Sesiones 12 y 13- Unidad de Instrucción Segmentada

Arquitectura de Computadores I (IS19)

Área de Arquitectura y Tecnología de Computadores
Departamento de Ingeniería y Ciencia de los Computadores

Curso 2002-2003

1. Objetivos

El objetivo de esta práctica es ver cómo funciona la ruta de datos segmentada del MIPS R2000 estudiada en clase de teoría. En primer lugar el alumno debe familiarizarse con la herramienta a emplear, en concreto el simulador *MIPSim* [1]. Esta herramienta presenta la misma arquitectura y ruta de datos que el procesador segmentado analizado en las clases teóricas. Otra ventaja importante es que el lenguaje ensamblador utilizado para su programación es el de los procesadores MIPS. Permite analizar de forma gráfica la ruta de datos de las instrucciones que componen un programa durante la ejecución de las mismas. Además, también permite comprobar el estado de las líneas de control y el contenido de las líneas de datos en cada etapa de la segmentación.

Una vez presentado el simulador se estudiará el comportamiento de una serie de programas que presentan una serie de riesgos propios de la segmentación, con el objetivo de detectar dichos problemas, y si cabe resolverlos.

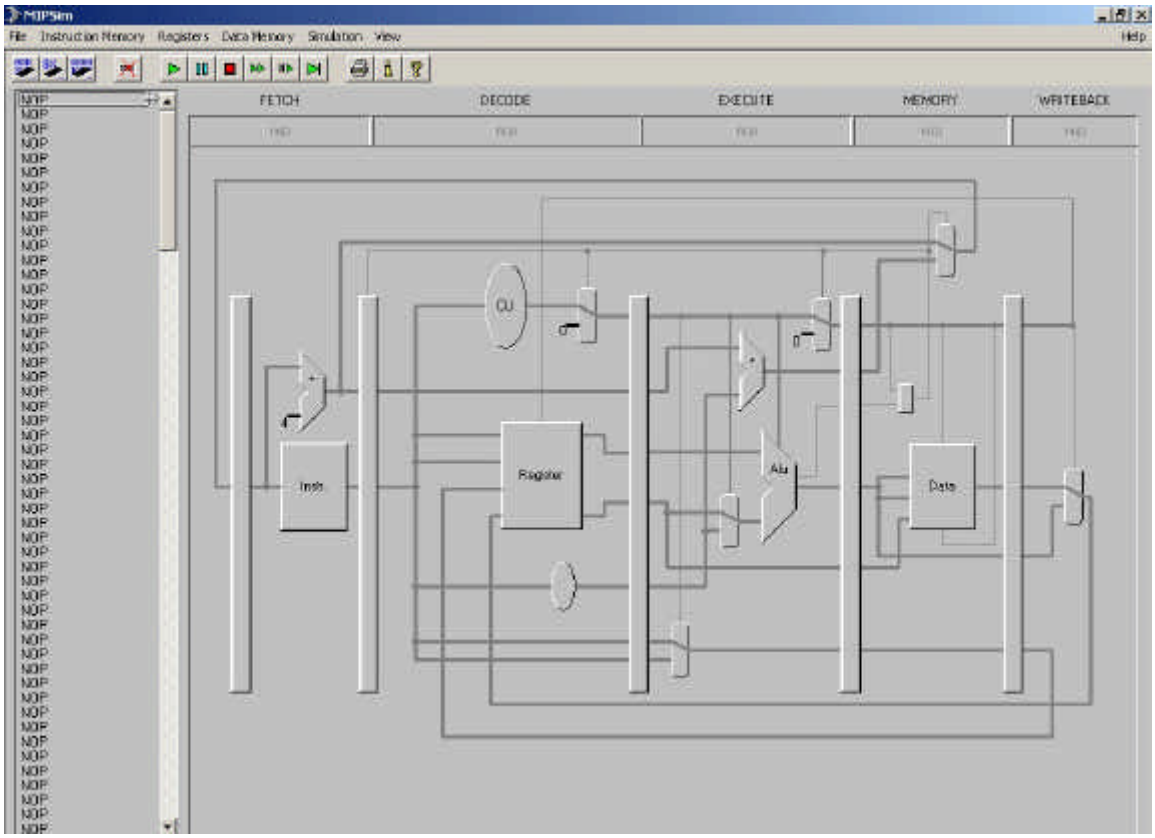
2. Material

Para el desarrollo de esta sesión de laboratorio se dispondrá de un computador personal PC, con el sistema operativo Windows, y el simulador de la ruta de datos

segmentada del MIPS R2000 MIPSim. Este simulador se puede descargar de la página web de la asignatura.

