PRÁCTICA 5

Estudio de la UPG.
Implementación de un multiplicador de 16 bits con la UPG y una UC específica.

Objetivos

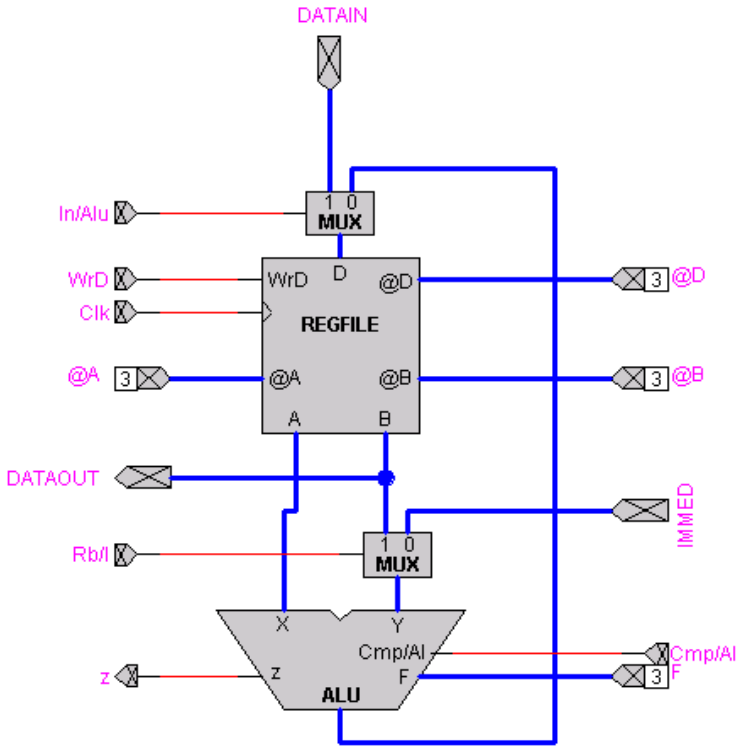
Después de realizar esta práctica, el alumno deberá:

1) Saber escribir la palabra de control de la UPG para que se realice una acción determinada, tanto en binario como mediante el mnemotécnico correspondiente
2) Determinar los cambios que se producen en los registros de la UPG cuando se ejecuta un conjunto de instrucciones
3) Diseñar una unidad de control específica para la UPG, distinguiendo la parte secuencial (el orden en el que se ejecutan las instrucciones en la UPG) de la parte combinacional (la que genera en cada ciclo la instrucción o palabra de control de la UPG)

Desarrollo 1

La unidad de proceso genérica (UPG):

En las siguientes figuras se muestra su estructura a nivel de bloques, la tabla que indica las operaciones que realiza la ALU en función del valor de los bits de selección F y Cmp/Al y por último los 32 bits que forman la palabra de control, esto es, los bits que en cada ciclo tiene que generar la unidad de control para que la UPG actúe como se desea. Todos los conectores que entran a la UPG llegan de la UC excepto DATAIN que viene del exterior, de donde llegan los operandos. De las señales que salen de la UPG, z va a la UC para indicarle si el resultado que sale por la ALU en ese ciclo es cero o no y DATAOUT va al exterior para entregar resultados. El banco de registros contiene 8 registros de 16 bits.



Funcionalidades de la ALU de la UPG

F				
b_2	b_1	b_0	Cmp/Al = 1	Cmp/Al = 0
0	0	0	CMPLT(X,Y)	AND(X,Y)
0	0	1	CMPLE(X,Y)	OR(X,Y)
0	1	0	---	XOR(X,Y)
0	1	1	CMPEQ(X,Y)	NOT(X)
1	0	0	CMPLTU(X,Y)	ADD(X,Y)
1	0	1	CMPLEU(X,Y)	SUB(X,Y)
1	1	0	---	SHA(X,Y)
1	1	1	---	SHL(X,Y)

In/Alu	WrD	@D			@A			Cmp/Al	F			Rb/I	@B			IMMED						
		b_2	b_1	b_0	b_2	b_1	b_0		b_2	b_1	b_0		b_2	b_1	b_0	b_{15}	b_{14}	b_{13}	…	b_2	b_1	b_0
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	…	x	x	x

Ejercicio: palabra de control:
Indicad la tira de bits que forma la palabra de control para cada una de las acciones (filas de la tabla). Indicad con una x los bits de la palabra de control que pueden valer tanto 1 como 0. Si la acción o acciones individuales que se especifican en una fila no pueden realizarse en la UPG en un único ciclo, indícalo con una línea que tache los bits de la palabra de control:

