APLICACIÓN PARA LA SIMULACIÓN Y APRENDIZAJE DEL FUNCIONAMIENTO DE UN MICROPROCESADOR SEGMENTADO

ANTONIO MORENO, MIGUEL ACHAERANDIO[1]
[1]*Dpto de Ingeniería Eléctrica y Electrónica. Universidad Pública de Navarra. Pamplona, España.*
Correo electrónico: acha@unavarra.es

Se presenta el diseño y desarrollo de la aplicación SimIVY, que simula el funcionamiento de una arquitectura RISC con segmentación ("pipelining") de forma gráfica, con el objeto de ser empleada como recurso didáctico en la enseñanza de arquitectura de computadores. La herramienta puede ser usada tanto en el aula como individualmente por el alumno.

1. Introducción

La arquitectura de computadores suele aprenderse mediante el estudio teórico de la materia y la aplicación de conocimientos en un microprocesador concreto. Usualmente ni los alumnos disponen de tiempo suficiente ni los centros poseen los recursos como para trabajar con varias arquitecturas diferentes con las cuales asimilar completamente la materia y así, en muchas ocasiones, la idea formada del funcionamiento básico de los ordenadores adolece de fallos conceptuales[1]. Para facilitar la docencia existe un buen número de herramientas que simulan el comportamiento de diversos microprocesadores, reales o ficticios[2,3]. Los simuladores de microprocesador, de los cuales existe una amplia oferta, son preferidos a los propios micros reales, ya que permiten la detección de errores, facilitan la manipulación de datos sin modificar elementos físicos y ofrecen en general mayores posibilidades. De este modo se permite a los alumnos incorporar programas propios escritos en lenguaje ensamblador y comprobar los resultados que produciría su ejecución sin los peligros de usar una máquina real.

Esta aproximación permite que los alumnos asimilen en relativamente poco tiempo algunos conceptos básicos de las arquitecturas de computadores, pero tiene la desventaja de que los alumnos no acaban de comprender los procesos reales que ocurren en el interior de un microprocesador, puesto que éste es visto como una "caja negra" de la cual no se sabe nada de su contenido. En otras palabras, con la mayoría de las herramientas tradicionales, los alumnos pueden entender *qué* ocurre, pero no *cómo* se llega a los resultados de la ejecución. En particular, no suele mostrarse el camino de datos de las instrucciones.

Por otro lado, muchas de estas aplicaciones se diseñaron hace bastante tiempo, y por ello su interfaz suele ser poco claro, limitándose en la mayoría de las ocasiones a mostrar los cambios producidos en los distintos registros internos accesibles por el programador. Pocas permiten mostrar los cambios en el estado de los elementos internos de un micro de forma visual, a pesar de que una representación gráfica animada permite ilustrar conceptos tales como la segmentación (*pipelining*) de forma sencilla. Además, la mayoría de las aplicaciones existentes deben ser ejecutados en un sistema operativo determinado, lo que limita su difusión.

Para tratar de solventar alguno de los inconvenientes encontrados, se ha diseñado la arquitectura denominada *IVY* que junto con la aplicación *SimIVY* permiten mostrar de forma gráfica cada una de los elementos de los que se compone el microprocesador, de forma que el alumno pueda analizar los cambios que se producen a medida que se simula la ejecución de instrucciones, así como introducir modificaciones en los componentes o las señales internas del sistema, comprobando los resultados que se obtienen al proseguir la simulación. El microprocesador simulado utiliza la

técnica de ejecución con segmentación (*pipelining*), aunque también soporta modos de ejecución más sencillos (mono y multiciclo) que pueden servir para introducir a los alumnos en su diseño. Obviamente, la arquitectura desarrollada no existe en el mercado, sino que incorpora elementos de varias ya existentes para poder explicar adecuadamente algunos conceptos. Ello permite generalizar los conocimientos adquiridos al precio de no poder aplicar los detalles específicos directamente a un micro real.

Se ha escogido el entorno de desarrollo Java de Sun Microsystems, y el proceso de desarrollo ha seguido las técnicas de diseño orientado a objetos. Al estar implementado mediante Java, *SimIVY* puede funcionar en cualquier combinación sistema operativo/computador que disponga de una máquina virtual. En la práctica, todos los sistemas operativos comunes actuales disponen de una y, por ello, *SimIVY* puede utilizarse en la mayoría de las situaciones, incluso en entornos heterogéneos en los que coexistan ordenadores con sistemas operativos diferentes.