3. Manual de uso del simulador MIPSim

Una vez iniciada la aplicación, el entorno de MIPSim es el siguiente:

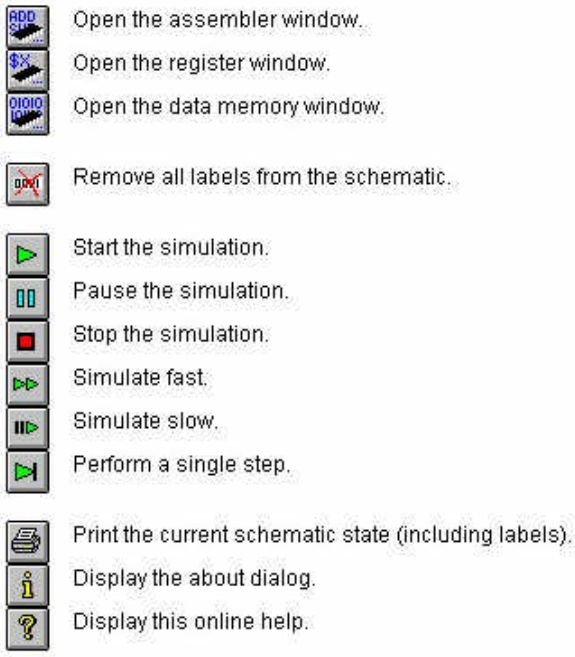


En la parte derecha se pueden observar las cinco etapas de la segmentación (fetch, decode, execute, memory y writeback). En cada etapa aparecen las unidades que forman parte de la misma:

- Fetch: Contador de programas y memoria de instrucciones.
- Decode: Unidad de control y banco de registros.
- Execute: Unidad Aritmético-Lógica
- Memory: Memoria de datos.

En la parte izquierda aparece una lista con las instrucciones de memoria en lenguaje ensamblador. A medida que se van ejecutando instrucciones se van "moviendo" por la ruta de datos segmentada, cada una de ellas con un color diferente. En la lista de instrucciones, la instrucción que va a comenzar a ejecutarse se marca con el símbolo 

En la parte superior existe una caja de herramientas con una serie de botones que permiten realizar diferentes operaciones comunes de forma más rápida. El significado de cada uno de los botones es el siguiente:

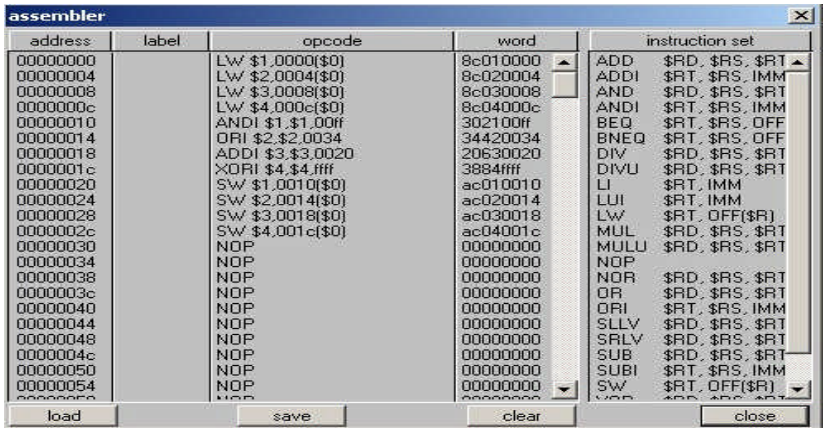


En el menú principal aparecen una serie de opciones:

-File: Para abrir, salvar e imprimir ficheros que contienen el programa escrito el lenguaje ensamblador del MIPS R2000. Dichos ficheros tienen la extensión (*.mp).
- Instruction Memory: Permite acceder a la memoria de instrucciones.
- Registers: Permite acceder al banco de registros.
- Data Memory: Permite acceder a la memoria de datos.
- Simulation: Para simular el programa.

