Taller entrega 3

IIC2343

Formalidades

º La entrega es el 29 de Abril **20:00 hrs**

º La calificación de esta entrega equivale al 50% de la nota final de proyecto, y se puede eliminar.

º Esta entrega ocupa como base la entrega 2

º Es posible sacarse un 7 aún si no cumple todos los requerimientos

Contenidos

º Motivación

º Componentes de esta entrega (Adder, SP y MUX IN)

º Pasos de desarrollo

º Assembler

º Simulación y testeo de esta entrega

º Formato de entrega

º Parte Práctica en el segundo módulo: Simulación de componentes

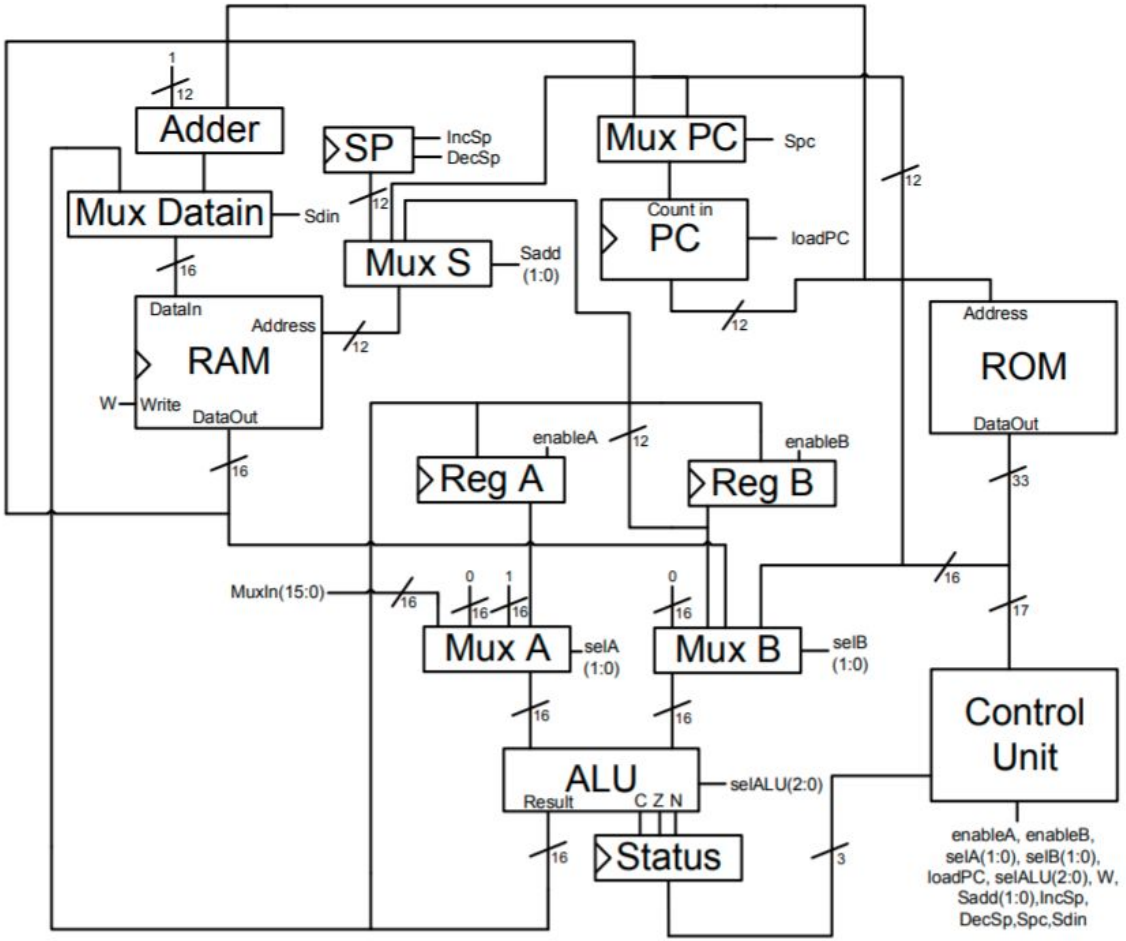
Motivación

º Hasta el momento la única forma de “interactuar” con nuestro computador era a través de la RAM. Sin embargo, un computador como nosotros lo conocemos tiene teclado, mouse, pantallas táctiles, etc.

