Herramienta de simulación para dar soporte a la enseñanza de arquitectura de computadoras

Maestría en Sistemas de Información

Ruiz Jose Maria

Director: Colombani Marcelo Alberto

2024

Contents

List of Tables

List of Figures

Resumen

Resumen aquí

Agradecimientos

Agradecimientos aquí.

1

Introducción

En nuestra vida cotidiana, dependemos de diversos dispositivos, desde máquinas de trabajo hasta teléfonos y relojes inteligentes, que funcionan gracias a la arquitectura de las computadoras. Comprender los componentes y su funcionamiento nos permite diseñar, desarrollar e implementar aplicaciones más eficientes.

Los alumnos de la asignatura Arquitectura de Computadoras no solo deben conocer la estructura y el funcionamiento interno de la computadora, sino que, idealmente, deben tener una experiencia práctica activa con dicha arquitectura.

Para brindar esta experiencia, es fundamental contar con un laboratorio equipado con el hardware necesario y dedicar el tiempo suficiente para que los alumnos adquieran competencia en el uso de herramientas prácticas. Por esta razón, se han desarrollado numerosos simuladores que ayudan a los estudiantes a comprender el funcionamiento y la estructura de las computadoras, proporcionando valiosas experiencias de aprendizaje. La presente tesis se enmarca dentro en la carrera Maestría en Sistemas de Información dictada en la Facultad de Ciencias de la Administración, se vincula directamente con el proyecto de investigación I/D novel PID-UNER 7065: "Enseñanza/aprendizaje de asignatura Arquitectura de Computadoras con herramientas de simulación de sistemas de cómputos", llevado a cabo en la Facultad de Ciencias de la Administración de la Universidad Nacional de Entre Ríos 1.

La asignatura arquitectura de computadoras forma parte del plan de estudios de la carrera Licenciatura en Sistemas de la Facultad de Ciencias de la Administración, Universidad Nacional de Entre Ríos. El objetivo de la asignatura Arquitectura de Computadoras es conocer la estructura y funcionamiento de las computadoras, entender la ejecución lógica de un programa a nivel de instrucciones máquinas.

Como primer nivel de entendimiento los alumnos deben ser capaces de comprender que las computadoras son máquinas que toman datos del exterior, los procesan y obtienen los resultados almacenándolos en la memoria o enviándolos a un dispositivo de entrada y salida.

El procesamiento lo realiza por medio del procesador o CPU, y es allí donde radica la mayor complejidad para el alumno y donde se pueden generar dificultades para comprender su funcionamiento.

Se pueden explicar las partes del mismo, su funcionamiento, la interacción que realiza cada componente, también se puede enseñar un lenguaje de programación ensamblador y realizar las prácticas de programación, pero en general se ve que el alumno no logra llegar a una comprensión del funcionamiento.

Sin embargo, la utilización de simuladores permite afianzar los conocimientos de los temas vistos en las clases teóricas, a fin de evitar que los estudiantes desvíen su atención hacia el aprendizaje del simulador propiamente, se debe buscar que los simuladores tengan un manejo simple, intuitivo y visualmente atractivo, simplificando su aprendizaje de su uso.

La simulación es un término de uso diario en muchos contextos: medicina, militar, entretenimiento, educación, etc., debido a que permite ayudar a comprender cómo funciona un sistema, responder preguntas como "qué pasaría si", con el fin de brindar hipótesis sobre cómo o por qué ocurren ciertos fenómenos.

Para continuar, se define simulación como el proceso de imitar el funcionamiento de un sistema a medida que avanza en el tiempo. Entonces para llevar a cabo una simulación, es necesario desarrollar previamente un modelo conceptual que representa las características o comportamientos del sistema, mientras que la simulación representa la evolución del modelo a medida que avanza en el tiempo 2, 3, 4.

Con los avances en el mundo digital, la simulación se ha convertido en una metodología de solución de problemas indispensable para ingenieros, docentes, diseñadores y gerentes. La complejidad intrínseca de los sistemas informáticos los hace difícil comprender y costosos de desarrollar sin utilizar simulación 3.

Muchas veces en el ámbito educativo, resulta difícil transmitir fundamentos teóricos de la organización y arquitectura interna de las computadoras debido a la complejidad de los procesos involucrados. Si sólo incorporamos los medios de enseñanza tradicionales, como ser una pizarra, un libro de texto o diapositivas, los mismos tienen una capacidad limitada para representar estos fundamentos. En consecuencia, es imprescindible un alto nivel de abstracción por parte del alumno para desarrollar un modelo mental adecuado para capturar la organización y arquitectura interna de las computadoras 5, 6, 7.

Es evidente la necesidad de utilizar nuevas tecnologías como recurso didáctico y como medio para la transferencia de conocimiento, ya que resultan de gran ayuda para que los alumnos relacionan conceptos abstractos con reales, permite situar al alumno en un contexto que imite algún aspecto de la realidad; en ese ambiente, el alumno podrá detectar problemáticas similares a las que podrían producirse en la realidad, logrando un mejor entendimiento por medio del trabajo exploratorio, inferencia, aprendizaje por descubrimiento y desarrollo de habilidades 8, 9.

Un simulador de arquitectura es una herramienta que imita el hardware de un sistema. El simulador se centra principalmente en la representación de los aspectos arquitectónicos y funciones del hardware simulado. El uso de herramientas de simulación permite realizar cambios, pruebas y ejecución de programas sin temor de dañar ningún componente o por falta de la computadora 10.

Algunas herramientas ofrecen una representación en forma visual e interactiva de la organización y arquitectura interna de la computadora, facilitando así la comprensión de su funcionamiento. En este sentido, los simuladores juegan una pieza clave en el campo de la Arquitectura de Computadores, permitiendo conectar fundamentos teóricos con la experiencia práctica, simplificando abstracciones y facilitando la labor docente 11, 12, 13, 14.

El repertorio de instrucciones de la arquitectura x86 es la más utilizada en computadoras de escritorio y servidores del mundo. Inició con el procesador Intel 8086 en el año 1978 como arquitectura de 16 bits. Después evolucionó hasta una arquitectura de 32 bits cuando apareció el procesador Intel 80386 en el año 1985, denominada i386 o x86-32. AMD amplió esta arquitectura de 32 bits a una de 64 bits. Intel adoptó las extensiones de la arquitectura de AMD de 64 bits, también denominada AMD64 o Intel 64 15, 16.

