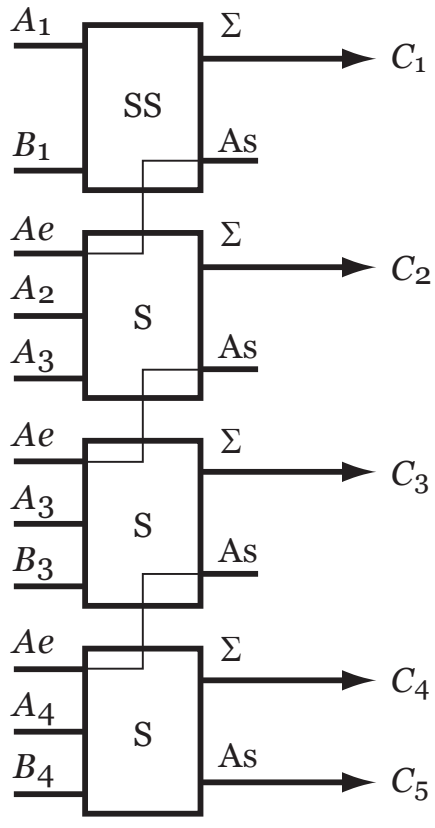
Clase práctica - Microarquitectura de la CPU

Juan Pablo Galeotti

Segundo Cuatrimestre de 2010 - Organización del Computador I

Introducción

`ADD R1, R0` ¿$\longrightarrow$?



Enunciado

Suponiendo que se encuentra resuelto el acceso a memoria principal y la decodificación de instrucciones, diseñar la *microarquitectura* de la máquina $\mathrm{ORGA1}$. No dibujar la unidad de control para simplificar el diagrama.
¿Qué podemos usar? Para esta tarea, suponer que puede utilizar los siguientes circuitos ya definidos:

(1) Registros

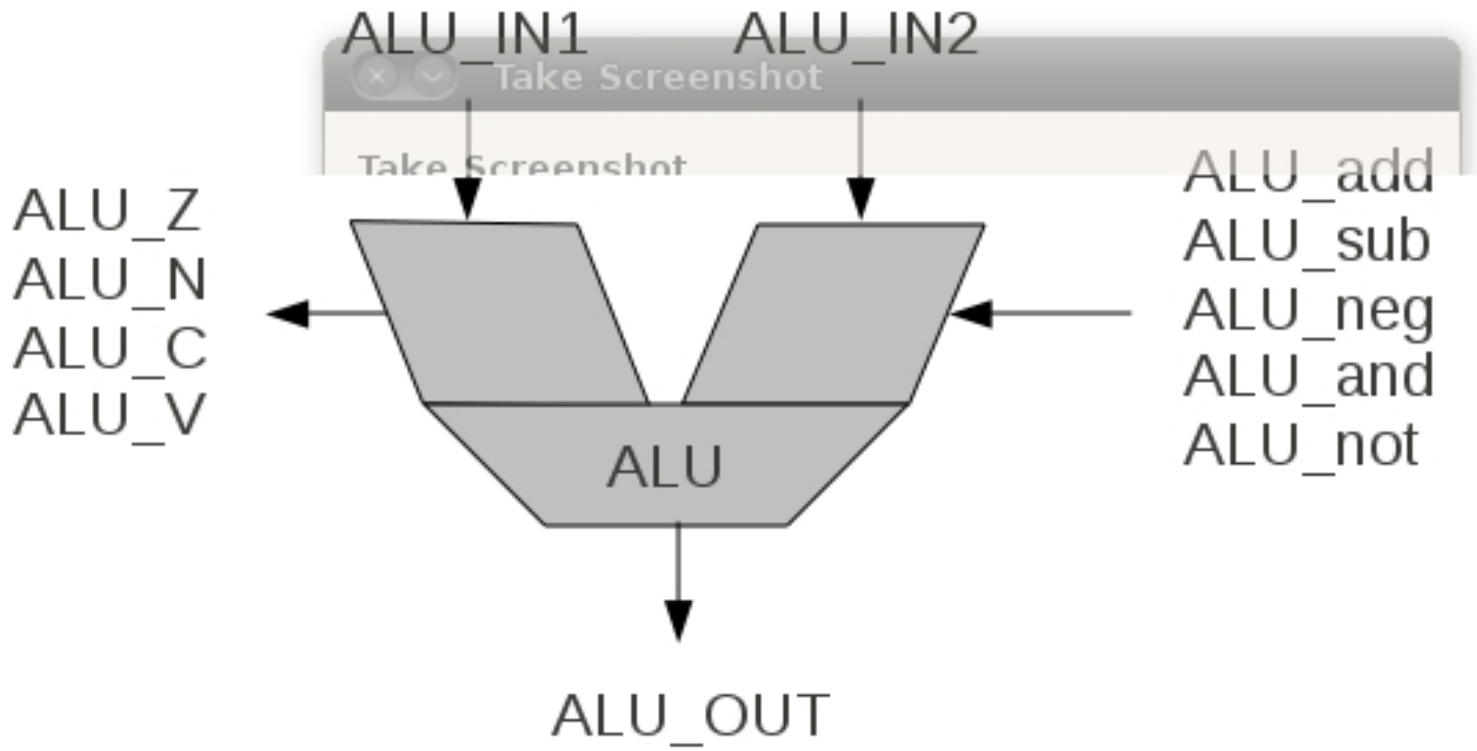
Nos permiten *almacenar* información dentro del CPU.

