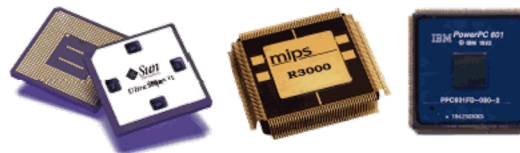


Tema 3: Microprocesador MIPS

Fundamentos de Ordenadores y Sistemas Operativos

Mario Martínez Zarzuela
marmar@tel.uva.es



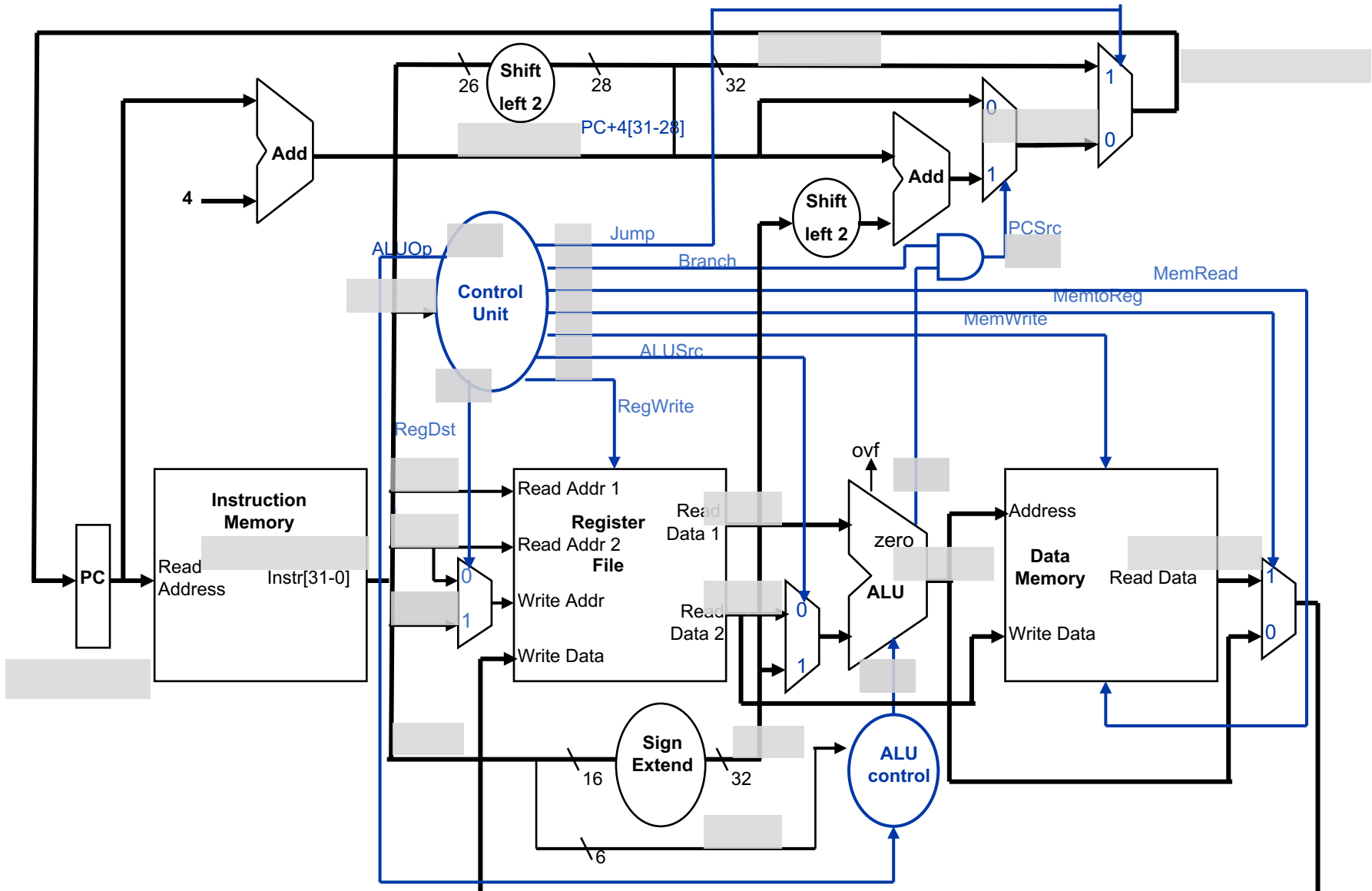
Preguntas a responder

1. ¿Cómo se convierten las instrucciones a **código máquina**?
2. ¿**Cómo se ejecuta** el código máquina en el microprocesador?