Un procesador x86-64 mantiene la compatibilidad con los modos x86 existentes de 16 y 32 bits, y permite ejecutar aplicaciones de 16 y 32 bits, como así también de 64 bits. Esta compatibilidad hacia atrás protege las principales inversiones en aplicaciones y sistemas operativos desarrollados para la arquitectura x86 15, 16, 17.

Por ello, la enseñanza de la arquitectura x86 es de gran relevancia en la asignatura Arquitecturas de Computadoras debido a los diferentes temas que aborda.

Los alumnos de la asignatura Arquitectura de Computadoras no solo deben conocer la estructura y el funcionamiento interno de la computadora, sino que, idealmente, deben tener una experiencia práctica activa con dicha arquitectura.

Para proporcionar esta experiencia es necesario un laboratorio con el hardware necesario y el tiempo para que los alumnos se vuelvan competentes en el uso de herramientas para trabajar con el hardware. Por este motivo, muchos simuladores han sido desarrollados, ayudando al alumno a comprender el funcionamiento y la estructura del computador proporcionando valiosas experiencias de aprendizaje 18.

1.1 Justificación

Aunque ya existen simuladores de la arquitectura x86 que apoyan la enseñanza en los cursos de Arquitectura de Computadoras 10, 11, estos suelen presentar una gran cantidad de contenidos preestablecidos. Aunque estos contenidos son relevantes, ofrecer

toda la especificación de la arquitectura x86 desde el principio puede ser abrumador para los estudiantes y dificultar su comprensión.

Sin embargo, desde esta tesis se propone un enfoque diferente, el objetivo principal es construir un herramienta de simulación de la arquitectura x86 para dar apoyo a la enseñanza de arquitectura de computadoras, más concretamente de los contenidos específicos que se enseñan en la currícula de arquitectura de computadoras basados en la arquitectura x86. Partiendo de una visión global de la estructura y funcionamiento de la computadora (CPU, memoria, módulo de E/S y buses), mostrando los micropasos necesarios para la realización del ciclo básico de una instrucción, ofreciendo un repertorio reducido de instrucciones que se habiliten las instrucciones a medida que se dictan en la asignatura, permitiendo la generación y ejecución de programas escritos en lenguaje ensamblador, ya sea paso a paso por instrucción o completa, gestión básica de interrupciones permitiendo la interacción con el teclado y la pantalla, comunicación con los módulos de entrada y salida e interacciones con los periféricos, y por último, medidas de rendimiento sobre la ejecución de un programa.

Con el objeto de ofrecer al alumno un simulador bien diseñado, robusto, modular y por tanto flexible y sencillo de modificar o ampliar, se explorará la utilización de técnicas formales de modelización y simulación como las redes de Petri o DEVS (Discrete Event System Specification). Estas técnicas permiten separar conceptualmente las capas de modelización y simulación y ofrecen por ello una separación ortogonal de ambas, facilitando la comprensión y modificación del software. Además, permiten el escalado transparente de las simulaciones, pudiéndose ejecutar en entornos de cómputo paralelo o distribuido sin necesidad de modificar el modelo en sí, lo que aporta grandes ventajas de escalado 19, 20, 21.

1.2 Objetivos

1.2.1 Objetivo General

El objetivo principal de esta tesis es desarrollar una herramienta de simulación de la arquitectura x86 para apoyar la enseñanza de arquitectura de computadoras, enfocándose específicamente en los contenidos de la currícula de Arquitectura de Computadoras basados en la arquitectura x86

1.2.2 Objetivos Específicos

Para ello se debe cumplir los siguientes objetivos específicos:

- Estudiar y evaluar diferentes herramientas actuales de simulación destinadas a dar apoyo a la enseñanza de la arquitectura x86.

- Construir una herramienta de apoyo para impartir los contenidos de la asignatura Arquitectura de Computadoras, para ello debe cumplir:
  - Ofrecer una visión global de la estructura y funcionamiento de la computadora.
  - Permitir la generación y ejecución de programas escritos en lenguaje ensamblador, ya sea paso a paso por instrucción o completa.
  - Ofrecer un repertorio de instrucciones x86 reducido donde se habiliten las instrucciones a medida que se desarrolle el contenido en la asignatura.
  - Simular de manera visual e interactiva los micros pasos que conlleva el ciclo básico de una instrucción.
  - Permitir la gestión básica de interrupciones permitiendo la interacción con el teclado y la pantalla.
  - Permitir la comunicación con los módulos de entrada y salida e interacciones con los periféricos.
  - Ofrecer medidas de rendimiento sobre la ejecución de un programa.

1.3 Metodología de Desarrollo

Teniendo en cuenta los objetivos propuestos en la sección anterior, se pretende alcanzar los mismos a través de los pasos que se describen en esta sección.

1.3.1 Etapas de la Investigación

a. Análisis bibliográfico. Se realizó una revisión continua de las publicaciones científicas y tecnológicas, libros e informes técnicos relacionados con el objeto de estudio.
b. Recopilación de simuladores. Se realizó un relevamiento del estado actual y las actualizaciones de los simuladores aplicados a la enseñanza de arquitectura de computadoras.
c. Estudio de los simuladores. En base a la documentación relevada de los simuladores se estudió en profundidad al menos 5 simuladores y se elaboró una comparativa de los simuladores seleccionados en cuanto a los contenidos que se imparten en la asignatura.
d. Construir el simulador. A través de métodos y técnicas de ingeniería de software se construyó un simulador de la arquitectura x86 donde abarque los aspectos más relevantes de la asignatura Arquitectura de Computadoras, permitiendo desarrollar los contenidos en una plataforma unificada, evitando así la pérdida de tiempo y dificultad que supone para el alumno habituarse a diferentes entornos. Se utilizarán para ello técnicas formales de modelización y simulación, que facilitan un desarrollo modular y enfocan el esfuerzo en la definición del

modelo de la arquitectura x86 más que en el protocolo de simulación, permitiendo además el escalado a entornos de ejecución paralelos o distribuidos sin necesidad de modificar el modelo de la arquitectura simulada.

1.4 Organización del Documento

El resto de este documento se organiza de la siguiente manera: el capítulo 2 define formalmente las características y el set de instrucciones de la arquitectura. Luego, el capítulo 3 repasa y motiva el interesante rol que la simulación desde un punto de vista didáctico puede desempeñar para el dictado de la asignatura donde se abordan estos tópicos. El capítulo 4 comparativo de los simuladores estudiados según criterios preestablecidos. Finalmente, el capítulo 5 adaptación de un simulador como soporte para el uso de la enseñanza y aprendizaje de arquitectura de computadora.

