

Microarquitectura del CPU

Organización del Computador I
Departamento de Computación
Universidad de Buenos Aires

¡Ejercicios!

Ejercicio 1 - Máquina Orga1

- 1) Diseñar el **camino de datos** de la arquitectura de la máquina ORGA1, suponiendo que se encuentra resuelta la decodificación y el acceso a memoria de la máquina. No dibujar la unidad de control para simplificar el diagrama.

Ejercicio 1 - Máquina Orga1

- 1) Diseñar el camino de datos de la arquitectura de la máquina ORGA1, suponiendo que se encuentra resuelta la decodificación y el acceso a memoria de la máquina. No dibujar la unidad de control para simplificar el diagrama.

Se cuentan con los siguientes circuitos:

- Una ALU con 2 registros de 16 bits (ALU_IN1 y ALU_IN2) que usa de entradas y 5 registros que usa de salida: ALU_OUT de 16 bits y (ALU_Z , ALU_N , ALU_C y ALU_V) de 1 bit. Sus señales de control son:

| Señal | Efecto |
|-------------|--|
| ALU_{add} | $ALU_OUT := ALU_IN1 + ALU_IN2$ |
| ALU_{sub} | $ALU_OUT := ALU_IN1 - ALU_IN2$ |
| ALU_{neg} | $ALU_OUT := - ALU_IN1$ |
| ALU_{and} | $ALU_OUT := ALU_IN1 \text{ AND } ALU_IN2$ |
| ALU_{not} | $ALU_OUT := \text{NOT } ALU_IN1$ |

- Un extensor de signo complemento a 2 ($SIGN_EXT$) con un registro de entrada de 8 bits (EXT_IN) y un registro de salida de 16 bits (EXT_OUT). Sus señales de control son:

| Señal | Efecto |
|------------------|---|
| $SIGN_EXT_{on}$ | activa la operación de extensión de signo de 8 bits a 16 bits |

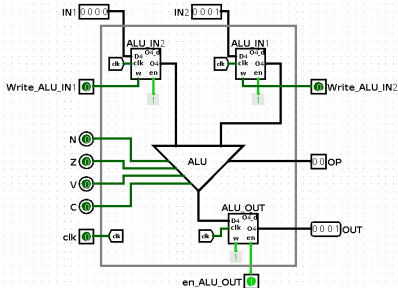
Datapath

Ejemplo de un componente ALU.

Datapath

Ejemplo de un componente ALU.

Componente completo

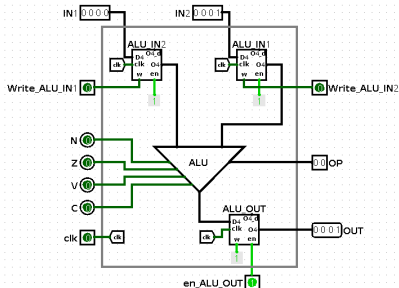


Abstracción

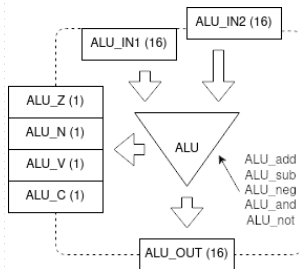
Datapath

Ejemplo de un componente ALU.

Componente completo



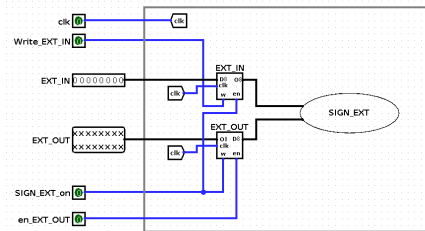
Abstracción



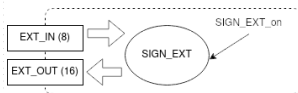
Datapath

Ejemplo de un componente EXTENSOR DE SIGNO.

