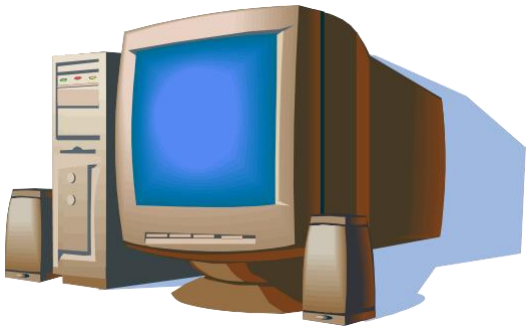
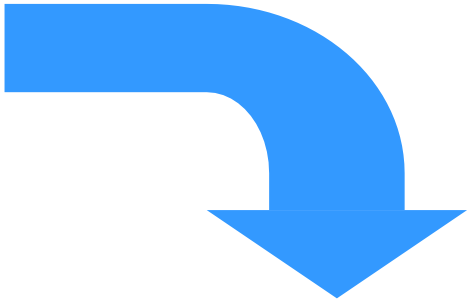
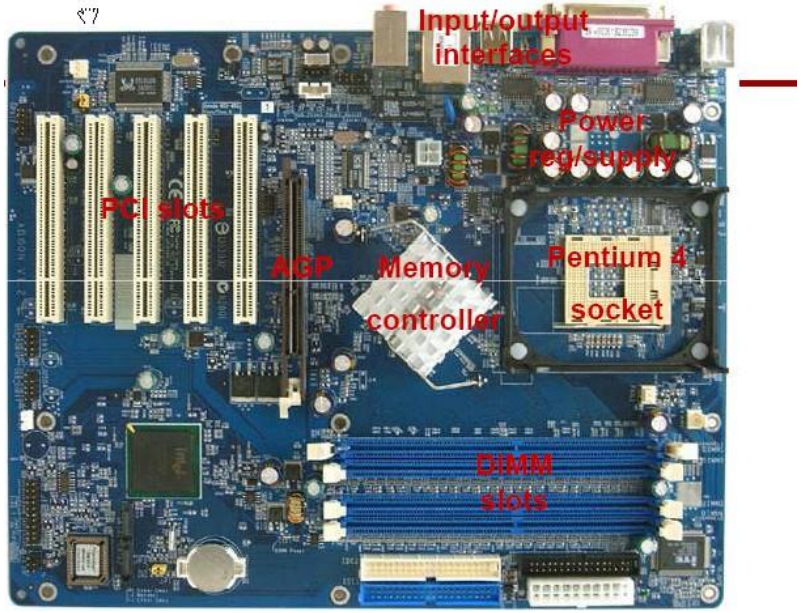
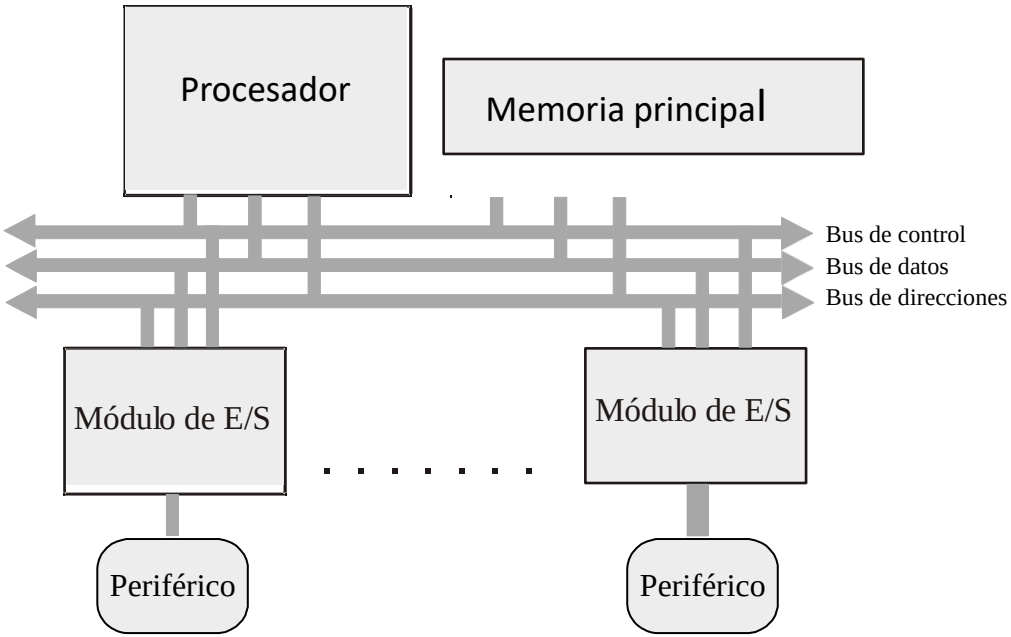
El Procesador I:
Representación y tipos de instrucciones