2

Estado del arte

2.1 Arquitectura de computadoras

La arquitectura de computadoras es el estudio y diseño de los componentes de hardware y software que componen una computadora, así como la interacción entre ellos. Este campo abarca desde el diseño de circuitos y componentes individuales hasta la integración de sistemas completos. La comprensión de estos fundamentos es crucial para el desarrollo de tecnologías eficientes y avanzadas. La arquitectura de computadoras es el campo que se encarga de diseñar y organizar los componentes fundamentales de un sistema informático, como la unidad central de procesamiento (CPU), la memoria y los dispositivos de entrada/salida. Estudiar esta disciplina nos brinda una comprensión profunda de cómo funcionan estos componentes y cómo se interconectan para formar un sistema informático. Una razón convincente para estudiar arquitectura de computadoras es que nos permite comprender cómo funcionan los sistemas informáticos desde el nivel más bajo. Nos proporciona una comprensión profunda de los principios fundamentales, las estructuras y los componentes clave de un sistema informático. Esta comprensión nos da una visión holística del funcionamiento de una computadora y nos permite aprovechar su potencial al máximo. Además, la arquitectura de computadoras está estrechamente relacionada con otros campos de la informática, como la programación y el desarrollo de software. Comprender la arquitectura del hardware nos permite escribir programas y algoritmos que aprovechen al máximo los recursos disponibles. Esto es especialmente importante en un mundo donde la optimización del rendimiento y la eficiencia energética son cada vez más relevantes. Además, la arquitectura de computadoras nos brinda una base sólida para comprender y aprovechar las nuevas tendencias tecnológicas, como la computación en la nube, la inteligencia artificial y el internet de las cosas. Estudiar esta disciplina nos prepara para ser parte de la innovación tecnológica y contribuir al desarrollo de soluciones avanzadas en estos campos. En resumen, estudiar arquitectura de computadoras nos proporciona un conocimiento profundo de los sistemas informáticos, nos capacita para diseñar sistemas eficientes y escalables,

y nos prepara para aprovechar las últimas tendencias tecnológicas. Es una disciplina esencial para aquellos que desean seguir una carrera en el campo de la informática y la tecnología, y contribuir al avance de la sociedad digital en la que vivimos. Además, el estudio de la arquitectura de computadoras nos brinda una base sólida para comprender y resolver problemas de rendimiento y eficiencia en los sistemas informáticos. Al adentrarnos en los conceptos de paralelismo, pipelining, caché y optimización de código, aprendemos a diseñar sistemas más eficientes y a mejorar el rendimiento de las aplicaciones. Otro motivo relevante para estudiar arquitectura de computadoras es la capacidad de contribuir al desarrollo de nuevas tecnologías. La innovación en esta área ha sido constante, desde la miniaturización de componentes hasta el diseño de arquitecturas avanzadas como las basadas en computación cuántica. Al obtener conocimientos en arquitectura, podemos ser parte de esta evolución tecnológica y contribuir a la creación de sistemas más poderosos y eficientes. Por último, el estudio de la arquitectura de computadoras nos proporciona una base sólida para comprender otros campos relacionados, como la seguridad informática y los sistemas embebidos. Estos conocimientos son cada vez más demandados en la industria, lo que abre oportunidades laborales en empresas de desarrollo de software, fabricantes de hardware, centros de investigación y muchas otras áreas relacionadas con la tecnología. En conclusión, estudiar arquitectura de computadoras es fundamental para comprender el funcionamiento de los sistemas informáticos, resolver problemas de rendimiento y eficiencia, contribuir a la innovación tecnológica y acceder a una amplia gama de oportunidades laborales. Es una disciplina emocionante que impulsa la evolución de la tecnología y nos permite ser parte activa de este cambio.

2.1.1 Arquitectura x86

La arquitectura x86 es una de las más influyentes y ampliamente utilizadas en el ámbito de las computadoras de escritorio y servidores. Su historia se remonta a 1978 con el lanzamiento del procesador Intel 8086, que introdujo una arquitectura de 16 bits. Esta arquitectura evolucionó significativamente con la introducción del Intel 80386 en 1985, que marcó el inicio de la era de 32 bits 15, 16, 17.

Contextualización Histórica y Evolución de la Arquitectura x86

La compatibilidad hacia atrás de la arquitectura x86 ha sido un factor clave en su éxito, permitiendo la ejecución de aplicaciones de 16, 32 y 64 bits en un mismo sistema. Esta característica ha protegido las inversiones en aplicaciones y sistemas operativos desarrollados para la arquitectura x86, consolidando su relevancia en la industria.

Table 2.1: Hitos procesadores x86

Procesador	Año.de.Lanzamiento	Número.de.Bits	Nuevas.Características
Intel 8086	1978	16	Arquitectura inicial
Intel 80386	1985	32	Memoria virtual
AMD64	2003	64	Extensiones de 64 bits

La evolución de la arquitectura x86 ha estado marcada por una serie de hitos importantes que han impulsado la informática hacia nuevas alturas. Tras el Intel 8086, el lanzamiento del Intel 80286 en 1982 introdujo modos de operación adicionales que mejoraron la eficiencia y el manejo de memoria. Luego, en 1989, el Intel 80486 incorporó una unidad de punto flotante integrada y una mejor caché, lo que aumentó significativamente el rendimiento. La serie Pentium, iniciada en 1993, llevó la arquitectura x86 a nuevos niveles de rendimiento y eficiencia, incorporando características avanzadas como la ejecución superescalar y la predicción de saltos. Los avances continuaron con el Pentium Pro en 1995, que mejoró la arquitectura con ejecución fuera de orden y una caché L2 integrada. En la década de 2000, la arquitectura x86 se adaptó a las demandas de la computación moderna con la introducción del Intel Core, que optimizó el rendimiento y la eficiencia energética. AMD también jugó un papel crucial con su serie Athlon y la introducción de AMD64, que llevó la arquitectura x86 a 64 bits, permitiendo un acceso a mayor memoria y mejorando el rendimiento en aplicaciones intensivas.