Componente completo



Abstracción



Microprogramas

- ▶ Datapath
- ▶ Lenguaje de microinstrucciones (RTL)

Microprogramas

- ▶ Datapath
 - ▶ Lenguaje de microinstrucciones (RTL)
-

Vamos a construir el micro con:

- ▶ Componentes
- ▶ Registros
- ▶ Líneas/buses

Microprogramas

- ▶ Datapath
 - ▶ Lenguaje de microinstrucciones (RTL)
-

Vamos a construir el micro con:

- ▶ Componentes
- ▶ Registros
- ▶ Líneas/buses

El lenguaje y Notación: Componentes

- ▶ Definimos **componentes** como circuitos con entradas, salidas y señales.
- ▶ Las **señales** son entradas que modifican el comportamiento de los circuitos.
- ▶ Las señales **se activarán** según como indique el *microprograma*.
- ▶ Se simbolizan con óvalos en el datapath.

Microprogramas

- ▶ Datapath
 - ▶ Lenguaje de microinstrucciones (RTL)
-

Vamos a construir el micro con:

- ▶ Componentes
- ▶ Registros
- ▶ Líneas/buses

El lenguaje y Notación: Registros

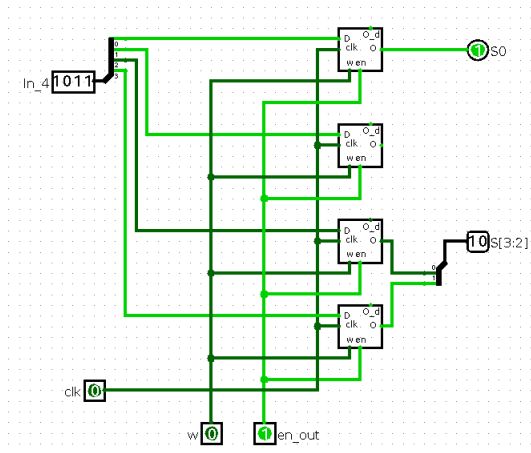
- ▶ Existen registros que almacenan conjuntos de valores.
- ▶ Los registros pueden ser usados por completo o parte de ellos.
- ▶ Se simbolizan con rectángulos en el datapath.

Ejemplos

- ▶ ALU_IN1: Registro ALU_IN1
- ▶ R0[0]: Bit 0 del registro R0
- ▶ R2[3:2]: Del bit 2 al bit 3 del registro R2

El lenguaje y Notación: Registros - O en circuito...

- R0[0]
- R2[3:2]



Microprogramas

- ▶ Datapath
 - ▶ Lenguaje de microinstrucciones (RTL)
-

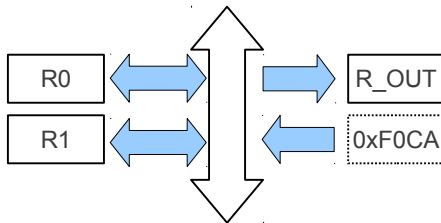
Vamos a construir el micro con:

- ▶ Componentes
- ▶ Registros
- ▶ Líneas/buses

El lenguaje y Notación: Líneas

- ▶ Los datos se mueven por caminos (líneas).
- ▶ Podemos mover un dato de un registro a otro si hay un camino directo entre ellos.
- ▶ Podemos asignar un valor constante almacenado a un registro.
- ▶ Se simbolizan con flechas en el datapath.

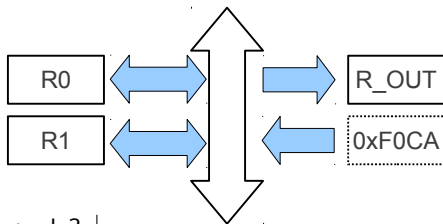
Ejemplos



El lenguaje y Notación: Líneas

- ▶ Los datos se mueven por caminos (líneas).
- ▶ Podemos mover un dato de un registro a otro si hay un camino directo entre ellos.
- ▶ Podemos asignar un valor constante almacenado a un registro.
- ▶ Se simbolizan con flechas en el datapath.