El Procesador
El Modelo Von-Newman

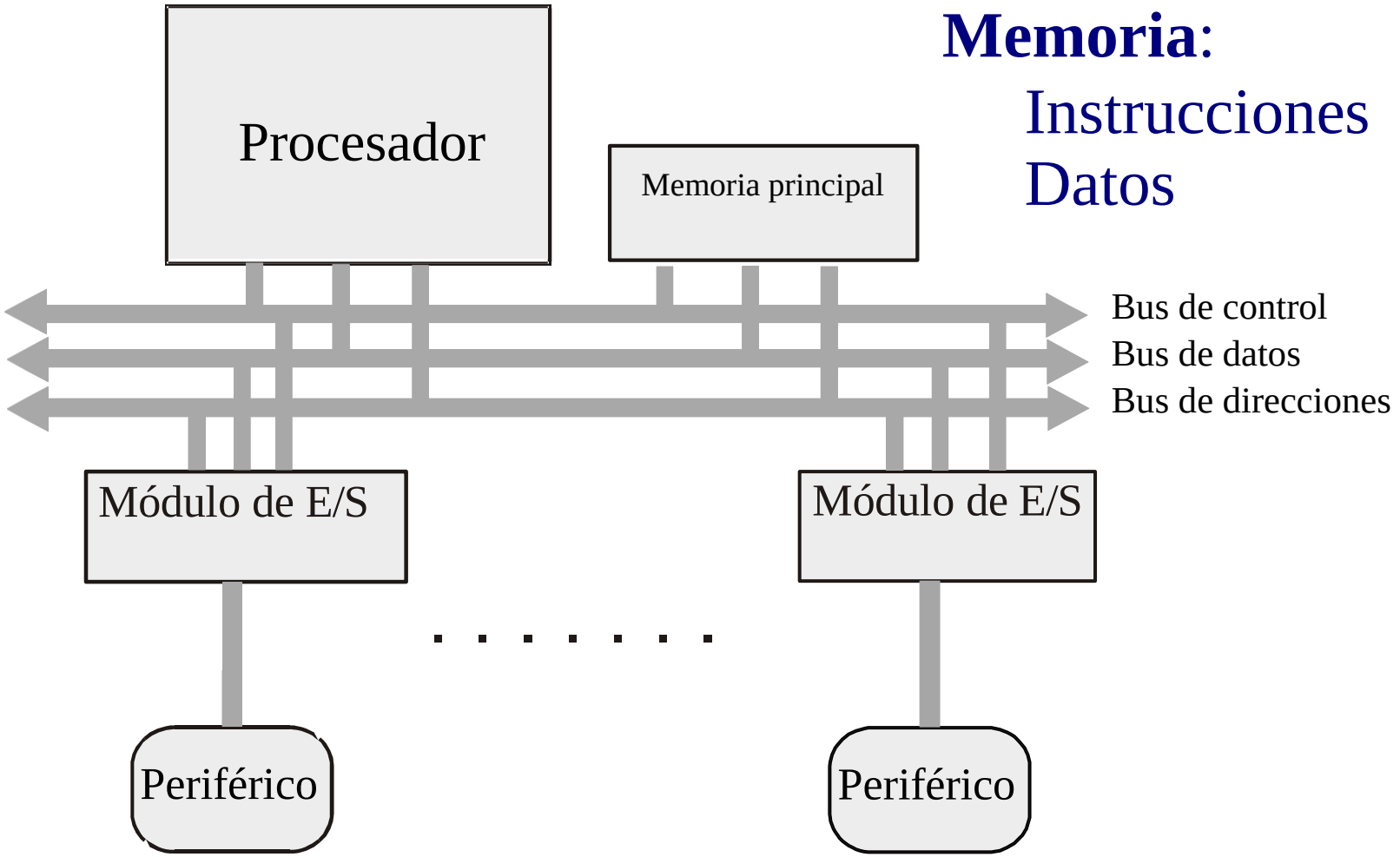






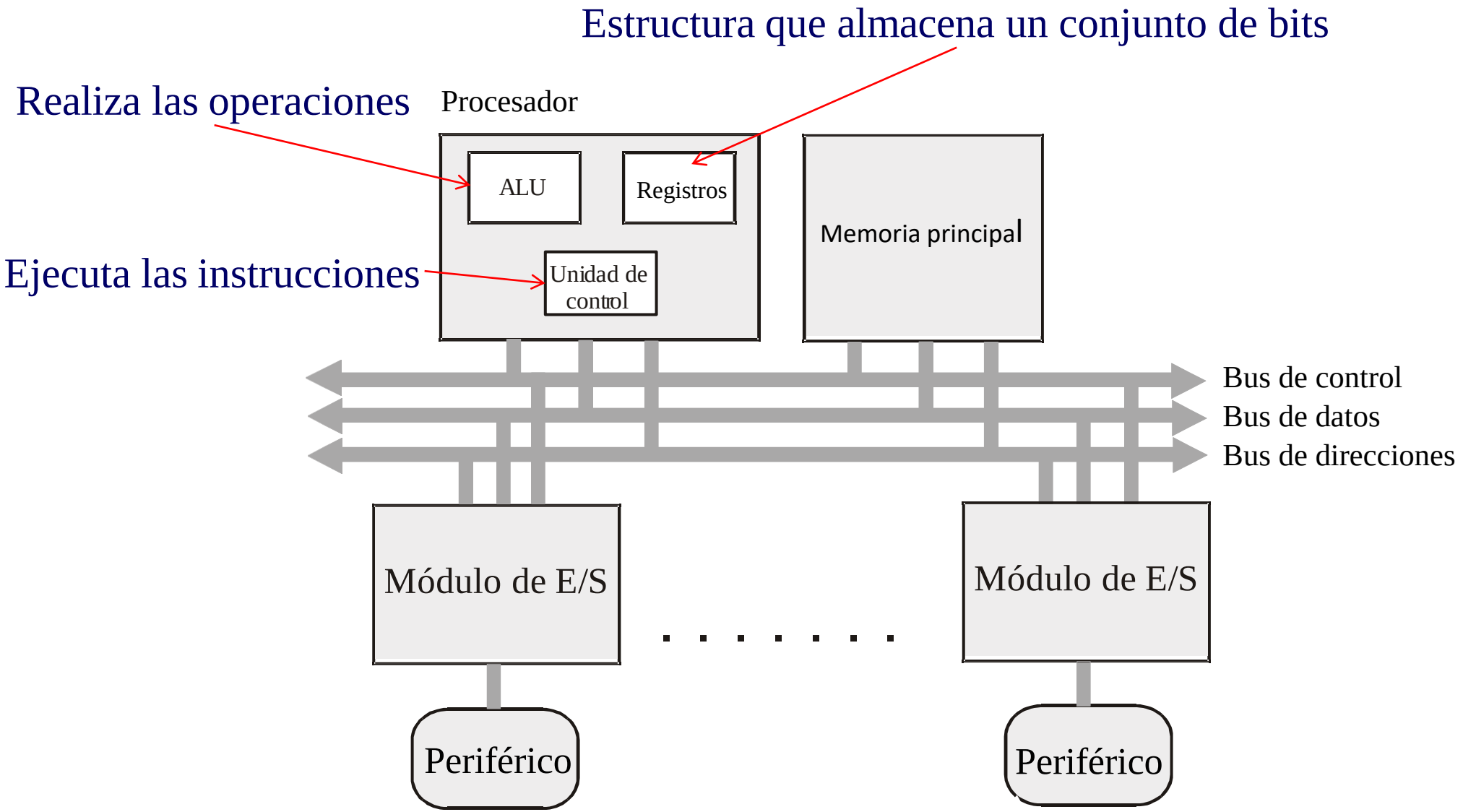


El Procesador
El Modelo Von-Newman



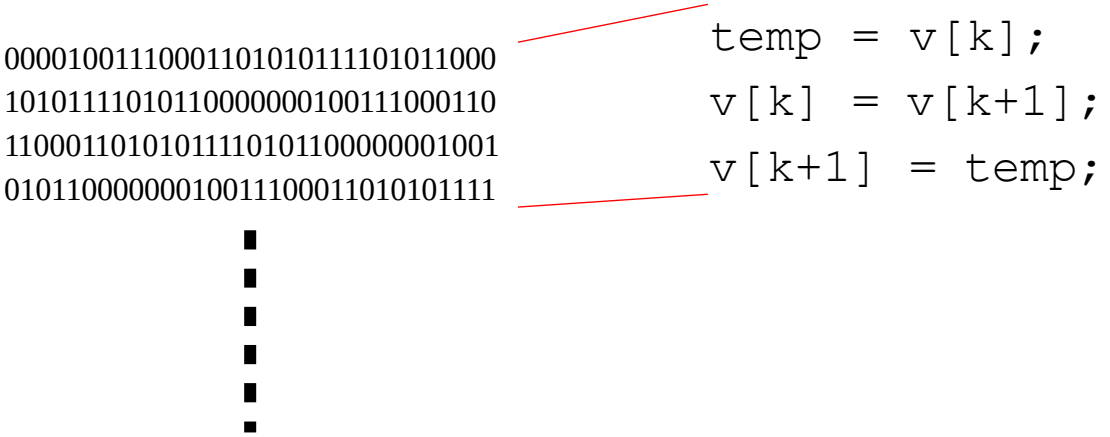
Memoria:
Instrucciones
Datos

El Procesador
Componentes principales del procesador

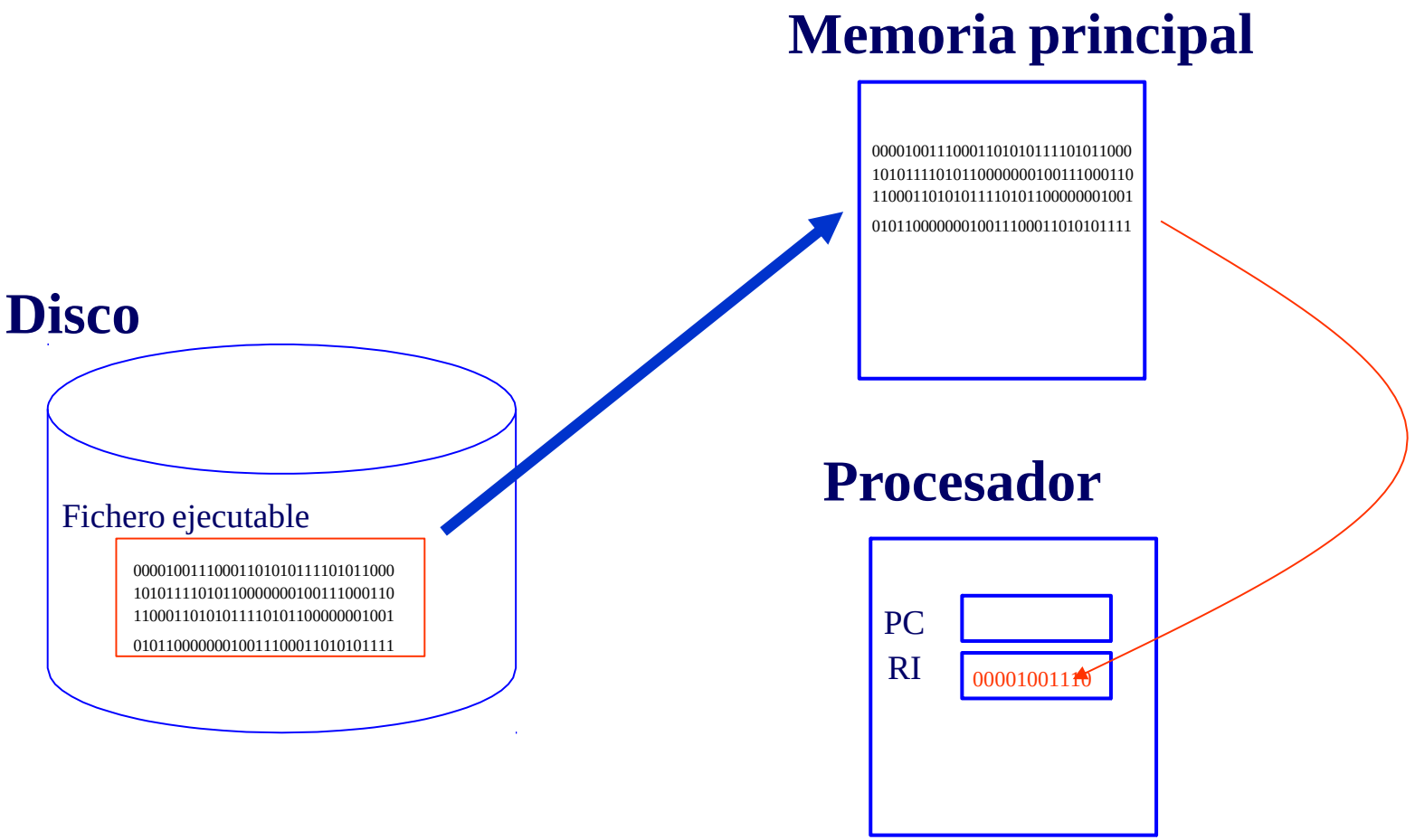


El Procesador
Carga y ejecución de un programa

Programa: Secuencia consecutiva de instrucciones máquina



El Procesador
Carga y ejecución de un programa



El Procesador
Tipos de instrucciones

- Tres arquitecturas de empotrados muy extendidas:
  - **MIPS**: De referencia en las clases de Teoría (MIPS32)
  - **ARM**: La que usamos en Prácticas (ARM Cortex M0)
  - **RISC-V**: es una arquitectura reciente que, a diferencia de todas las anteriores, es abierta.

    Es por esta razón que la UAB ha apostado en introducir esta arquitectura, tanto en asignaturas de software como de hardware.

El Procesador
Un ejemplo de juego instrucciones

Características:

- Tamaño de una posición de memoria: **16 bits**
- Tamaño de la instrucción: **16 bits**
- Código de operación: **3 bits**
- ¿Cuántas instrucciones diferentes puede tener este computador? **$2^3 = 8$**
- Número de registros de propósito general: **4**
- ¿Cuántos bits se necesitan para representar 4 registros? **2**
- Identificadores simbólicos:
  - Tipo ARM: **r0, r1, r2, r3**
  - Tipo MIPS32 [1]: **t0, ..., t3**
  - Tipo RISC-V [2] : **t0,…,t3**

[1]Arquitectura MIPS32 para empotrados: http://en.wikichip.org/wiki/mips/mips32_instruction_set