Puede leer/escribir el contenido del registro.

(2) ALU

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

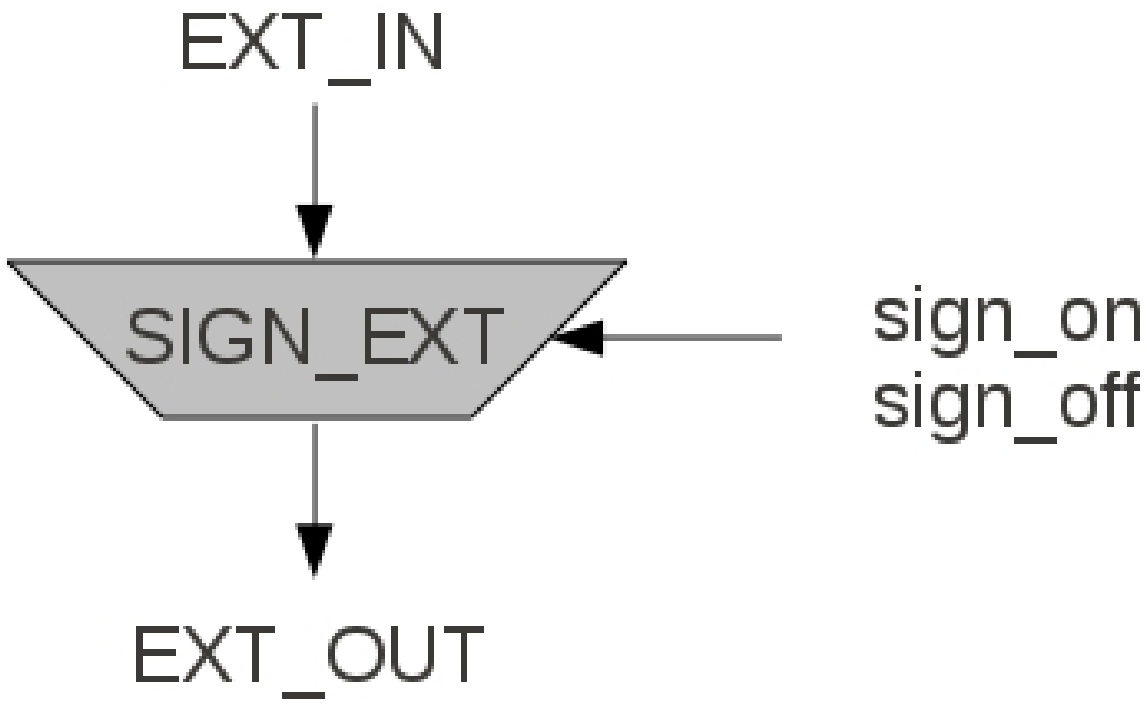
Señal	Efecto
ALU_{add}	ALU_OUT := ALU_IN1 + ALU_IN2
ALU_{sub}	ALU_OUT := ALU_IN1 - ALU_IN2
ALU_{neg}	ALU_OUT := - ALU_IN1
ALU_{and}	ALU_OUT := ALU_IN1 AND ALU_IN2
ALU_{not}	ALU_OUT := NOT ALU_IN1