2. Arquitectura desarrollada

El microprocesador ficticio *IVY-1* desarrollado se basa en una arquitectura RISC (R*educed* I*nstruction* S*et* C*omputer*) de 32 *bits* y de tipo *LOAD-STORE*, muy similar en el diseño de los formatos de instrucción a la arquitectura MIPS (M*icroprocessor without* I*nterlocked* P*ipeline* S*tages*)[4,5]. El conjunto de instrucciones y su codificación se han diseñado específicamente para la aplicación, y han sido elegidos de modo que sean fácilmente asimilables por un alumno con unos mínimos conocimientos teóricos.

IVI-1 es un microprocesador segmentado, pero soporta modos de ejecución más sencillos (mono y multiciclo), que permiten trabajar tanto sin el concepto de segmentación como su introducción gradual. Los principios fundamentales del diseño, el proceso seguido para la codificación de las distintas instrucciones y la elección del *hardware* así como el lenguaje ensamblador desarrollado están documentados en el manual de usuario, de modo que unos conocimientos someros de electrónica digital permiten entender los principales problemas que aparecen durante el desarrollo de un microprocesador y las decisiones y compromisos necesarios.

2.1. Especificaciones de la arquitectura

Las especificaciones principales son las siguientes:

- Arquitectura del tipo *LOAD-STORE*.
  Esto significa que las únicas instrucciones que pueden acceder a la memoria son precisamente las instrucciones *LOAD* y *STORE*, y que el resto de instrucciones (aritméticas, lógicas,...) utilizan direccionamiento Registro-Registro. Esta decisión está relacionada con la idea de escoger instrucciones que realicen una sola operación, para facilitar el diseño de un microprocesador segmentado.
- Por sencillez, sólo incluye una unidad de enteros. No permite trabajar en punto flotante.
- Se utilizan registros de propósito general, que sirven tanto para almacenar direcciones de memoria, como datos.
- Existen muy pocos formatos de instrucción.
  Esta decisión tiene dos ventajas: por una parte, se simplifica su aprendizaje a los alumnos; por otra, se facilita enormemente la descodificación de instrucciones por parte del microprocesador.
- Las instrucciones tienen un tamaño fijo y están alineadas en la memoria.

  Con ello el diseño del procesador es mucho más simple: puesto que se sabe siempre dónde comienza y dónde termina cada instrucción, es más sencillo leerlas de la memoria.

Además de estos requisitos iniciales, se añadieron algunos impuestos por el diseño de las arquitecturas reales o por la búsqueda de la facilidad en la comprensión de la arquitectura:

- La arquitectura dispone de una pila controlada por su propio registro.
  Este mecanismo ha ido desapareciendo, pero sigue siendo utilizado en los microprocesadores basados en arquitecturas más antiguas, así que se incluye para poder explicarlo a los alumnos.
- Direccionamiento absoluto
  Aunque el direccionamiento absoluto de los datos en memoria con las instrucciones LOAD y STORE es imposible en toda la memoria (debido al uso de palabras de instrucción de tamaño fijo, de 32 *bits*), se ha introducido un direccionamiento absoluto limitado, pensando en su sencillez inicial para los alumnos.

2.2. Registros de la arquitectura *IVY*

IVY posee 32 registros de propósito general de 32 *bits*. Además, existe un registro contador de programa que almacena la dirección de la siguiente instrucción que se va a ejecutar. Se ha incluido un registro específico para implementar la pila (ello permite simplificar el banco de registros de propósito general, ya que un registro específico hace innecesario poseer dos puertos de escritura para las instrucciones de carga de datos de la pila).

Además, existe un registro de estado de cuatro *bits*, que almacena información sobre la última instrucción ejecutada, y dos registros adicionales, RHI, y RLO, que permiten almacenar el resultado de algunas operaciones aritméticas (las multiplicaciones y las divisiones).

