Herramientas de software para dar soporte en la enseñanza y aprendizaje de la arquitectura x86

Marcelo A. Colombani, José M. Ruiz, Amalia G. Delduca, Marcelo A. Falappa
Universidad Nacional de Entre Ríos, Av. Tavella 1424, (E3202KAC), Concordia, Entre Ríos, Argentina
Universidad Nacional del Sur Av. Alem 1253, (B8000CPB) Bahía Blanca, Argentina
{marcelo.colombani, jose.ruiz, amalia.delduca}@uner.edu.ar, mfalappa@cs.uns.edu.ar

RESUMEN

Existe un consenso creciente en 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, simplificando abstracciones y haciendo más rica la labor docente.

La arquitectura x86 es ampliamente utilizada en computadoras de escritorio y servidores. Este documento pretende realizar una comparación de los simuladores x86 que más se adecuan en el dictado de la asignatura Arquitectura de Computadoras de la carrera Licenciatura en Sistemas, establecer los criterios de evaluación y evaluar los simuladores seleccionados de acuerdo con estos criterios.

La presente investigación surge en el marco del proyecto de investigación I/D novel PID-UNER 7065: “Enseñanza/aprendizaje de asignatura Arquitectura de Computadoras con herramientas de simulación de sistemas de cómputos”. El Proyecto es llevado a cabo en la Facultad de Ciencias de la Administración de la Universidad Nacional de Entre Ríos, se vincula directamente con la asignatura Arquitectura en Computadoras que se dicta en segundo año de la carrera Licenciatura en Sistemas perteneciente a la Facultad de Ciencias de la Administración de la Universidad Nacional de Entre Ríos.

***Palabras clave*:** x86, simulador, aprendizaje, enseñanza, arquitectura de computadoras.

1. INTRODUCCIÓN

Simulación

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 cómo 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. [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].

Aporte pedagógico

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 puede 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 [4,5].

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 relacionen 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 [6,7].

Simuladores

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].

Repertorio de instrucciones x86

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 [14,16].

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 [14,16].

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.

Asignatura Arquitectura de Computadoras

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 [17].

2. DESARROLLO

Objetivos

El trabajo tiene como objetivos:

- Estudiar y evaluar diferentes herramientas de simulación de procesadores de la arquitectura x86.

- Definir criterios de evaluación de las herramientas de simulación para su utilización en el dictado de clases.
- Comparar diferentes herramientas de simulación según los criterios de evaluación preestablecidos.
- Analizar el impacto de la jerarquía de memorias en la ejecución de un programa utilizando las herramientas de simulación de procesadores.
- Generalizar dichos conceptos a otras arquitecturas, como por ejemplo las actuales basadas en 64 bits.

Criterios de evaluación

Se trabajó sobre el segundo objetivo; "Definir criterios de evaluación de las herramientas de simulación para su utilización en el dictado de clases".

Se han definido 7 criterios de evaluación:

1. **Usabilidad:** se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso. Escala de usabilidad (difícil-media-fácil).
2. **Editor:** soporte para escribir código fuente en lenguaje ensamblador. Escala de editor (baja-media-alta).
3. **Documentación**: disponibilidad de soporte para el aprendizaje, repertorio de instrucciones, manual de usuario. Escala de documentación (mínima-media-completa)
4. **Ejecución de simulación:** facilidad para controlar la simulación. Escala de ejecución de simulación (baja-media-alta).
5. **Nivel de especificación de la Organización y Arquitectura del sistema simulado:** nivel de implementación del set de instrucciones, memoria, módulos de E/S, etc. Escala de nivel de especificación x86 (mínima-media-completa).
6. **Características del desarrollo del producto software:** tipo de licencia *open source* o privativas, fecha de última versión, Web/Escritorio, uso académico. Escala de producto software (mala-buena-muy buena).
7. **Cobertura de los contenidos preestablecidos en las currícula:** se busca que la herramienta abarque o se ajuste a la mayor cantidad de tópicos de la asignatura Arquitectura de Computadoras, se define la escala cobertura contenido (baja-media-alta) y se considera los siguientes tópicos de la asignatura:
   - Estructura y funcionamiento de la computadora.
   - Repertorio de instrucciones.
   - Tipos de datos y formato de representación a nivel máquina.
   - Ciclo de la instrucción.
   - Programación en lenguaje ensamblador.
   - Módulos de entrada y salidas y su comunicación con los periféricos.
   - Gestión de interrupciones.
   - Medidas de rendimiento del procesador.