Ejemplos



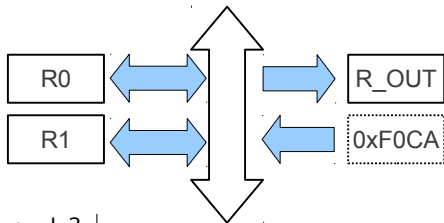
¿vale?

R1:= R0

El lenguaje y Notación: Líneas

- ▶ Los datos se mueven por caminos (líneas).
- ▶ Podemos mover un dato de un registro a otro si hay un camino directo entre ellos.
- ▶ Podemos asignar un valor constante almacenado a un registro.
- ▶ Se simbolizan con flechas en el datapath.

Ejemplos

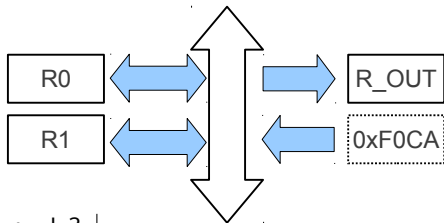


| | ¿vale? | |
|-----------|--------|--------------------------------|
| R1:= R0 | ✓ | Copia el contenido de R0 en R1 |
| R_OUT:=R0 | | |

El lenguaje y Notación: Líneas

- ▶ Los datos se mueven por caminos (líneas).
- ▶ Podemos mover un dato de un registro a otro si hay un camino directo entre ellos.
- ▶ Podemos asignar un valor constante almacenado a un registro.
- ▶ Se simbolizan con flechas en el datapath.

Ejemplos

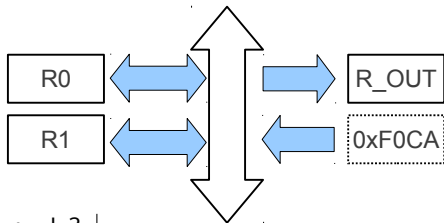


| | ¿vale? | |
|------------|--------|-----------------------------------|
| R1:= R0 | ✓ | Copia el contenido de R0 en R1 |
| R_OUT:=R0 | ✓ | Copia el contenido de R0 en R_OUT |
| R0:=0xF0CA | | |

El lenguaje y Notación: Líneas

- ▶ Los datos se mueven por caminos (líneas).
- ▶ Podemos mover un dato de un registro a otro si hay un camino directo entre ellos.
- ▶ Podemos asignar un valor constante almacenado a un registro.
- ▶ Se simbolizan con flechas en el datapath.

Ejemplos

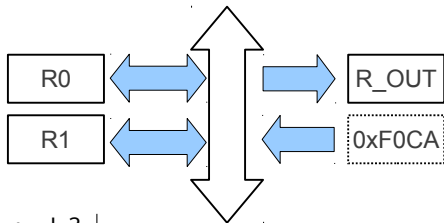


| | ¿vale? | |
|------------|--------|-----------------------------------|
| R1:= R0 | ✓ | Copia el contenido de R0 en R1 |
| R_OUT:=R0 | ✓ | Copia el contenido de R0 en R_OUT |
| R0:=0xF0CA | ✓ | Copia la constante 0xF0CA a R0 |
| R0:=R_OUT | | |

El lenguaje y Notación: Líneas

- ▶ Los datos se mueven por caminos (líneas).
- ▶ Podemos mover un dato de un registro a otro si hay un camino directo entre ellos.
- ▶ Podemos asignar un valor constante almacenado a un registro.
- ▶ Se simbolizan con flechas en el datapath.