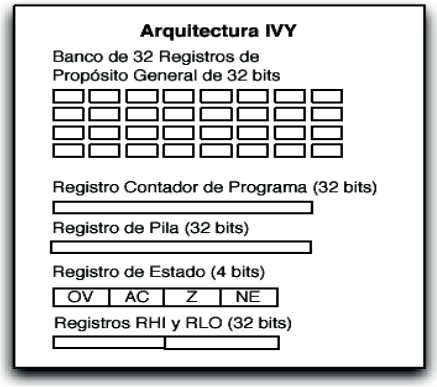

Figura 1. Registros de la arquitectura *IVI*

2.3. Datos

La arquitectura soporta tres tipos de tamaños de datos:

Tamaño	Número de bits	Direcciones
.b (byte)	8 bits	Cualquier dirección
.hw (media palabra)	16 bits	Direcciones acabadas en 0
.w (palabra)	32 bits	Direcciones acabadas en 00

Tabla 1. Tamaños de datos de la arquitectura *IVI*

Los datos se alinean en la memoria (lo cual simplifica el acceso a la memoria y, por lo tanto, el diseño del microprocesador), y siguen el ordenamiento *big endian (extremante mayor)*. Para que el alumno asimile las implicaciones del modo de ordenamiento, éste puede cambiarse de forma simple. El bus de datos externo es de 32 *bits*.

2.4. Formatos de instrucción

Dado que parte del objetivo es hacer comprender no sólo el funcionamiento sino las opciones de diseño, también se explican en el manual los formatos de instrucción. Se han usado tres formatos, muy similares a los usados en la arquitectura MIPS, y denominados A, B y C.

El formato A tiene tres campos para operandos, y permite especificar instrucciones con dos operandos de lectura, y uno de destino. Por ello, es el utilizado para las instrucciones aritméticas y lógicas comunes. Además, posee 6 *bits* destinados a una extensión del código de operación, y por eso, es el formato utilizado cuando la instrucción no requiere ninguna característica especial (para así dejar libre más espacio de códigos de operación).

El formato B incorpora dos campos para operandos, y un campo con un desplazamiento (o valor inmediato). Este formato se utiliza para las instrucciones LOAD y STORE (que utilizan direccionamiento relativo a un registro), y también para las aritméticas y lógicas que utilizan direccionamiento inmediato.

El formato C simplemente incorpora un gran campo con un desplazamiento, y se utiliza para las operaciones de salto.

Formato A: instrucciones entre registros y de manejo de la pila

opcode	Tipo de operación	R1	R2/n bits	RR
6 *bits*	11 *bits*	5 *bits*	5 *bits*	5 *bits*

Formato B: instrucciones de acceso a memoria y de valor inmediato

opcode	RB	Desplazamiento	RR
6 *bits*	5 *bits*	16 *bits*	5 *bits*

Formato C: instrucciones de salto

opcode	Desplazamiento
6 *bits*	26 *bits*

Tabla 2. Formatos de instrucción de la arquitectura *IVI*

2.5. Repertorio de instrucciones

Juego de Instrucciones completo			
Formato A		Formato B	Formato C
MOVE.b	OR.b	LOAD.b (sin registro)	JUMP
MOVE.hw	OR.hw	LOAD.hw (sin registro)	JSUB
MOVE.w	OR:w	LOAD.w (sin registro)	JAC
MOVEHI	NOT.b	LOAD.b (con registro)	JOV
MOVELO	NOT.hw	LOAD.hw (con registro)	JZ
ADD.b	NOT.w	LOAD.w (con registro)	JNEG
ADD.hw	SHIFTL.b	STORE.b (sin registro)	JSAC
ADD.w	SHIFTL.hw	STORE.hw (sin registro)	JOSV
SUB.b	SHIFTL.w	STORE.w (sin registro)	JSZ
SUB.hw	SHIFTR.b	STORE.b (con registro)	JSNEG
SUB.w	SHIFTR.hw	STORE.hw (con registro)	RET
MULU.b	SHIFTR.w	STORE.w (con registro)	RETE
MULU.hw	ROTL.b	LOADST	
MULU.w	ROTL.hw	STOREST	
MULS.b	ROTL.w	MOVEI	
MULS.hw	ROTR.b	MOVEIST	
MULS.w	ROTR.hw	ADDI.b	
DIVU.b	ROTR.w	ADDI.hw	
DIVU.hw	PUSH.b	ADDI.w	
DIVU.w	PUSH.hw	SUBI.b	
DIVS.b	PUSH.w	SUBI.hw	
DIVS.hw	POP.b	SUBI.w	
DIVS.w	POP.hw	ANDI	
AND.b	POP.w	ORI	
AND.hw		JUMPR	
AND.w		JSUBR	

Tabla 3. Repertorio de instrucciones de la arquitectura *IVI*

3. El microprocesador *IVI*-1

IVY-1 es un microprocesador con 5 etapas de segmentación, de ejecución estrictamente en orden, y que ejecuta instrucciones de la arquitectura *IVY* descrita previamente. La segmentación viene dada porque la ejecución de las instrucciones está dividida en varias fases independientes entre sí.