Exploración y selección de simuladores

A partir de una exploración en internet acerca de herramientas de simulación de la arquitectura x86 utilizadas para la enseñanza, se detectar los siguientes simuladores:

- Assembly debugger (x86).
- Simple 8-bit Assembler Simulator.
- Microprocessor Simulator.
- Simulador de ensamblador de 16 bits.
- Emu8086.
- VonSim.
- Orga1.
- Qsim.

De dicho conjunto de simuladores se tomaron los más significativos y que a priori contemplan la mayor cantidad de criterios a evaluar. Se seleccionaron el Emu8086, el VonSim y el Simple 8-bit Assembler Simulator.

Análisis comparativo

De los criterios de evaluación preestablecidos para comparar los simuladores seleccionados se observa:

Simple 8-bit Assembler Simulator

1. La usabilidad se considera nivel media. Presenta todos los componentes del simulador en una sola pantalla, para un usuario inicial puede implicar un exceso de información.

El formato de representación de valores para la memoria y los registros es en hexadecimal y decimal sin signo.

Muestra información de las etiquetas definidas en el programa a la dirección de memoria donde realizan el salto, ya que muestra una tabla con la correspondencia entre etiqueta, dirección destino y contenido.

Implementa sintaxis del lenguaje ensamblador basada en NASM, pero no incluye las directivas para definir programas ejecutables.

2. Las herramientas que ofrece el editor del simulador son consideradas de nivel bajo. Incluye aviso de errores de sintaxis únicamente al momento de ensamblar, no posee resaltado de sintaxis, ni breakpoints. El simulador no brinda un repositorio o mecanismos para guardar y cargar programas.

3. La documentación es considerada de nivel mínimo, consta de un manual de instrucciones implementadas en el simulador, no posee tutorial para el aprendizaje de la herramienta, ni manual de uso.

4. El control de la simulación es de nivel media, el simulador permite configurar la velocidad del reloj de CPU (1 Hz, 4 Hz, 8 Hz y 16 Hz), resalta en el editor y en la memoria principal la instrucción en curso, también resalta cuando se actualizan los valores de la pila en memoria principal. En cuanto a los controles de simulación consta de un botón para cargar en memoria (Assemble) y los botones run/stop, step y reset para controlar la simulación. No posee control de simulación para volver a la instrucción anterior (step back).

5. El detalle de la arquitectura es de nivel mínima, simplifica la arquitectura x86 en un CPU de 8 bits, es decir, todos los registros del CPU son de 8 bits, implementa cuatro registros de propósito general (A, B, C y D), un registro puntero de instrucción (IP), un puntero de pila SP y el registro de estado únicamente los flags Zero (Z), Carry (C) y falta (F) que representa,, por ejemplo una división por cero. Este último flag F no existe en la arquitectura x86, ya que las faltas o excepciones se gestionan mediante interrupciones. La memoria direccionable es de tan solo 256 bytes. En cuanto a los dispositivos de entrada y salidas únicamente consta de una pantalla que puede mostrar 24 caracteres.

Al ser una arquitectura de 8 bits solo puede manejar valores tipo byte lo cual implica que no se puede ejemplificar el concepto de *little-endian*, una memoria de 256 posiciones representada en una matriz 16 filas y 16 columnas donde cada posición ocupa un byte, y el set de instrucciones solo se implementan instrucciones básicas. En cuanto a los dispositivos de entrada y salida cuenta únicamente con una pantalla que muestra como máximo 24 caracteres. No implementa las instrucciones IN y OUT y para comunicarse con la pantalla se reserva los últimos 24 bytes de memoria, al copiar un valor en el video RAM reservada se despliega en pantalla el carácter. No cuenta con la implementación de la instrucción INT ni la representación del vector de interrupciones.

Por simplicidad el formato de las instrucciones es de 1, 2 y 3 bytes, donde el primer byte representa el código de operación, el segundo el primer operando y el tercero el último operando.

Soporta los siguientes modos de direccionamiento: inmediato, registro, directo, indirecto basado en el registro BX, indirecto + desplazamiento.