Ejemplos

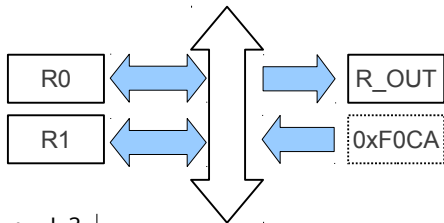


| | ¿vale? | |
|------------|--------|--|
| R1:= R0 | ✓ | Copia el contenido de R0 en R1 |
| R_OUT:=R0 | ✓ | Copia el contenido de R0 en R_OUT |
| R0:=0xF0CA | ✓ | Copia la constante 0xF0CA a R0 |
| R0:=R_OUT | × | El registro R_OUT es de solo escritura |
| R1:=x | | |

El lenguaje y Notación: Líneas

- ▶ Los datos se mueven por caminos (líneas).
- ▶ Podemos mover un dato de un registro a otro si hay un camino directo entre ellos.
- ▶ Podemos asignar un valor constante almacenado a un registro.
- ▶ Se simbolizan con flechas en el datapath.

Ejemplos



| | ¿vale? | |
|------------|--------|---|
| R1:= R0 | ✓ | Copia el contenido de R0 en R1 |
| R_OUT:=R0 | ✓ | Copia el contenido de R0 en R_OUT |
| R0:=0xF0CA | ✓ | Copia la constante 0xF0CA a R0 |
| R0:=R_OUT | × | El registro R_OUT es de solo escritura |
| R1:=x | × | No es posible asignar una constante <i>cualquiera</i> |

Microprogramas

- ▶ Datapath
 - ▶ Lenguaje de microinstrucciones (RTL)
-

Vamos a construir el micro con:

- ▶ Componentes
- ▶ Registros
- ▶ Líneas/buses

Ejercicio 1 - Máquina Orga1

- 1) Diseñar el camino de datos de la arquitectura de la máquina ORGA1, suponiendo que se encuentra resuelta la decodificación y el acceso a memoria de la máquina. No dibujar la unidad de control para simplificar el diagrama.

Se cuentan con los siguientes circuitos:

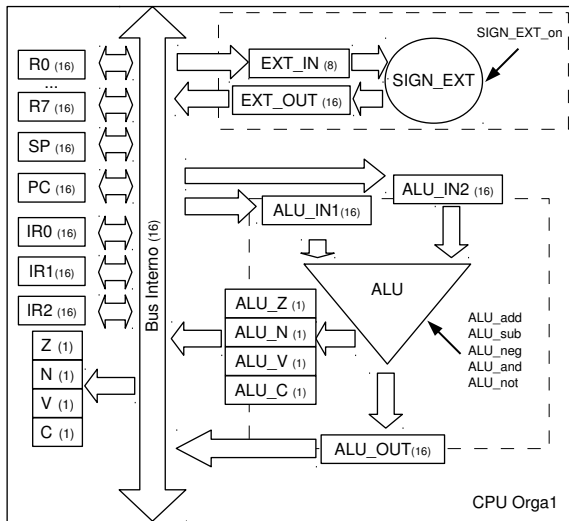
- Una ALU con 2 registros de 16 bits (ALU_IN1 y ALU_IN2) que usa de entradas y 5 registros que usa de salida: ALU_OUT de 16 bits y (ALU_Z , ALU_N , ALU_C y ALU_V) de 1 bit. Sus señales de control son:

| Señal | Efecto |
|-------------|--|
| ALU_{add} | $ALU_OUT := ALU_IN1 + ALU_IN2$ |
| ALU_{sub} | $ALU_OUT := ALU_IN1 - ALU_IN2$ |
| ALU_{neg} | $ALU_OUT := - ALU_IN1$ |
| ALU_{and} | $ALU_OUT := ALU_IN1 \text{ AND } ALU_IN2$ |
| ALU_{not} | $ALU_OUT := \text{NOT } ALU_IN1$ |