(3) Extensor de signo

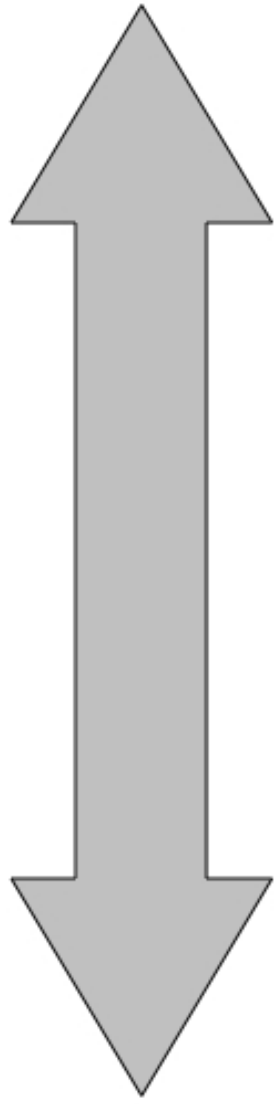
Un extensor de signo complemento a 2 (SIGN_EXT) con una entrada de 16 *bits* (EXT_IN) y una salida de 16 *bits* (EXT_OUT). Sus señales de control son:

Señal	Efecto
$sign_{on}$	activa la operación de extensión de signo de 8 *bits* a 16 *bits*
$sign_{off}$	copia el contenido de EXT_IN en ALU_IN2



(4) Buses

Permiten interconectar componentes y transferir datos entre ellos.