º Nuestra placa no tiene nada de esas formas avanzada de interacción, pero de la entrega 1 nosotros ya conocemos dos formas de input, botones y switches!

º Si finalizan completamente esta entrega ¡habrán hecho un computador con el cual pueden realmente interactuar y soporta funciones! (al menos de forma simulada)

Diagrama:


Adder
SP
IncSp
DecSp
Mux PC
Spc
Mux Datain
Sdin
Count in
PC
loadPC
Mux S
Sadd
(1:0)
DataIn
Address
RAM
W
Write
DataOut
Address
ROM
DataOut
enableA
enableB
Reg A
Reg B
MuxIn(15:0)
Mux A
selA
(1:0)
Mux B
selB
(1:0)
Control
Unit
ALU
Result
C Z N
selALU(2:0)
Status
enableA, enableB,
selA(1:0), selB(1:0),
loadPC, selALU(2:0), W,
Sadd(1:0),IncSp,
DecSp,Spc,Sdin

Componentes de esta entrega: Adder

º Componente muy similar a lo hecho para la ALU

º Su entrada y su salida tiene distinta cantidad de bits

Entradas		Salidas
a(11:0)	b(11:0)	result(15:0)
*	*	a + b

Figura 5: Tabla de verdad de Adder.

Componentes de esta entrega: SP

º Es un registro con propiedades de decremento e incremento

º Al igual que Status y PC de la entrega pasada se recomienda usar instancias de Reg para su implementación

º Considere que si el contador de su implementación parte en cero, ser´a necesario que al principio de cada programa se disminuya en una unidad SP de manera que este parta en 111111111111

Entradas			Salidas
clock	up	down	dataout
Flanco Subida	*	*	dataout
Flanco Subida	1	0	dataout + 1
Flanco Subida	0	1	dataout - 1

Figura 4: Tabla de verdad de SP.

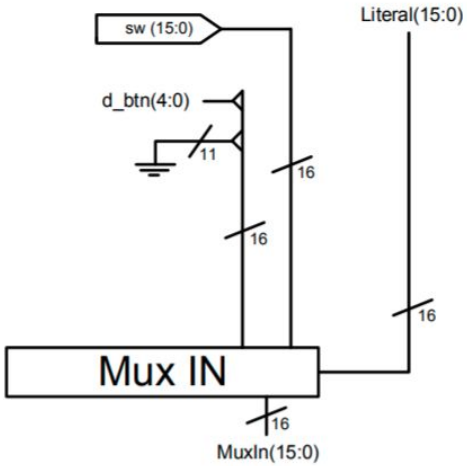
Componentes de esta entrega: MUX IN

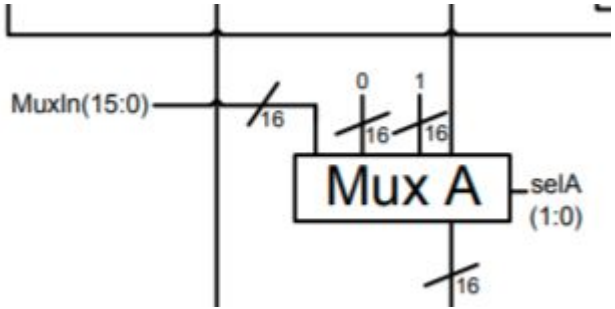
º Un selector como el que siempre han hecho.

º Selecciona desde el literal señales de input de botones y switches. El largo es de 16 bits, lo que implica que algunas señales serán “expandidas”.

Puerto	Input
0	Switches
1	Botones
*	Nada

Figura 3: Tabla de puertos Input.





Pasos de Desarrollo: Tener completada la entrega 2

º Partir de la base que se tiene todos los requerimientos de dicha entrega para poder avanzar correctamente en el desarrollo de esta entrega. En caso de que falte algún componente o instrucción no testeada, el grupo debe asegurarse de completar y testear el computador de la entrega pasada.

