

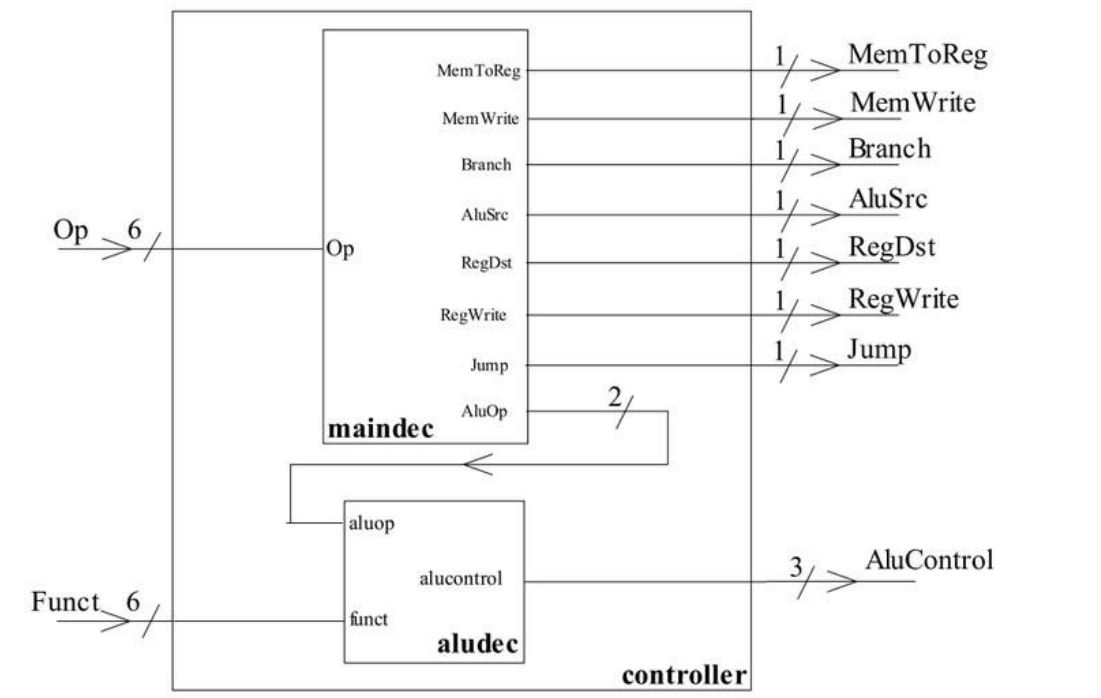
Arquitectura de Computadoras 2015

Práctico N° 3

Para todos los ejercicios se pide que respeten los nombres de las entidades, entradas y salidas. Representación de datos siempre en *std_logic* o *std_logic_vector*.

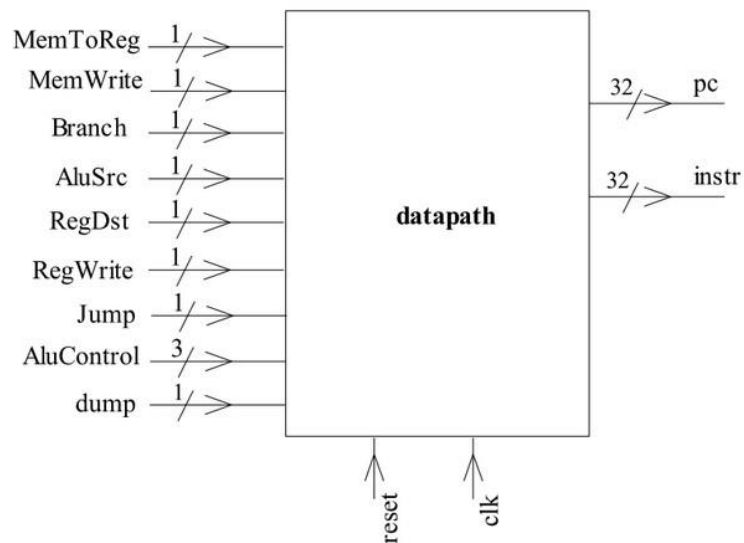
Internamente, se les da la libertad para que utilicen el diseño que consideren más adecuado siempre y cuando la salida sea la esperada.

Ejercicio 1) Diseñe un componente llamado controller utilizando los componentes *aludec* y *maindec* del práctico 1 según el diagrama.

