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: "Computer Organization and Design: The Hardware/Software Interface ", por David A.Patterson y John L. Hennessy. Se recomienda seguir el libro para realizar esta práctica, ya que se sigue este libro con bastante fidelidad. En concreto, se sigue el modelo segmentado del MIPS, detallado en el capítulo 4.

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

Realizar la implementación de un procesador segmentado completo que admite las siguientes instrucciones: *add*, *sub*, *and*, *or*, *lw*, *sw*, *slt*, *beq, addi, andi* y *ori*. En cualquier caso, la instrucción *beq*, que implica riesgos de control por ser un salto, funcionará “anómalamente” en la versión básica del ejercicio obligatorio.

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	**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	**SUB** 1 0 0 0 1 0
6	5	5	5	5	6

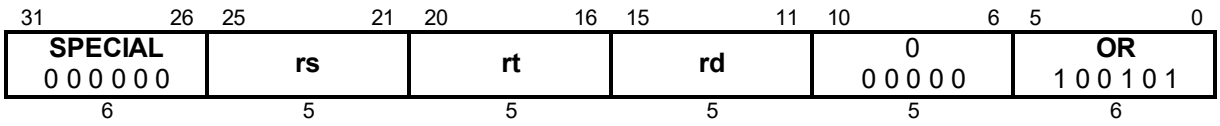
Formato: SUB rd, rs, rt
Descripción: rd ← rs - rt

AND

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	**AND** 1 0 0 1 0 0
6	5	5	5	5	6

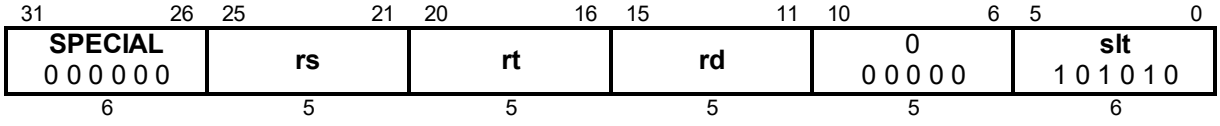
Formato: AND rd, rs, rt
Descripción: rd ← rs AND rt

OR



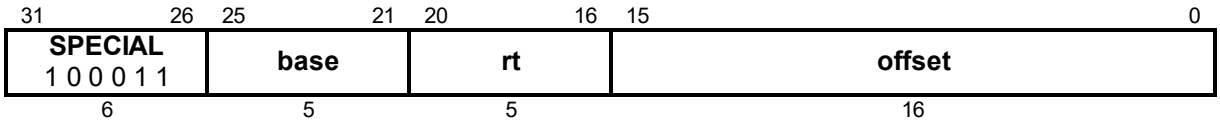
Formato: OR rd, rs, rt
Descripción: rd ← rs OR rt

SLT (Set on Less Than)



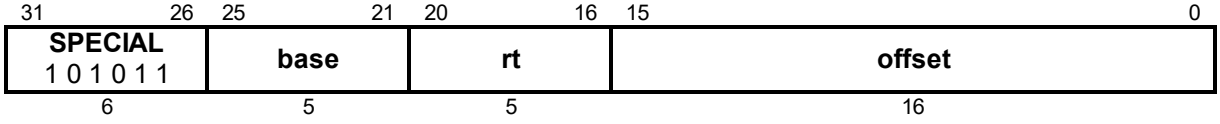
Formato: SLT rd, rs, rt
Descripción: rd ← (rs < rt)

LW (Load Word)



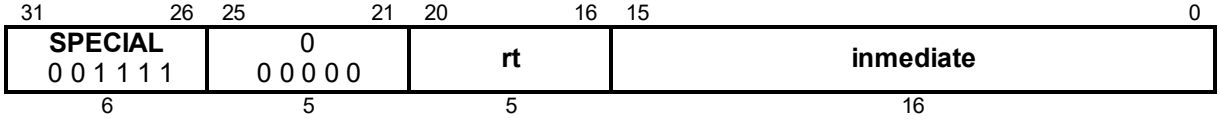
Formato: LW rt, offset(base)
Descripción: rt ← memory[base+offset]

SW (Store Word)



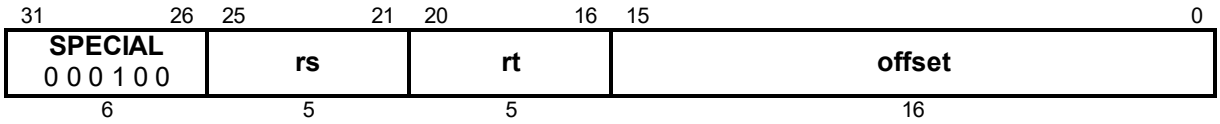
Formato: ST rt, offset(base)
Descripción: memory[base+offset] ← rt