Figura 2. Etapas de segmentación del cauce de *IVI-1*

Como ya se ha indicado, *IVY-1* puede funcionar en tres modos de ejecución diferentes que hemos denominado monociclo, multiciclo y *pipelining*. A pesar de ser un diseño único, estos tres modos permiten al *IVY-1* comportarse de forma muy diferente.

- En modo *monociclo IVY-1* se comporta de forma efectiva como si no fuera un diseño segmentado. Cada instrucción tarda exactamente un ciclo en completarse, y en ese ciclo recorre las cinco fases de segmentación.
- En modo *multiciclo IVY-1* se comporta como un diseño segmentado, pero no permite que haya más de una instrucción a la vez en el cauce de ejecución. De esta forma, cada instrucción tarda exactamente cinco ciclos en ejecutarse, y deben pasar cinco ciclos desde que entra una instrucción a ejecutarse, hasta que puede entrar la siguiente.
- En modo *pipelining IVY-1* se comporta como un diseño segmentado, permitiendo además que, a medida que las instrucciones anteriores progresan en su ejecución, vayan entrando nuevas instrucciones en el cauce. En este modo, cada instrucción tarda cinco ciclos en ejecutarse, y puede haber hasta cinco instrucciones en el cauce a la vez (una en cada etapa).

El microprocesador se diseñó desde el principio para este último método de ejecución, pero dado que el modo *pipelining* es el más avanzado de los tres y por lo tanto también el más complejo, se juzgó necesario incorporar modos de ejecución adicionales para facilitar la labor de la explicación del proceso de ejecución. Con la presencia de los tres modos, el profesor puede elegir cuál se ajusta mejor en cada momento a lo que pretende explicar.

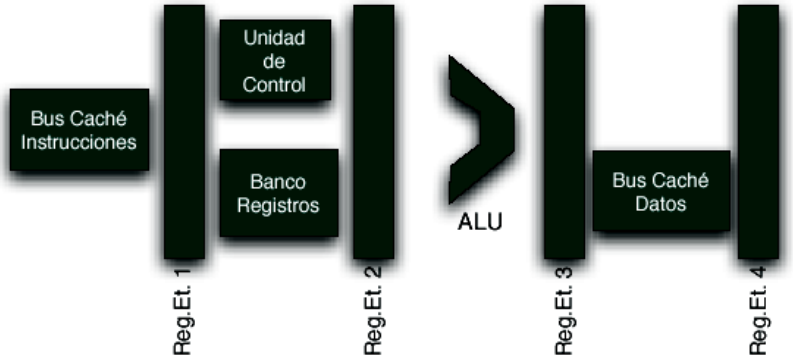

Figura 3. Esquema simplificado del microprocesador *IVY-1*

3. Herramienta *software* de simulación

SimIVY ha sido desarrollado en Java por ser éste un lenguaje gratuito multiplataforma. Presenta una interfaz gráfica sencilla e intuitiva que permite ocultar y mostrar distintas partes del diseño, utiliza un código de colores para mostrar los cambios en el estado de los componentes y posee elementos de ayuda en línea que describen la función que ejerce cada elemento del micro. Permite al usuario manipular prácticamente todos los elementos que componen el *IVY-1* (registros internos específicos, registros de propósito general, señales y buses internos), así como editar la memoria principal del sistema para incorporar instrucciones o datos a la misma, de forma manual o semiautomatizada. Cualquier dato puede verse en binario, decimal o hexadecimal.

Se permite direccionar cualquier posición de memoria en todo el rango de 0 a 2^{32}. Dado el objetivo docente del simulador el uso de tanta memoria sería absurdo y, además, incluirla memoria en el simulador sería inmanejable. Aún así, el programa permite trabajar con ella mediante el uso de ficheros que contienen bloques dinámicos de memoria.

La aplicación simula la ejecución de instrucciones en los tres modos diferentes de ejecución ya descritos en la sección anterior, de forma continua o paso a paso. Además, permite volver a estados anteriores de ejecución del microprocesador deshaciendo los cambios realizados, así como grabar el estado ya sea del sistema completo o sólo de la memoria para su uso posterior. Las instrucciones pueden introducirse directamente en la memoria en su codificación numérica, o bien escribirse mediante los mnemónicos correspondientes. Además, pueden crearse programas escritos en ficheros de texto, que son ensamblados por el propio *SimIVY*.

El programa dispone de dos modos de visualización, entre los que puede cambiarse en cualquier momento. El modo externo permite ver al microprocesador desde un punto de vista exterior, y sus conexiones con la memoria.

