa sí(X) no( )
Aprobado por:	Luis Rosales
Cargo:	Cliente
Fecha:	2-jul-19

Tabla 70. Descripción de prueba realizada 11, elaboración propia.

Código de prueba:	P11
Requerimiento(s):	R6, R8, R10
Objetivo:	Mostrar Especulación fallida.
Resultados esperados:	El prototipo despliega una pestaña con los valores porcentuales de Especulación fallida: Predicciones de salto fallidas y Limpieza de segmentación.
Resultados obtenidos.	Prueba exitosa sí(X) no( )
Aprobado por:	Luis Rosales
Cargo:	Cliente
Fecha:	2-jul-19

Tabla 71. Descripción de prueba realizada 12, elaboración propia.

Código de prueba:	P12
Requerimiento(s):	R6, R8, R10
Objetivo:	Desplegar Límite de backend
Resultados esperados:	El prototipo despliega una pestaña con los valores porcentuales de Límite de backend: Límite de memoria y Límite de core.
Resultados obtenidos.	Prueba exitosa sí(X) no( )
Aprobado por:	Luis Rosales
Cargo:	Cliente
Fecha:	2-jul-19

Tabla 72. Descripción de prueba realizada 13, elaboración propia.

Código de prueba:	P13
Requerimiento(s):	R7, R8, R9, R10
Objetivo:	Guardar síntesis
Resultados esperados:	Un archivo con formato de imagen con la síntesis seleccionada se encuentra guardado en la ubicación del sistema destino deseada
Resultados obtenidos.	Prueba exitosa sí(X) no( )
Aprobado por:	Luis Rosales
Cargo:	Cliente
Fecha:	2-jul-19