Table 2.2: Línea de Tiempo de la Evolución de la Arquitectura x86

Año	Procesador	Innovación_Principal
1978	Intel 8086	Introducción de la arquitectura x86, 16 bits
1982	Intel 80286	Modos de operación adicionales
1985	Intel 80386	Arquitectura de 32 bits, memoria virtual
1989	Intel 80486	Unidad de punto flotante integrada, mejor caché
1993	Intel Pentium	Ejecución superescalar, predicción de saltos
1995	Intel Pentium Pro	Ejecución fuera de orden, caché L2 integrada
2003	AMD64	Extensiones a 64 bits, mayor acceso a memoria
2006	Intel Core	Optimización de rendimiento y eficiencia energética

Comparación con Otras Arquitecturas

La arquitectura x86 ha dominado el mercado de las computadoras de escritorio y servidores durante décadas, pero existen otras arquitecturas importantes que también han tenido un impacto significativo en la informática. Comparar x86 con arquitecturas como ARM, MIPS y RISC-V nos permite entender mejor sus ventajas y desventajas.

Table 2.3: (#tab:tabla-comparacion_arquitecturas)Comparación de Arquitecturas

Característica	x86	ARM	MIPS	RISC.V
Eficiencia Energética	Moderada	Alta	Moderada	Alta
Complejidad ISA	Alta	Baja	Moderada	Baja
Rendimiento	Alto	Moderado	Moderado	Variable
Compatibilidad	Alta (hacia atrás)	Moderada	Moderada	Alta
Áreas de Aplicación	Escritorio, servidores	Dispositivos móviles	Sistemas embebidos	Investigación, embebidos

La arquitectura ARM, conocida por su eficiencia energética, ha ganado popularidad en dispositivos móviles y sistemas embebidos. MIPS, aunque menos prominente en la actualidad, se ha utilizado en sistemas embebidos y educación. RISC-V, una arquitectura abierta y libre, ha surgido como una alternativa flexible y eficiente, especialmente en investigación y aplicaciones embebidas.

Aplicaciones y Casos de Uso de la Arquitectura x86

La arquitectura x86 ha demostrado ser extremadamente versátil y ha encontrado aplicaciones en una amplia variedad de campos. Su capacidad para manejar tareas complejas y su compatibilidad hacia atrás han sido factores clave en su adopción.

\begin{table}

\caption{(#tab:casos_uso_x86)Casos de Uso Notables de la Arquitectura x86}

Servidores: La arquitectura x86 domina el mercado de servidores debido a su capacidad para manejar gr
Computadoras Personales: Desde su introducción, la arquitectura x86 ha sido la columna vertebral de la
Sistemas Embebidos: Con la miniaturización y la eficiencia mejorada, los procesadores x86 se han integr

\end{table}

Impacto de la Arquitectura x86 en la Industria del Software

La arquitectura x86 ha tenido un impacto profundo en el desarrollo de software. Los sistemas operativos populares como Windows y Linux están optimizados para x86, lo que ha influido en el desarrollo y la optimización de aplicaciones para esta

arquitectura. Influencia en el Desarrollo de Software: Optimización del Rendimiento: Los desarrolladores de software han trabajado estrechamente con las características de la arquitectura x86 para optimizar el rendimiento de sus aplicaciones. Esto incluye el uso de instrucciones específicas de x86 y la optimización para cachés y pipelines. Compatibilidad y Soporte: La compatibilidad hacia atrás de x86 ha permitido la continuidad de aplicaciones y sistemas operativos, protegiendo las inversiones en software y facilitando las actualizaciones. Ecosistema de Desarrollo: Un amplio ecosistema de herramientas de desarrollo, bibliotecas y frameworks ha sido construido alrededor de la arquitectura x86, facilitando el desarrollo de aplicaciones de alto rendimiento y su depuración.

Futuro de la Arquitectura x86

La arquitectura x86 sigue evolucionando para enfrentar los desafíos de la computación moderna. Las tendencias emergentes como la computación cuántica, la inteligencia artificial y la computación en la nube presentan tanto oportunidades como desafíos para x86. Desafíos y Oportunidades Futuras: Computación Cuántica: Aunque aún en sus primeras etapas, la computación cuántica podría complementar las arquitecturas tradicionales como x86, especialmente en aplicaciones que requieren capacidades de procesamiento extremo. Inteligencia Artificial: La integración de aceleradores de IA y el soporte para operaciones de aprendizaje profundo en hardware x86 están siendo explorados para mejorar el rendimiento en aplicaciones de IA. Computación en la Nube: La arquitectura x86 sigue siendo relevante en la infraestructura de la nube, aunque enfrenta competencia de arquitecturas más eficientes como ARM. En resumen, la arquitectura x86 ha sido fundamental en la evolución de la informática, proporcionando una plataforma versátil y poderosa para una amplia variedad de aplicaciones. Su continua evolución y adaptación a las nuevas tecnologías aseguran su relevancia en el futuro.

Repertorio de instrucciones x86

El repertorio de instrucciones, o ISA (Instruction Set Architecture), es fundamental para la interacción entre el software y el hardware de una computadora. En el caso de la arquitectura x86, este conjunto de instrucciones ha evolucionado para incluir una amplia gama de operaciones que permiten un control detallado del hardware. La ISA x86 es conocida por su complejidad y flexibilidad, lo que la hace ideal para una variedad de aplicaciones. Por ello, la enseñanza de la arquitectura x86 es de gran relevancia en la asignatura Arquitecturas de Computadoras debido a los diferentes temas que aborda.

Ejemplos de Instrucciones x86

El repertorio de instrucciones x86 incluye una variedad de instrucciones para realizar operaciones aritméticas, lógicas, de control y de manejo de memoria. Algunos ejemplos son:

- `MOV`: Mueve datos de una ubicación a otra.
- `ADD`: Suma dos valores.
- `SUB`: Resta un valor de otro.
- `JMP`: Salta a una dirección específica.