- Un extensor de signo complemento a 2 ($SIGN_EXT$) con un registro de entrada de 8 *bits* (EXT_IN) y un registro de salida de 16 *bits* (EXT_OUT). Sus señales de control son:

| Señal | Efecto |
|------------------|---|
| $SIGN_EXT_{on}$ | activa la operación de extensión de signo de 8 <i>bits</i> a 16 <i>bits</i> |

Ejercicio 1 - Máquina Orga1 - Solución



Ejercicio 1 - Máquina Orga1 - Solución

- ▶ 16 bits: R0, ..., R7, SP, PC, IR0, IR1, IR2, EXT_OUT, ALU_IN1, ALU_IN2, ALU_OUT
- ▶ 8 bits: EXT_IN
- ▶ 1 bit: Z, N, V, C, ALU_Z, ALU_N, ALU_V, ALU_C
- ▶ Bus interno: 16 líneas.
- ▶ Los flags están conectados a las 4 líneas menos significativas del bus.
- ▶ El registro EXT_IN está conectado a las 8 líneas menos significativas del bus.

Ejercicio 1 - Máquina Orga1

Se cuentan con los siguientes circuitos:

- Una ALU con 2 registros de 16 bits (ALU_IN1 y ALU_IN2) que usa de entradas y 5 registros que usa de salida: ALU_OUT de 16 bits y (ALU_Z , ALU_N , ALU_C y ALU_V) de 1 bit. Sus señales de control son:

| Señal | Efecto |
|-------------|--|
| ALU_{add} | $ALU_OUT := ALU_IN1 + ALU_IN2$ |
| ALU_{sub} | $ALU_OUT := ALU_IN1 - ALU_IN2$ |
| ALU_{neg} | $ALU_OUT := - ALU_IN1$ |
| ALU_{and} | $ALU_OUT := ALU_IN1 \text{ AND } ALU_IN2$ |
| ALU_{not} | $ALU_OUT := \text{NOT } ALU_IN1$ |

- Un extensor de signo complemento a 2 ($SIGN_EXT$) con un registro de entrada de 8 *bits* (EXT_IN) y un registro de salida de 16 *bits* (EXT_OUT).
Sus señales de control son:

| Señal | Efecto |
|------------------|---|
| $SIGN_EXT_{on}$ | activa la operación de extensión de signo de 8 <i>bits</i> a 16 <i>bits</i> |

Ejercicio 1 - Máquina Orga1

Se cuentan con los siguientes circuitos:

- ▶ Una ALU con 2 registros de 16 bits (ALU_IN1 y ALU_IN2) que usa de entradas y 5 registros que usa de salida: ALU_OUT de 16 bits y (ALU_Z , ALU_N , ALU_C y ALU_V) de 1 bit. Sus señales de control son:

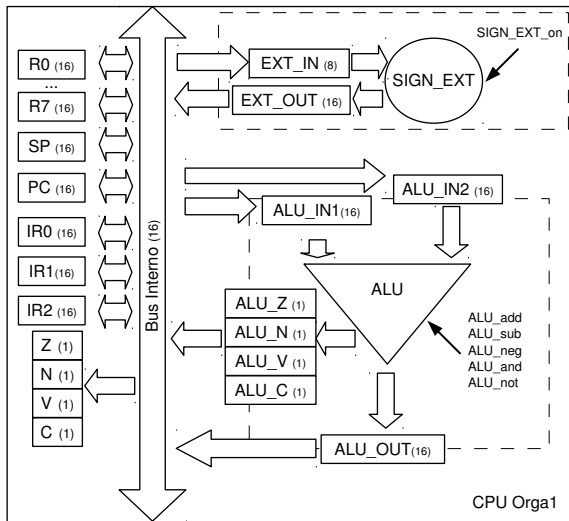
| Señal | Efecto |
|-------------|--|
| ALU_{add} | $ALU_OUT := ALU_IN1 + ALU_IN2$ |
| ALU_{sub} | $ALU_OUT := ALU_IN1 - ALU_IN2$ |
| ALU_{neg} | $ALU_OUT := - ALU_IN1$ |
| ALU_{and} | $ALU_OUT := ALU_IN1 \text{ AND } ALU_IN2$ |
| ALU_{not} | $ALU_OUT := \text{NOT } ALU_IN1$ |

