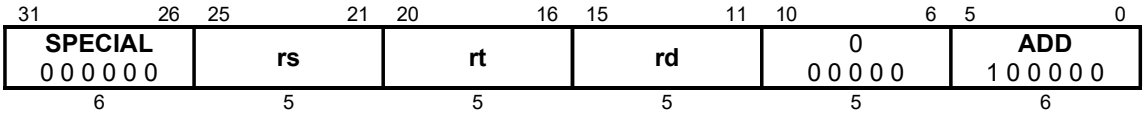
Práctica Especial: Microprocesador MIPS uniciclo

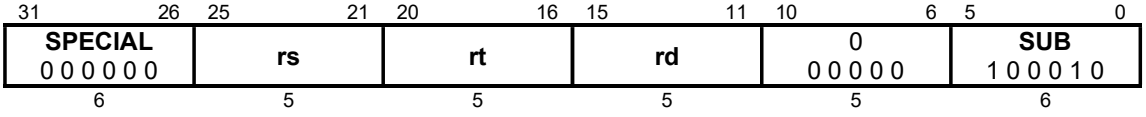
El objetivo de esta práctica es implementar el microprocesador MIPS uniciclo (visto en el curso) en VHDL. Diseñe e implemente el MIPS uniciclo que admita las siguientes instrucciones: *add*, *sub*, *and*, *or*, *lw*, *sw*, *slt*, *beq* y *j*.

ADD (Add Word)



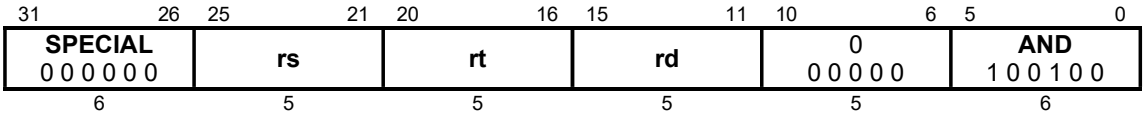
Formato: ADD rd, rs, rt
Descripción: rd ← rs + rt

SUB (Substract Word)



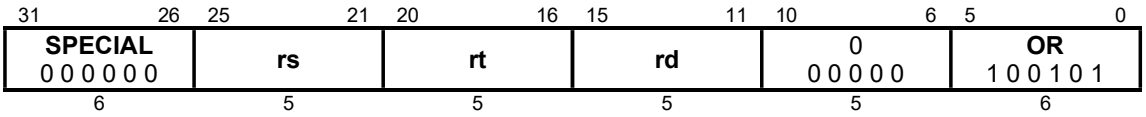
Formato: SUB rd, rs, rt
Descripción: rd ← rs - rt

AND



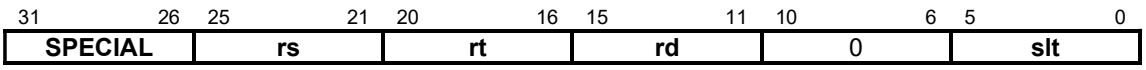
Formato: AND rd, rs, rt
Descripción: rd ← rs AND rt

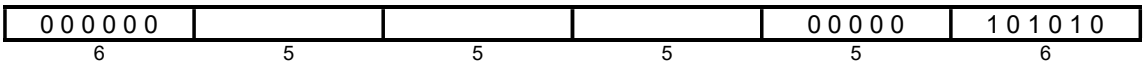
OR



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

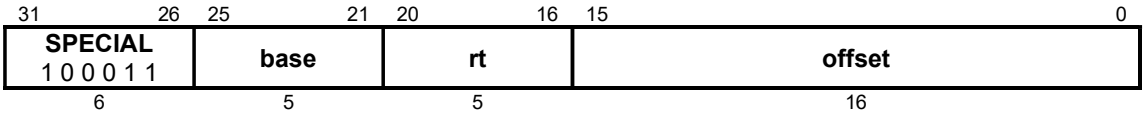
SLT (Set on Less Than)





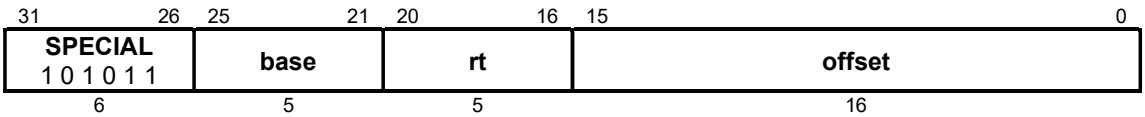
Formato: SLT rd, rs, rt
Descripción: rd ← (rs < rt)