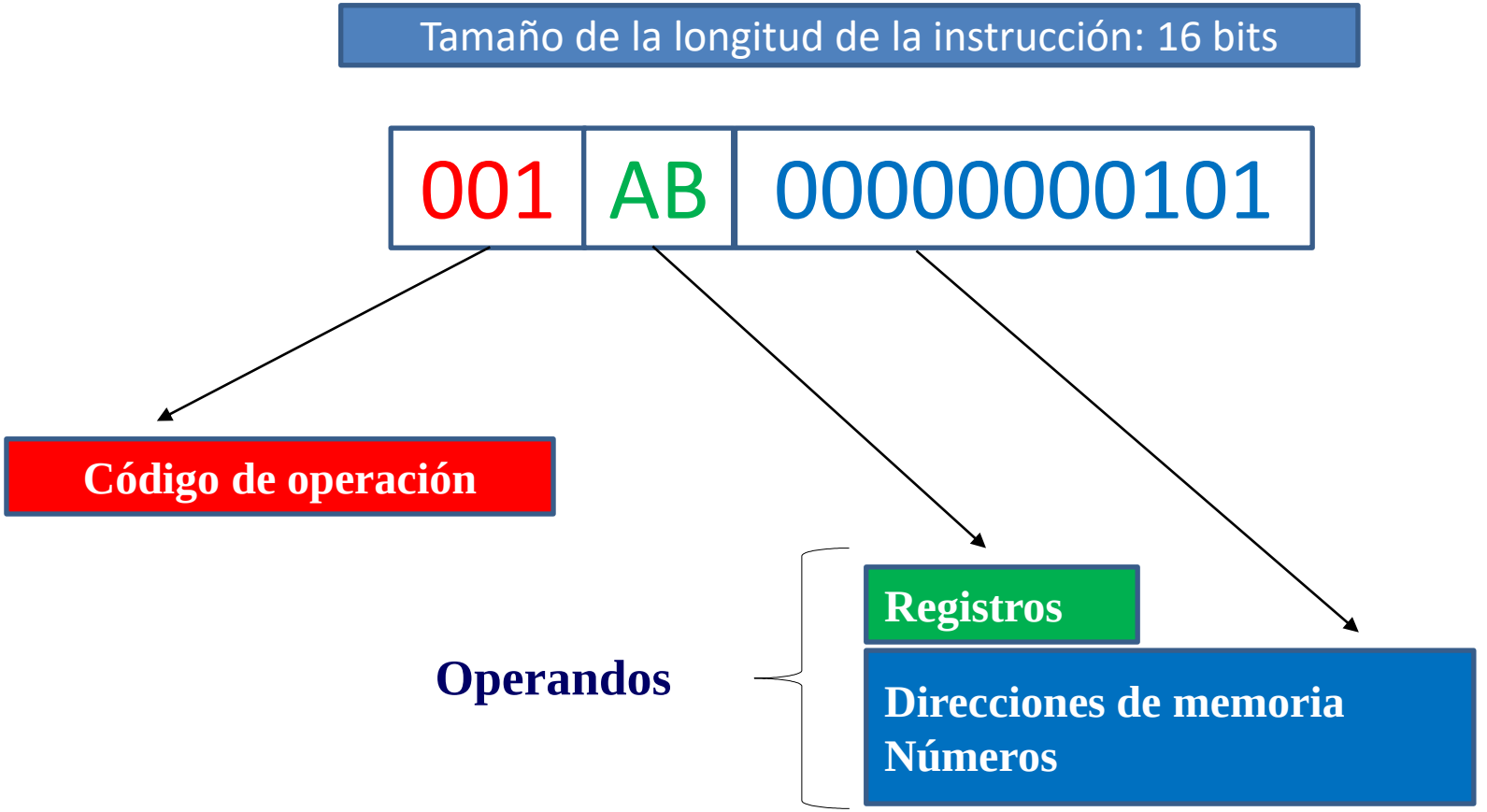
[2] Guía práctica de RISC-V: http://riscvbook.com/spanish/guia-practica-de-risc-v-1.0.5.pdf. Página 36

El Procesador
Un ejemplo de juego instrucciones

INSTRUCCIÓN	DESCRIPCIÓN
000EFABCDXXXXXXX	Suma el registro AB con el CD y deja el resultado en EF
001AB00000000101	Almacena en el registro AB el valor 00000000101
010AB00000001001	Almacena en el registro AB el valor almacenado en la posición de memoria 00000001001
011AB00000001001	Almacena en la posición de memoria 00000001001 el contenido del registro AB
100000000001001	Se salta a ejecutar la instrucción almacenada en la posición de memoria 0000000001001
101ABCD000001001	Si el contenido del registro AB es igual al del registro CD se salta a ejecutar la instrucción almacenada en 000001001

Siendo A,B, C, D, E, F = 0 o 1

El Procesador
Un ejemplo de juego instrucciones



No usado

El Procesador
Un ejemplo de juego instrucciones

INSTRUCCIÓN	DESCRIPCIÓN
0000100010XXXXXXX	Suma el registro 00con el 10y deja el resultado en 01
0010100000000101	Almacena en el registro 01el valor 00000000101
0100100000001001	Almacena en el registro 01el valor almacenado en la posición de memoria 00000001001
0110100000001001	Almacena en la posición de memoria 00000001001 el contenido del registro 01
1000000000001001	Se salta a ejecutar la instrucción almacenada en la posición de memoria 0000000001001
1010100000001001	Si el contenido del registro 01es igual al del registro 00 se salta a ejecutar la instrucción almacenada en 000001001

El Procesador
Un ejemplo de juego instrucciones

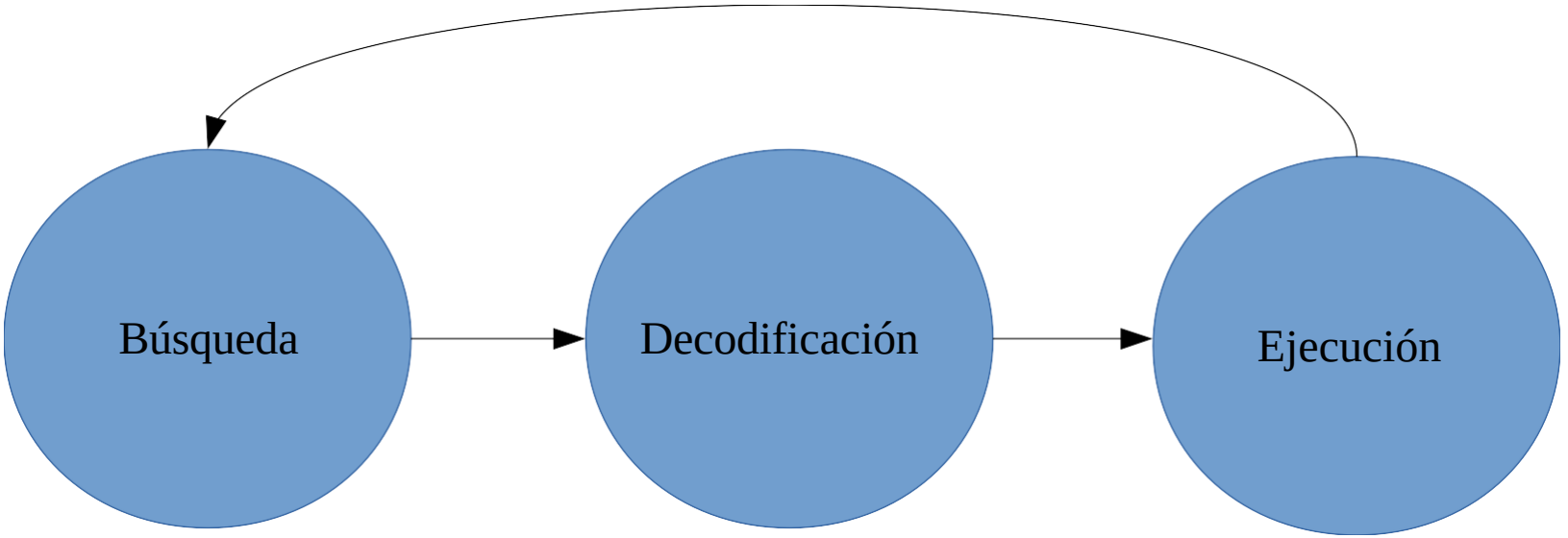
- ¿Instrucción que almacena un 5 en el registro 00?
- ¿Instrucción que almacena un 7 en el registro 01?
- ¿Instrucción que suma el contenido del registro 00 y el registro 01 y deja el resultado en el registro 10?
- ¿Instrucción que almacena el resultado anterior en la posición de memoria 1027 (en decimal)?

El Procesador
Un ejemplo de juego instrucciones

- ¿Instrucción que almacena un 5 en el registro 00?
  - **0010000000000101**
- ¿Instrucción que almacena un 7 en el registro 01?
  - **0010100000000111**
- ¿Instrucción que suma el contenido del registro 00 y el registro 01 y deja el resultado en el registro 10?
  - **0001000001XXXXXX**
- ¿Instrucción que almacena el resultado anterior en la posición de memoria 1027 (en decimal)?
  - **0111010000000011**

El Procesador
Fases de ejecución de instrucciones

Secuencia básica de ejecución de instrucciones



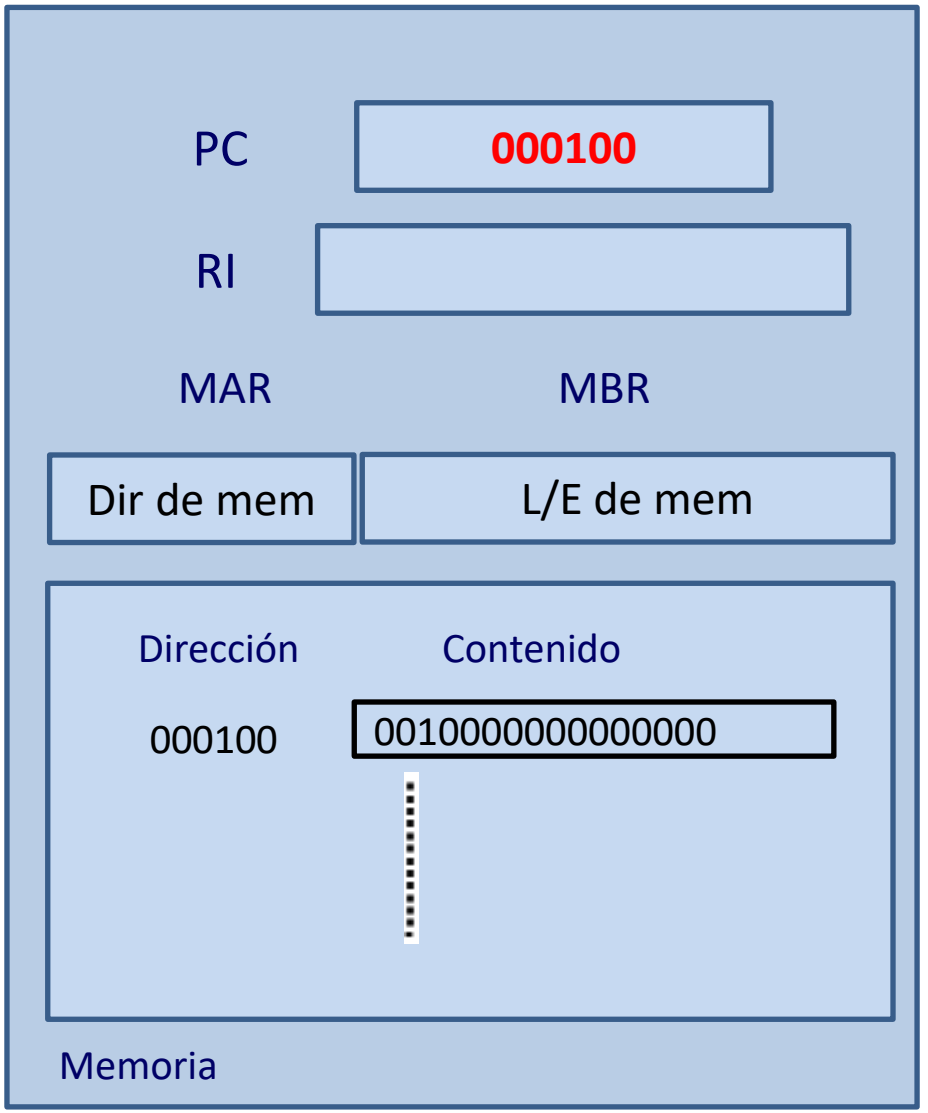
El Procesador
Fases de ejecución de instrucción

