UNIVERSIDAD DE COSTA RICA
ESCUELA DE INGENIERÍA ELÉCTRICA
TAREAS #4 y #5

IE0521 – ESTRUCTURAS DE COMPUTADORES DIGITALES II

I CICLO LECTIVO DEL AÑO 2015

Propósito de Aprendizaje

Que el estudiante desarrolle los conocimientos de un procesador con pipeline mediante la implementación de un modelo en Verilog.

Consideraciones Esenciales

Esta tarea, a pesar de consistir en una única asignación, se le asignará el puntaje como si fuera la tarea #4 y la #5, esto debido a que su complejidad es superior a la de las tareas anteriores.

Esta tarea se realizará en grupos de 3 personas tal y como lo explica el documento *EvaluacionLaboratorios.pdf* de la página virtual. La funcionalidad del circuito a diseñar se relaciona con el tema 4 del curso, así como también con algunos aprendizajes obtenidos en el curso de Estructuras de Computadoras Digitales I. Se implementará un procesador que cuenta con un pipeline de cinco etapas como se ha visto en clase.

Este laboratorio está diseñado para darle al estudiante experiencia en:

1. El marco de trabajo en el modelado y verificación de hardware en Verilog.
2. Principios de diseño como jerarquía, modularidad y encapsulación.
3. Diseño de circuitos digitales.

Desarrollo de la Tarea

Algunas de las características que tendrá el procesador a desarrollar son las siguientes:

1. Estará basado en un procesador con pipeline de 5 etapas.
2. No tendrá direccionamiento indirecto.
3. Tendrá dos registros de uso general, llamados A y B.
4. No se harán operaciones de A con Memoria, ni de B con Memoria, únicamente entre A, B y constantes.
5. La memoria de datos estará separada de la de instrucciones.
6. Cada posición de la memoria de datos guardará un byte.
7. Cada posición de la memoria de instrucciones guardará dos bytes.
8. La memoria de datos tendrá 1024 posiciones (10 bits para indexarla).
9. La memoria de instrucciones tendrá 1024 posiciones (10 bits para indexarla).
10. No habrá subrutinas.
11. No se implementará atención a interrupciones.
12. No se implementará una pila.
13. Se supondrán únicamente números sin signo, por lo que no habrá bandera de rebase.

Cada posición en la memoria de instrucciones representa un microcódigo, compuesto por la siguiente información:

1. Los 10 bits menos significativos portan información adicional, como posiciones me memoria, constantes o saltos relativos, en caso de no ser necesaria, esta información es irrelevante.

2. Los 6 bits más significativos portan el código de la instrucción, esta información permite saber qué instrucción se va a realizar, en la Tabla 1 de muestra la codificación utilizada.

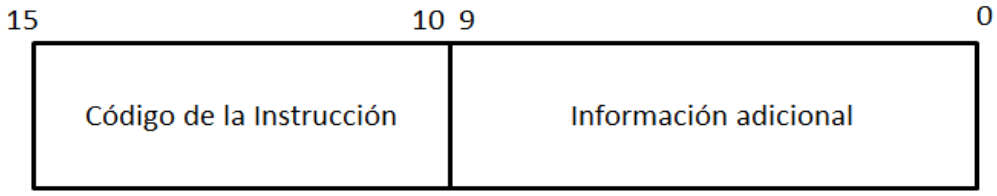

Figura 1 Microcódigo para instrucciones

La información adicional puede ser cualquiera de las siguientes:

1. Dirección de Memoria: En este caso se utilizan los 10 bits para determinar la dirección de la que se desea leer, escribir, o a la que se desea mover el Contador de Programa.
2. Constante: Se utilizan los 8 bits menos significativos (del 0 al 7) para que quede determinada la constante a utilizar.
3. Salto Relativo: Los 6 bits menos significativos (del 0 al 5) indican la magnitud del salto y el siguiente bit (el 6) indica si es hacia adelante o hacia atrás, (0 adelante, 1 atrás).

Se utilizará el siguiente set de instrucciones:

Tabla 1 Codificación de Instrucciones

#	Instrucción	Código	Funcionalidad	Función	Información Adicional
0	LDA	$00	Carga en A lo que hay en la dirección MEM	A ← (MEM)	Dirección de Memoria (MEM)
1	LDB	$01	Carga en B lo que hay en la dirección MEM	B ← (MEM)	Dirección de Memoria (MEM)
2	LDCA	$02	Carga en A la constante CONST	A ← CONST	Constante (CONST)
3	LDCB	$03	Carga en B la constante CONST	B ← CONST	Constante (CONST)
4	STA	$04	Guarda en la dirección MEM lo que hay en A	MEM ← (A)	Dirección de Memoria (MEM)
5	STB	$05	Guarda en la dirección MEM lo que hay en B	MEM ← (B)	Dirección de Memoria (MEM)