LW (Load Word)



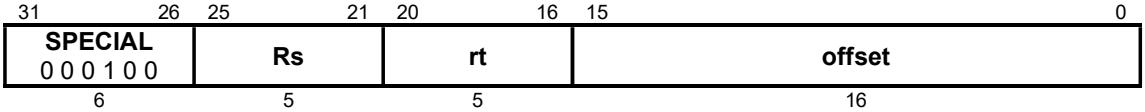
Formato: LW rt, offset(base)
Descripción: rt ← memory[base+offset]

SW (Store Word)



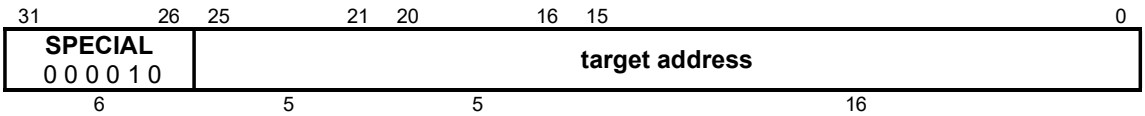
Formato: SW rt, offset(base)
Descripción: memory[base+offset] ← rt

BEQ (Branch on Equal)



Formato: BEQ rs, rt, offset
Descripción: if (rs=rt) then PC ← PC +4 + (offset << 2)

J (Jump)



Formato: j target address
Descripción: PC ← (PC+4)[31:28] & (target address << 2)

La cátedra proporciona los componentes *ALU* y *Registers*, correspondiente respectivamente a la *Unidad Artimética Lógica* y *Banco de Registros* del procesador.

Unidad Artimética Lógica.

Esta ALU opera con datos de 32 bits, dispone del *flag zero* para indicar si el resultado es igual a cero. Sus operaciones aritmético/lógicas básicas son suma, resta, and, or, "menor que" y desplazamiento a la izquierda. En el caso de introducir un código de operación (control) que no corresponda con los prefijados en la tabla, la señal de salida *result* se pondrá a 0. Los códigos de operación se corresponderán con la siguiente tabla:

control	*operation*
000	*a* and *b*
001	*a* or *b*
010	*a* + *b*
100	*b* << 16 (lógico)
110	*a* - *b*
111	*a* < *b*

El *flag zero* se pondrá a 1 cuando el resultado de la operación es igual a cero. En caso contrario debe permanecer siempre a 0. El código de operación 111 pondrá en la salida *result* un 1 si *a* es menor que *b* y un 0 en caso contrario.

Entidad: *ALU*
Entradas: *a*(31 ***downto*** 0), *b*(31 ***downto*** 0) y *control*(2 ***downto*** 0).
Salidas: *result*(31 ***downto*** 0) y *zero*

Banco de Registros

El banco de registros del MIPS tiene 32 registros de propósito general de 32 bits cada uno, de los cuales el primero (*r*0) está siempre en 0. Este banco permite el acceso simultáneo a tres registros, cuyas direcciones son: *reg1_rd*, *reg2_rd* y *reg_wr*. El acceso a los registros direccionados por *reg1_rd* y *reg2_rd* será de lectura, mientras que el asociado a *reg_wr* será de escritura. La escritura será síncrona, en flanco de bajada y habilitada con la señal *wr* a 1, mientras que la lectura será siempre combinacional. El *reset* del circuito es asíncrono y activo a nivel alto.

Entidad: *Registers*.
Entradas: *clk*, *reset*, *wr*, *reg1_rd* (4 ***downto*** 0), *reg2_rd* (4 ***downto*** 0), *reg_wr*(4 ***downto*** 0) y *data_wr*(31 ***downto*** 0).
Salidas: *data1_rd* (31 ***downto*** 0) y *data2_rd* (31 ***downto*** 0).

Realice

1- Implemente el procesador MIPS uniciclo en VHDL

2-Analice por qué la *Memoria de Programa* y el *Banco de Registros* trabajan en flanco descendente, y la *Memoria de Datos* y *Program Counter* (PC) en flanco ascendente. Muestre un diagrama en donde se detalle el análisis realizado.

Nota: En el caso de que se deseen probar más programas, para el desarrollo del código ensamblador de los mismos, se recomienda el uso de la herramienta MARS disponible en courses.missouristate.edu/KenVollmar/MARS/download.htm

Material a entregar

- Archivos VHDL
- Archivos ".s" en el caso que pruebe más programas
- Archivos de memoria de instrucción
- Informe