2.2 Lenguaje ensamblador

Un procesador puede interpretar y ejecutar solo instrucciones máquina. Estas instrucciones son simplemente secuencias de números binarios almacenados en la computadora y son leídas por el procesador e interpretadas por él. Si un programador quisiera programar directamente en el lenguaje máquina, necesita introducir los programas como datos binarios. Esto requeriría escribir las sentencias que necesita realizar el procesador directamente en binarios, por lo tanto, conocer la secuencia de ceros y unos para cada operación escribiéndose ordenadamente, además de respetar la estructura de memoria y direccionamientos del procesador. Esto sin duda es un trabajo complejo, difícil, pesado y muy susceptible a errores. Adicionalmente, una vez que se requiera realizar modificaciones, su lectura implica ir traduciendo estas secuencias de ceros y unos a su correspondiente instrucción, lo que es doblemente dificultoso. El lenguaje ensamblador es un lenguaje de programación de bajo nivel que permite a los programadores escribir instrucciones comprensibles por el procesador. A diferencia del lenguaje máquina, que utiliza secuencias de números binarios, el ensamblador emplea mnemónicos simbólicos que hacen que el código sea más legible y manejable para los humanos. Cada arquitectura de procesador tiene su propio lenguaje ensamblador que usualmente es definida por el fabricante de hardware, por lo tanto es específica de cierta arquitectura de la computadora física. Un programa llamado ensamblador es usado para traducir sentencias del lenguaje ensamblador al código de máquina de la computadora. El ensamblador realiza una traducción casi directa uno a uno desde las sentencias mnemónicas a las instrucciones y datos de máquina. Esto está en contraste con los lenguajes de alto nivel, en los cuales una sola declaración generalmente da lugar a muchas instrucciones de máquina. El código fuente generado en mnemotécnicos debe ser traducido a lenguaje máquina para poder ser ejecutado por la computadora. Este proceso lo realizan programas denominados ensambladores. Si nos ubicamos en la arquitectura x86 encontraremos diferentes lenguajes ensambladores. De ahí los diferentes productos ofrecidos, entre ellos, TASM (Turbo Assembler), MASM (Microsoft Macro Assembler) y NASM (Netwide

Assembler). Cada uno ofrece diferentes características y beneficios, adaptándose a diversas necesidades y preferencias de los programadores. Estos ensambladores convierten el código simbólico en código máquina, permitiendo su ejecución en el hardware específico de la arquitectura x86.

2.3 Simulación

La simulación es una herramienta esencial en múltiples campos como la medicina, el ámbito militar, el entretenimiento y la educación. Permite a los profesionales comprender el funcionamiento de un sistema, realizar análisis predictivos y responder preguntas del tipo "qué pasaría si", generando hipótesis sobre cómo o por qué ocurren ciertos fenómenos. Según Banks et al. (2001), la simulación se define como el proceso de imitar el comportamiento de un sistema a lo largo del tiempo. Este proceso requiere primero desarrollar un modelo conceptual que capture las características y comportamientos del sistema real, mientras que la simulación representa la evolución de este modelo conforme el tiempo avanza. La capacidad de replicar y analizar sistemas complejos sin necesidad de intervenir directamente en ellos es lo que hace que la simulación sea una metodología indispensable para ingenieros, diseñadores y gerentes en el mundo digital actual [1,3]. Con los avances en el mundo digital, la simulación se ha convertido en una metodología de solución de problemas indispensables para ingenieros, docentes, diseñadores y gerentes. La complejidad intrínseca de los sistemas informáticos los hace difícil comprender y costosos de desarrollar sin utilizar simulación [3].

En la industria automotriz, la simulación es fundamental para el diseño y prueba de sistemas de seguridad, como airbags y frenos. Un modelo virtual del automóvil y sus componentes permite realizar pruebas de colisión y evaluar el rendimiento de los sistemas de seguridad sin la necesidad de llevar a cabo costosas pruebas físicas. Además, la simulación permite optimizar el diseño de motores, analizar el flujo aerodinámico y prever el comportamiento de los materiales en condiciones extremas. Ejemplo Adicional: En el campo de la aviación, la simulación se utiliza para entrenar pilotos en simuladores de vuelo que replican condiciones reales sin riesgos. También se emplea en el diseño de aeronaves para evaluar la aerodinámica y el rendimiento de nuevos diseños bajo diversas condiciones de vuelo. Estos ejemplos demuestran cómo la simulación puede reducir costos, mejorar la seguridad y acelerar el desarrollo de productos complejos.

2.3.1 Herramientas de Simulación en la Educación

Herramientas como Simulink, Proteus, y Logisim permiten a los estudiantes interactuar con modelos virtuales de circuitos y sistemas informáticos, facilitando la comprensión de la arquitectura interna y las operaciones de una computadora.

2.4 Simulación aplicada a la enseñanza de arquitectura de computadoras

Los estudiantes de la asignatura Arquitectura de Computadoras deben no solo conocer la estructura y el funcionamiento interno de las computadoras, sino también tener una experiencia práctica activa con dicha arquitectura. Esta experiencia práctica es crucial para desarrollar competencias en el uso de herramientas y técnicas relacionadas con el hardware. Sin embargo, la implementación de laboratorios físicos con el hardware necesario puede ser costosa y requiere tiempo considerable para que los alumnos adquieran las habilidades necesarias. Para superar estas limitaciones, se han desarrollado numerosos simuladores que proporcionan experiencias de aprendizaje valiosas al replicar el funcionamiento y la estructura de las computadoras. Simuladores como SimpleScalar, SPIM y GEM5 permiten a los estudiantes experimentar con arquitecturas complejas y técnicas avanzadas como el pipelining y la ejecución fuera de orden. Estas herramientas facilitan la comprensión de los conceptos teóricos a través de la interacción práctica, proporcionando un entorno seguro y accesible para la exploración y el aprendizaje [17]. ### Uso de la Simulación para Enseñar Pipelining El pipelining es una técnica utilizada en las CPUs modernas para mejorar el rendimiento. Utilizando simuladores como SimpleScalar, los estudiantes pueden visualizar cómo las instrucciones se ejecutan en diferentes etapas del pipeline, y cómo se manejan las dependencias de datos y los riesgos de control.

2.5 El Formalismo DEVS (Discrete Event System Specification)

DEVS, la abreviación de Discrete Event System Specification, es un formalismo modular y jerárquico para el modelado y análisis de sistemas que pueden ser representados como sistemas de eventos discretos, sistemas de estado continuo o sistemas híbridos. Este formalismo, desarrollado por Bernard P. Zeigler en los años 70, extiende el concepto de las máquinas de Moore al proporcionar una estructura para modelar sistemas complejos mediante la utilización de eventos cronometrados. ### Descripción del Formalismo DEVS El formalismo DEVS define el comportamiento de un sistema real utilizando eventos de entrada y salida, y transiciones entre estados concretos. Un sistema en DEVS está compuesto por modelos atómicos y acoplados. Los modelos atómicos representan las unidades básicas de comportamiento, mientras que los modelos acoplados consisten en combinaciones de modelos atómicos y/o otros modelos acoplados. Esta estructura jerárquica facilita la gestión y análisis de sistemas complejos, permitiendo la prueba de subsistemas de manera aislada antes de integrarlos en el sistema completo. Bajo un punto de vista general, un modelo DEVS está caracterizado por generar eventos

de salida Y , en relación con el estado en el que se encuentre S y las entradas recibidas X, cada cierto tiempo.