A continuación se presentan con más detalle la memoria de instrucciones, el banco de registros y la memoria de datos.

Memoria de instrucciones: Su estructura es la siguiente:



En la parte izquierda existen cuatro columnas:

- Address: Dirección de memoria de la instrucción.
- Label: Etiqueta empleada en las instrucciones de salto.
- Opcode: La instrucción en lenguaje ensamblador.
- Word: La instrucción en lenguaje máquina.

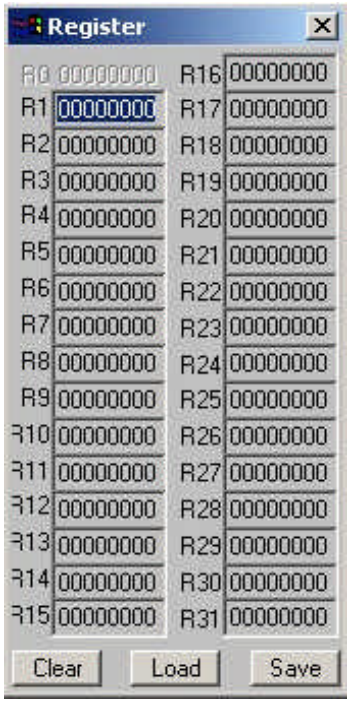
Se puede cambiar el contenido de los campos anteriores, salvo el campo address que sólo es de lectura, seleccionándolo y editando el nuevo texto. Cuando se introduce una nueva instrucción en lenguaje ensamblador, el simulador comprueba la sintaxis de la misma, de forma que si existe algún error aparece el correspondiente mensaje advirtiendo de tal situación.

En la parte derecha aparece una lista con el conjunto de instrucciones del MIPS R2000.

En la parte inferior aparecen una serie de botones cuya finalidad es la siguiente:

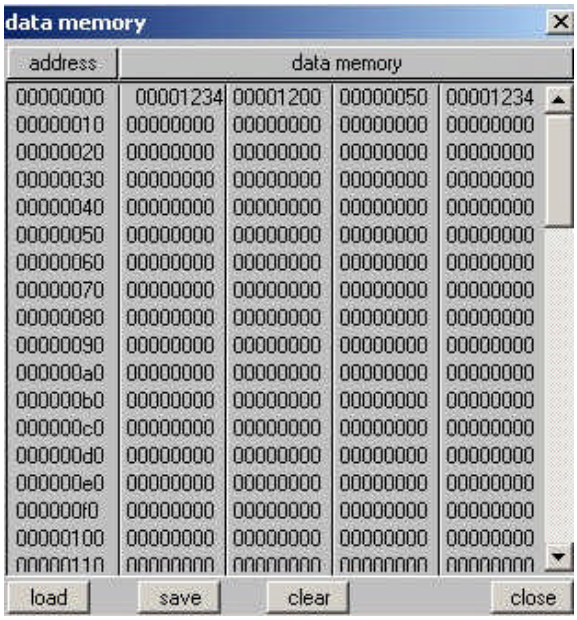
- Load: Para abrir un fichero con un programa en lenguaje ensamblador (*.mp).
- Save: Salvar el programa actual.
- Clear all: Pone a cero toda la memoria de instrucciones, es decir instrucciones nop.

Banco de registro: Su estructura es la siguiente:



Presenta los valores de los registros, que también pueden ser modificados por el usuario, haciendo “clic” en el correspondiente registro a modificar. Es posible cargar o salvar una configuración del banco de registros mediante ficheros con la extensión (*.mr).