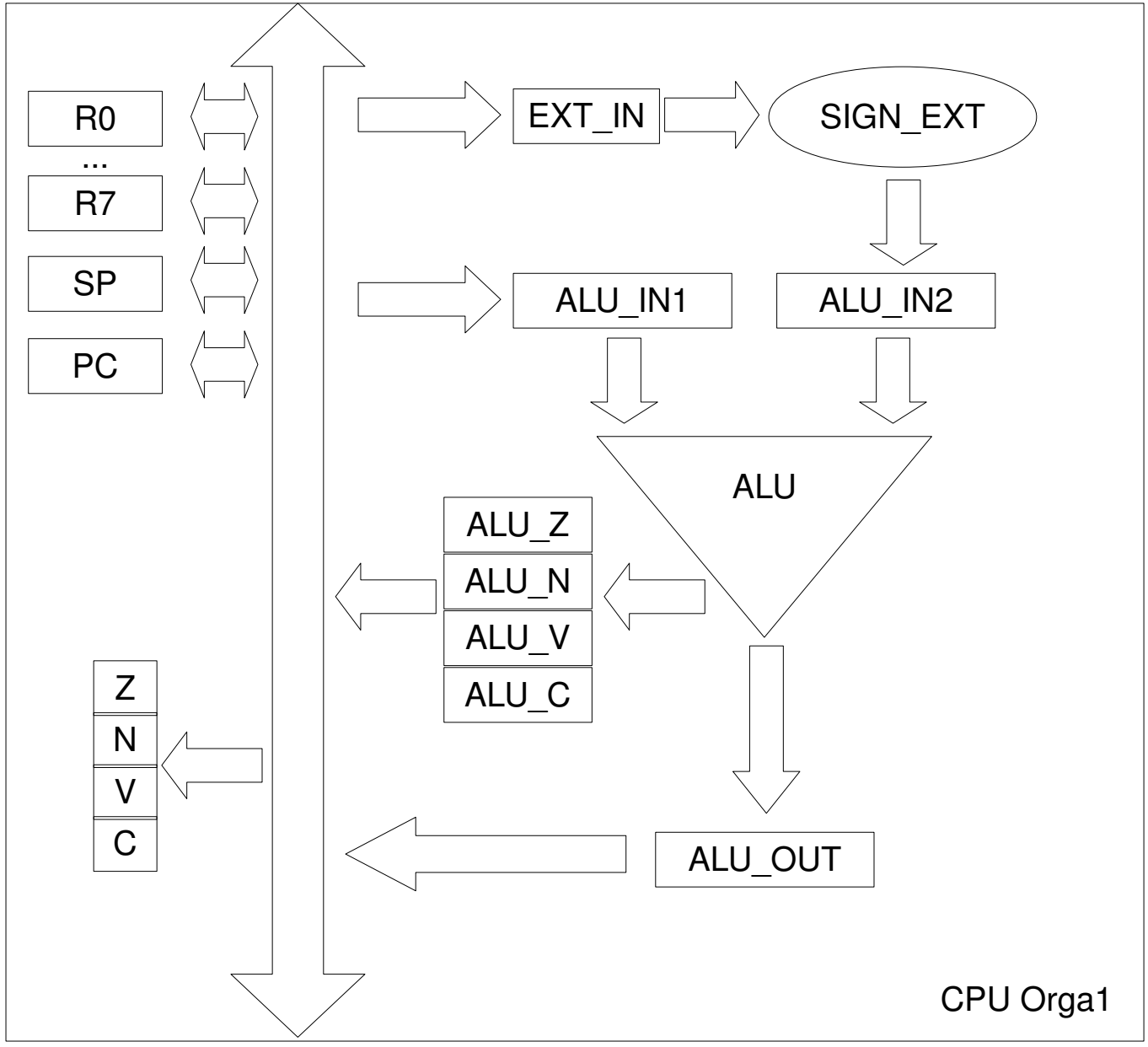


Enunciado (de vuelta)

Suponiendo que se encuentra resuelto el acceso a memoria principal y la decodificación de instrucciones, diseñar la *microarquitectura* de la máquina $\mathrm{ORGA1}$. No dibujar la unidad de control para simplificar el diagrama.
¿Qué podemos usar?

- Registros
- ALUs
- SIGN_Ext
- Buses

Una solución



Enunciado (cont.)

Indicar cuál es la secuencia de señales (o microoperaciones) que debe realizar la unidad de control para ejecutar las siguientes instrucciones:

a) MOV R5, R1

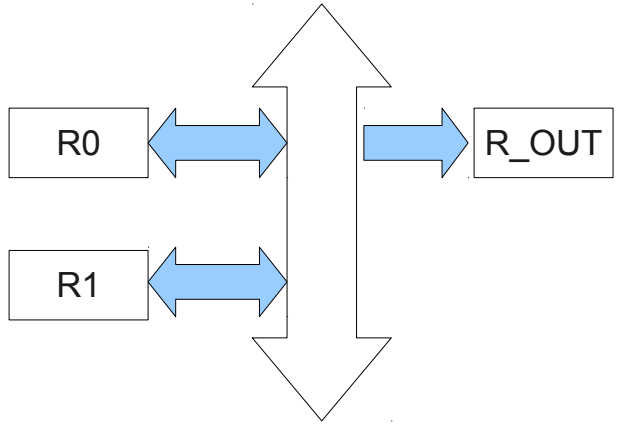
b) AND R7, R1

c) NEG R4

El lenguaje RTL (1)

Nos permite describir secuencias de microoperaciones.
Por ejemplo:

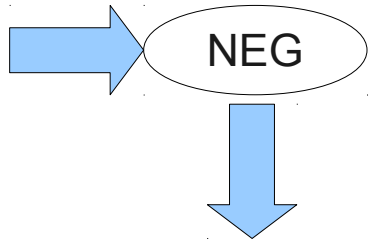
- Podemos mover un dato de un registro a otro si hay un camino directo entre ellos



VALE	NO VALE
R1:= R0	R0:=R_OUT
R_OUT:=R0	R1:=R_OUT
⋮	

El lenguaje RTL (2)

- Podemos activar señales



`NEG_on: niega los bits de la entrada y los pone a la salida`

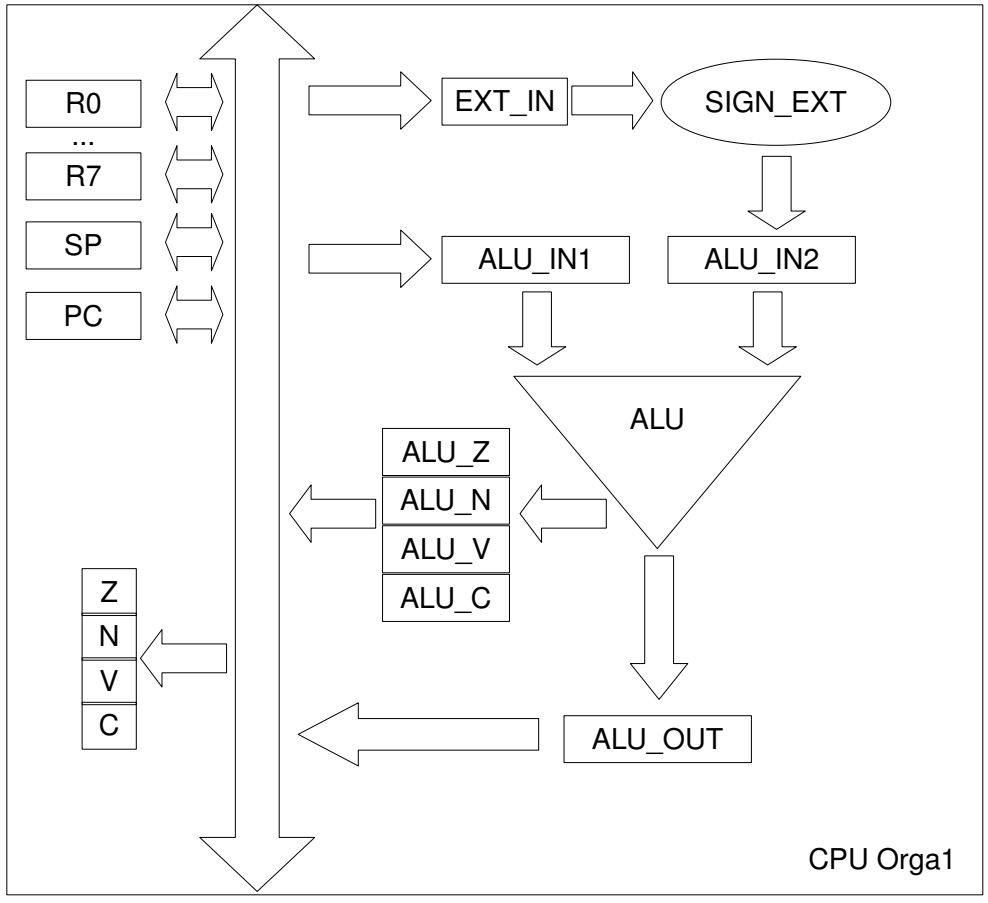
`NEG_off: pasa los bits sin modificarlos`

El lenguaje RTL (3)

- Podemos analizar registro o parte de ellos y actuar en consecuencia

```
if R1[3:0]=0x0
    R_OUT:=R3
else
    R_OUT:=R2
endif
```