6. Las características del desarrollo del producto software es considerada de nivel buena, posee Licencia MIT, fecha de última versión 2015, desarrollado sobre plataforma web, no hay registro de uso académico.

7. La cobertura de los contenidos preestablecidos en este criterio es baja, debido a que no implementa memoria independiente para los módulos de entrada y salida, rutinas de tratamientos de interrupciones, ciclo de instrucción y medidas de rendimiento.

VonSim

1. Nivel medio de usabilidad, presenta todos los componentes del simulador mediante solapas, pero aun así, muestra todos los componentes disponibles al usuario inicial, esto puede implicar un exceso de información.

El formato de representación de valores para la memoria principal muestra en hexadecimal; sin embargo, al ubicarse sobre la posición se despliega su valor equivalente en formato binario, complemento a dos, decimal sin signo, ASCII. Sin embargo, en los registros del CPU solo se muestra en formato hexadecimal.

Es difícil vincular etiquetas de un programa con sus correspondientes direcciones de destino, para ello se debe buscar el destino del salto en la memoria principal.

Implementa la sintaxis de lenguaje ensamblador basada en MASM, pero no incluyen directivas para definir las partes de un programa ejecutable, sino que en su lugar aconseja almacenar datos del programa a partir de la posición de memoria 1000h e instrucciones a partir de la posición 2000h. Lo que dificulta poder llegar a implementar los programas en alguna máquina real sin antes incorporar estás directivas.

Las interrupciones por software preestablecidas en el simulador no son compatibles con algún sistema operativo conocido.

Permite cambiar el idioma del simulador inglés o español.

2. Las herramientas que ofrece el editor del simulador son consideradas de nivel medio. Incluye aviso de errores de sintaxis al momento de ensamblar y al terminar de escribir una instrucción, posee resaltado de sintaxis en la instrucción, no posee la opción de breakpoints interactivo sino mediante interrupción por software. El simulador no brinda un repositorio o mecanismos para guardar y cargar programas.

3. La documentación es considerada de nivel media, posee manual de uso y tutorial para el aprendizaje interactivo con ejemplos y auto evaluaciones. Consta de un repertorio de instrucciones implementadas alojadas en el repositorio donde se encuentra el simulador, esto puede ser de difícil acceso para el usuario inicial.

4. El control de la simulación es de nivel medio, permite configurar la velocidad del reloj de CPU (1 Hz, 4 Hz, 8 Hz, 16 Hz y 32 Hz), posee resaltado de línea de ejecución en curso, en cuanto a los controles de simulación consta de un botón para cargar en memoria y ejecución completa del programa (ejecución rápida) y los botones abort/debug, stop y step para controlar la simulación. No posee control de simulación para volver a la instrucción anterior (step back).

5. El detalle de la arquitectura es de nivel medio, el simulador representa una simplificación del procesador 8088 que es compatible con la arquitectura x86, posee una arquitectura interna de 16 bits y externa de 8 bits, es decir, su bus de datos es de 8 bits, sin embargo, el CPU puede manejar valores de un byte y dos bytes (word), lo que implica que se puede ejemplificar el concepto de *little-endian*.

Los registros del CPU son de 16 bits, cuatro registros de propósito general (AX BX CX DX), que pueden ser tratados también como registros de 8 bits (AH, AL, BH, BL...), un registro puntero de instrucción (IP), un puntero de pila (SP), un registro buffer de direcciones (MAR), un registro buffer de datos (MBR), un registro de instrucciones (IR) y un registro de estado que representa los flags Zero (Z), Signo (S), Carry (C), Overflow (O) y Interrupt (I). La memoria direccionable 16 KiB. Cuenta tanto con dispositivos internos: PIO, TIMER, PIC, CDMA. Y externos: llaves de luces y led, monitor, teclado, impresora.

El procesador 8088 puede direccionar 2^{16} posiciones de memoria (64 KiB); sin embargo, el simulador permite direccionar 2^{14} posiciones de memoria (16 KiB) representada en un vector 16384 posiciones de 1 byte cada una.

El simulador implementa las interrupciones por hardware que son disparadas por dispositivos y gestionadas a través del PIC, las subrutinas de atención de la interrupción por hardware deben ser implementadas por el usuario.