	In/Alu	WrD	@D			@A			Cmp/Al	F			Rb/I	@B			IMMED (hexa)				Palabra de control (hexa)
ADD R2, R1, R0																					
ADD I R3, R3, 5																					
ADDI -, R7, -1 // IN R3																					
OUT R5 // IN R6																					
SHL R4, R4, R5																					
SHLI R4, R4, -1 // OUT R5																					
CMPLTU R5, R3, R4																					
AND R3, R2, R1 // IN R3																					
XOR R2, R4, R4																					

Ejercicio: modificación de los registros de la UPG:
Suponiendo que inicialmente todos los registros contienen el valor 3, especifica los cambios que se producen después de la llegada del flanco de reloj que indica el final de cada ciclo, cuando la UPG ha estado controlada por las palabras de control que se muestran a continuación (estas palabras llegan a la UPG en ciclos consecutivos):

Ciclo	*Palabra de Control*	*Nuevo valor de los registros*
1	ADD R2, R1, R0	
2	ADDI R3, R3, 5	
3	SHL R4, R4, R5	
4	SHLI R4, R4, -1	
5	CMPLTU R5, R3, R4	
6	XOR R2, R4, R4	
7	IN R0	

Desarrollo 2

Diseño de un multiplicador binario de 16 bits, utilizando la UPG:

El objetivo es similar al de la práctica 4, pero en lugar de diseñar una UP específica para el algoritmo de multiplicación ahora disponemos de la UPG. Por tanto, solo nos tenemos que ocupar de crear una UC que genere en cada ciclo la palabra de control (instrucción) para que la UPG trabaje adecuadamente. Sin embargo, no se pueden hacer cumplir algunas de las especificaciones del multiplicador de la práctica4: por ejemplo, no se pueden cargar los datos *X e Y* en el mismo ciclo, porque la UPG tiene un único bus de entrada de datos.

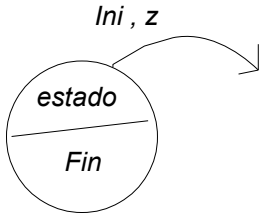
Entonces, supongamos la siguientes características para el procesador:

Entradas: datos → *DATAIN*
control → *Ini*
Salidas: datos → *DATAOUT*
control → *Fin*

Funcionamiento:

1. En el ciclo en el que la señal *Ini* vale 1, el dato en el bus *DATAIN* es válido (dato X).
2. Supongamos que el procesador es más rápido que el usuario: entonces el circuito podría leer los datos X e Y cuando el usuario solo ha escrito el valor de X en el bus de entrada. Para evitar esta circunstancia, supongamos que el procesador solo da validez al segundo (dato Y) cuando la señal *Ini* vale 0. Es decir, el usuario escribe el dato X, pone *Ini* = 1, a continuación escribe el dato Y y pone *Ini* = 0.
3. Mientras el procesador este operando los datos, los posibles cambios en la señal Ini no serán tenidos en cuenta.
4. El procesador opera los datos, y cuando tenga el resultado lo muestra en el bus *DATAOUT* y avisa al usuario poniendo la señal *Fin* a 1 en ese mismo ciclo.
   Para que el usuario no pierda el resultado, el procesador se mantiene en este estado si la señal *Ini* vale 1. Por tanto, para comenzar de nuevo el proceso y operar otros datos, *Ini* debe estar a 0

Grafo de estados de la UC:



Palabra de control de cada estado:

Desarrollo 3

Implementación de la UC:

El circuito *Mult-UPG* es un adelanto del procesador que pretendemos construir: se observan tres bloques diferenciados, UPG, UC, y entradas/salidas externas. A su vez, dentro de la UC se distinguen dos partes:

1- La parte secuencial, encargada de generar el estado siguiente a partir del estado actual y las entradas Ini, z (es decir, la parte que decide en qué orden se envían las instrucciones a la UPG), que implementaremos con una ROM (Rom-secuencia) cuyo contenido obtendremos de la tabla de transiciones:

Estado actual	***Ini z***	***Estado siguiente***

2- La parte combinacional, que genera la palabra de control (instrucción) correspondiente a cada estado de la UC, que también implementamos con una ROM (Rom-instrucciones). Aprovechamos esta ROM para implementar la salida de la UC *Fin*:

Estado	***Fin + Instrucción (hexa)***

Desarrollo 4

Ejercicio propuesto: diseñar un procesador que se encarga de contar el número de unos de un vector de 16 bits

Realizar el diseño de forma análoga al del multiplicador anterior, con señales *Ini, Fin* de forma que la entrada del dato y la salida del resultado sean similares al del circuito multiplicador.