Lectura de la instrucción (ciclo de *fetch*)

- MAR ← PC
- Lectura
- MBR ← Memoria
- PC ← PC + 1
- RI ← MBR

Decodificación de la instrucción

Ejecución de la instrucción
Volver a *fetch*



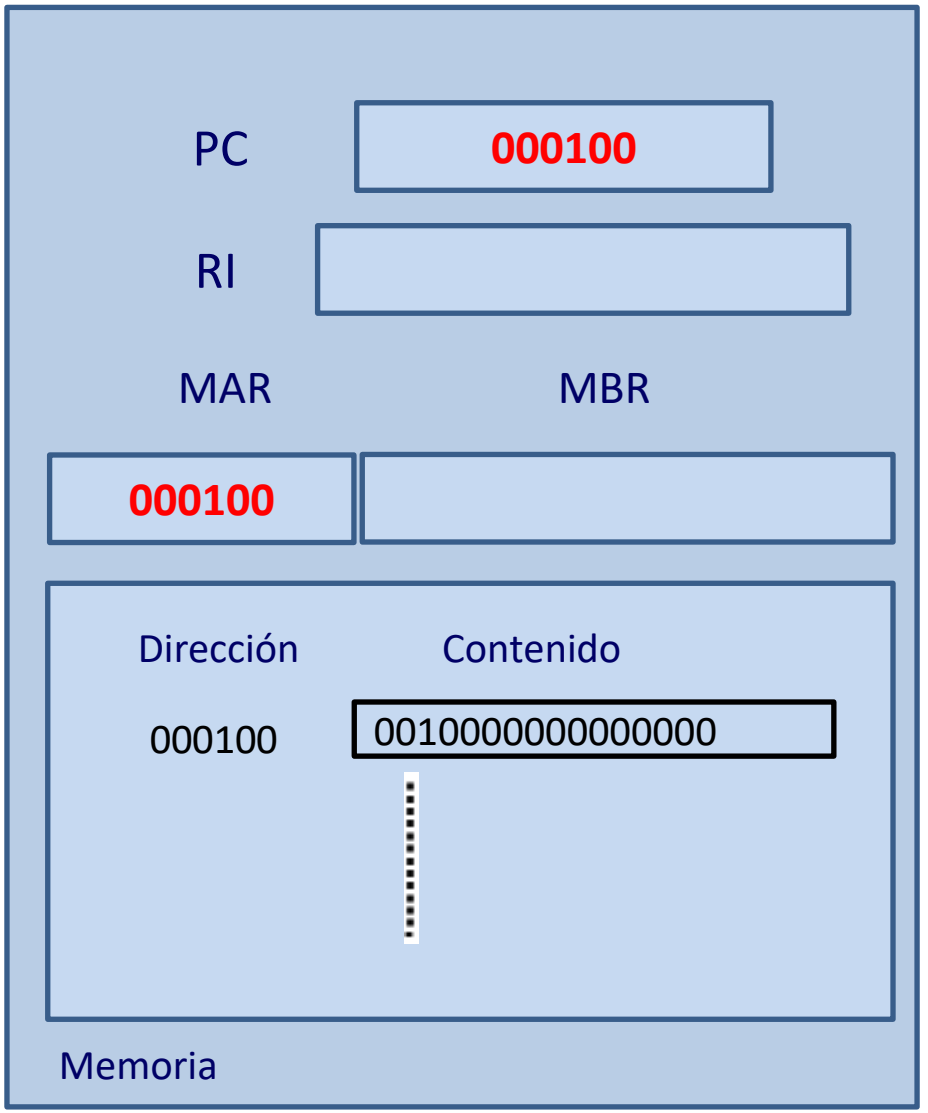
El Procesador
Fases de ejecución de instrucción

Lectura de la instrucción (ciclo de *fetch)*

- **MAR** ← **PC**
- Lectura
- MBR ← Memoria
- PC ← PC + 1
- RI ← MBR

Decodificación de la instrucción

Ejecución de la instrucción
Volver a *fetch*



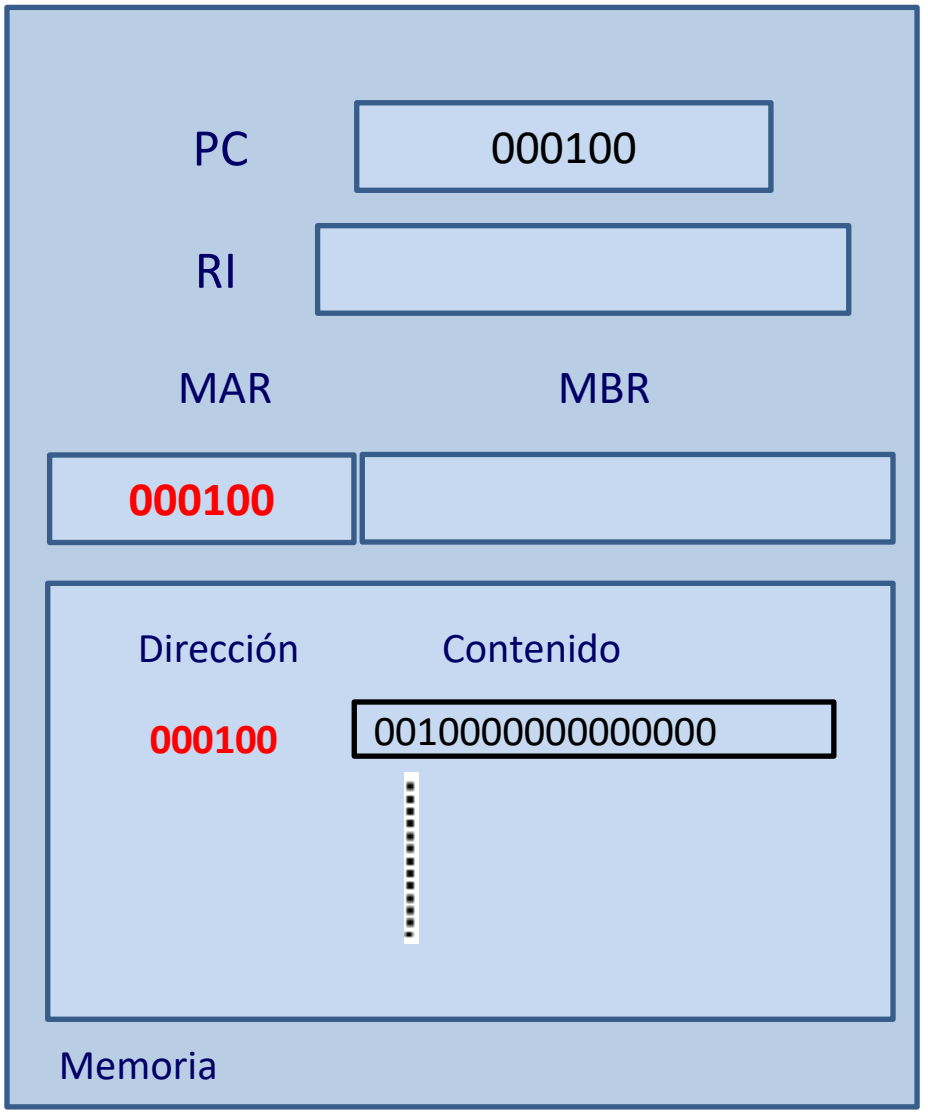
El Procesador
Fases de ejecución de instrucción

Lectura de la instrucción (ciclo de *fetch)*

- MAR ← PC
- **Lectura**
- MBR ← Memoria
- PC ← PC + 1
- RI ← MBR

Decodificación de la instrucción

Ejecución de la instrucción
Volver a *fetch*



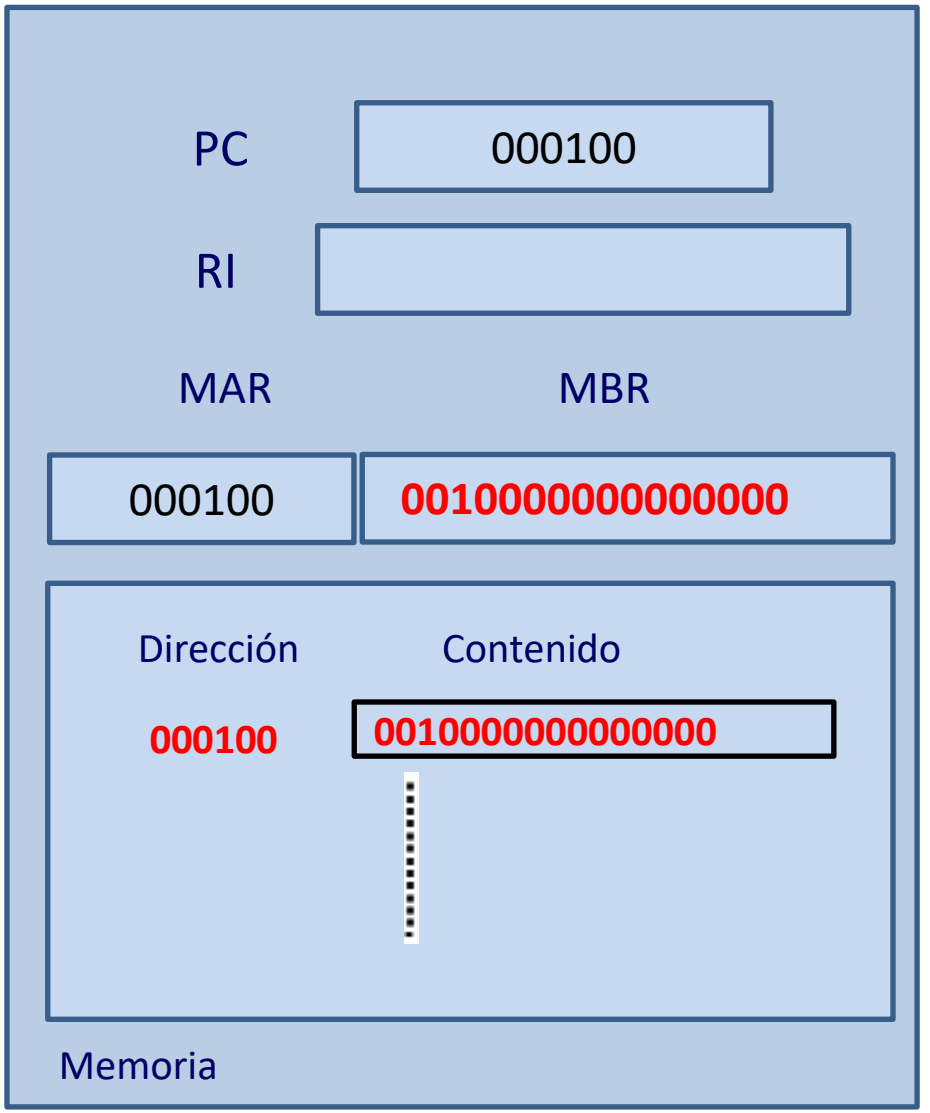
El Procesador
Fases de ejecución de instrucción