Permite configurar el PIC para gestionar las interrupciones por hardware, acceder a los puertos de los dispositivos internos mediante las instrucciones IN y OUT.

Los dispositivos internos que posee son:

- **PIO**: su comportamiento es similar al Intel 8255 programado en modo 0. Puerto paralelo de entrada/salida.
- **HAND-SHAKE**:similar al modo 2 del Intel 8255. Periférico de Handshaking.
- **Controlador de interrupciones(PIC):** pretende asemejarse al Intel 8259. Controlador de interrupciones.
- **Controlador de acceso directo a memoria (DMA)**: semejante al Intel 8237. Controlador de Acceso Directo a Memoria.
- **Timer**: contador de eventos.

Los dispositivos externos que posee son:

- **Barra de leds**.
- **Barra de microconmutadores**.
- **Impresora**: interfaz Centronics.
- **Tecla F10**.
- **Teclado**.
- **Monitor**.

Las interrupciones por software sirven para hacer llamadas al sistema. Se invocan desde el programa mediante la instrucción int N, y hacen que se ejecute una subrutina del sistema operativo (cuyo código está oculto al usuario), posee 4 subrutinas que se invocan de la siguiente manera:

- **INT 0**: termina el programa.
- **INT 3**: colocar un punto de parada (breakpoint).
- **INT 6**: leer caracteres por teclado, la subrutina espera para leer un carácter del teclado y lo almacena en la posición de memoria cuya dirección se indica en el registro BX.
- **INT 7**: escribe una cadena de caracteres por pantalla, recibe como parámetro en el registro BX la dirección de comienzo de la cadena, y en el registro AL la cantidad de caracteres que componen la cadena.

Tanto las interrupciones por software y hardware, el mecanismo mediante el cual la CPU produce la ejecución de una subrutina en base al identificador de interrupción es el mismo, se busca en el vector de interrupción es la dirección de la subrutina de tratamiento de la interrupción.

La tabla de vector de interrupciones es el nexo entre un tipo de interrupción (0...255) y el procedimiento que ha sido designado para atenderla. Cada interrupción vectorizada tiene un código que la identifica para la CPU. Existen por lo tanto 256 entradas en el vector, una para cada tipo de interrupción, 4 están destinadas para las interrupciones por software o llamadas al sistema operativo, quedando 251 interrupciones libres en el vector de interrupciones para ser utilizadas por el usuario.

La tabla de vector de interrupciones se ubica en los primeros 1024 bytes de la memoria y cada entrada de la tabla ocupa 4 bytes. Por ende, para ubicar una entrada en la tabla el CPU necesita realizar la siguiente operación $4*N$, donde N es el tipo de interrupción para la CPU.

En el bloque de la ALU se muestra la operación aritmética o lógica en curso y tres registros internos de almacenamiento temporal de 16 bits representados en formato binario.

En cuanto al formato de las instrucciones pueden ser de 1, 2, 3, 4 y 6 bytes, y respetan el formato de la arquitectura x86.

Soporta los siguientes modos de direccionamiento: inmediato, registro, directo e indirecto basado en el registro BX.

6. Las características del desarrollo del producto software es considerada de nivel muy buena, posee Licencia GNU Affero General Public License v3.0, fecha de última versión 2020, desarrollado sobre plataforma web y una extensa evidencia de uso académico.

7. La cobertura de los contenidos preestablecidos en este criterio es media, debido a que no implementa de manera visual herramientas para desarrollar contenidos como ciclo de instrucción y medidas de rendimiento.

Emu8086

1. Nivel fácil de usabilidad debido a que al iniciarse el simulador solo aparece el editor, y al cargar el programa en memoria (emulate), se muestran los registros del CPU y la memoria, luego el usuario puede ir activando los distintos componentes del simulador.

El formato de representación de valores para la memoria principal se muestra en hexadecimal, decimal y ASCII, pero al hacer clic sobre la posición de memoria se puede ver en binario, decimal con signo/sin signo, octal y ASCII agrupado de 8 y 16 bits. Sin embargo, en los registros del CPU solo se muestra por default en formato hexadecimal, pero al hacer clic sobre el registro se puede ver en binario, decimal con signo/sin signo, octal y ASCII agrupado de 8 y 16 bits. Actualiza los valores de los registros del CPU y la memoria principal durante la ejecución de cada instrucción.