º Se recomienda revisar talleres pasados, issues y realizar todos los test de dicha entrega.

Pasos de Desarrollo: Opcodes y Señales

º Analizar y crear la forma como la ROM entrega las instrucciones a la Control Unit. (Cuantos bits ocupar, que significará cada bit, o grupo de bits, ¡ocupen sus bits de forma inteligente!)

º Se recomienda usar tablas que identifica el código de operación y sus respectivas señales asociadas.

INSTRUCCIÓN	SEÑAL 1	SEÑAL 2	...	SEÑAL N
MOV A,B	1	0		1

Pasos de Desarrollo: Opcodes y Señales

- Los nuevos opcodes soportados son los siguientes:

```
MOV A,(B)      (cargar Mem[B] en A)
    B,(B)      (cargar Mem[B] en B)
    (B),A      (guarda A en Mem[B])
    (B),Lit    (guarda Lit en Mem[B])

ADD A,(B)      (guarda A+Mem[B] en A)
    B,(B)      (guarda A+Mem[B] en B)

SUB A,(B)      (guarda A-Mem[B] en A)
    B,(B)      (guarda A-Mem[B] en B)

AND A,(B)      (guarda A and Mem[B] en A)
    B,(B)      (guarda A and Mem[B] en B)

OR  A,(B)      (guarda A or Mem[B] en A)
    B,(B)      (guarda A or Mem[B] en B)
```

```
XOR A,(B)      (guarda A xor Mem[B] en A)
    B,(B)      (guarda A xor Mem[B] en B)

NOT (B),A      (guarda not A en Mem[B])

SHL (B),A      (guarda shift left A en Mem[B])

SHR (B),A      (guarda shift right A en Mem[B])

INC (B)        (incrementa Mem[B] en una unidad)

CMP A,(B)      (hace A-Mem[B])

PUSH A          Mem[SP] = A, SP--
     B          Mem[SP] = B, SP--

POP  A          SP++, A = Mem[SP]
     B          SP++, B = Mem[SP]

CALL Dir        Mem[SP] = PC, PC = Dir, SP--
```

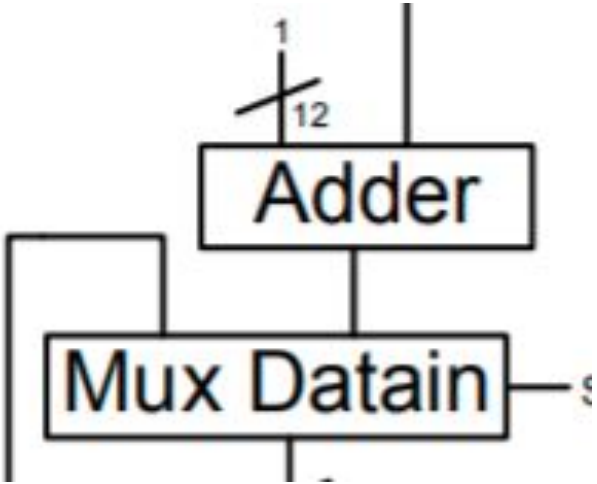
```
RET             SP++, PC = Mem[SP]

IN   A,Lit      A = Input[Lit]
     B,Lit      B = Input[Lit]
     (B),Lit    Mem[B] = Input[Lit]
```

Pasos de Desarrollo: Comportamientos de los Componentes

º Además de los componentes antes vistos tenemos nuevos selectores:

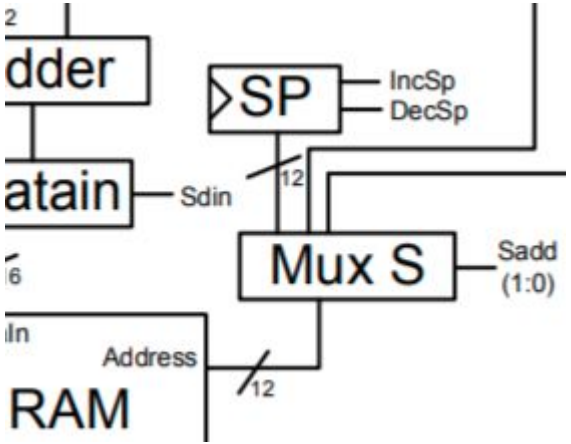
MUX Datain: selector que selecciona el dato de entrada de la Main Memory, ambos de 16 bits.