Lectura de la instrucción (ciclo de *fetch)*

- MAR ← PC
- Lectura
- **MBR ← Memoria**
- PC ← PC + 1
- RI ← MBR

Decodificación de la instrucción

Ejecución de la instrucción
Volver a *fetch*



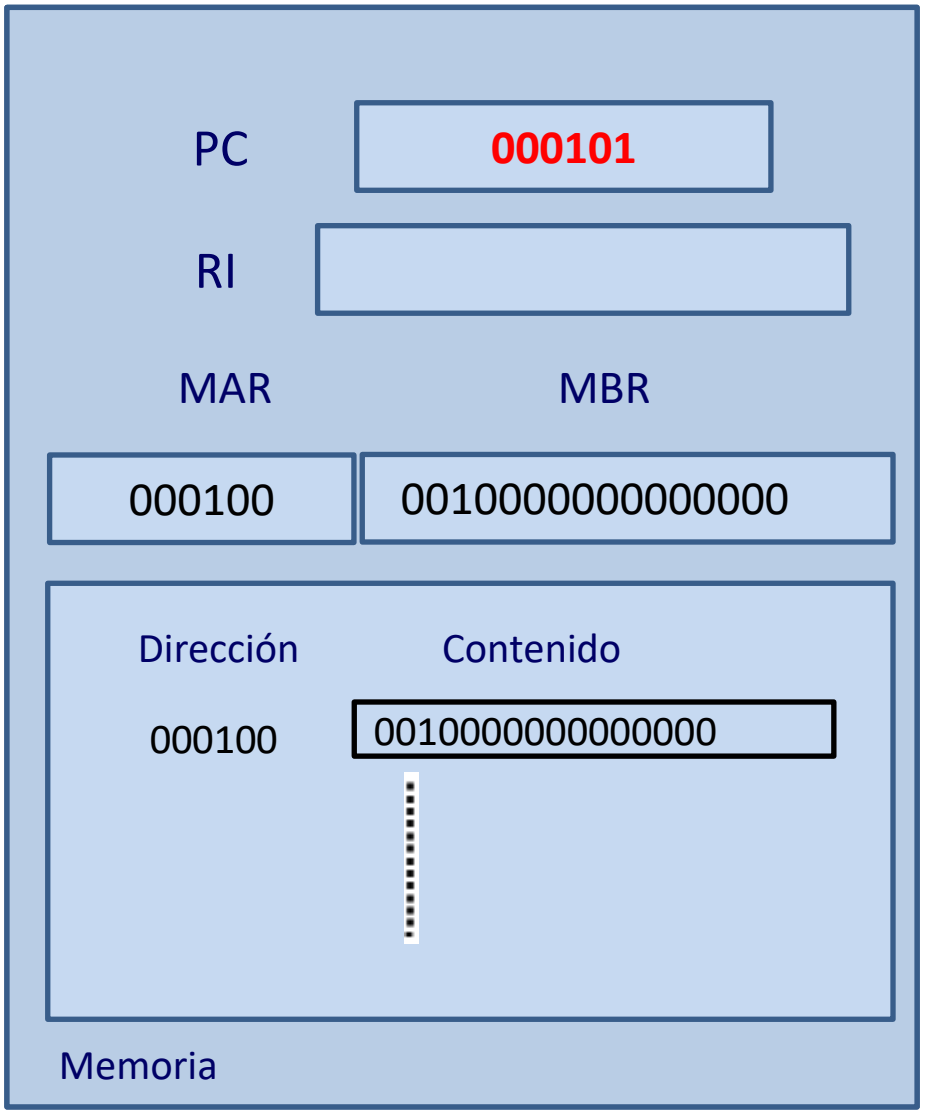
El Procesador
Fases de ejecución de instrucción

Lectura de la instrucción (ciclo de *fetch)*

- MAR ← PC
- Lectura
- MBR ← Memoria
- **PC ← PC + 1**
- RI ← MBR

Decodificación de la instrucción

Ejecución de la instrucción
Volver a *fetch*



El Procesador
Fases de ejecución de instrucción

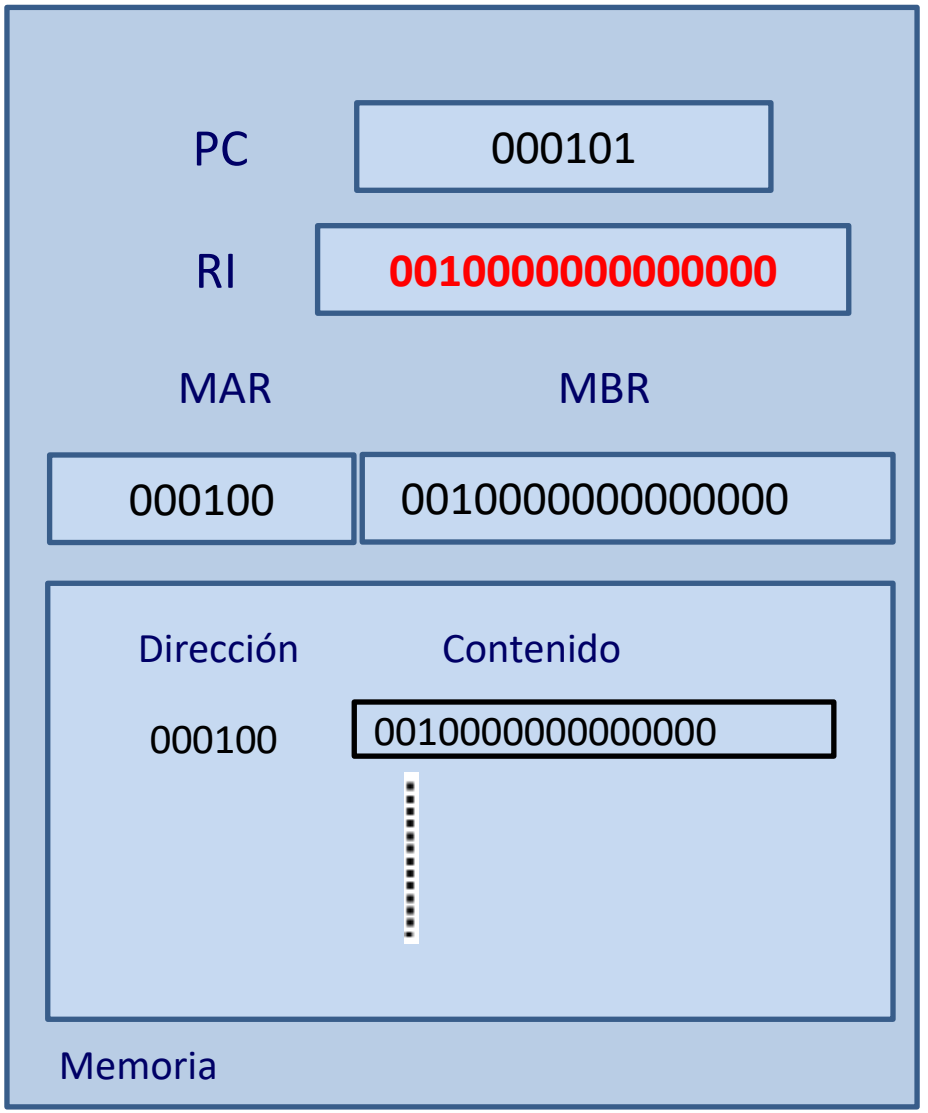
Lectura de la instrucción (ciclo de *fetch)*