- ▶ Un extensor de signo complemento a 2 ($SIGN_EXT$) con un registro de entrada de 8 *bits* (EXT_IN) y un registro de salida de 16 *bits* (EXT_OUT).
Sus señales de control son:

| Señal | Efecto |
|------------------|---|
| $SIGN_EXT_{on}$ | activa la operación de extensión de signo de 8 <i>bits</i> a 16 <i>bits</i> |

- 2) Indicar cuál es la secuencia de señales (o microoperaciones) que debe realizar la unidad de control para ejecutar las siguientes instrucciones:
- ▶ $MOV\ R5, R1$
 - ▶ $AND\ R7, R1$
 - ▶ $JE\ 0xFF$

Ejercicio 1 - Máquina Orga1 - Solución



Ejercicio 1 - Solución: Secuencias de microoperaciones

► MOV R5,R1

1. R5 := R1

► AND R7, R1

1. ALU_IN1 := R7
2. ALU_IN2 := R1
3. ALU_{and}
4. R7 := ALU_OUT
5. Z := ALU_Z
6. N := ALU_N
7. C := ALU_C
8. V := ALU_V

► JE 0xFF

1. IF Z = 1
2. EXT_IN := IR0[7:0]
3. SIGN_EXT_on
4. ALU_IN_1 := PC
5. ALU_IN_2 := EXT_OUT
6. ALU_{add}
7. PC := ALU_OUT

Ejercicio 2 - Máquina Orga1

Se cuenta con una memoria con palabra y direccionamiento de 16 *bits*. El CPU se comunicará con ella utilizando un controlador de bus. Este es un dispositivo que posee 2 registros de entrada de 16 *bits* (ADDR, WRT_DATA) y 1 de salida de 16 *bits* (RD_DATA). Sus señales de control son:

- ▶ MEM_WRITE: Activa la microoperación de escritura del contenido del registro WRT_DATA en la dirección de memoria indicada por el ADDR
 - ▶ MEM_READ: Activa la microoperación de lectura del contenido de la dirección de memoria indicada por el ADDR, colocando el valor en el registro RD_DATA.
1. Extender el camino de datos de la arquitectura de la máquina ORGA1. No dibujar la unidad de control para simplificar el diagrama.

Ejercicio 2 - Máquina Orga1

Se cuenta con una memoria con palabra y direccionamiento de 16 *bits*. El CPU se comunicará con ella utilizando un controlador de bus. Este es un dispositivo que posee 2 registros de entrada de 16 *bits* (ADDR, WRT_DATA) y 1 de salida de 16 *bits* (RD_DATA). Sus señales de control son:

- ▶ MEM_WRITE: Activa la microoperación de escritura del contenido del registro WRT_DATA en la dirección de memoria indicada por el ADDR
 - ▶ MEM_READ: Activa la microoperación de lectura del contenido de la dirección de memoria indicada por el ADDR, colocando el valor en el registro RD_DATA.
1. Extender el camino de datos de la arquitectura de la máquina ORGA1. No dibujar la unidad de control para simplificar el diagrama.
 2. ¿Qué componentes del camino de datos se encuentran dentro del CPU y fuera de él?