Muestra las instrucciones de assembler pero con las direcciones y etiquetas ya resueltas por parte del ensamblador (código desensamblado).

Implementa la sintaxis de lenguaje ensamblador compatible con MASM/TASM, y mediante directiva es compatible con Flat Assembler (FASM).

El simulador ofrece plantillas para los siguientes tipos de estructura de programas COM, EXE, BIN y BOOT.

El emu8086 permite conectar con dispositivos virtuales y simular una comunicación con el espacio de E/S. Para esto, el emu8086 cuenta con una serie de dispositivos virtuales preexistentes en el software base, listos para ser utilizados, entre los que se encuentran una impresora, floppy disk, termómetro y calentador, control de tráfico mediante semáforos, un motor paso a paso, etc. No obstante, permite agregar nuevos dispositivos adicionales con características particulares.

2. Las herramientas que ofrece el editor del simulador son consideradas de nivel alto. Incluye aviso de errores de sintaxis únicamente al momento de ensamblar, posee resaltado de sintaxis en la instrucción, posee la opción de breakpoints cuando se está depurando programa o ejecutar el programa hasta el lugar donde se encuentra el cursor (“run until”). Al ser un simulador de escritorio permite guardar y cargar los programas generados. Luego ejecutar una instrucción el valor de los registros del CPU que se hayan modificado se verán resaltados en azul.

3. La documentación es de nivel completo, consta de un manual de instrucciones e interrupciones con ejemplo de uso, posee tutorial para el aprendizaje con ejemplos, que incluye un manual de uso.

4. El control de la simulación es de nivel alto. El simulador permite configurar la velocidad del reloj de CPU (1 ms, 100 ms, 200 ms, 300 ms y 400 ms), posee resaltado de línea de ejecución en curso, en cuanto a los controles de simulación consta de un botón “emulate” que carga en memoria el programa y abre el emulador, el emulador posee los siguientes controles load, reload (reinicia el programa y comienza a ejecutar el mismo desde la primer instrucción de código y todos los registros inicializan nuevamente), single step, step back (retrocede una instrucción) y run/stop. Se considera que tiene un nivel de ejecución de la simulación media.

Durante la ejecución de un programa se puede modificar dinámicamente el contenido de los registros del CPU, esto es útil cuando se necesita forzar alguna situación particular durante la ejecución del programa.

El simulador permite ver el estado de los Flag, mapa de memoria, estado de la pila, variables, etc.

5. El detalle de la arquitectura es de nivel completo, ya que posee un nivel de especificación detallada de la arquitectura del procesador 8086, que consta de una arquitectura interna de 16 bits, un bus de datos es de 16 bits y un bus de direcciones de 20 bit que permite direccionar 2^{20} posiciones de memoria (1 MiB). Por lo tanto, el CPU puede manejar valores de un byte y 2 bytes (una word), lo que implica que se puede ejemplificar el concepto de *little-endian*.

El software representa de manera exacta los componentes y el funcionamiento del procesador 8086, los registros del CPU son de 16 bits, cuatro registros de propósito general (AX BX CX DX), que pueden ser tratados también como registros de 8 bits (AH, AL, BH, BL...), un registro puntero de instrucción (IP), un puntero base (BP), un puntero de pila (SP), un índice fuente (SI), un índice destino (DI), un segmento de datos (DS), un segmento extendido (ES), un segmento de pila (SS), segmento de código (CS) y un registro de estado que representa los flags Zero (Z), Signo (S), Auxiliary Carry (A), Overflow (O), Carry (C), Direction (D), Interrupt (I), Trap (T) y Parity (P). La memoria direccionable 1 MiB.

También se representa la tabla de vector de interrupciones (Interrupt Vector Table), donde los tipos de interrupciones van de 0 a 255, esta tabla cuenta con 256 posiciones, y en cada una de ellas se encuentran dos valores CS e IP.

Se implementan interrupciones por software (INT identificador), antes de poder ejecutar la subrutina asociada a una interrupción específica debe obtener el CS e IP donde se encuentra la misma, para ello, el procesador debe calcular la entrada de la tabla del vector de interrupciones asociado con esa interrupción, multiplica el identificador de interrupción por cuatro, por ejemplo interrupción por software INT 5, se calcula 4 x 5 = 20 (14h), este resultado será la dirección física en la cual se encuentra el IP (00014h) y dos posiciones más arriba el CS (00016h).