Memoria de datos: Su estructura es la siguiente:



Se puede emplear para visualizar y modificar el contenido de la memoria de datos. Cada línea contiene cuatro columnas de datos (en hexadecimal), precedidas de la dirección de memoria de la primera palabra. Es posible cargar o salvar una configuración de la memoria de datos mediante ficheros con la extensión (*.md).

4. Desarrollo de la práctica

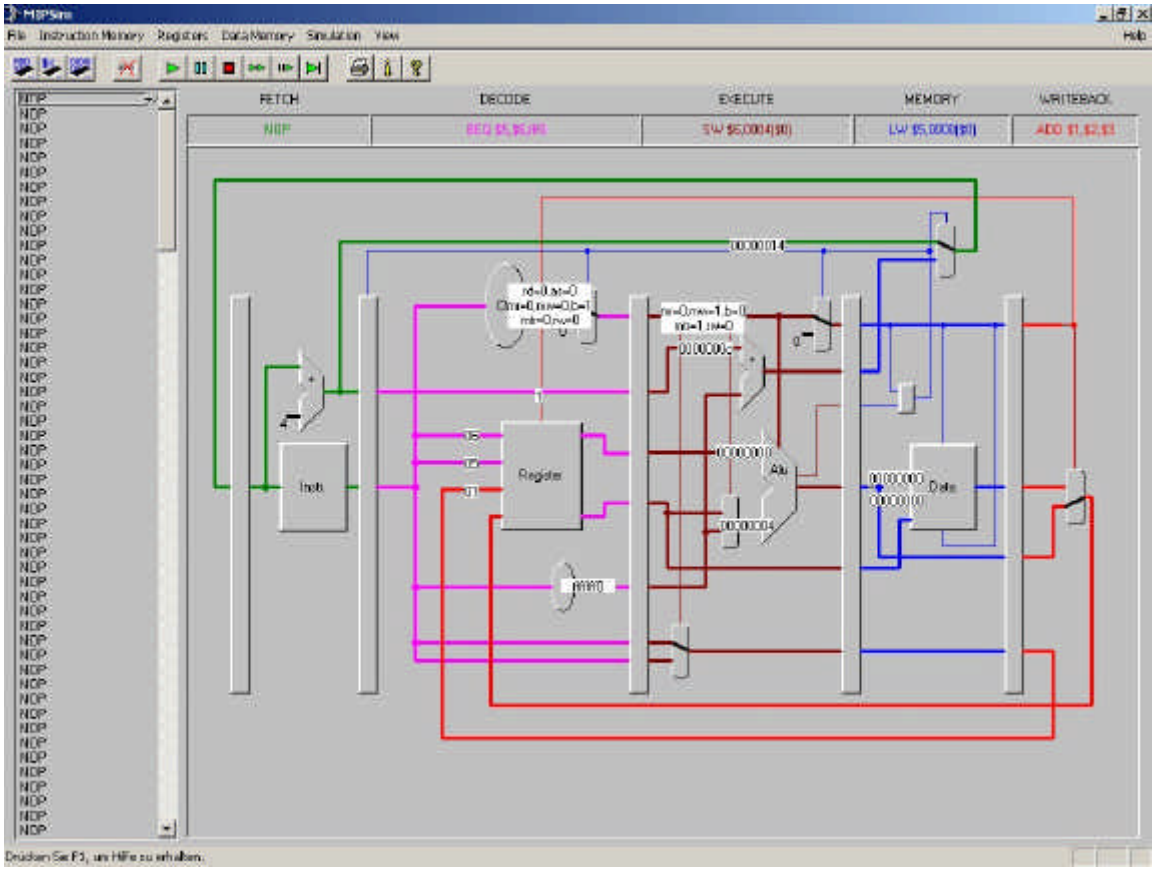
4.1- Estudio de la ejecución de las instrucciones estudiadas en clase de teoría.

En este apartado se pretende observar qué operaciones se realizan en la ejecución de las instrucciones estudiadas en clase de teoría. Se deberá determinar qué se realiza en cada una de las etapas de las mismas, así como observar los valores que toman las señales de control y la líneas existentes en la ruta de datos segmentada.