Ejercicio 2 - Máquina Orga1

Se cuenta con una memoria con palabra y direccionamiento de 16 *bits*. El CPU se comunicará con ella utilizando un controlador de bus. Este es un dispositivo que posee 2 registros de entrada de 16 *bits* (ADDR, WRT_DATA) y 1 de salida de 16 *bits* (RD_DATA). Sus señales de control son:

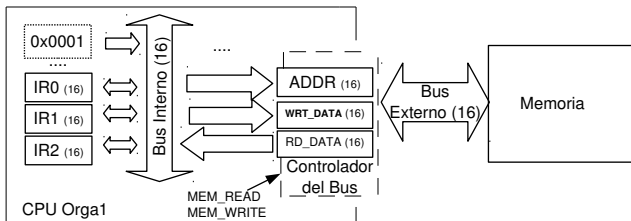
- ▶ MEM_WRITE: Activa la microoperación de escritura del contenido del registro WRT_DATA en la dirección de memoria indicada por el ADDR
 - ▶ MEM_READ: Activa la microoperación de lectura del contenido de la dirección de memoria indicada por el ADDR, colocando el valor en el registro RD_DATA.
1. Extender el camino de datos de la arquitectura de la máquina ORGA1. No dibujar la unidad de control para simplificar el diagrama.
 2. ¿Qué componentes del camino de datos se encuentran dentro del CPU y fuera de él?
 3. Indicar cuál es la secuencia de señales (o microoperaciones) que debe realizar la unidad de control para ejecutar las siguientes instrucciones:
 - ▶ MOV R2, [R5]
 - ▶ MOV [0xFF00], [0xFF01]

Ejercicio 2 - Máquina Orga1

Se cuenta con una memoria con palabra y direccionamiento de 16 *bits*. El CPU se comunicará con ella utilizando un controlador de bus. Este es un dispositivo que posee 2 registros de entrada de 16 *bits* (ADDR, WRT_DATA) y 1 de salida de 16 *bits* (RD_DATA). Sus señales de control son:

- ▶ MEM_WRITE: Activa la microoperación de escritura del contenido del registro WRT_DATA en la dirección de memoria indicada por el ADDR
 - ▶ MEM_READ: Activa la microoperación de lectura del contenido de la dirección de memoria indicada por el ADDR, colocando el valor en el registro RD_DATA.
1. Extender el camino de datos de la arquitectura de la máquina ORGA1. No dibujar la unidad de control para simplificar el diagrama.
 2. ¿Qué componentes del camino de datos se encuentran dentro del CPU y fuera de él?
 3. Indicar cuál es la secuencia de señales (o microoperaciones) que debe realizar la unidad de control para ejecutar las siguientes instrucciones:
 - ▶ MOV R2, [R5]
 - ▶ MOV [0xFF00], [0xFF01]
 4. Como Tarea: Describa la secuencia de microoperaciones que realiza la unidad de control para realizar un *fetch* de una instrucción de la máquina Orga1.

Ejercicio 2 - Máquina Orga1: Solución



Ejercicio 2 - Solución: secuencias de microoperaciones

- ▶ MOV R2, [R5]
 1. ADDR := R5
 2. MEM_READ
 3. R2 := RD_DATA

- ▶ MOV [0xFF00], [0xFF01]
 1. ADDR := IR2
 2. MEM_READ
 3. WRT_DATA := RD_DATA
 4. ADDR := IR1
 5. MEM_WRITE

Ejercicio 3

La computadora STACK1 es una máquina de pila con direccionamiento a byte, tamaño de palabra de 16 bits y direcciones de memoria de 12 bits. Trabaja con aritmética complemento a 2 de 16 bits. Posee el siguiente set de instrucciones:

| Instrucción | CodOp | Significado |
|-------------|-------|---|
| PUSH [M] | 0000 | push [M] |
| POP [M] | 0001 | [M] := pop |
| ADD | 0010 | push(pop+pop) |
| SUB | 0011 | push(pop-pop) |
| JUMP | 0100 | PC := pop (sólo los 12 bits menos significativos) |
| SKIP_N | 0101 | ignora la próxima instrucción si top es < 0 |
| SKIP_Z | 0110 | ignora la próxima instrucción si top es 0 |
| SKIP_GE | 0111 | ignora la próxima instrucción si top es >= 0 |

