Arquitectura de Computadoras I

Ingeniería de Sistemas – Curso 2018

Práctica de Laboratorio: Microprocesador MIPS Segmentado

El objetivo de esta práctica es implementar el microprocesador MIPS (visto en clase de teoría) en VHDL. En concreto, se va a realizar la versión **segmentada** del microprocesador, cuyos detalles se pueden encontrar en el caítulo 4 del libro "Computer Organization and Design: The Hardware/Software Interface ", por David A.Patterson y John L. Hennessy.

Punto de partida

El modelo de las memorias de datos y programas proporcionado (archivo memory.vhd) no introduce ciclos de espera y responde en el mismo ciclo. Además, utiliza dos archivos separados para el contenido inicial de cada memoria, archivo llamado “program1” para memoria de instrucciones y “data” para memoria de datos. Se proporcionan un archivo de ejemplo (program1.s) para utilizar junto con el testbench de la práctica, si bien se pueden generar otros archivos correspondientes a otros códigos para hacer más pruebas.

Ejercicio

Se pide implementar el microprocesador MIPS en su versión segmentada con Unidad de Adelantamiento (Forwarding Unit). El resultado debe ser un micro capaz de realizar en el caso ideal, una instrucción por ciclo de reloj. Para el ejercicio básico, no es necesario que el modelo soporte riesgos (salvo el estructural de acceso a memorias separadas de instrucciones y datos).

No se pide que el microprocesador soporte todo el juego de instrucciones completo, sino solamente las siguientes instrucciones: *add*, *sub*, *and*, *or*, *lw*, *sw*, *slt*, *lui* y *beq*. En cualquier caso, la instrucción *beq*, que implica riesgos de control por ser un salto, funcionará “anómalamente” en la versión básica del ejercicio obligatorio.

Se recomienda utilizar el esquema del libro, reflejado en la siguiente figura 1. Todos los registros deben resetearse asíncronamente y funcionar por flanco de subida del reloj. Las instrucciones a implementar tienen los siguientes códigos de operación. Los códigos de operación se pueden encontrar también en el apéndice B del libro.

ADD (Add Word)

31 – 26	25 – 21	20 – 16	15 – 11	10 – 6	5 – 0
SPECIAL 0 0 0 0 0 0	**rs**	**rt**	**rd**	0 0 0 0 0 0	**ADD** 1 0 0 0 0 0
6	5	5	5	5	6

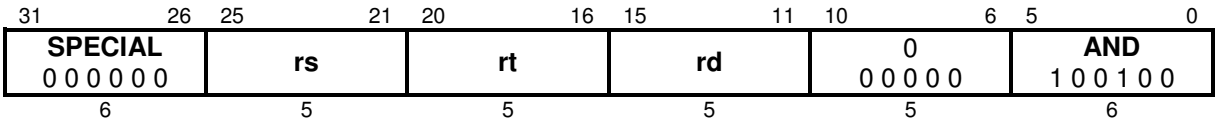
Formato: ADD rd, rs, rt
Descripción: rd ← rs + rt

SUB (Substract Word)

31 – 26	25 – 21	20 – 16	15 – 11	10 – 6	5 – 0
SPECIAL 0 0 0 0 0 0	**rs**	**rt**	**rd**	0 0 0 0 0 0	**SUB** 1 0 0 0 1 0
6	5	5	5	5	6

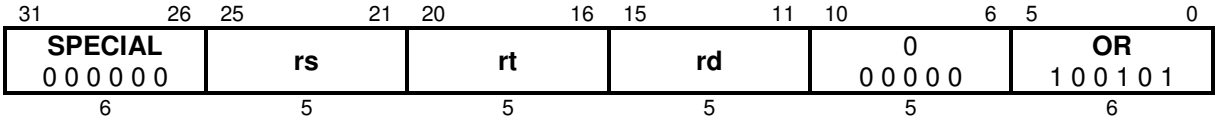
Formato: SUB rd, rs, rt
Descripción: rd ← rs - rt

AND



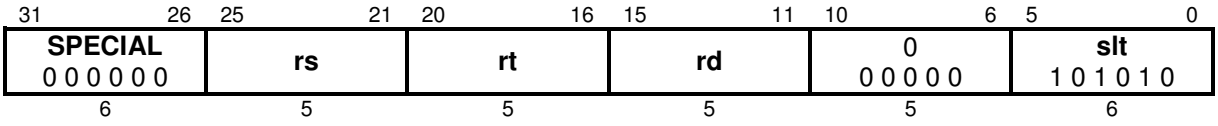
Formato: AND rd, rs, rt
Descripción: rd ← rs AND rt

OR



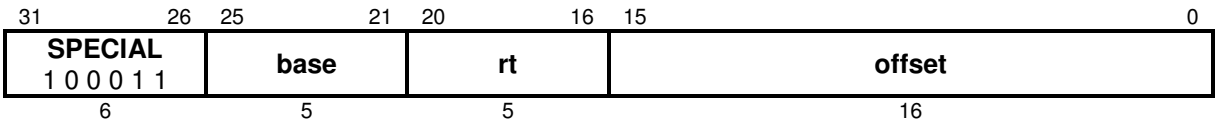
Formato: OR rd, rs, rt
Descripción: rd ← rs OR rt