Para realizar este apartado se ejecutará el siguiente código, bien introduciéndolo directamente en la memoria de instrucciones, o bien escribiéndolo en un fichero con la extensión (*.mp) y después cargándolo en el simulador:

```
salto:   add$ 1,$2,$3
         lw   $5,0($0)
         sw   $6,4($0)
         beq  $5,$6,salto
```

Es posible ver el estado de cada una de las líneas (señales de control y datos) simplemente con hacer clic en dicha línea, apareciendo un esquema como el siguiente:



4.2- Riesgos en la segmentación.

En este apartado se ejecutarán una serie de programas en los que se deberá comprobar que los resultados obtenidos son los esperados, y en caso contrario se deberán resolver esos problemas que surgen con la versión segmentada.

Programa 1: Ejecuta el siguiente programa

```
lw  $1,0($0)
lw  $2,4($0)
add $3,$1,$2
sub $4,$1,$2
sw  $3,8($0)
sw  $4,c($0)
```

Inicialmente en las posiciones de memoria de datos 0x00000000 y 0x00000004 deben aparecer los valores 8 y 4, respectivamente.

Cuestiones:

- ¿Qué resultado se ha obtenido?
- Si no es el esperado:
  - ¿qué tipo de riesgo de segmentación ha aparecido?
  - ¿Cómo se podría modificar el código para resolver dicho problema?

Programa 2: Ejecuta el siguiente programa

```
lw   $2,0($1)
lw   $3,4($1)
addi $4,$4,1
sw   $3,0($1)
sw   $2,4($1)
```

Inicialmente en las posiciones de memoria de datos 0x00000000 y 0x00000004 deben aparecer los valores 8 y 4, respectivamente. Asimismo el registro $1 está inicializado a cero y el registro $4 tiene el valor 5.

Cuestiones:

- ¿Qué resultado se ha obtenido?
- Si no es el esperado:
  - ¿qué tipo de riesgo de segmentación ha aparecido?
  - ¿Cómo se podría modificar el código para resolver dicho problema?

Programa 3: Ejecuta el siguiente programa

```
      beq   $1,$2,salto
      add   $3,$1,$2
      sub   $4,$1,$2
      addi  $5,$5,1
salto:sw    $3,0($6)
```

Inicialmente, las posiciones de memoria deben estar inicializadas a cero. Asimismo los registros $1, $2, $3 y $6 deben tener los valores 7, 7, 5 y 0, respectivamente.

Cuestiones:

- ¿Qué resultado se ha obtenido?
- Si no es el esperado:
  - ¿qué tipo de riesgo de segmentación ha aparecido?
  - ¿Cómo se podría modificar el código para resolver dicho problema?

A continuación se ejecuta el mismo programa, pero con el valor 6 en el registro $3

Cuestiones:

- ¿Qué resultado se ha obtenido?
- Si no es el esperado:
  - ¿qué tipo de riesgo de segmentación ha aparecido?
  - ¿Cómo se podría modificar el código para resolver dicho problema?

Programa 4: Dado el siguiente programa

```
lw  $1,0($0)
lw  $2,4($0)
nop
nop
add $3,$1,$2
lw  $4,8($0)
lw  $5,c($0)
nop
nop
sub $6,$4,$5
sw  $3,10($0)
```

donde inicialmente se tienen los siguientes valores en memoria:

0x00000000 = 4 ; 0x00000004 = 8; 0x00000008 = 3 ; 0x0000000c = 2

Cuestiones:

- ¿Cómo se podría modificar el código para eliminar algunas de las instrucciones nop?

5. Evaluación

Para realizar la evaluación de esta práctica, el profesor de laboratorio al inicio de la última sesión entregará al alumno un cuestionario sobre la práctica. El alumno deberá rellenar dicho cuestionario y devolver la hoja donde habrá anotado los resultados obtenidos.

6. Bibliografía

[1] MIPSim. Prof. Herbert Grünbacher, Vienna University of Technology, Institut für Technische Informatik, http://www.ecs.tuwien.ac.at/