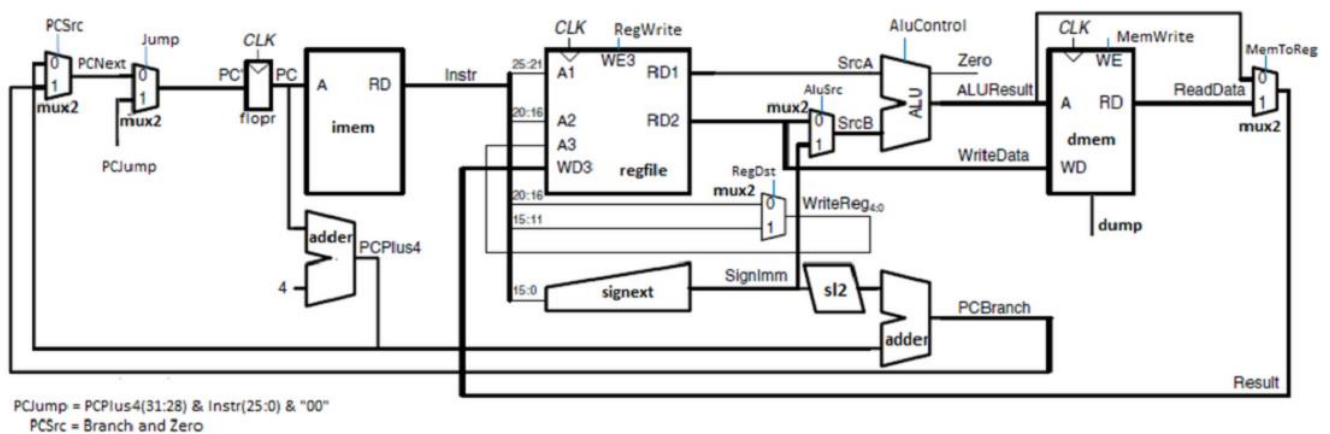


Implemente en VHDL, diseñe un testbench adecuado, simule con GHDL y GTKWave.

Ejercicio 2) Diseñe un componente llamado *datapath*. Respete los nombres de las señales y las conexiones según los diagramas adjuntos.

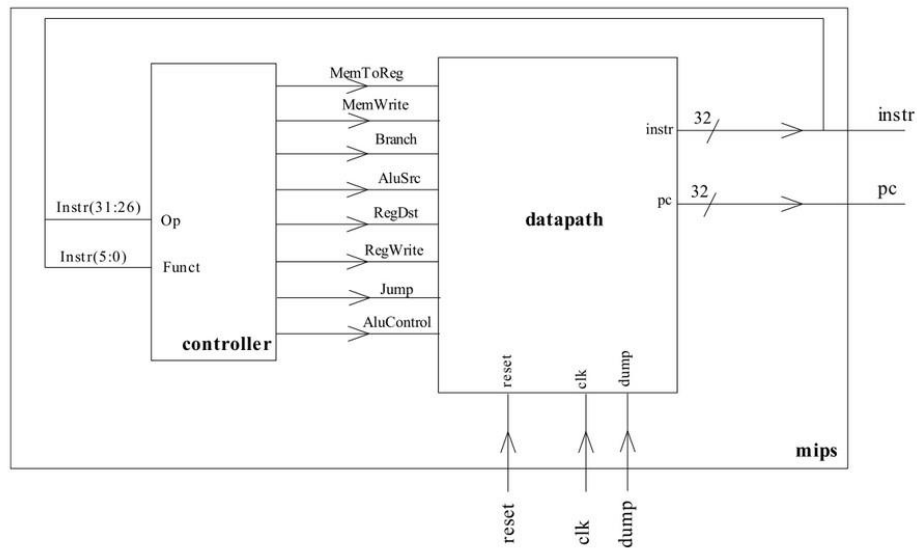


Deberá reutilizar los componentes *adder*, *flopr*, *mux2*, *imem*, *signext*, *sl2*, *alu*, *regfile*, y *dmem* del práctico anterior.



Implemente en VHDL, diseñe un testbench adecuado, simule con GHDL y GTKWave.

Ejercicio 3) Diseñe un módulo llamado *mips* utilizando los componentes *controller* y *datapath* según el diagrama.



Implemente en VHDL, diseñe un testbench adecuado, simule con GHDL y GTKWave.

Una vez terminada la implementación de la entidad de más alto nivel, “mips”, ha completado el desarrollo de un microprocesador MIPS de un solo ciclo. Es hora de probar su funcionamiento; utilice el simulador MARS de MIPS, desarrolle y compile pequeños programas en *assembly* para luego cargarlos en la memoria de su MIPS (utilizando *imem.vhd* y *dmem.vhd* provistos (o “a ser provistos”) por la cátedra).