Pasos de Desarrollo: Comportamientos de los Componentes

º Además de los componentes antes vistos tenemos nuevos selectores:

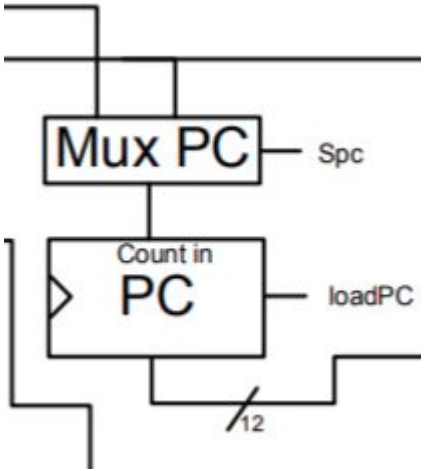
MUX S: selecciona la dirección a que apuntará la Main Memory, los valores de selección de 12 bits. Este puede ser la salida del SP, el literal de la ROM o la salida del Registro B.



Pasos de Desarrollo: Comportamientos de los Componentes

º Además de los componentes antes vistos tenemos nuevos selectores:

MUX PC : selector que seleccionen entre dos entradas para el valor de entrada del PC, cada una de 12 bits. Puede ser la salida de la RAM o el literal que sale de la ROM.



Pasos de Desarrollo: Assembly a Binario

° Existen principalmente dos formas de pasar Assembly a Binario:

1.- Manualmente (No recomendado)

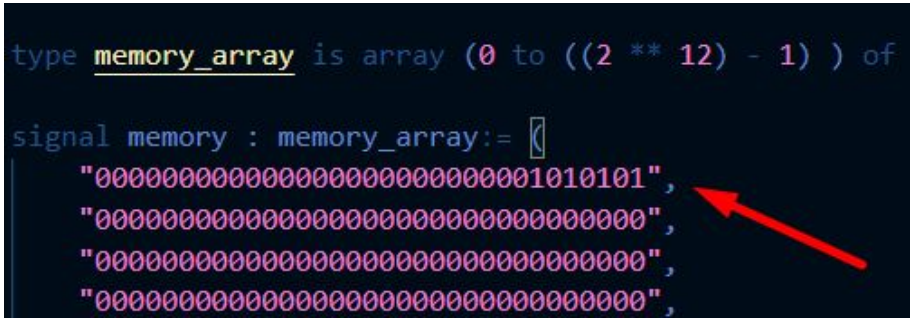
2.- Utilizando un Assembler (Recomendado)

Pasos de Desarrollo: Assembly a Binario

1.- La manera manual consiste en que ustedes transcriban a mano su assembly, instrucción por instrucción, utilizando la codificación correspondiente a su palabra de control.

A modo de ejemplo, si codificaron MOV A, B como 1010101...

MOV A, B → 1010101 →



Pasos de Desarrollo: Assembly a Binario

Manera manual:

> Pros:

- No usan Assembler.

> Contras:

- Transcripción toma mucho tiempo.
- Propenso a error humano.
- Seguimiento de errores es costoso en tiempo.

Pasos de Desarrollo: Assembly a Binario

2.- El uso de un Assembler consiste en que ustedes creen un programa/script que automatice la transcripción de assembly a binario.

¡El assembler no es evaluado!

```
C:\>python assembler.py assembly.txt ROM.vhd
```

Pasos de Desarrollo: Assembly a Binario

Uso de Assembler:

> Pros:

- Pueden programarlo en su lenguaje de preferencia
- Transcripción assembly-binario prácticamente instantánea.
- Fácilmente expandible

> Contras:

- Tienen que programarlo...