SLT (Set on Less Than)



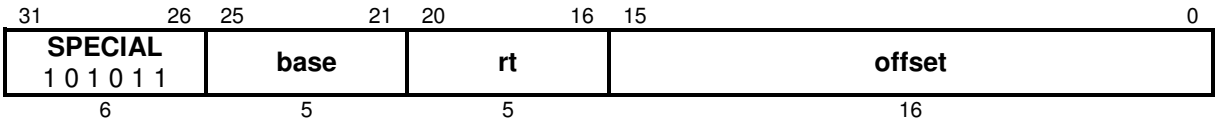
Formato: SLT rd, rs, rt
Descripción: rd ← (rs < rt)

LW (Load Word)



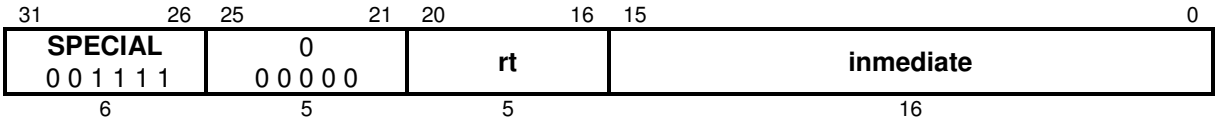
Formato: LW rt, offset(base)
Descripción: rt ← memory[base+offset]

SW (Store Word)



Formato: ST rt, offset(base)
Descripción: memory[base+offset] ← rt

LUI (Load Upper Inmediate)



Formato: LUI rt, inmediate
Descripción: rt ← immediate & 0^{16} (rt ← immediate << 16)

BEQ (Branch on Equal)

31 – 26	25 – 21	20 – 16	15 – 0
SPECIAL 0 0 0 1 0 0	**rs**	**rt**	**offset**
6	5	5	16

Formato: BEQ rs, rt, offset
Descripción: if (rs=rt) then PC ← PC + (offset << 2)

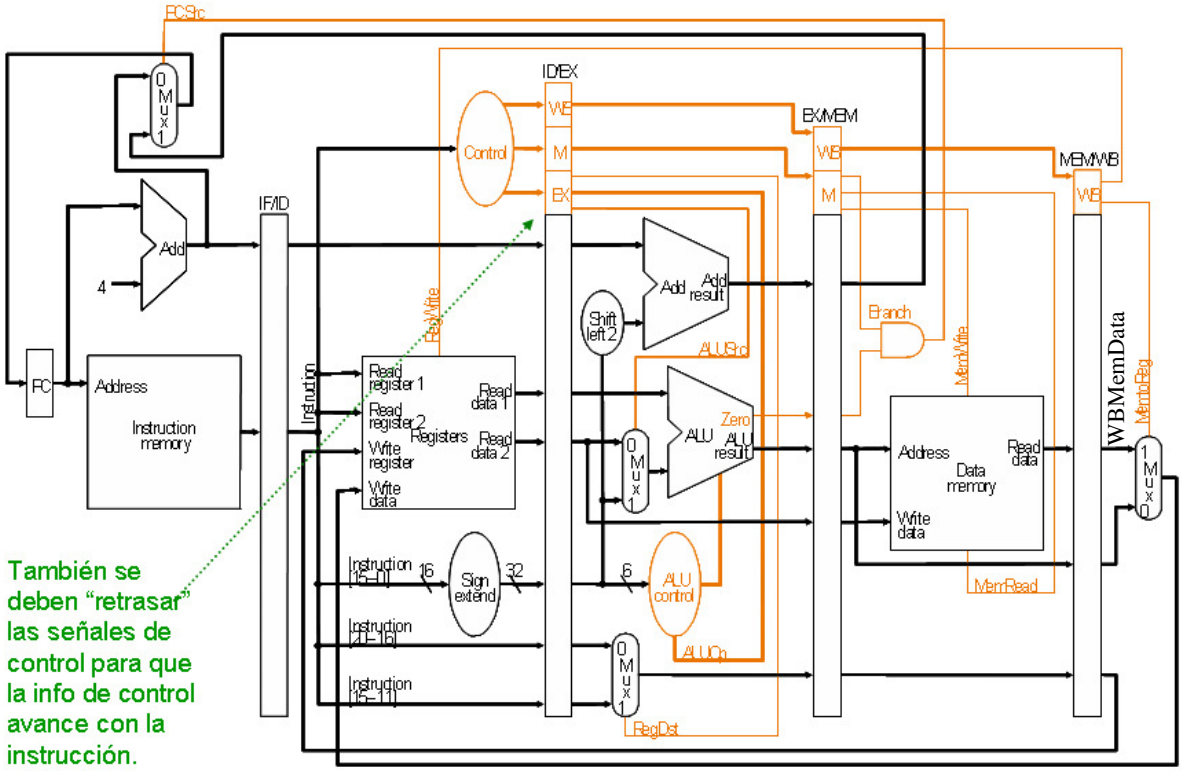


Figura 1. Modelo de microprocesador segmentado (sin unidad de adelantamiento de datos).

