Proyecto
Fase IV: Demonstración de Operación de PPU para RISC-V

Tarea:

En esta fase demostrarán la operación del microprocesador mediante una simulación en Verilog del microprocesador ejecutando dos programas de prueba en código RISC-V compatible con las especificaciones arquitecturales.

Programa de validación preliminar:

Antes de la demostración final de la operación del PPU es altamente recomendable que ejecuten el programa que se describe más adelante y se adjunta como documento de esta fase del proyecto. Ya que su comportamiento es conocido, el programa les servirá como una herramienta de "debugging".

Instrucción	Assembly	Address	Hazard
00000011 00000000 00000010 10010011	ADDI r5, r0,48	0	
00000000 00000010 10000000 10000011	LB r1, 0(r5)	4	data
00000000 00010010 11000001 00000011	LBU r2, 1(r5)	8	data
00000000 00100010 11000001 10000011	LBU r3, 2(r5)	12	data
00000000 00000000 11010011 01100011	BGE r1, r0, 3	16	data/control
01000000 00100001 10000011 00110011	SUB r6, r3, r2	20	data
00000000 00000000 00000010 01100011	BEQ r0, r0, 2	24	control
00000000 00100001 10000011 00110011	ADD r6, r3, r2	28	
00000000 01100010 10000001 10100011	SB r6, 3(r5)	32	data
00000000 00000000 00000000 01100011	BEQ r0, r0, 0	36	control
00000000 00000000 00000000 00000000	NOP	40	
00000000 00000000 00000000 00000000	NOP	44	
11111100 00010011 00100000 00000000		48	

El programa lee un byte de la localización 48, un segundo byte de la localización 49 y un tercer byte de la localización 50. Si el primer byte es positivo el segundo byte y el tercer byte se suman y el resultado se guarda en la localización 51. En cambio, si el primer byte es negativo el segundo byte se resta del tercer byte y el resultado se guarda en la localización 51. En la ejecución del programa se deben detectar cuatro data hazards y un posible control Hazard si se da la condición del brinco BGE.

Las instrucciones del programa transcurren por la unidad de ejecución pipelined de la siguiente manera:

IF	ID	EXE	MEM	WB	PC
ADDI					0
LB	ADDI				4

LBU	**LB**	**ADDI**			**8 forwarding desde EX**
LBU	**LBU**	**LB**	**ADDI**		**12 forwarding desde MEM**
BGE	**LBU**	**LBU**	**LB**	**ADDI**	**16 forwarding desde WB**
SUB	**BGE**	**LBU**	**LBU**	**LB**	**20 forwarding desde WB**
BEQ	**SUB**	**BGE**	**LBU**	**LBU**	**24 forwarding desde MEM/WB**
ADD	**BEQ**	**SUB**	**BGE**	**LBU**	**28**
SB	**ADD**	**BEQ**	**SUB**	**BGE**	**32 control hazard**
SB	**canceled**	**canceled**	**BEQ**	**SUB**	**32**
BEQ	**SB**	**canceled**	**canceled**	**BEQ**	**36 forwarding desde EX**
NOP	**BEQ**	**SB**	**canceled**	**canceled**	**40**
NOP	**NOP**	**BEQ**	**SB**	**canceled**	**44 control hazard**
BEQ	**canceled**	**canceled**	**BEQ**	**SB**	**36**
NOP	**BEQ**	**canceled**	**canceled**	**BEQ**	**40**
NOP	**NOP**	**BEQ**	**canceled**	**canceled**	**44 control hazard**
BEQ	**canceled**	**canceled**	**BEQ**	**canceled**	**36**
NOP	**BEQ**	**canceled**	**canceled**	**BEQ**	**40**
NOP	**NOP**	**BEQ**	**canceled**	**canceled**	**44 control hazard**
BEQ	**NOP**	**NOP**	**BEQ**	**canceled**	**36**

Como el comportamiento del programa es conocido pueden saber qué valores se deben producir a la salida de los muxes de data forwarding de la etapa ID, a la salida del Second Operand Handler y el ALU en la etapa EX y en caso de instrucciones load/store qué se lee o qué se escribe y en cuál localización en la etapa MEM. Cuando su simulación no aparezca estar trabajando correctamente deben monitorear esas señales para verificar que tengan los valores esperados para las instrucciones que se encuentran en cada etapa.

Para demostrar que el programa trabaja correctamente deben mostrar (mediante una instrucción de $monitor), el contenido de lo siguiente: **PC** y el contenido de **r1**, **r2**, **r3**, **r5** y **r6**, todas en decimal.

Una vez el programa de prueba finalice (llega a un loop infinito) deben imprimir el contenido del Word en la localización 48 en binario.

Demostración de programas de prueba:

Para la demostración de esta fase se les proveerán dos programas de prueba en un archivo de puro texto cada uno, que deben utilizar para pre cargar las dos memorias del PPU. Cada archivo contiene bytes (en binario) separados por espacios o CR. El primer byte corresponde a la localización 0, el segundo a la 1, el tercero a la 2 y así sucesivamente hasta un eof (como los archivos de pre carga de memorias utilizados a las Fases I y III).

La simulación debe comenzar inicializando **Clk** en cero a tiempo cero. Entonces, debe cambiar de estado cada dos unidades de tiempo de manera perpetua. La señal **Reset** debe tener un valor de 1 a tiempo cero y cambiar a 0 en tiempo 3. La simulación debe culminar cuando el programa llegue a un loop infinito.

Para el primer programa de prueba su simulador debe mostrar (mediante una instrucción de $monitor), el contenido de lo siguiente: **PC**, el address que recibe la memoria de data y el contenido de **r1**, **r3**, **r4**, **r5**, **r8**, **r10**, **r11** y **r12**, todas en decimal. Todas las señales deben estar

declaradas en una sola instrucción de $monitor. Sin embargo, cuando sea requerido, su simulador debe tener la opción de mostrar las señales de control en las diferentes etapas y la instrucción que llega a la etapa de ID (todas en binario).

Para que se le facilite el monitoreo de señales deben instanciar e interconectar los componentes del PPU en el módulo de prueba. De esa manera cualquier señal de interconexión estará disponible para mostrarlas con una instrucción de monitor de ser necesario.

Una vez el programa de prueba finalice (llega a un loop infinito) deben imprimir el contenido de la memoria de data en binario. El contenido de la memoria lo deben imprimir en bytes separados por un espacio (cuatro bytes por línea).

Reglas de juego:

El/la estudiante que abandone el grupo en esta etapa recibirá una calificación de cero en la misma.

Entrega:

- Subir a NEO un archivo comprimido con lo siguiente:

  1. Una portada identificando los/las integrantes del grupo (pdf).
  2. Versión final del diagrama de bloque del microprocesador (pdf).
  3. Código final de la implementación del circuito y su correspondiente módulo de prueba en Verilog

- De manera individual, cada miembro del grupo debe entregar una evaluación de su contribución al proyecto y de la contribución de cada uno de sus compañeros(as). **Sugiera una calificación para cada uno de ellos(as).** Estas evaluaciones deben ser sometidas individualmente.

Rúbrica de evaluación:

- Se adjudicarán 15 puntos si se demuestra una operación correcta del programa de validación.
- Se adjudicarán 20 puntos si se demuestra una operación correcta del primer programa de prueba.
- Se adjudicarán 20 puntos si se demuestra una operación correcta del segundo programa de prueba.
- Se podrían adjudicar puntos parciales dependiendo del nivel de desarrollo del programa de simulación en caso de que uno o ninguno de los programas corra correctamente.