Pasos de Desarrollo: Assembly a Binario

Uso de Assembler:

Nota: La funcionalidad de su assembler depende de ustedes!

- Si les basta con que genere texto para copiar-pegar, esta bien ✔️

- Si quieren que su assembler sea más sofisticado y realice el manejo de archivos, también está bien ✔️

En conclusión: Es su assembler, hecho a su manera, para ustedes 😊

Pasos de Desarrollo: Simulación

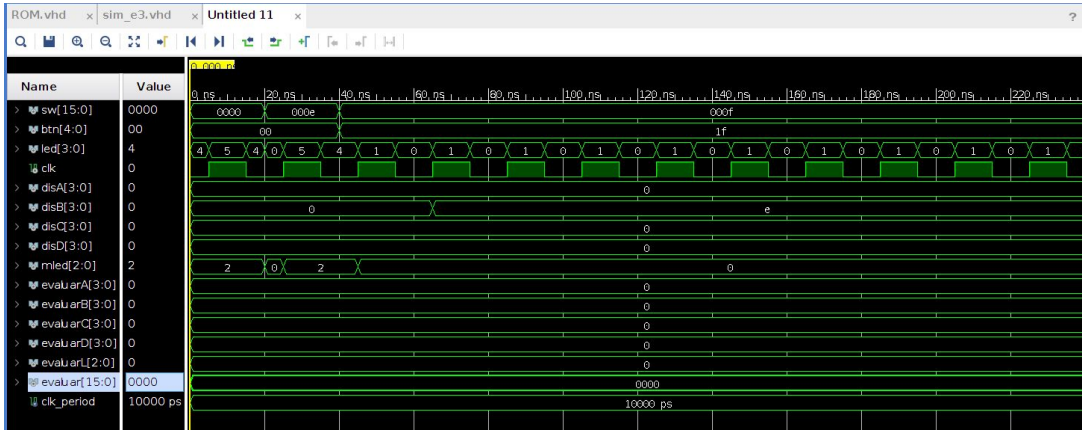
º Finalmente, teniendo todo lo anterior listo, deberán simular y observar si los valores transitivos y finales de los display corresponden a los valores esperados o no.

º Para esto se le entregar un archivo de simulación listo con explicaciones de un archivo de simulación listo con explicaciones de cómo deberán probar sus test. (Estos serán subidos en el transcurso de la semana).

Pasos de Desarrollo: Simulación

º Ejemplo 7 del enunciado en simulación:

```
-- MOV B,0
btn <= "00000";
sw <= "0000000000001110"; -- esto se guardara e
evaluarA <= (disA xor "0000");
evaluarB <= (disB xor "0000");
evaluarC <= (disC xor "0000");
evaluarD <= (disD xor "0000");
wait for 20 ns;
-- IN (B),0
btn <= "11111";
sw <= "0000000000001111"; -- esto no se guarda
evaluarA <= (disA xor "0000");
evaluarB <= (disB xor "0000");
evaluarC <= (disC xor "0000");
evaluarD <= (disD xor "0000");
wait for 20 ns;
-- IN A,2
btn <= "11111";
```



Lo que deben entregar

º En sus repositorios de Github deben subir la carpeta del proyecto de Vivado

º En dicho debe estar contenido los archivos srcs, con las entidades desarrolladas a lo largo de esta entrega y las anteriores

º Un informe en formato Markdown donde se detalla lo que hizo cada persona como también muestras de las simulaciones de que su entidad funciona correctamente.

º Los archivos ROM, un archivo por cada programa de ejemplo.

Evaluación de Pares

º Esta evaluación se hará durante el desarrollo de la entrega hasta el miércoles 27 de mayo a las 23:00 horas

º El objetivo es poder lograr una comunicación efectiva con todo el equipo, y que si sea necesario una intervención del cuerpo docente sea antes de entregar para evitar perjudicar la nota final

º Si presenta alguna complicación no dude en informar lo más pronto posible para poder tomar las medidas pertinentes.

¿Dudas?