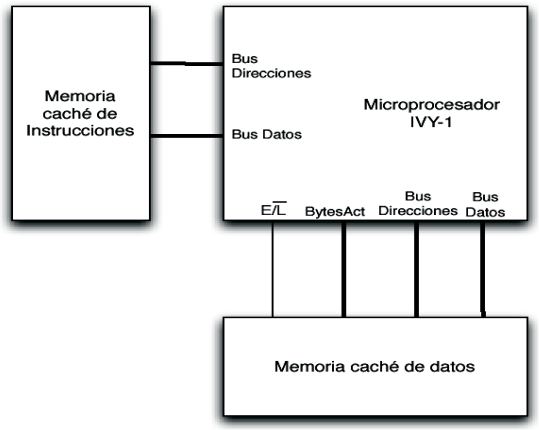

Figura 4. Vista externa de *SimIVY*

El modo interno permite ver todos los registros, los buses y señales internas, y los elementos principales de los que consta el diseño (unidades aritmético-lógicas, unidad de control,...). En cada momento, el usuario puede ver el valor de los registros internos, así como el valor de las señales, simplemente seleccionándolas con el ratón.

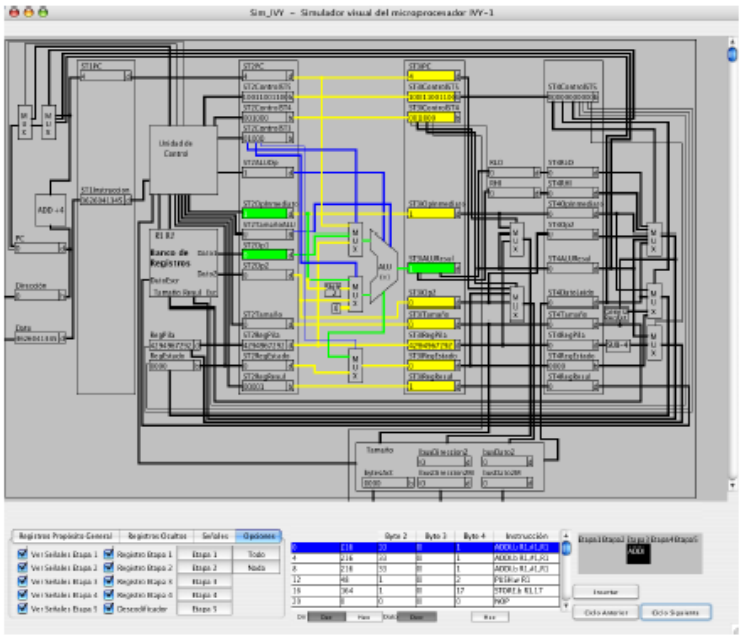

Figura 5. Vista interna de *SimIVY*, en modo multiciclo

En cada modo de ejecución distinto los códigos de colores y los mensajes del sistema son diferentes. En el modo *pipelining* se usan códigos de colores específicos para cada una de las instrucciones que están simultáneamente en ejecución en el microprocesador.