Figura 3.4: Representación simple de un modelo DEVS ### Aplicaciones del Formalismo DEVS El formalismo DEVS es aplicable a una amplia gama de sistemas, desde redes de comunicación hasta procesos de manufactura. Por ejemplo, en una red de comunicación, un modelo DEVS puede simular el enrutamiento de paquetes de datos y la gestión de congestiones. En la manufactura, un modelo DEVS puede representar el flujo de materiales y el control de calidad en una línea de producción, ayudando a identificar cuellos de botella y optimizar procesos. ### Ejemplo de Modelo DEVS Consideremos un sistema de colas en un banco, donde los clientes llegan, esperan en la fila y son atendidos por cajeros. Un modelo DEVS puede representar los eventos de llegada de clientes, la asignación a los cajeros y el tiempo de servicio, permitiendo evaluar el tiempo de espera y la utilización de los recursos. Este ejemplo ilustra cómo DEVS puede ser utilizado para mejorar la eficiencia operativa y la satisfacción del cliente mediante la optimización del flujo de trabajo y la asignación de recursos.

2.5.1 Aporte pedagógico

En el ámbito educativo, transmitir los fundamentos teóricos de la organización y arquitectura interna de las computadoras puede ser un desafío debido a la complejidad de los procesos involucrados. Los métodos tradicionales de enseñanza, como el uso de pizarras, libros de texto y diapositivas, a menudo tienen una capacidad limitada para representar estos conceptos de manera efectiva. Esto requiere que los estudiantes tengan un alto nivel de abstracción para desarrollar un modelo mental adecuado para capturar la organización y arquitectura interna de las computadoras [4,5]. La integración de nuevas tecnologías como recurso didáctico es crucial para facilitar el aprendizaje. Las herramientas de simulación permiten a los estudiantes relacionar conceptos abstractos con situaciones reales, situándose en un contexto que imita aspectos de la realidad. Este enfoque pedagógico facilita la detección de problemáticas y el desarrollo de habilidades a través del trabajo exploratorio, la inferencia y el aprendizaje por descubrimiento. Simuladores como Simulink, Proteus y Logisim juegan un papel esencial en la enseñanza de la arquitectura de computadoras, proporcionando una representación visual e interactiva que enriquece la comprensión teórica y práctica de los estudiantes [6,7]. El uso de herramientas de simulación en la enseñanza para procesos dinámicos complejos, como las operaciones intrínsecas de la computadora, que permiten representar de forma visual e interactiva la organización y arquitectura interna de la computadora, facilitando así la comprensión de su funcionamiento por parte de los alumnos y el desarrollo de los temas por parte del docente. En este contexto, los simuladores juegan una pieza clave en el campo de la Arquitectura de Computadoras, permitiendo conectar fundamentos teóricos con la experiencia práctica, implicando abstracciones y haciendo más rica la labor docente.

3

Comparativa de simuladores

Este capítulo realiza una comparación de los simuladores x86 más adecuados para la asignatura de Arquitectura de Computadoras de la Licenciatura en Sistemas. Se establecen los criterios de evaluación y se analizan los simuladores seleccionados de acuerdo con estos criterios.

3.1 Simuladores de software

Un simulador de arquitectura es un software que imita una situación del mundo real y, en este contexto, puede imitar el hardware de un sistema de cómputo. El simulador se centra principalmente en la representación de los aspectos arquitectónicos y funciones del hardware simulado. El uso de herramientas de simulación permite realizar cambios, pruebas y ejecución de programas sin temor de dañar ningún componente o por falta de la computadora [8]. Algunos softwares ofrecen una representación en forma visual e interactiva de la organización y arquitectura interna de la computadora, facilitando así la comprensión de su funcionamiento, como ser los simuladores Assembly debugger (x86), Simple 8-bit Assembler Simulator, Microprocessor Simulator, Simulador de ensamblador de 16 bits y Emu8086. En este sentido, los simuladores juegan una pieza clave en el campo de la Arquitectura de Computadoras, permitiendo conectar fundamentos teóricos con la experiencia práctica, simplificando abstracciones y facilitando la labor docente [9,13].

3.2 Simuladores similares

Cabe destacar que existen estudios comparativos que evalúan diferentes simuladores aplicados a la enseñanza de los cursos de arquitectura de computadoras: “A survey

and evaluation of simulators suitable for teaching courses in computer architecture and organization", 2009 [9]: Otros estudios evalúan diferentes simuladores para abordar diferentes temas en el dictado de los cursos de Arquitectura de Computadoras, en general estos estudios evalúan simuladores en términos de dos categorías predefinidas: una referida a las características de la simulación, como ser granularidad, usabilidad, disponibilidad, presentación visual, flujo de simulación, etc., y otra sobre la cobertura de los contenidos preestablecidos en las currículas. "Survey and evaluation of simulators suitable for teaching for computer architecture and organization Supporting undergraduate students at Sir Syed University of Engineering & Technology", 2012 [10]: usabilidad, disponibilidad, Fundamentos de arquitectura informática, jerarquía de sistemas de memoria, comunicación e interfaz y diseño de sistemas de procesadores. Este trabajo se diferencia al proponer un enfoque diferente: evaluar los simuladores x86 bajo criterios de evaluación basados en características de simulación y en contenidos específicos de la asignatura Arquitectura de Computadoras de la carrera de Licenciatura en Sistemas de la Universidad Nacional de Entre Ríos.

3.3 Criterios de evaluación

Se han definido 7 criterios de evaluación: Usabilidad: Se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario. Indicador: facilidad de uso (difícil-media-fácil). Editor: Soporte para escribir código fuente en lenguaje ensamblador. Indicador: funcionalidades del editor (baja-media-alta). Documentación: Disponibilidad de soporte para el aprendizaje, repertorio de instrucciones, manual de usuario. Indicador: documentación disponible (mínima-media-completa). Ejecución de simulación: Facilidad para controlar la simulación. Indicador: controles de simulación (baja-media-alta). Nivel de especificación de la Organización y Arquitectura del sistema simulado: Nivel de implementación del set de instrucciones, memoria, módulos de E/S, etc. Indicador: adecuación del repertorio de instrucciones del simulador a una arquitectura x86 real (mínima-media-completa). Características del desarrollo del producto software: Tipo de licencia (open source o privativas), fecha de última versión, Web/Escritorio, uso académico. Indicador: características del producto software (mala-buena-muy buena). Cobertura de los contenidos preestablecidos en la currícula: Ajuste a los tópicos de la asignatura Arquitectura de Computadoras. Indicador: características del simulador para dar soporte a contenidos específicos (baja-media-alta).