- MAR ← PC
- Lectura
- MBR ← Memoria
- PC ← PC + 1
- **RI ← MBR**

Decodificación de la instrucción

Ejecución de la instrucción

Volver a *fetch*



El Procesador
Fases de ejecución de instrucción

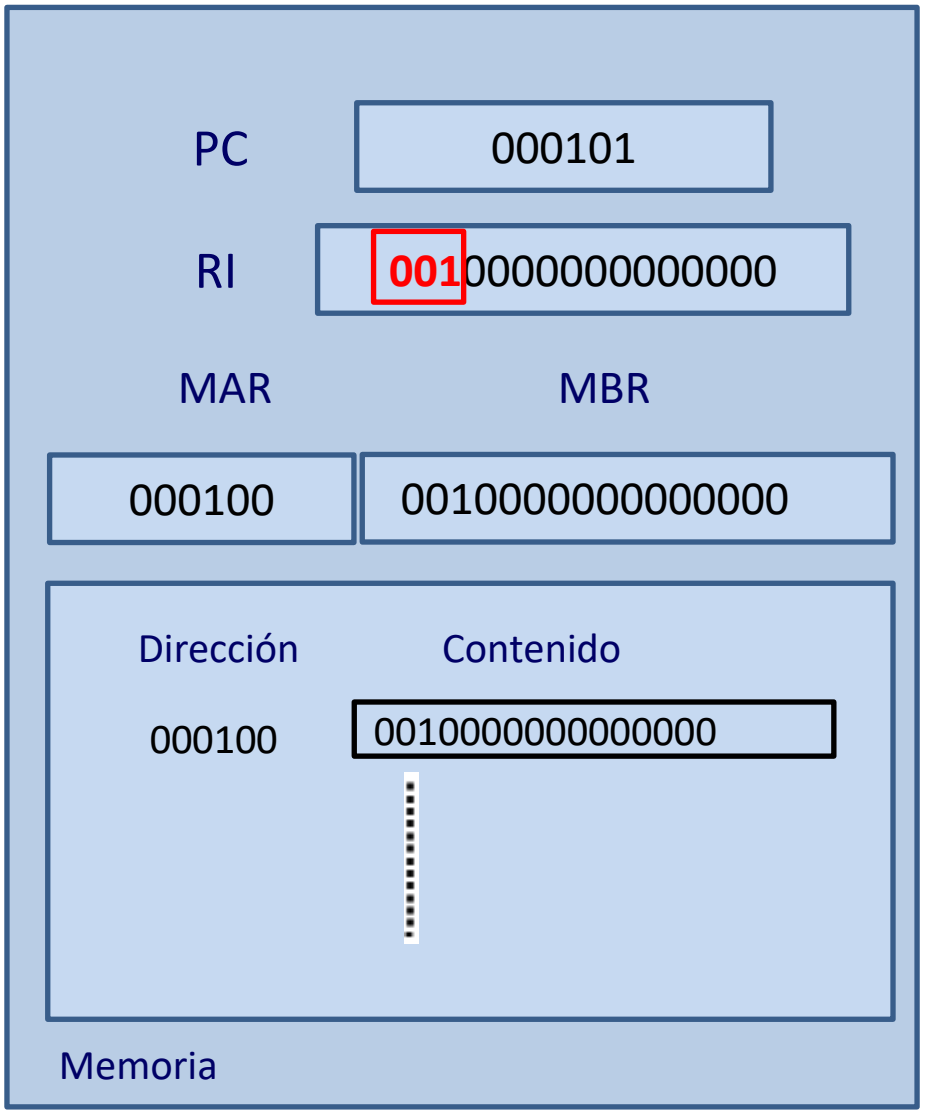
Lectura de la instrucción (ciclo de *fetch)*

- MAR ← PC
- Lectura
- MBR ← Memoria
- PC ← PC + 1
- RI ← MBR

Decodificación de la instrucción

Ejecución de la instrucción

Volver a *fetch*



El Procesador
Fases de ejecución de instrucción

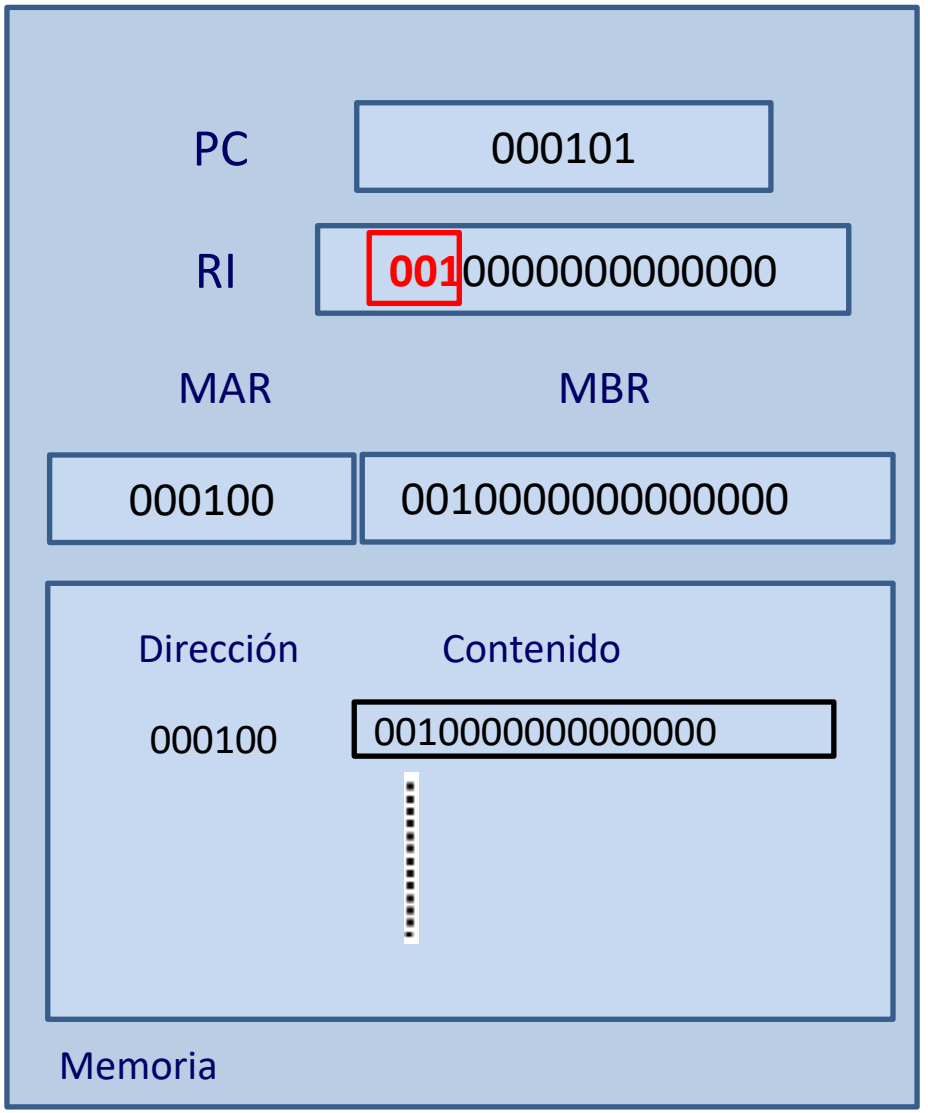
Lectura de la instrucción (ciclo de *fetch)*

- MAR ← PC
- Lectura
- MBR ← Memoria
- PC ← PC + 1
- RI ← MBR

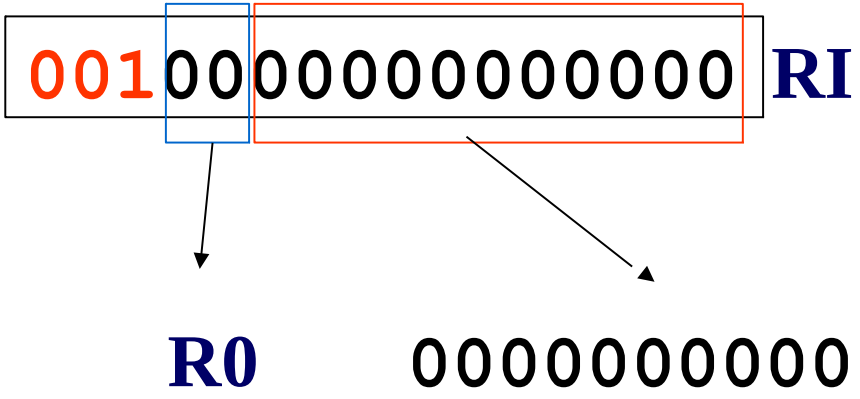
Decodificación de la instrucción

Ejecución de la instrucción

Volver a *fetch*



El Procesador
Ejecución de la instrucción

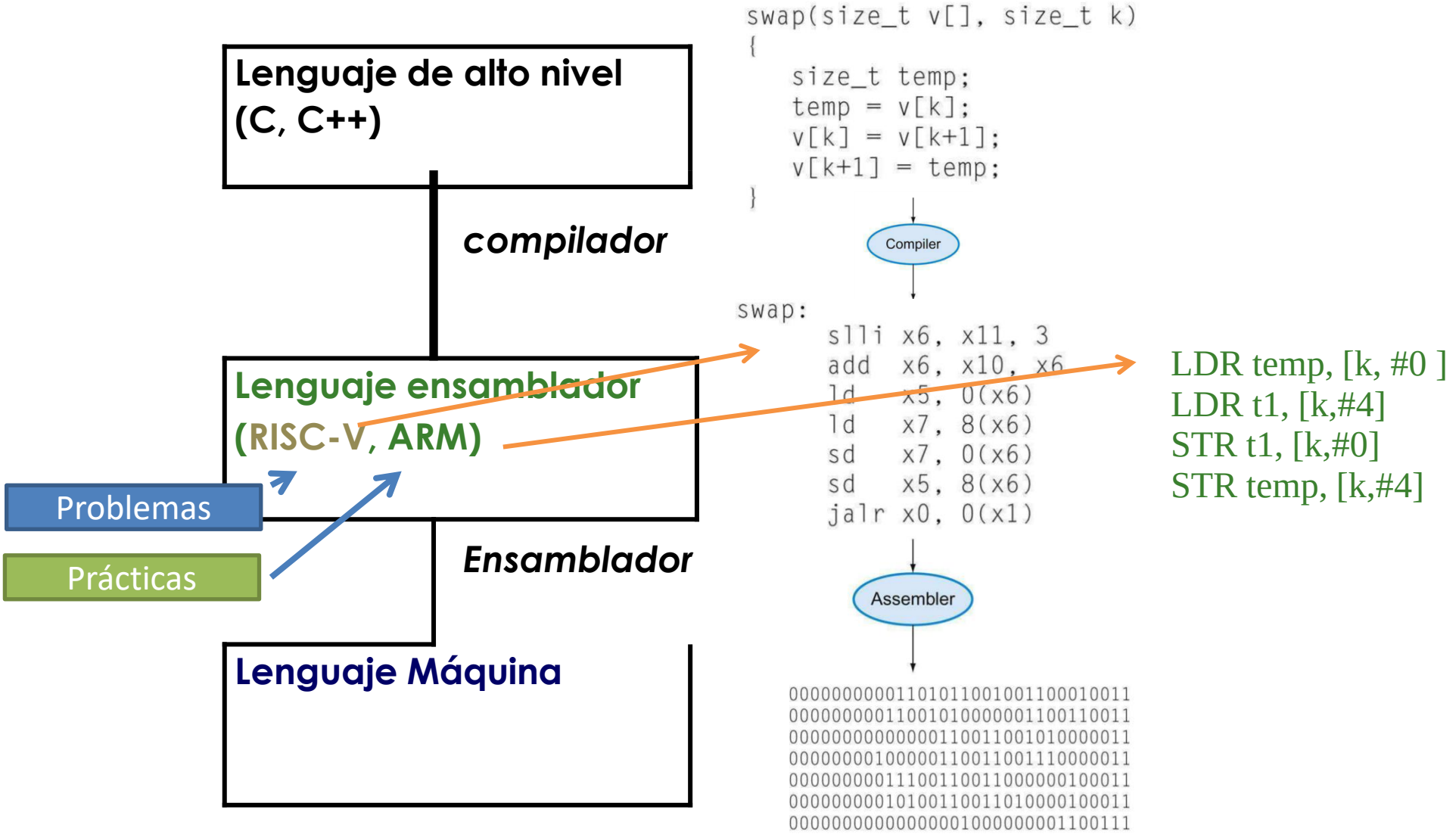


Resultado de las fases de ejecución de la instrucción anterior

Se carga en R0 el valor 0

El procesador

Un ejemplo de programa con el juego de instrucciones



Ejemplo1

Programa en lenguaje C:

```
i=0;
s = 0;
while (i < 4)
{
    s = s + 1;
    i = i + 1;
}
```