El formato de instrucción de STACK1 es el que sigue:

| CodOp | Dirección |
|--------|-----------|
| 4 bits | 12 bits |

- 1) Definir el camino de datos y la organización del CPU de STACK1 para soportar la implementación de, al menos, estas instrucciones.
- 2) Describa la secuencia de microoperaciones que realiza la unidad de control para realizar un *fetch* de una instrucción.
- 3) Implementar las siguientes instrucciones:

I. JUMP

II. SKIP_Z

III. PUSH [0xFAC]

IV. ADD

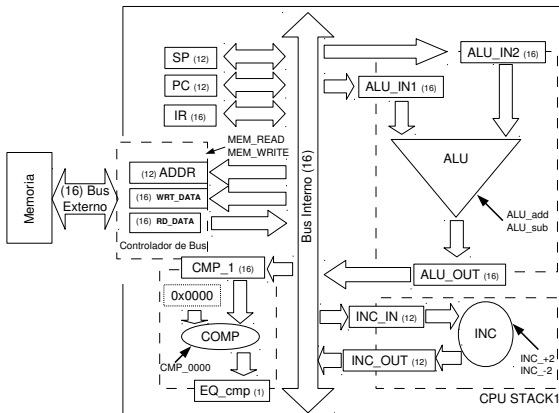
Ejercicio 3: Solución

Se utiliza un circuito incrementador con 2 señales: INC_{+2} que suma 2 a la entrada, y INC_{-2} que resta 2 a la entrada.

ADDR, INC_IN, INC_OUT, SP y PC son registros de 12 bits.

IR, CMP_1, ALU_IN1, ALU_IN2, ALU_OUT, WRT_DATA y RD_DATA y los buses INTERNO y EXTERNO son de 16 bits.

EQ_CMP es de 1 bit. Las 12 líneas de los registros correspondientes están conectadas a las líneas menos significativas del BUS.



Ejercicio 3 - Solución: *fetch*

1. ADDR := PC
2. MEM_READ
3. IR := RD_DATA // cargo el IR
4. INC_IN := PC
5. INC₊2
6. PC := INC_OUT // incremento PC

Ejercicio 3 - Solución: secuencia de microoperaciones

► JUMP

1. INC_IN := SP
2. INC₊₂
3. SP := INC_OUT
4. ADDR := SP
5. MEM_READ
6. PC := RD_DATA[11:0]

► SKIP_Z

1. INC_IN := SP
2. INC₊₂
3. ADDR := INC_OUT
4. MEM_READ
5. CMP_1 := RD_DATA
6. CMP_0000
7. if EQ_cmp = 1
8. INC_IN := PC
9. INC₊₂
10. PC := INC_OUT
11. endif

Ejercicio 3 - Solución: secuencia de microoperaciones

► PUSH [X]

1. ADDR := IR[11:0]
2. MEM_READ
3. WRT_DATA := RD_DATA
4. ADDR := SP
5. MEM_WRITE
6. INC_IN := SP
7. INC₋₋₂
8. SP := INC_OUT

► ADD

1. INC_IN := SP
2. INC₊₂
3. SP := INC_OUT
4. ADDR := SP
5. MEM_READ
6. ALU_IN1 := RD_DATA // primer operando
7. INC_IN := SP
8. INC₊₂
9. SP := INC_OUT
10. ADDR := SP
11. MEM_READ
12. ALU_IN2 := RD_DATA // segundo operando
13. ALU_{-add}
14. WRT_DATA := ALU_OUT
15. ADDR := SP
16. MEM_WRITE // push resultado
17. INC_IN := SP
18. INC₋₋₂
19. SP := INC_OUT

Cómo sigue Orga1...

Con lo visto hoy pueden hacer toda la práctica 4.

- ▶ Jueves 14: Taller de Microprogramación - parte 1
- ▶ Martes 21: Taller de Microprogramación - parte 2 + Consultas

FIN