UNIVERSIDAD DE COSTA RICA
ESCUELA DE INGENIERÍA ELÉCTRICA
TAREAS #4 y #5
IE0521 – ESTRUCTURAS DE COMPUTADORES DIGITALES II

6	ADDA	$06	Suma A con B y lo guarda en A	A ← (A) + (B)	-
7	ADDB	$07	Suma A con B y lo guarda en B	B ← (A) + (B)	-
8	ADDCA	$08	Suma A con una constante y lo guarda en A	A ← (A) + CONST	Constante (CONST)
9	ADDCB	$09	Suma B con una constante y lo guarda en B	B ← (B) + CONST	Constante (CONST)
10	SUBA	$0A	Hace A menos B y lo guarda en A	A ← (A) - (B)	-
11	SUBB	$0B	Hace B menos A y lo guarda en B	B ← (B) - (A)	-
12	SUBCA	$0C	Hace A menos una constante y lo guarda en A	A ← (A) - CONST	Constante (CONST)
13	SUBCB	$0D	Hace B menos una constante y lo guarda en B	B ← (B) - CONST	Constante (CONST)
14	ANDA	$0E	Hace A & B y lo guarda en A	A ← (A) & (B)	-
15	ANDB	$0F	Hace B & A y lo guarda en B	B ← (A) & (B)	-
16	ANDCA	$10	Hace A & una constante y lo guarda en A	A ← (A)& CONST	Constante (CONST)
17	ANDCB	$11	Hace B & una constante y lo guarda en B	B ← (B)& CONST	Constante (CONST)
18	ORA	$12	Hace A | B y lo guarda en A	A ← (A) | (B)	-
19	ORB	$13	Hace B | A y lo guarda en B	B ← (A) |(B)	-
20	ORCA	$14	Hace A | una constante y lo guarda en A	A ← (A) | CONST	Constante (CONST)
21	ORCB	$15	Hace B | una constante y lo guarda en B	B ← (B) | CONST	Constante (CONST)
22	ASLA	$16	Desplazamiento aritmético de A hacia la izquierda y lo guarda en A	A←{(A_6),(A_5), (A_4), (A_3), (A_2), (A_1), (A_0),0}	-
23	ASRA	$17	Desplazamiento aritmético de A hacia la derecha y lo guarda en A	A←{0,(A_7), (A_6), (A_5), (A_4), (A_3), (A_2), (A_1)}	-

24	JMP	$18	Mueve el PC hacia la dirección MEM	PC←MEM	Dirección de Memoria (MEM)
25	BAEQ	$19	Salta si $Z_A = 1$	(Z_A)=1? PC+SR; PC+1	Salto Relativo (SR)
26	BANE	$1A	Salta si $Z_A = 0$	(Z_A)=0? PC+SR; PC+1	Salto Relativo (SR)
27	BACS	$1B	Salta si $C_A = 1$	(C_A)=1? PC+SR; PC+1	Salto Relativo (SR)
28	BACC	$1C	Salta si $C_A = 0$	(C_A)=0? PC+SR; PC+1	Salto Relativo (SR)
29	BAMI	$1D	Salta si $N_A = 1$	(N_A)=1? PC+SR; PC+1	Salto Relativo (SR)
30	BAPL	$1E	Salta si $N_A = 0$	(N_A)=0? PC+SR; PC+1	Salto Relativo (SR)
31	BBEQ	$1F	Salta si $Z_B = 1$	(Z_B)=1? PC+SR; PC+1	Salto Relativo (SR)
32	BBNE	$20	Salta si $Z_B = 0$	(Z_B)=0? PC+SR; PC+1	Salto Relativo (SR)
33	BBCS	$21	Salta si $C_B = 1$	(C_B)=1? PC+SR; PC+1	Salto Relativo (SR)
34	BBCC	$22	Salta si $C_B = 0$	(C_B)=0? PC+SR; PC+1	Salto Relativo (SR)
35	BBMI	$23	Salta si $N_B = 1$	(N_B)=1? PC+SR; PC+1	Salto Relativo (SR)
36	BBPL	$24	Salta si $N_B = 0$	(N_B)=0? PC+SR; PC+1	Salto Relativo (SR)

Aclaraciones sobre la Tabla 1:

- Cuando un registro o posición de memoria está entre paréntesis, se está haciendo referencia al contenido en esa posición.
- La función lógica Y se representa con un & y la O con |.
- Se representa como Z_X, C_X, N_X a las banderas cero, acarreo y signo del registro X.
- En los saltos condicionales, la función (X)=0? Y; W se lee: “Si el contenido de X es igual a cero, brinco a Y, sino a W”.

En la Figura 3 se muestra un diagrama de bloques a muy alto nivel de cómo se podría implementar el pipeline del procesador propuesto.