En cada iteración del bucle se suma 1 al contenido de la posición de memoria asignada a la variable s

Ejemplo 1

AYUDA

LDR Rd, [Rb]	Almacena el contenido de la posición de memoria almacenada en el registro Rb en el registro Rd: Rd ← Memoria(Rb)
STR Rf, [Rb]	Almacena el contenido del registro Rf en la posición de memoria almacenada en el registro Rb: Rf → Memoria(Rb)
CMP	Comparación de los registros
B etiqueta	Salto incondicional a etiqueta (dirección de memoria)
BEQ etiqueta	Salto a etiqueta si el contenido de los registros es igual

Ejemplo 1- Solución

Programa en lenguaje C:

```
i=0;
s = 0;
while (i < 4)
{
    s = s + 1;
    i = i + 1;
}
```

ARM

```
    mov R0, #0
    mov R1, #4
    mov R2, #1
    ldr R3, [R4]
Bucle:
    cmp R0,R1
    beq fin
    add R3, R3, R2
    add R0,R0,R2
    B bucle
Fin: str R3, [R4]
```

branch if equal

branch

store word

AYUDA

R0 → i
R1 → contiene el valor 4
R2 → valor 1 para el incremento
R3 → s
R4 → contiene la dirección de la variable s

Ejemplo1.1

Programa en lenguaje C: en RISC-V

```
i=0;
s = 0;
while (i < 4)
{
    s = s + 1;
    i = i + 1;
}
```

El programa almacena el resultado en un registro.

Ejemplo 1.1

AYUDA

Li rd, immediate	Load immediate
Mv rd, rs	Copiar registro
ADD rd, rs1, rs2	rd = rs1 + rs2
ADDI rd, rs1, imm	rd = rs1 + imm

Ejemplo 1.1 - Solución

Programa en lenguaje C: en RISC-V

```
i=0;
s = 0;
while (i < 4)
{
    s = s + 1;
    i = i + 1;
}
```

```
        li   t0, 0
        li   t1, 4
        li   t2, 1
        li   t3, 0
bucle:  beq  t0, t1, fin
        add  t3, t3, t2
        add  t0, t0, t2
        j bucle
fin:    mv t4, t3
```

load integer

brach if equal

El programa almacena el resultado en un registro.

El Procesador
Tipo de instrucciones

- **De transferencia**
- **Aritmética**
- **Lógicas**
- **De desplazamiento**
- **De rotación**
- **De comparación**
- **De bifurcación**

El Procesador
Tipo de instrucciones

Transferencia de datos

- Copia un dato en un registro: carga inmediata
  - load r1, 5 r1 ⟵ 5
- Registro a registro
  - mov r2,r1 r2 ⟵ r1
- Instrucciones de acceso a memoria(se explicará en los modos de direccionamiento)
  - ➢ De registro a memoria
  - ➢ De memoria a registro

El Procesador
Tipo de instrucciones

Aritméticas

- Realiza operaciones aritméticas de enteros (ALU) y/o aritmética de números en coma flotante (FPU)
- Ejemplos de operaciones con enteros
  - Sumar

    add r0, r1, r2 r0 ⟵ r1 + r2

    addi r0, r1, 5 r0 ⟵ r1 + 5
  - Restar

    sub r0, r2, r0
  - Multiplicar

    mul r0, r1, r2
  - Dividir

    div r0, r1, r2 r0 ⟵ r1 / r2 división entera

El Procesador
Tipo de instrucciones

Ejemplo 2

```
int a = 5;
int b = 7;
int c = 8;
int d;

d = a * (b + c)
```

El Procesador
Tipo de instrucciones

Ejemplo 2-Solución

```
int a = 5;
int b = 7;
int c = 8;
int d;

d = a * (b + c)
```

```
load      r0,   5
load      r1,   7
load      r2,   8

add  r1,   r1,    r2
mul  r3,   r1,    r0
```

El Procesador
Tipo de instrucciones

Ejemplo 3

```
int a = 5;
int b = 7;
int c = 8;
int d;

d=-(a*(b-10)+c)
```

El Procesador
Tipo de instrucciones
Ejemplo 3-Solución

```
int a = 5;
int b = 7;
int c = 8;
int d;

d=-(a*(b-10)+c)
```

```
load r0, 5
load r1, 7
load r2, 8
load r3, 10
sub r4, r1, r3
mul r4, r4, r0
add r4, r4, r2
load r5, -1
mul r4, r4, r5
```

El Procesador
Tipo de instrucciones

Lógicas

- Operaciones booleanas
- Ejemplos:
  - AND

    AND r0, r1, r2 (r0 = r1 & r2)

  - OR

    OR r0, r1, r2 (r0 = r1 | r2)

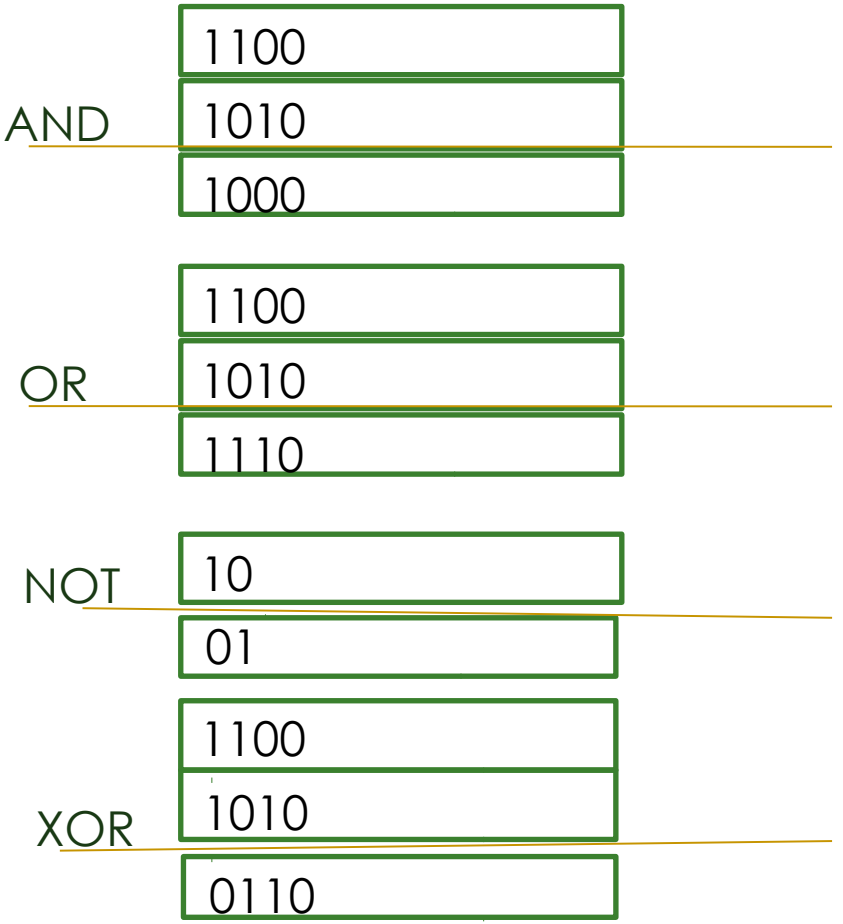
    ORI r0, r1, 80 (r0 = r1 | 80)

  - NOT (r0 = ! r1)

    not r0, r1

  - XOR (r0 = r1 ^ r2)

    xor r0, r1, r2



El Procesador
Tipo de instrucciones

Ejemplo 4

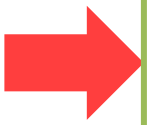
```
load    R0,     #5
load    R1,     #8
and     R2, R1, R0
```

¿Cuál es el valor de R2?

El Procesador
Tipo de instrucciones

Ejemplo 4- Solución

```
load    R0,     #5
load    R1,     #8
and     R2, R1, R0
```



¿Cuál es el valor de R2?

000 …. 0101 R0

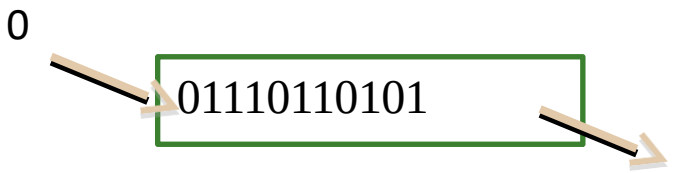
000 ….. 1000 R1

000 ….. 0000 R2

Desplazamientos

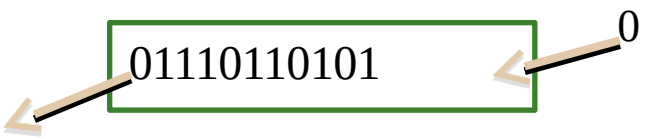
- De movimiento de bits
- Ejemplos:
  - Desplazamiento lógico a la derecha

    SRL r0, r0, 4 (r0 = r0 >> 4 bits)



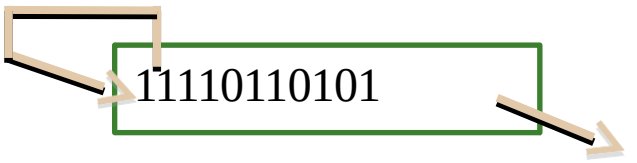
  - Desplazamiento lógico a la izquierda

    SLL r0, r0, 5 (r0 = r0 << 5 bits)



  - Desplazamiento aritmético a la derecha mantiene el signo

    SRA r0, r0, 2 (r0 = r0 >> 2 bits)



El Procesador
Tipo de instrucciones

Ejemplo 5

```
load    R0,     #5
load    R1,     #6
sra     R0, R1, 1
```

¿Cuál es el valor de R0?

