PRÁCTICO 9 - Implementación de la ISA

Antes de empezar el practico: Resumen para entender que es la ISA:

La ISA es la interfaz entre el hardware y el software de una computadora. Define el set de instrucciones que el procesador puede ejecutar, así como los registros y las instrucciones básicas que se pueden utilizar para escribir programas. Es un contrato entre el software y el hardware.

El datapath simple con la unidad de control es una representación de cómo se implementan físicamente las instrucciones de la ISA en el hardware del procesador. Muestra los componentes principales del procesador y cómo están interconectados para ejecutar las instrucciones.

Ejercicio 1:

Marque con una barra invertida e indique el número de bits que representa cada una de las líneas del data path. (me dio paja)

Ejercicio 2:

Considerando la siguiente distribución de instrucciones en un programa:

R-type	I-Type	LDUR	STUR	CBZ	B
24%	28%	25%	10%	11%	2%

a) Qué porcentaje de todas las instrucciones utiliza data memory?

Para hacer esto necesitamos identificar las instrucciones que acceden a la memoria en este caso son:

- LDUR
- STUR

Por lo tanto sumando los porcentajes de cada uno obtener el porcentajes de todas las instrucciones que utilizan data memory: 25% + 10% = 35%

b) Qué porcentaje de todas las instrucciones utiliza instruction memory?

En este caso debemos identificar las instrucciones que acceden a la memoria de instrucciones.

Practicamente todas las instrucciones acceden a la memoria de instrucciones ya que deben ser buscadas desde la memoria de instrucciones para ser ejecutadas.

Dado que todas las instrucciones del programa de ejemplo acceden a la memoria de instrucciones el porcentaje es el 100%.

c) Qué porcentaje de todas las instrucciones utiliza el sign extend?

Los tipos de instrucción que utilizan el bloque de extensión de signo son:

●Tipo I = 28%
●Tipo CB = 11%
●Tipo B = 2%
●Tipo D = 25%

Por lo tanto, el 76% de las instrucciones utilizan sign extend (generalmente lo vemos por los tipos de instrucciones que tienen un campo inmediato)

EL sign extend se usa para extender los bits de un campo. Para cualquier tipo de instrucción decodifica la instrucción que ingrese y determina en que bits de la instrucción esta realmente el inmediato,

Ejercicio 3:

Complete la tabla con el estado de las señales sin mirar el libro.
Indique con X las condiciones no-importa.

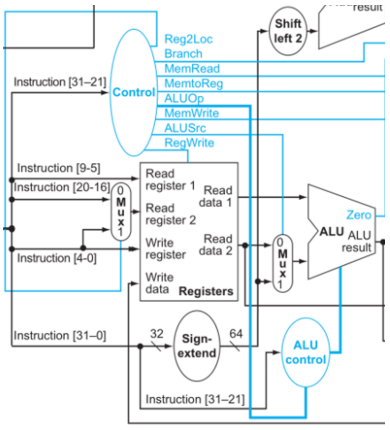
Instr	Reg2Loc	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch
R-type	**0**	**0**	**0**	**1**	**0**	**0**	**0**
LDUR	**X**	**1**	**1**	**1**	**1**	**0**	**0**
STUR	**1**	**1**	**X**	**0**	**0**	**1**	**0**
CBZ	**1**	**1**	**X**	**0**	**0**	**0**	**1**

La tabla es como una tabla de verdad del datapath pero simplificada.

Empecemos viendo la instrucción tipo “R” que son operaciones entre dos registros y el resultado va a parar a otro registro, por lo tanto tiene 3 argumentos.

Entonces viendo la green card sabemos que el campo del 4..0 es para rd, entonces ese campo lo usamos, el campo del 9..5 es para Rn que tambien lo usamos y por ultimo el campo del 20..16 es para Rm y lo usamos en las tipo “r” por lo tanto el multiplexor tiene que ser 0, la entrada de selección del multiplexor es Reg2Loc, por lo tanto Reg2Loc es las instrucciones tipo “r” tiene que ser 0.

Como sabemos que la instruccion de tipo “R” opera entre dos registros, el read data 2 tiene que entrar a la ALU, por lo tanto el multiplexor tiene que ser 0, para que este sea 0, el ALUSrc tiene que ser 0.