3.4 Selección de simuladores

A partir de una exploración en internet sobre herramientas de simulación de la arquitectura x86 utilizadas para la enseñanza, se detectaron los siguientes

simuladores: Assembly debugger (x86) Simple 8-bit Assembler Simulator Microprocessor Simulator Simulador de ensamblador de 16 bits Emu8086 VonSim Orga1 Qsim De estos simuladores, se seleccionaron tres que a priori cubrían la mayor cantidad de contenidos curriculares establecidos en el criterio de evaluación "Cobertura de los contenidos preestablecidos en la currícula": Emu8086, VonSim y Simple 8-bit Assembler Simulator.

3.5 Análisis comparativo

Se analizaron los simuladores seleccionados con base en los criterios de evaluación definidos: Simple 8-bit Assembler Simulator Usabilidad: Nivel medio. Presenta todos los componentes en una sola pantalla, lo cual puede ser abrumador para usuarios iniciales. Editor: Nivel bajo. Incluye aviso de errores de sintaxis al ensamblar, sin resaltado de sintaxis ni breakpoints. No permite guardar o cargar programas. Documentación: Nivel mínimo. Consta solo de un manual de instrucciones implementadas. Ejecución de simulación: Nivel medio. Permite configurar la velocidad del reloj de CPU y ofrece controles básicos de simulación. Nivel de especificación: Nivel mínimo. Simplifica la arquitectura x86 en un CPU de 8 bits, con memoria de 256 bytes y sin soporte para IN y OUT. Desarrollo del producto: Nivel bueno. Licencia MIT, última versión en 2015, desarrollado sobre plataforma web. Cobertura de contenidos: Nivel bajo. No implementa memoria independiente para módulos de entrada y salida, rutinas de tratamiento de interrupciones ni ciclo de instrucción. VonSim Usabilidad: Nivel medio. Presenta componentes mediante solapas, lo cual puede ser abrumador para usuarios iniciales. Editor: Nivel medio. Incluye aviso de errores de sintaxis, resaltado de sintaxis y breakpoints mediante interrupción por software. Documentación: Nivel medio. Posee manual de uso y tutorial interactivo. Ejecución de simulación: Nivel medio. Permite configurar la velocidad del reloj de CPU y ofrece controles básicos de simulación. Nivel de especificación: Nivel medio. Representa una simplificación del procesador 8088 con arquitectura de 16 bits y memoria direccionable de 16 KiB. Desarrollo del producto: Nivel muy bueno. Licencia GNU Affero General Public License v3.0, última versión en 2020, desarrollado sobre plataforma web con extensa evidencia de uso académico. Cobertura de contenidos: Nivel media. Implementa dispositivos internos y externos, pero no desarrolla contenidos visuales para ciclo de instrucción y medidas de rendimiento. Emu8086 Usabilidad: Nivel fácil. Presenta inicialmente el editor y permite activar componentes del simulador a medida que se cargan programas. Editor: Nivel alto. Incluye aviso de errores de sintaxis, resaltado de sintaxis y opciones de breakpoints. Permite guardar y cargar programas. Documentación: Nivel completo. Incluye manual de instrucciones, tutorial de aprendizaje y manual de uso. Ejecución de simulación: Nivel alto. Permite configurar la velocidad del reloj de CPU y ofrece controles avanzados de simulación, incluyendo "step back". Nivel de especificación: Nivel completo. Representa detalladamente la arquitectura del procesador 8086 con memoria direccionable de 1

MiB y soporte para interrupciones por software y hardware. Desarrollo del producto: Nivel bueno. Licencia privativa, última versión en 2023, desarrollado sobre plataforma de escritorio. Cobertura de contenidos: Nivel alto. Soporta todos los modos de direccionamiento y permite emular el booteo de una IBM PC desde el floppy disk, entre otros.

Tabla comparativa según criterios de evaluación preestablecidos. Criterios de evaluación Simple 8-bit VonSim Emu8086 1. Usabilidad (Difícil * - Media ** - Fácil ***)*** 2. Editor (Baja * - Media ** - Alta ***)*** ***3. Documentación (Mínima* - **Media** - Completa ***)*** ***4. Ejecución de simulación (Baja* - **Media** - Alta ***)*** 5. Nivel de especificación x86 (Mínima * - Media ** - Completa ***)*** ***6. Producto software (Mala* - **Buena** - Muy buena ***)*** ***7. Cobertura contenidos (Baja* - **Media** - Alta ***)*** **

3.6 Conclusiones del análisis

En la asignatura se utiliza la herramienta emu8086 versión 4.08 desde el año 2018. En el análisis de la herramienta se observó que la herramienta tiene un periodo de evaluación gratuito de 14 días, luego de esta se debe adquirir la licencia para su uso. Desde la asignatura se incentiva el uso de simuladores para dar apoyo a los proceso de enseñanza y aprendizaje, pero también se incentiva que los contenidos desarrollados puedan volcarse en máquinas reales, en este sentido consideramos que el enfoque planteado por la herramienta emu8086 es el más adecuado para la asignatura, ya que facilita mecanismos para implementar los programas en máquinas reales. Sin embargo presenta el inconveniente que genera ejecutables dependientes del sistema operativo MS-DOS, la mayoría de los sistemas operativos actuales no permiten la ejecución de dichos programas, obligando a la utilización de emuladores de MS-DOS para poder correrlos, siendo esto otro elemento más que se incorpora a los procesos de enseñanza y aprendizaje. Utilizar lenguaje NASM (Netwide Assembler) garantiza soporte tanto para Linux como Windows a través de herramientas libres como GCC (GNU Compiler Collection), generando programas para la arquitectura x86 de 16, 32 y 64 bits. En el criterio de evaluación uno emu8086 se destaca del resto debido a que su interfaz se despliega solo si es necesaria para su uso, por ejemplo pila, flags, teclado, pantalla, etc. a medida que se usan. En cambio las otras herramientas muestran todos sus componentes en su interfaz. En el criterio de evaluación dos emu8086 se destaca del resto debido a que el editor presenta punto de ruptura, que permiten correr la ejecución del programa hasta cierto punto, además incluye la facilidad de guardar y recuperar un programa desde el propio editor. En el criterio de evaluación tres el emu8086 se destaca del resto debido a que ofrece los tres tipos de documentación establecidas en este criterio, un repertorio de instrucciones con ejemplos para cada tipo de instrucción, un manual que explica las partes de las herramientas y un tutorial para aprender a programar en ensamblador En el criterio de evaluación cuatro el emu8086 se destaca