El Procesador
Tipo de instrucciones

Ejemplo 5

```
load    R0,     #5
load    R1,     #6
sra     R0, R1, 1
```

¿Cuál es el valor de R0?

… 16 8 4 2 1

000 …. 0110 R1

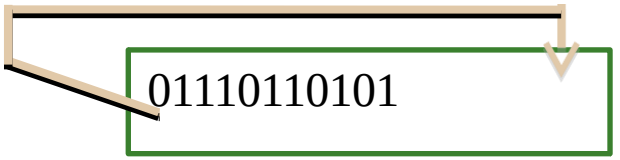
Se desplaza 1 bit a la derecha

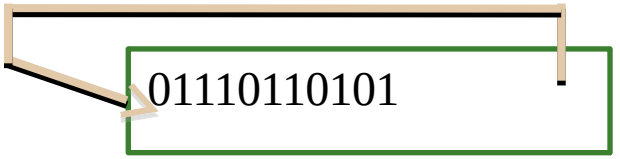
000 ….. 0011 R0

El Procesador
Tipo de instrucciones

Rotaciones

- De movimiento de bits
- Ejemplos:
  - Rotación a la izquierda
    rol R0, R0, #4 rotan 4 bits
  - Rotación a la derecha
    ror R0,R0, #5 rotan 5 bits





El Procesador
Tipo de instrucciones

Comparación y salto

- **CMP** R1,R2 — compara el contenido de dos registros y actualiza la palabra de estado del procesador

➢ B	b etiqueta	actualiza el registro PC con el valor de la etiqueta
➢ Beq	beq etiqueta	actualiza el registro PC si R1 = R2
➢ Bge	bge etiqueta	actualiza el registro PC si R1 >= R2
➢ Ble	ble etiqueta	actualiza el registro PC si R1 <= R2
➢ Bne	bne etiqueta	actualiza el registro PC si R1 ≠ R2

El Procesador
Tipo de instrucciones

De bifurcación (salto)

Estos valores se comparan con la instrucción cmp vista anteriormente

- Distintos tipos:
  - Bifurcación o salto condicional:
    beq dirección
    - Salta a la posición dirección, si valor R0== R1

    beqz dirección
    - Salta a la instrucción etiqueta con dirección si R1== 0
  - Bifurcación o salto incondicional:
    - El salto se realiza siempre

      j dirección

      b dirección
  - Llamadas a funciones:

    jal direccion .....

    jr registro

Llamadas a funciones se verá más adelante

El Procesador
Tipo de instrucciones

Ejemplo 6

Pseudocódigo	Ensamblador

```
if (R1 > R2)
      R3 = R4 + R5;
else
      R3 = R4 - R5
sigue la ejecución normal
```

```
          CMP R1, R2
          BLE else
          ADD R3, R4, R5
          B fin_if
else:     SUB R3, R4, R5
fin_if:   sigue la ejecución normal
```

El Procesador
Tipo de instrucciones

Ejemplo 7

```
Int i;
i=0;
while (i < 10)
{
    /* acción */
     i = i + 1 ;
}
```

El Procesador
Tipo de instrucciones

Ejemplo 7-Solución

```
Int i;
i=0;
while (i < 10)
{
    /* acción */
     i = i + 1 ;
}
```

```
            load r0, 0          #i
            load r1, 10
while:      cmp r0, r1
            bge fin
             # acción
            addi r0, r0, 1
            b while
fin:
            ...
```

El Procesador
Tipo de instrucciones

Ejemplo 8

- Calcular la suma 1 + 2 + 3 + …. + 10

```
   i=0;
   s=0;
   while (i <= 10)
   {
      s = s + i;
      i = i + 1;
   }
}
```

- Resultado en R1

El Procesador
Tipo de instrucciones

Ejemplo 8- Solución

- Calcular la suma 1 + 2 + 3 + …. + 10

```
  i=0;
  s=0;
  while (i <= 10)
  {
     s = s + i;
     i = i + 1;
  }
}
```

```
          Load R0,0        #i
          Load R1,0        #s
          Load R2,10
While:    cmp R0,R2
          bgt fin
          add R1, R1, R0
          addi R0, R0, 1
          b while
Fin:      …
```

- Resultado en R1

El Procesador
Tipo de instrucciones

Ejemplo 9

- Calcular el número de 1´s que hay en un registro "R1". Resultado en R3

```
  i = 0;
  n = 45;   #numero
  s=0;
  while (i < 32)
  {
    b = ultimo bit de n
    s = s + b;
    se desplaza n un bit
    a la derecha
    i = i + 1 ;
  }
}
```

El Procesador
Tipo de instrucciones

Ejemplo 9-Solución

- Calcular el número de 1´s que hay en un registro "R1". Resultado en R3

```
  i = 0;
  n = 45;   #numero
  s=0;
  while (i < 32)
  {
    b = ultimo bit de n
    s = s + b;
    se desplaza n un bit
    a la derecha
    i = i + 1 ;
  }
}
```

```
            load  R0, 0    #i
            load  R1, 45   #n
            load  R2, 32
            load  R3, 0     #s
while:      cmp   R0, R2
            bge   fin
            and   R4, R1, 1
            add   R3, R3, R4
            srl   R1, R1, 1
            addi  R0, R0, 1
            b     while
fin:        ...
```

Repertorio de instrucciones RISC-V

Categoría	Nombre	Fmt	RV32I Base	
Shifts	Shift Left Logical	R	SLL	rd,rs1,rs2
	Shift Left Log. Imm.	I	SLLI	rd,rs1,shamt
	Shift Right Logical	R	SRL	rd,rs1,rs2
	Shift Right Log. Imm.	I	SRLI	rd,rs1,shamt
	Shift Right Arithmetic	R	SRA	rd,rs1,rs2
	Shift Right Arith. Imm.	I	SRAI	rd,rs1,shamt
Aritmética	ADD	R	ADD	rd,rs1,rs2
	ADD Immediate	I	ADDI	rd,rs1,imm
	SUBtract	R	SUB	rd,rs1,rs2
	Load Upper Imm	U	LUI	rd,imm
	Add Upper Imm to PC	U	AUIPC	rd,imm
Lógica	XOR	R	XOR	rd,rs1,rs2
	XOR Immediate	I	XORI	rd,rs1,imm
	OR	R	OR	rd,rs1,rs2
	OR Immediate	I	ORI	rd,rs1,imm
	AND	R	AND	rd,rs1,rs2
	AND Immediate	I	ANDI	rd,rs1,imm
Comparación	Set <	R	SLT	rd,rs1,rs2
	Set < Immediate	I	SLTI	rd,rs1,imm
	Set < Unsigned	R	SLTU	rd,rs1,rs2
	Set < Imm Unsigned	I	SLTIU	rd,rs1,imm
Branches	Branch =	B	BEQ	rs1,rs2,imm
	Branch ≠	B	BNE	rs1,rs2,imm
	Branch <	B	BLT	rs1,rs2,imm
	Branch ≥	B	BGE	rs1,rs2,imm
	Branch < Unsigned	B	BLTU	rs1,rs2,imm
	Branch ≥ Unsigned	B	BGEU	rs1,rs2,imm
Jump & Link	J&L	J	JAL	rd,imm
	Jump & Link Register	I	JALR	rd,rs1,imm
Sinc.	Synch thread	I	FENCE	
	Synch Instr & Data	I	FENCE.I	
Ambiente	CALL	I	ECALL	
	BREAK	I	EBREAK	

Loads	Load Byte	I	LB	rd,rs1,imm
	Load Halfword	I	LH	rd,rs1,imm
	Load Byte Unsigned	I	LBU	rd,rs1,imm
	Load Half Unsigned	I	LHU	rd,rs1,imm
	Load Word	I	LW	rd,rs1,imm
Stores	Store Byte	S	SB	rs1,rs2,imm
	Store Halfword	S	SH	rs1,rs2,imm
	Store Word	S	SW	rs1,rs2,imm

Guía Práctica de RISC-V. El Atlas de una Arquitectura Abierta
http://riscvbook.com/spanish/

Repertorio de instrucciones ARM

Mnemonic	Instruction	Action
ADC	Add with carry	Rd := Rn + Op2 + Carry
ADD	Add	Rd := Rn + Op2
AND	AND	Rd := Rn AND Op2
B	Branch	R15 := address
BIC	Bit Clear	Rd := Rn AND NOT Op2
BL	Branch with Link	R14 := R15, R15 := address
BX	Branch and Exchange	R15 := Rn, T bit := Rn[0]
CDP	Coprocesor Data Processing	(Coprocessor-specific)
CMN	Compare Negative	CPSR flags := Rn + Op2
CMP	Compare	CPSR flags := Rn - Op2
EOR	Exclusive OR	Rd := (Rn AND NOT Op2) OR (op2 AND NOT Rn)
LDC	Load coprocessor from memory	Coprocessor load
LDM	Load multiple registers	Stack manipulation (Pop)
LDR	Load register from memory	Rd := (address)
MCR	Move CPU register to coprocessor register	cRn := rRn {<op>cRm}
MLA	Multiply Accumulate	Rd := (Rm * Rs) + Rn
MOV	Move register or constant	Rd : = Op2
MRC	Move from coprocessor register to CPU register	Rn := cRn {<op>cRm}
MRS	Move PSR status/flags to register	Rn := PSR
MSR	Move register to PSR status/flags	PSR := Rm
MUL	Multiply	Rd := Rm * Rs
MVN	Move negative register	Rd := 0xFFFFFFFF EOR Op2
ORR	OR	Rd := Rn OR Op2
RSB	Reverse Subtract	Rd := Op2 - Rn
RSC	Reverse Subtract with Carry	Rd := Op2 - Rn - 1 + Carry
SBC	Subtract with Carry	Rd := Rn - Op2 - 1 + Carry
STC	Store coprocessor register to memory	address := CRn
STM	Store Multiple	Stack manipulation (Push)
STR	Store register to memory	<address> := Rd
SUB	Subtract	Rd := Rn - Op2
SWI	Software Interrupt	OS call
SWP	Swap register with memory	Rd := [Rn], [Rn] := Rm
TEQ	Test bitwise equality	CPSR flags := Rn EOR Op2
TST	Test bits	CPSR flags := Rn AND Op2

Estructura de Computadores: ESI Ciudad Real
https://slideplayer.es/slide/4288181/