LUI (Load Upper Inmediate)



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

BEQ (Branch on Equal)



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

ADDI (Addtion Immediate)

31 — 26	25 — 21	20 — 16	15 — 0
SPECIAL 0 0 1 0 0 0	**rs**	**rt**	**inmediate**
6	5	5	16

Formato: ADDI rt, rs, immediate
Descripción: rt ← rs + SIGN_EXTEND(immediate)

ANDI (And Immediate)

31 — 26	25 — 21	20 — 16	15 — 0
SPECIAL 0 0 1 1 0 0	**rs**	**rt**	**inmediate**
6	5	5	16

Formato: ADDI rt, rs, immediate
Descripción: rt ← rs AND SIGN_EXTEND(immediate)

ORI (Or Immediate)

31 — 26	25 — 21	20 — 16	15 — 0
SPECIAL 0 0 1 1 0 1	**Rs**	**rt**	**inmediate**
6	5	5	16

Formato: ANDI rt, rs, immediate
Descripción: rt ← rs OR SIGN_EXTEND(immediate)

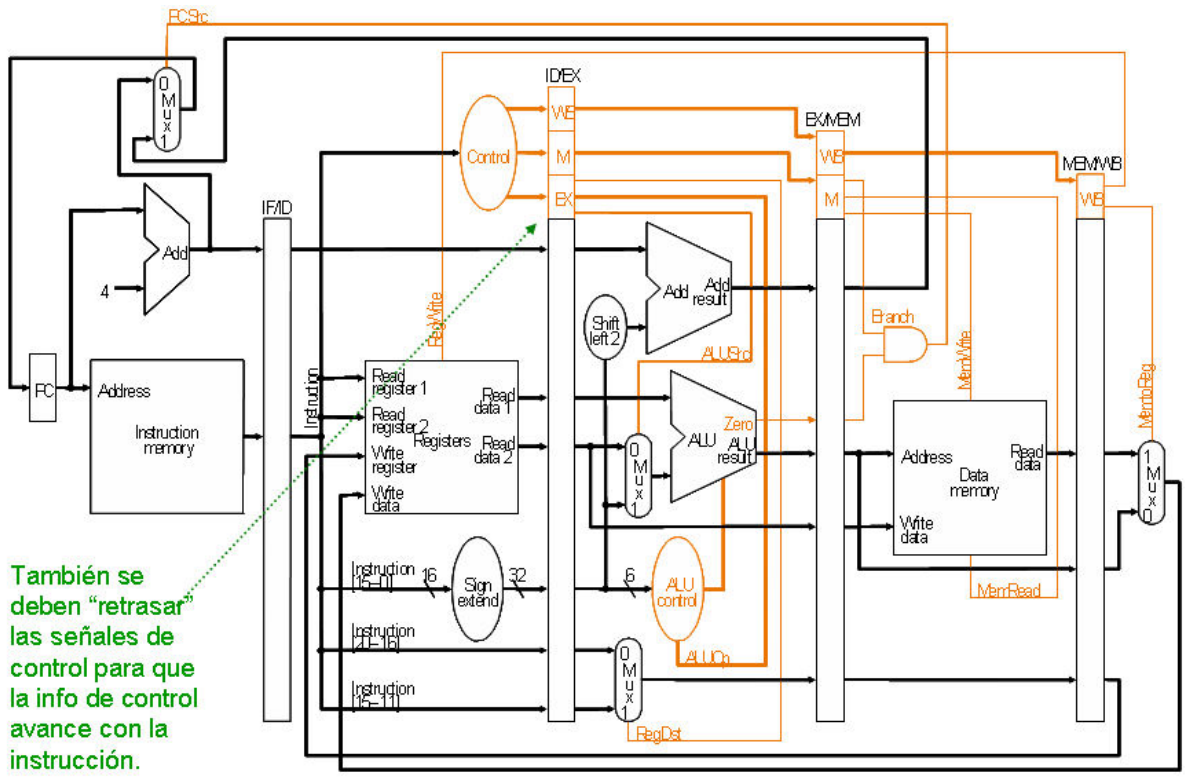


Figura 1. Modelo de microprocesador segmentado (faltan instrucciones inmediatas).

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

Material a entregar

- Archivos VHDL
- Archivos “.s”
- Archivos de memoria de instrucción

Ayudas y avisos

- Los archivos “programa” y “datos” 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 “program.s” proporcionado en la práctica no incluye riesgos y prueba todas las instrucciones del ejercicio básico. El archivo “programa” es el resultado del ensamblado del “program.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.