La unidad de adelantamiento de datos permite reducir los riesgos de datos y se integra al procesador tal como se muestra en la siguiente figura:

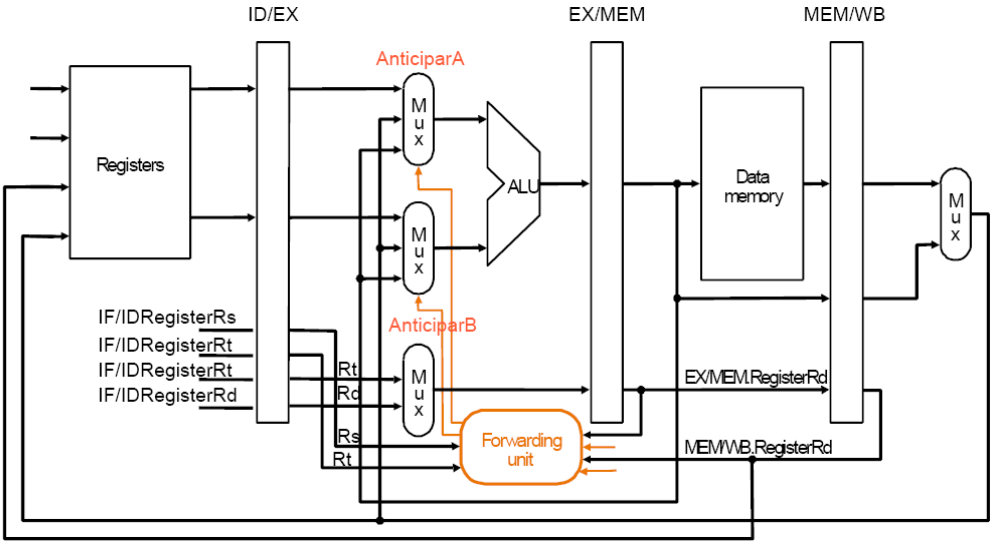


Figura 2. Unidad de adelantamiento de datos.

Para la aprobación del curso deberá:

1. Verificar el diseño utilizando el archivo “program1” proporcionado por la cátedra.

2. Ensamblar y verificar el diseño utilizando el archivo “bit_counter.s” proporcionado por la cátedra. El programa implementa un algoritmo que cuenta el número de bits en estado alto (‘1’) de los 8 bits menos significativos de un dato almacenado en la dirección 4 de memoria. El programa sólo utiliza 3 registros del procesador y utiliza la memoria de datos para almacenamiento de datos temporales. El estado inicial de la memoria de datos es el siguiente:

   Dirección 0x0000 0000: 0x000000001
   Dirección 0x0000 0004: 0x0000000YY (YY = dato)

   El resultado se almacena en la posición 8 de la memoria.

3. Realizar las modificaciones necesarias al programa “bit_counter.s” considerando la existencia de la unidad de adelantamiento. Compilar y verificar el correcto funcionamiento.

4. ¿Cuál es la demora de ejecución para la versión del programa sin adelantamiento de datos? ¿Cuál es la demora utilizando la unidad de adelantamiento? ¿Cuál es la aceleración obtenida al incorporar la unidad de adelantamiento?

Material a entregar

- Archivos VHDL
- bit_counter.s corregido
- Archivos de memoria de instrucción y de datos
- Archivo “readme.txt” con los comentarios respecto al diseño que considere adecuados, y las respuestas del ejercicio 4.

Ayudas y avisos

- Los archivos “program1” y “data” que contienen las memorias de instrucciones y datos respectivamente deben estar en el directorio de trabajo. Si no es así, en el archivo procesador_TB.vhd se puede dar la ruta completa de dichos archivos cambiando las líneas de código:

```
C_ELF_FILENAME    => "programa",
...
C_ELF_FILENAME    => "datos",
```

por otras donde indique la ruta completa a ambos archivos, por ejemplo:

```
C_ELF_FILENAME    => "D:\nombredirectorio\programa",
..
C_ELF_FILENAME    => "D:\nombredirectorio\datos",
```

- El programa de prueba “program1.s” proporcionado en la práctica no incluye riesgos y prueba todas las instrucciones del ejercicio básico. El archivo “program1” es el resultado del ensamblado del “program1.s” que se usará para probar el ejercicio básico. El archivo “datos” contiene los datos que se usaran por el “programa” en el ejercicio básico.

- El archivo memory.vhd contiene la memoria que se usará en el ejercicio básico. El archivo processor.vhd contiene la entidad del micro que se deberá implementar. El archivo processor_tb.vhd contiene el test bench para probar el ejercicio básico.

- La tabla contenida en el archivo “registers.html” proporcionada en la práctica muestra la traducción de los nombres de registros usados en ensamblador al número de registro en el micro, del 0 al 31.

-Se recomienda utilizar el programa “MARS” para la generación de la memoria de programa. En la página del curso se encuentra un tutorial y el enlace de descarga de esta herramienta.