UNIVERSIDAD DE COSTA RICA
ESCUELA DE INGENIERÍA ELÉCTRICA
TAREAS #4 y #5
IE0521 – ESTRUCTURAS DE COMPUTADORES DIGITALES II

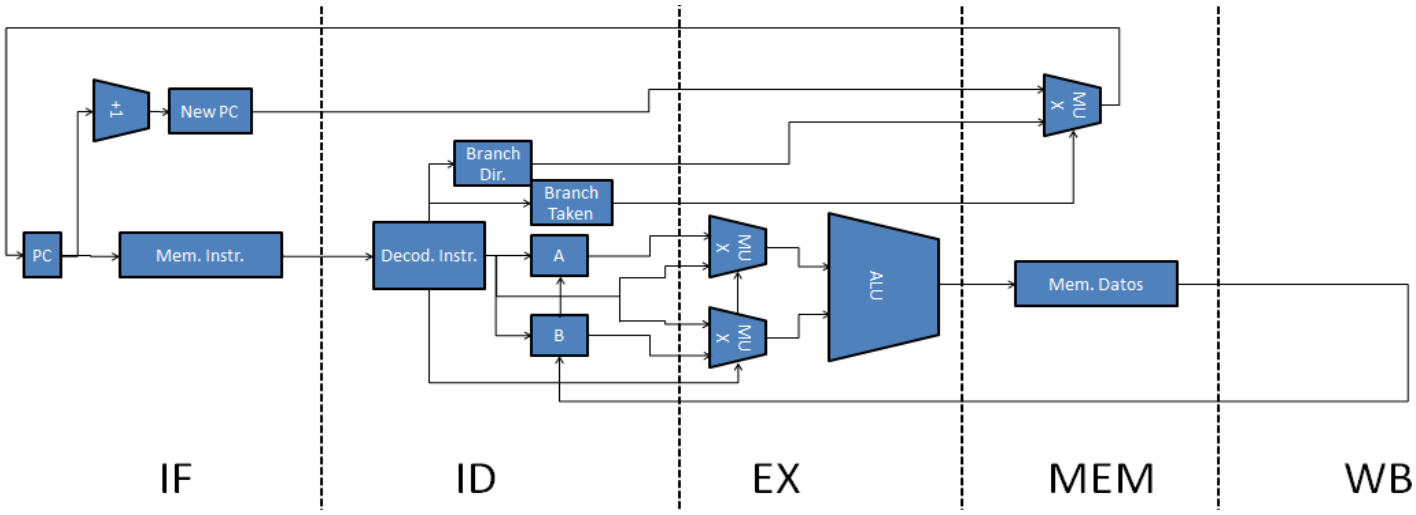

Figura 2 Pipeline Sugerido

1. Durante Instruction Fetch (IF) se lee el valor de PC y se lee el contenido de esa dirección en la memoria de instrucciones. Se le suma 1 al valor del PC para saber el próximo valor que tendrá PC.
2. Durante Instruction Decode (ID) se decide qué operación se llevará a cabo, y se pone en los registros A y/o B los valores necesarios. En caso de que la instrucción sea un salto, se calcula la dirección nueva y si el salto será tomado o no.
3. Durante Execution (EX) se realiza la operación en la ALU utilizando A, B y las constantes necesarias.
4. Durante Memory (MEM) se escribe o se carga de la memoria de datos el dato a utilizar.
5. Durante Write Back (WB) se escribe de vuelta a los registros el valor calculado o el valor leído.

Este diseño se dividirá en tantos módulos como los estudiantes consideren necesarios.
Los estudiantes deberán agregar las señales y lógica que consideren necesaria para el correcto funcionamiento del circuito.
Se debe asegurar el correcto funcionamiento del procesador por medio de una estrategia de pruebas, tal y como lo describe el archivo de evaluación de los laboratorios.
Para escribir los datos en las memorias deben utilizar el comando readmem de Verilog.

Retos

1. Implemente optimizaciones para evitar todos los hazards que encuentre mientras hace sus pruebas.

Consideraciones Adicionales

UNIVERSIDAD DE COSTA RICA
ESCUELA DE INGENIERÍA ELÉCTRICA
TAREAS #4 y #5

IE0521 – ESTRUCTURAS DE COMPUTADORES DIGITALES II

1. La tarea deberá ser entregada mediante el sitio virtual del curso en la fecha designada ahí mismo.
2. Deberán entregar un archivo .zip o .tar.gz que tendrá la siguiente estructura:
    a. Directorio Principal *"<CARNÉ1>_<CARNÉ2>_<CARNÉ3>"*
        i. Directorio de Código *"CÓDIGO"*
            1. Archivo(s) con la descripción del circuito (*.v)
            2. Archivo(s) de pruebas (*.v)
            3. Makefile, que permita compilar y ejecutar el programa de forma sencilla.
            4. README, que incluya instrucciones de uso del Makefile.
        ii. Reporte *"<CARNÉ1>_<CARNÉ2>_<CARNÉ3>.pdf"*
3. Se asume que a este nivel, el estudiante tiene conocimientos de programación en Verilog y uso de Makefile, cualquier refrescamiento del lenguaje será responsabilidad del estudiante.