Se emula el manejo de excepciones que permiten resolver situaciones de error al ejecutar una determinada instrucción, por ejemplo, la presencia de una división por cero dispara automáticamente una interrupción con identificador 0.

Se permite emular interrupciones por hardware, representa 256 interrupciones mediante el archivo “emu8086.hw”, si en la posición se almacena 1 indica que se generó una interrupción por hardware. Las interrupciones por hardware se encuentran deshabilitadas cuando el Flag de Interrupciones (IF) se encuentra en 0. Cuando el IF está en 1, el emu8086 verifica continuamente los primeros 256 bytes del archivo “emu8086.hw”, si alguno de los bytes leídos es distinto de cero, por ejemplo el byte 15, el procesador transfiere el control a la subrutina de atención de la interrupción con el identificador 15 en base a lo configurado en la IVT (Interrupt vector table).

Cuando se produce una interrupción por hardware el simulador lo indica mediante un mensaje.

El emu8086 permite utilizar el espacio de E/S y emular la comunicación con los dispositivos virtuales mediante las instrucciones IN y OUT. Para esto, el emu8086 utiliza el archivo “emu8086.io” como medio común para la comunicación con los dispositivos virtuales, permitiendo gestionar 65535 puertos de un byte cada uno, tanto procesador como los dispositivos virtuales leen y escriben en este archivo, así por ejemplo el puerto 100 de E/S corresponde con el byte 100 del archivo “emu8086.io”.

La comunicación entre los dispositivos externos y el procesador se realiza a través de los archivos “emu8086.io” y “emu8086.hw”, a través de las instrucciones IN y OUT.

Permite emular el booteo de una IBM PC desde el floppy disk.

En cuanto al formato de las instrucciones pueden ser de 1, 2, 3, 4 y 6 bytes, y respetan el formato de la arquitectura x86.

Soporta los diferentes modos de direccionamiento implementados en el procesador 8086: inmediato, registro, directo e indirecto basado en los siguientes registros BX o BP más un desplazamiento en los siguientes registros SI o DI, más un desplazamiento inmediato de 8 o 16 bits.

6. Las características del desarrollo del producto software es considerada de nivel mala. El simulador tiene un límite de hasta 14 días de período de evaluación gratuito, después del período de evaluación debe comprar la clave de licencia digital. Existe evidencia de su uso en el ámbito académico.

7. La cobertura de los contenidos preestablecidos en este criterio es media, debido a que no implementa manera visual para desarrollar contenidos como ciclo de instrucción y medidas de rendimiento.

Tabla comparativa según criterios de evaluación preestablecidos.

Criterios de evaluación	Simple 8-bit	VonSim	Emu 8086
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. CONCLUSIONES

A lo largo de este trabajo hemos ido siguiendo la hipótesis inicial que el uso de los simuladores en el aula juegan una pieza clave en el campo de la Arquitectura de Computadores.

A partir de los criterios de evaluación definidos se buscaron herramientas que abarquen o se ajusten a la mayor cantidad de tópicos de la asignatura Arquitectura de Computadoras.

Al evaluar los simuladores seleccionados según los criterios preestablecidos se concluyó, que el simulador simple 8-bit carece de diferentes recursos necesarios para desarrollar los diferentes tópicos que se dictan en la asignatura de Arquitectura de Computadoras.

El simulador VonSim se caracteriza por ser un producto software que se encuentra en continuo desarrollo y con un extenso historial académico y se destaca la variedad de dispositivos internos implementados. Por lo tanto, se considera una buena opción para el dictado de los contenidos de la arquitectura x86.

Como punto en contra, es que no representa la arquitectura x86 completa, esto limita poder volcar directamente los programas generados a una máquina real. Es muy difícil poder llegar implementar en una máquina hardware real los programas escritos sin hacer las correspondientes modificaciones, como ser, la manera de llamar a los servicios de interrupciones que dependen del sistema operativo y directivas del lenguaje ensamblador para generar archivos ejecutables para un sistema operativo. Y por último, no cubre todos los contenidos preestablecidos en la asignatura.

El emulador emu8086 tiene como punto a favor que representa de manera exacta las características del procesador 8086 y el sistema operativo MS-DOS, lo que permite correr los programas escritos en el simulador directamente en una máquina real.