del resto debido presenta una mayor cantidad de controladores para gestionar el flujo de ejecución, por ejemplo permite retroceder la ejecución de una instrucción, otra opción es de recargar el programa actual. En el criterio de evaluación cinco el emu8086 se destaca del resto debido a que ofrece una mayor especificidad de la arquitectura x86, además implementa interrupciones del sistema operativo MS-DOS, a través del cual se pueden ejecutar los programas en una máquina real. En el criterio de evaluación seis VonSim se destaca del resto debido a que es licencia libre y posee una comunidad que respalda el proyecto. En cuanto al último criterio, ninguna de las herramientas evaluadas cubre todos los contenidos que se pretende desarrollar con la ayuda de una herramienta, quedando excluido pasos del ciclo de instrucción y medidas de rendimientos (tiempo de CPU y CPI: ciclo por instrucción).

4.6 Publicación El resultado de esta comparativa fue publicado en el XVII Congreso de Tecnología en educación y Educación en Tecnología año 2022 bajo el título de "Herramientas de software para dar soporte en la enseñanza y aprendizaje de la arquitectura x86". (Cita) Durante la elaboración de este análisis comparativo establecí contacto con uno de los docentes de la Universidad Nacional de la Plata (UNLP) que habían desarrollado un simulador web VonSim, con el acuerdo de docente se generó una solicitud de incorporación de cambios (pull request) en el repositorio de Github del simulador VonSim y en agosto-2023 salió una nueva versión donde se implementan animaciones de la ejecución de instrucciones y documentación on-line.

4

Diseño y construcción del simulador

El objetivo principal de esta tesis es construir una herramienta de simulación de la arquitectura x86 para apoyar la enseñanza de arquitectura de computadoras. Para lograr este objetivo, se plantean los siguientes objetivos específicos:

Bibliografía

[1] M. A. Colombani, M. A. Falappa, A. G. Delduca, and J. M. Ruiz, "PID novel 7065: Enseñanza/aprendizaje de asignatura Arquitectura de Computadoras con herramientas de simulación de sistemas de cómputos." Feb. 2022. Accessed: Jul. 10, 2024. Available: https://proyectos.uner.edu.ar/aplicacion.php?ah=st668e6d47663eb&ai=gestion_extinv||23000105

[2] J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol, *Discrete-event system simulation*, 5th ed. Prentice Hall, 2010.

[3] A. M. Law, *Simulation Modeling & Analysis*, 5th ed. New York, NY, USA: McGraw-Hill, 2015.

[4] S. Robinson, *Simulation: The Practice of Model Development and Use*, 2nd edition. 2014.

[5] C. Lion, "Los simuladores. Su potencial para la enseñanza universitaria," *Cuadernos de Investigación Educativa*, vol. 2, no. 12, pp. 53–66, 2005.

[6] G. Contreras, R. G. Torres, and M. S. R. Montoya, "Uso de simuladores como recurso digital para la transferencia de conocimiento," *Apertura: Revista de Innovación Educativa*, vol. 2, no. 1, pp. 86–100, 2010.

[7] A. Garcia-Garcia, J. C. Saez, J. L. Risco-Martin, and M. Prieto-Matias, "PBBCache: An open-source parallel simulator for rapid prototyping and evaluation of cache-partitioning and cache-clustering policies," *Journal of Computational Science*, vol. 42, p. 101102, 2020.

[8] B. Nova, J. C. Ferreira, and A. Araújo, "Tool to support computer architecture teaching and learning," in *Engineering Education (CISPEE), 2013 1st International Conference of the Portuguese Society for*, IEEE, 2013, pp. 1–8.

[9] B. Mustafa, "Evaluating A System Simulator For Computer Architecture Teaching And Learning Support," *Innovation in Teaching and Learning in Information and Computer Sciences*, vol. 9, no. 1, pp. 100–104, 2010, doi: 10.11120/ital.2010.09010100.

[10] Z. Radivojevic, M. Cvetanovic, and J. Đordevic, "Design of the simulator for teaching computer architecture and organization," in *2011 Second Eastern European Regional Conference on the Engineering of Computer Based Systems*, IEEE, 2011, pp. 124–130.

[11] B. Nikolic, Z. Radivojevic, J. Djordjevic, and V. Milutinovic, "A Survey and Evaluation of Simulators Suitable for Teaching Courses in Computer Architecture and Organization," *IEEE Transactions on Education*, vol. 52, no. 4, pp. 449–458, Nov. 2009, doi: 10.1109/TE.2008.930097.

[12] R. Hasan and S. Mahmood, "Survey and evaluation of simulators suitable for teaching for computer architecture and organization Supporting undergraduate students at Sir Syed University of Engineering & Technology," in *Control (CONTROL), 2012 UKACC International Conference on*, IEEE, 2012, pp. 1043–1045.

[13] J. L. Hennessy and D. A. Patterson, *Computer architecture: A quantitative approach*, Fifth Edition. Elsevier, 2012.

[14] W. Stallings, *Computer organization and architecture: Designing for performance*, Eleventh Edition. Pearson, 2013.

[15] Intel, "64 and IA-32 architectures software developers manual," *325462-060US*, vols. 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C and 3D, p. 4670, 2016, Available: https://software.intel.com/en-us/articles/intel-sdm

[16] AMD, "Developer Guides, Manuals & ISA Documents." Apr. 2019. Accessed: Apr. 21, 2019. Available: https://developer.amd.com/resources/developer-guides-manuals/

[17] P. Abel, *IBM PC Assembly Language and Programming*, Fifth Edition. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000.

[18] D. Skrien, "CPU Sim 3.1: A tool for simulating computer architectures for computer organization classes," *Journal on Educational Resources in Computing (JERIC)*, 2001.

[19] J. L. Peterson, *Petri net theory and the modeling of systems*. Prentice Hall PTR, 1981.

[20] B. Zeigler, H. Prähofer, and T. G. Kim, "Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems," vol. 2, Jan. 2000.

[21] B. P. Zeigler, A. Muzy, and E. Kofman, *Theory of Modeling and Simulation: Discrete Event & Iterative System Computational Foundations*. Academic Press, 2018.