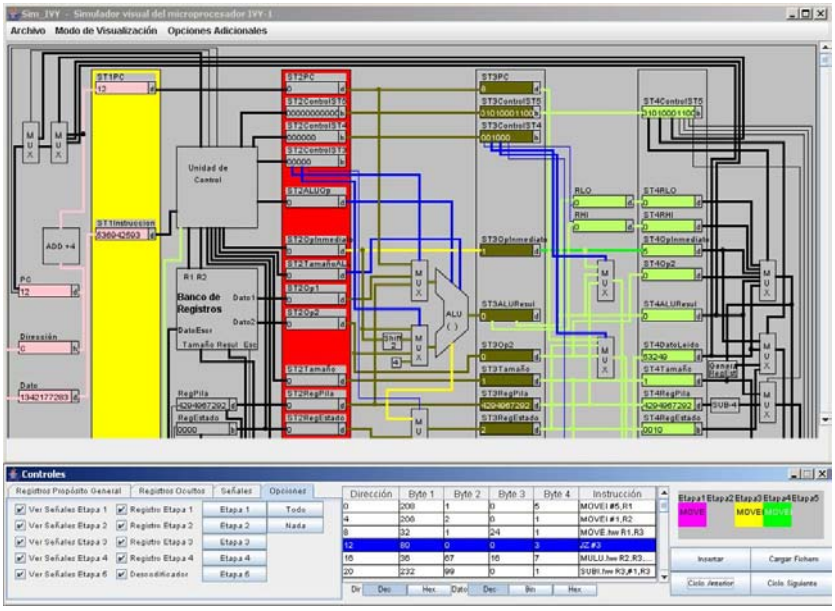

Figura 6. Vista interna de *SimIVY,* en modo *pipelining*

Para facilitar a los alumnos la comprensión de la técnica de ejecución con *pipelining* se añadió un cuarto modo de ejecución, que se denominó "*Pipelining sin detección de fallos",* que permite ilustrar de forma gráfica los problemas que posee este modo de ejecución. Este sistema funciona igual que el modo *pipelining* convencional, con la diferencia de que no comprueba las dependencias entre instrucciones ni las consecuencias de los saltos. De este modo, en muchas ocasiones el

resultado de la ejecución de instrucciones utilizando este modo será errónea. Este modo de ejecución puede ser utilizado por el profesor para demostrar los problemas derivados de la segmentación del cauce y el uso de la técnica de ejecución con *pipelining*.

Si bien la ejecución de las instrucciones es simulada, muchas instrucciones se han programado pensando en una ampliación del simulador. Por ejemplo, las operaciones de suma o resta no están simplemente implementadas internamente mediante las correspondientes instrucciones JAVA, sino que se llevan a cabo mediante algoritmos binarios. Se pretende en un futuro que *SimIVY* permita también demostrar algunos conceptos básicos de electrónica digital.

4. Aplicaciones para la docencia

Sim-IVY está diseñado para ayudar en la docencia de dos formas distintas. Por un lado, puede ser usado por el profesor para iniciar a los alumnos en el diseño y funcionamiento de un microprocesador segmentado simple. Para ello se incluyen distintos programas de ejemplo que muestran la utilidad los elementos internos y que presentan los problemas que aparecen en la cola de segmentación junto con algunas posibles soluciones. En este proceso son muy útiles los diversos modos de ejecución que existen: *multiciclo* puede ser muy útil para ir explicando, paso a paso, las funciones que cumple cada etapa, mientras que "*pipelining" sin detección de fallos* puede servir para ilustrar los problemas derivados de la segmentación.

Por otro lado, los alumnos pueden utilizarlo en un laboratorio, para realizar prácticas de lenguaje ensamblador que les permitirían entender, al mismo tiempo, el proceso de ejecución de las instrucciones que ellos mismos han introducido. Junto con la aplicación se incluyen los correspondientes manuales de la arquitectura RISC *IVY* y del diseño del microprocesador *IVY-1*.

5. Conclusiones y líneas futuras

SimIVY es una aplicación multiplataforma desarrollada en JAVA, de muy sencillo manejo e indicada para ser usada en el aula o en el laboratorio, que puede servir tanto de ayuda al profesor en la explicación del funcionamiento de un microprocesador segmentado como de soporte para prácticas realizadas por los alumnos. Simula un microprocesador ficticio *IVY-1* basado en una arquitectura también ficticia *IVY* de tipo RISC y similar a MIPS.

En un futuro próximo se espera integrar en el simulador las funciones necesarias para la enseñanza del funcionamiento de un subsistema de memoria basado en caché, que podrá funcionar junto con el resto del simulador o de forma independiente. Además, se desea completar el microprocesador incluyendo una unidad de punto flotante.

Referencias

[1] L.Cassel, M.Holliday, D.Kumar, J.Impagliazzo, K.Bolding, M.Pearson, J.Davies, G.S.Wolffe, W.Yurcik, *Distributed expertise for teaching computer organization & architecture*, *ACM SIGCSE Bull.*,v33 n2,2001.

[2] W.Yurcik, G. S.Wolffe, M.A.Holliday, "*A Survey of Simulators Used in Computer Organization / Architecture Courses"*, *Proc. of the 2001 Summer Computer Simulation Conference, (*2001).

[3] Yurcik, W. *CAALE: The computer architecture and assembly language education homepage. A compendium of computer simulators*. http://www.sosresearch.org/caale/caalesimulators.html.

[4] J.Hennessy, N.Jouppi, S.Przybylski, C.Rowen, T.Gross, F.Baskett, J.Gill, "*MIPS: A microprocessor architecture*", *Proc of the 15th annual workshop on Microprogramming*, pp.17-22 (1982).

[5] J.L.Hennesy, D.A.Patterson, "*Computer Architecture, A Quantitative Approach*", Morgan Kaufmann Publishers, 3 ª ed. (2003).