Como puntos en contra, funciona solo bajo el sistema operativo Windows como aplicación de escritorio. Implementa el servicio de interrupciones del sistema operativo MS-DOS, por lo tanto, sólo permite crear programas para una máquina que ejecute dicho sistema operativo. No permite escribir programas en lenguaje ensamblador NASM.

El sitio web y mail de contacto del producto no funcionan o responden, y se desconocen las últimas actualizaciones del producto. Además, no cubre todos los contenidos preestablecidos en la asignatura.

Teniendo en cuenta el trabajo de investigación, queda abierto al desarrollo de un simulador que pueda funcionar sobre una plataforma web, que implemente servicio de interrupciones de un sistema operativo Linux, que permita programar en lenguaje ensamblador NASM, que esté organizado en capas, partiendo de una arquitectura reducida x86, donde a medida que se avanza en

complejidad se habiliten más capas según su complejidad. Por ejemplo, agregar instrucciones, llamada a subrutinas, manejo de la pila, manejo de interrupciones, comunicación con los módulos de E/S y dispositivos externos, etc. Así se reduciría la complejidad del simulador a primera vista y a medida que se avanza con los diferentes temas de la asignatura ir habilitando las diferentes vistas en el simulador, que presente un nivel de granularidad necesario para representar los micropasos que conlleva el ciclo básico de la instrucción (Búsqueda y ejecución) mostrando los componentes que intervienen por cada ciclo de reloj, permitiendo ver gráficamente estos micropasos a través de la composición de la ruta de datos de una CPU, donde se pueda consultar los datos que se transmiten en los buses y las señales de control para cada instrucción, ya que la visualización del ciclo de instrucción mejora el proceso de aprendizaje, y por último, que permita abordar los diferentes tópicos de la asignatura Arquitectura de computadoras.

4. BIBLIOGRAFÍA

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

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

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

[4] C. Lion, «Los simuladores. Su potencial para la enseñanza universitaria», Cuad. Investig. Educ., vol. 2, n.o 12, pp. 53–66, 2005.

[5] G. Contreras, R. G. Torres, y M. S. R. Montoya, «Uso de simuladores como recurso digital para la transferencia de conocimiento», Apert. Rev. Innov. Educ., vol. 2, n.o 1, pp. 86–100, 2010.

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

[7] B. Mustafa, «Evaluating A System Simulator For Computer Architecture Teaching And Learning Support», Innov. Teach. Learn. Inf. Comput. Sci., vol. 9, n.o 1, pp. 100-104, 2010.

[8] Z. Radivojevic, M. Cvetanovic, y J. Đordevic, «Design of the simulator for teaching computer architecture and organization», en 2011 Second Eastern European Regional Conference on the Engineering of Computer Based Systems, 2011, pp. 124–130.

[9] B. Nikolic, Z. Radivojevic, J. Djordjevic, y V. Milutinovic, «A Survey and Evaluation of Simulators Suitable for Teaching Courses in Computer Architecture and Organization», IEEE Trans. Educ., vol. 52, n.o 4, pp. 449-458, nov. 2009.

[10] R. Hasan y 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», en Control (CONTROL), 2012 UKACC International Conference on, 2012, pp. 1043–1045.

[11] J. L. Hennessy y D. A. Patterson, Computer architecture: A quantitative approach, Fifth Edition. Elsevier, 2012.

[12] W. Stallings, Computer organization and architecture: designing for performance, Eleventh Edition. Pearson, 2013.

[13] P. BEHROOZ, Computer Architecture From Microprocessors to Supercomputers. McGraw Hill, 2007.

[14] Intel, «64 and IA-32 architectures software developers manual», 325462-060US, vol. 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C and 3D, p. 4670, 2016.

[15] AMD, «Developer Guides, Manuals & ISA Documents». [En línea]. Disponible en: https://developer.amd.com/resources/developer-guides-manuals/. [Accedido: 21-abr-2019].

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

[17] D. Skrien, «CPU Sim 3.1: A tool for simulating computer architectures for computer organization classes», J. Educ. Resour. Comput. JERIC, 2001.

[18] A. Akram y L. Sawalha, «A comparison of x86 computer architecture simulators», 2016.