Siguiendo con el flujo de datos, en las instrucciones de tipo “R” no tienen que entra a la memoria a buscar algo, entonces tienen que pasar por el siguiente multiplexor, y guardar el resultado, por lo tanto el ultimo multiplexor tiene que ser 0, y para que este sea 0, el Memtoreg tiene que ser 0

Y ya esta para el tipo “R” no tiene mucho flujo. Pero analicemos las otras:

- RegWrite: Determina si un dato debe escribirse en el registro (habilita la escritura de un registro), si la senal es 1 se escribe el registro en el banco de registros y si la sena es 0 no se escribe ningún registro, por lo tanto en la instrucción “R” es 1.
- MemRead: Determina si un dato debe leerse de la data memory, en la instrucción “R”, como en esta instrucción no leemos un dato de memoria, por lo tanto es 0.
- MemWrte: Determina si un data debe escribirse en la data memory, si se scribe el dato es 1 y si no se escribe es 0, por lo tanto es 0, ya que el resultado va a parar a un registor no a memoria.
- Branch: Como la instrucción “r” no hace salto, y la parte de “zero” de la alu no me asegura que sea 0, la instrucción de branch tiene que ser 0.

Los mismo análisis los vamos haciendo para las otras instrucciones.

Para el LDUR, recordemos que se utiliza para cargar (leer) un valor desde la memoria y almacenarlo en un registro.

Para el STUR, recordemos que se utiliza para almacenar (escribir) un valor desde un registro en una ubicación específica de la memoria.

Ejercicio 4:

Cuando se fabrican los chips de silicio, defectos en los materiales y errores en la fabricación pueden generar circuitos defectuosos. Un defecto común es que un cable de señal se rompa y siempre registre un ‘0’ lógico. Esto se conoce comúnmente como “stuck-at-0 fault”

a) ¿Qué instrucciones operarían de forma incorrecta si el cable MemToReg está atascado en ‘0’?

SI el cable de MemToReg esta atascado en “0” signifca que todas las instrucciones que leen la memoria van a operar de forma incorrecta. Es **ldur**.

b) ¿Qué instrucciones operarían de forma incorrecta si el cable ALUSrc está atascado en ‘0’?

En este caso si ALUSrc esta atascado en 0, las que funcionarian incorrectamente seria **LDUR y STUR**, ya que no podrán utilizar el desplazamiento inmediato `imm` para calcular la dirección de memoria. En lugar de eso, la ALU usaría un registro como el segundo operando, llevando a cálculos incorrectos de la dirección de memoria y, por ende, a operaciones incorrectas de carga y almacenamiento. Por lo tanto si tambien hubiese operaciones de tipo “i” las que usan inmediato, tampoco andarian bien.

c) ¿Qué instrucciones operarían de forma incorrecta si el cable Reg2Loc está atascado en ‘0’?

Si Reg2Loc esta atascado en “0”, si yo necesito leer el registor que esta almacenado en los bits del 4-0 va a fallar, por ejemplo la **STUR y CBZ** se rompen ya que necesito leer esos registros.

Ejercicio 5:

Agregando una compuerta en diagrama del data path & control, cambie la implementación de la instrucción CBZ a CBNZ.

Simplemente lo que hay que hacer es poner una negación en el cable de “zero” de la alu, entonces se cambia la implementación.

Ejercicio 6: (mucho texto, estoy cansada)

En este ejercicio analizaremos en detalle cómo se ejecuta una instrucción en el single-cycle datapath, asumiendo que la palabra de instrucción que ejecuta el procesador es: 0xf8014062, dado que PC=0x100.

Los que nos dice esto es que la salida viendo en el datapath en el PC es 0x100 y la instruccion que sale es 0xf8014062,

a) ¿Cuáles son las salidas de los bloques Sign-extend y Shift left 2 para esta palabra de instrucción?

b) ¿Cuáles son los valores de entrada a la unidad ALU control para esta palabra de instrucción?

c) ¿Cuál es la nueva dirección en el PC después de ejecutar esta instrucción?

d) Mostrar los valores de las entradas y salidas de cada Mux durante la ejecución de esta instrucción. Para los valores que son salidas de Registers, utilizar “Reg [Xn]”.

e) ¿Cuáles son los valores de entrada de la ALU y las dos unidades Add?

f) ¿Cuáles son los valores de todas las entradas del bloque Registers?

Ejercicio 7:

Agregar a la implementación de la ISA la instrucción de salto incondicional B, a partir de una nueva señal que sale de Control, denominada UncondBranch.