Ejercicio 1 - Datapath



a) MOV R5, R1

b) AND R7, R1

c) NEG R4

Ejercicio 1 - Solución: Secuencias de microoperaciones

- MOV R5,R1
  1. R5 := R1
- AND R7, R1
  1. ALU_IN1 := R7
  2. EXT_IN := R1
  3. SIGN_EXT_{off}
  4. ALU_{and}
  5. R7 := ALU_OUT
  6. Z := ALU_Z
  7. N := ALU_N
  8. C := 0
  9. V := 0

- NEG R4
  1. ALU_IN1 := R4
  2. ALU_{neg}
  3. R0 := ALU_OUT
  4. Z := ALU_Z
  5. N := ALU_N
  6. C := ALU_C
  7. V := ALU_V

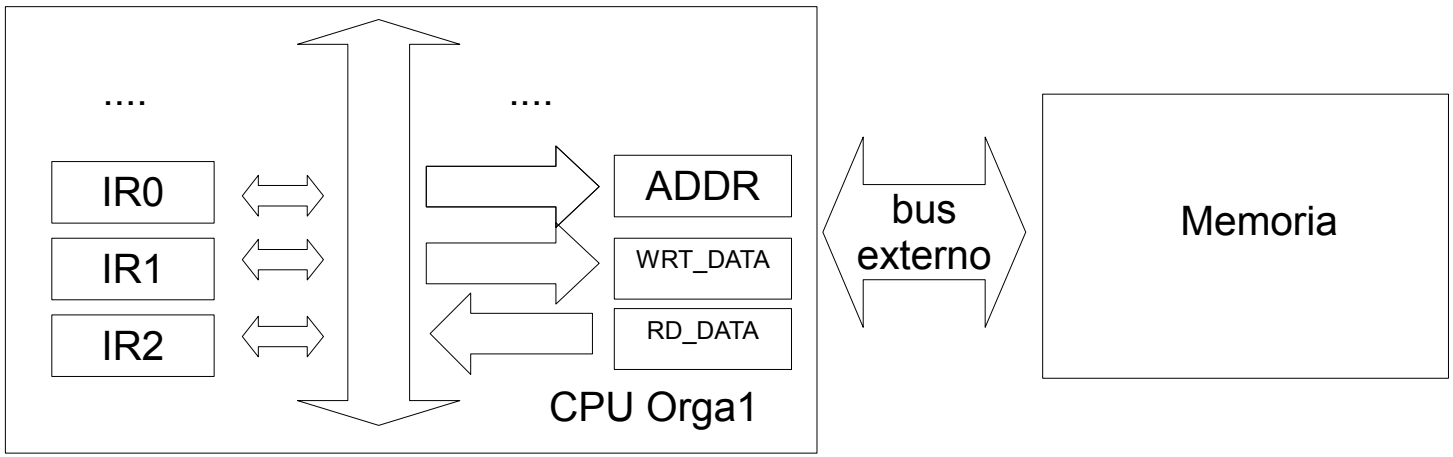
Ejercicio 2

Se cuenta con una memoria con palabra y direccionamiento de 16 *bits*. Posee 2 entradas de 16 *bits* (ADDR, WRT_DATA) y 1 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 R2, [R5]
   - MOV R2, [0xFF00]
   - MOV [0xFF00], [0xFF01]

Ejercicio 2: Solución



Ejercicio 2 - Solución: secuencias de microoperaciones

- MOV R2, R5
  1. R2 := R5
- MOV R2, [R5]
  1. ADDR := R5
  2. MEM_READ
  3. R2 := RD_DATA
- MOV R2, [0xFF00]
  1. ADDR := IR1
  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 [X]
`POP [M]`	0001	[X] := 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:

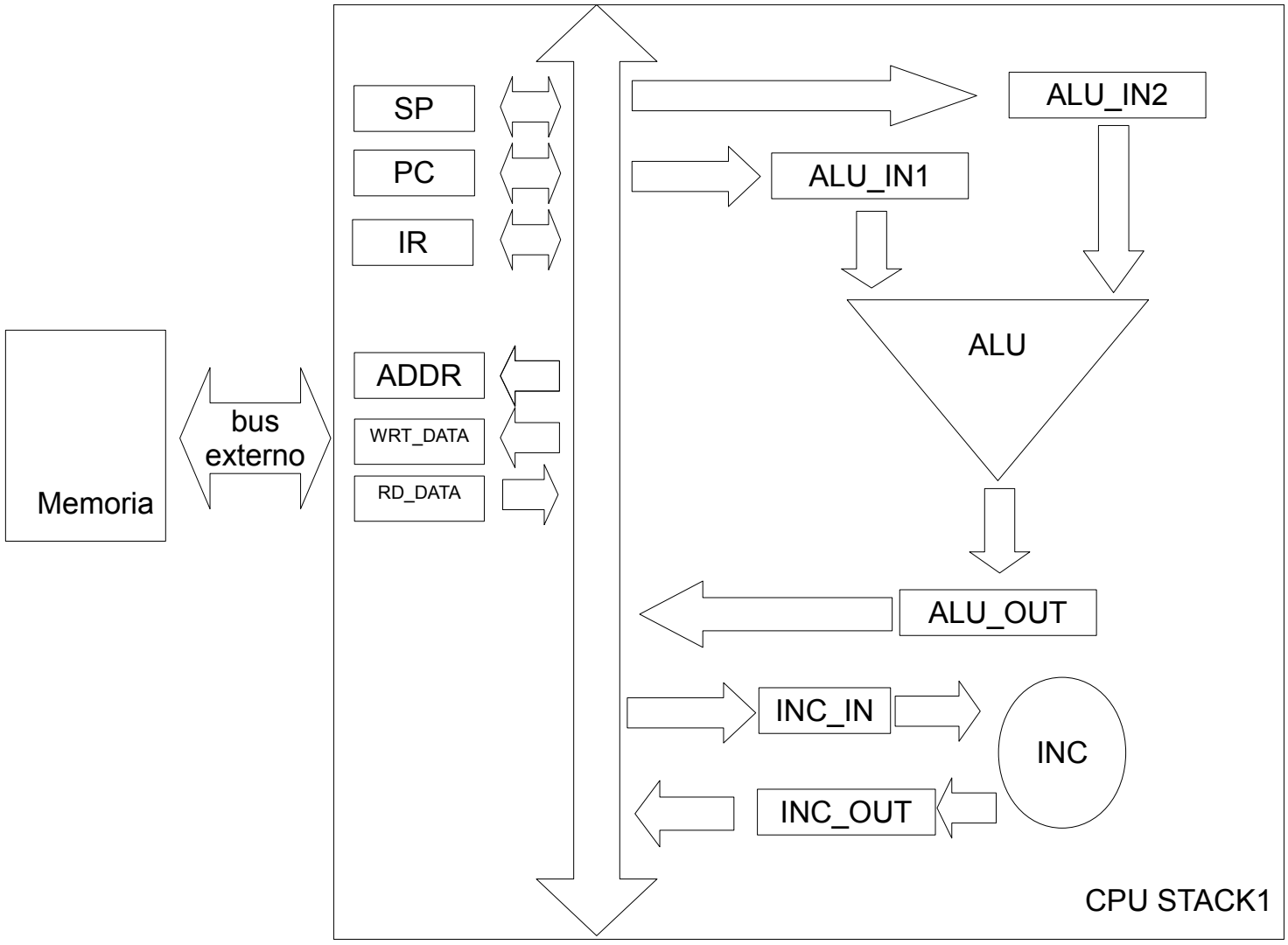
CodOp	Dirección
4 bits	12 bits

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 [X]`
IV) `ADD`

Ejercicio 3: Solución

Se utiliza un circuito incrementador con 2 señales: $\texttt{INC\_+2}$ que suma 2 a la entrada, y $\texttt{INC\_-2}$ que resta 2 a la entrada. `ADDR`, `INC_IN`, `INC_OUT`, `SP` y `PC` son registros de 12 bits.



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. if RD_DATA = 0x0000
  6. INC_IN := PC
  7. INC_{-+2}
  8. PC := INC_OUT
  9. 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