Ejemplo:

Inicializar los registros \$t0 a \$t7 y luego colocar su valor en memoria RAM (en las direcciones 0x00 a 0x07).

| Dirección | Instrucción de MIPS | Código para <i>imem</i> |
|-----------|--------------------------|-------------------------|
| 0x00 | <i>addi \$t0, \$0, 0</i> | 20080000 |
| 0x01 | <i>addi \$t1, \$0, 1</i> | 20090001 |
| 0x02 | <i>addi \$t2, \$0, 2</i> | 200a0002 |
| 0x03 | <i>addi \$t3, \$0, 3</i> | 200b0003 |
| 0x04 | <i>addi \$t4, \$0, 4</i> | 200c0004 |
| 0x05 | <i>addi \$t5, \$0, 5</i> | 200d0005 |
| 0x06 | <i>addi \$t6, \$0, 6</i> | 200e0006 |
| 0x07 | <i>addi \$t7, \$0, 7</i> | 200f0007 |
| 0x08 | <i>sw \$t0, 0(\$0)</i> | ac080000 |
| 0x09 | <i>sw \$t1, 4(\$0)</i> | ac090004 |
| 0x0A | <i>sw \$t2, 8(\$0)</i> | ac0a0008 |
| 0x0B | <i>sw \$t3, 12(\$0)</i> | ac0b000c |
| 0x0C | <i>sw \$t4, 16(\$0)</i> | ac0c0010 |
| 0x0D | <i>sw \$t5, 20(\$0)</i> | ac0d0014 |
| 0x0E | <i>sw \$t6, 24(\$0)</i> | ac0e0018 |
| 0x0F | <i>sw \$t7, 28(\$0)</i> | ac0f001c |

En su *testbench* debe colocar la señal *reset* -> 1 por dos ciclos; correr el programa anterior por 17 ciclos, y luego colocar la señal *dump* -> 1 por tres ciclos, para obtener el archivo con el *dump* de la memoria. Verificar en el archivo de salida que el contenido de las siete primeras posiciones de la memoria contiene su índice (es decir, el contenido de la posición 0 es 0; el contenido de la posición 1 es 1, y así sucesivamente hasta la posición 7).


Hint 1: Es bueno agregar un bucle infinito del estilo “exit: j exit” para evitar que el programa vuelva a comenzar y la memoria se pise con valores erróneos.

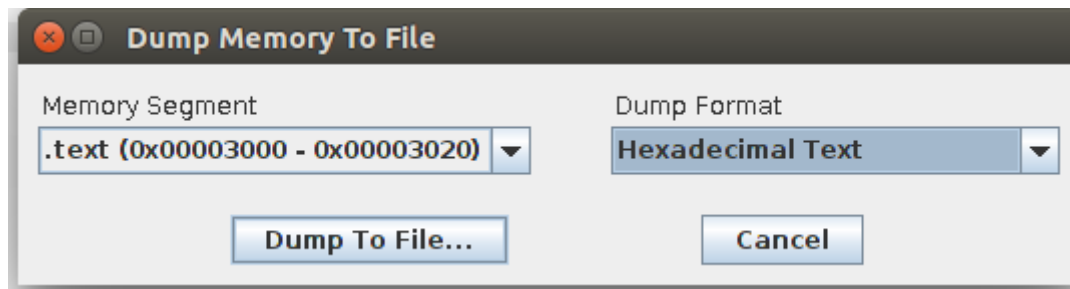
Hint 2: Tenga en cuenta los tiempos de simulación, un “stop-time” o un “wait for” en su *testbench* “pequeños” podrían acortar la simulación, de manera que el programa finalice la ejecución de instrucciones antes de escribir el resultado en memoria.

Hint 3 de MARS: Chequear que la memoria de datos comience en 0:

Settings -> Memory Configuration -> Compact Data at Address 0.

Hint 4 de MARS: Una vez que compilo el programa en MIPS, debe realizar un *dump*

del código  y luego seleccionar el tipo de salida: “Hexadecimal Text”:



Para los siguientes ejercicios, debe utilizar las instrucciones MIPS *assembly* (que soporte su implementación de MIPS: *addi*, *j*, *beq*, etc.), simular en MARS, hacer un *dump* de las instrucciones, correr su implementación de MIPS con ese programa (*imem* deberá leer ese archivo, para que luego *dmem* genere un archivo de salida), corroborar que el archivo *dump* de la memoria RAM contenga el valor esperado.

Recuerde comentar su código.

Ejercicio 4) Con la menor cantidad de registros e instrucciones de MIPS, inicialice con el valor de su índice las primeras N posiciones de memoria (comenzando en la dirección 0).

Ejercicio 5) Realice la sumatoria de las primeras N posiciones de memoria, y guarde el resultado en la posición N+1;

Ejercicio 6) Realice la multiplicación de dos registros: \$t0 y \$t1. Guarde el resultado en \$t3. Suponga que el resultado "cabe" en el registro. Guarde el resultado en la posición 0 de la memoria. Inicialice los valores de los registros debidamente.