Camino de Datos y Unidad de Control

Imprime y revisa esta plantilla

R

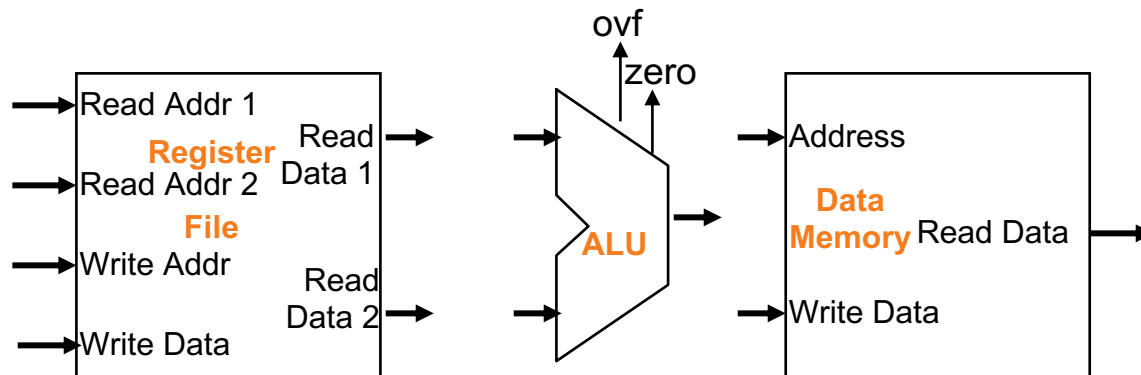


Contenidos

- Repaso de conocimientos previos
- Instrucciones aritméticas y de carga/almacenamiento
 - Codificación de las instrucciones
 - Construcción del camino de datos
- Instrucciones de salto condicional/no condicional
 - Codificación de las instrucciones
 - Construcción del camino de datos
- Señales de control según instrucción
- Conclusiones y bibliografía

Tema 2: Carga y almacenamiento

- Pasos necesarios para realizar cálculos con datos dentro del microprocesador
 - **Cargar (load)** los operandos
 - Desde la memoria de datos hacia los registros
 - **Operar** en la Unidad Aritmético Lógica (ALU)
 - **Almacenar (store)** el resultado
 - Desde los registros hacia la memoria



Tema 2: Tipos de instrucciones

- **Tipo R:** todos los operandos en registros (**r**)

```
add  rd, rs, rt      # addition      rd←rs+rt
sub  rd, rs, rt      # subtraction  rd←rs-rt
```

- **Tipo I:** incluyen algún dato inmediato (**immediate**)

```
addi rt, rs, imm     # addition      rd←rs+imm
subi rt, rs, imm     # subtraction  rd←rs-imm
```

```
lw   rt, imm(rs)     # load word from memory
sw   rt, imm(rs)     # store word to memory
```

```
beq  rs, rt, imm     # branch if rs==rt
```

```
bne  rs, rt, imm     # branch if rs!=rt
```

- **Tipo J:** para salto (**jump**)

```
j      addr          #jump
```

Repaso Tema 2

Completa la tabla

R

- Pon un ejemplo de las siguientes instrucciones e indica qué operación se realiza

Instrucción	Ejemplo	Explicación
	<code>add \$s2,\$s0,\$zero</code>	
<code>addi rt,rs,imm</code>		<code>\$zero + 3 → \$s2</code>
<code>lw rt,imm(rs)</code>		Carga <i>var</i> en <code>\$s2</code>
	<code>sw \$s2,var(\$zero)</code>	
<code>beq rs,rt,imm</code>		Si <code>\$s1 == \$s2</code> salta a <i>loop</i>
		salta a <i>fin</i>

Repaso Tema 2

Completa la tabla


- Pon un ejemplo de las siguientes instrucciones e indica qué operación se realiza


Instrucción	Ejemplo	Explicación
	<code>add \$s2,\$s0,\$zero</code>	
<code>addi rt,rs,imm</code>		<code>\$zero + 3 → \$s2</code>
<code>lw rt,imm(rs)</code>		Carga <i>var</i> en <code>\$s2</code>
	<code>sw \$s2,var(\$zero)</code>	
<code>beq rs,rt,imm</code>		Si <code>\$s1 == \$s2</code> salta a <i>loop</i>
		salta a <i>fin</i>

Tema 2: Completa las instrucciones

V

```
.data
num1: .word    11
num2: .word    12
res:  .word    -1

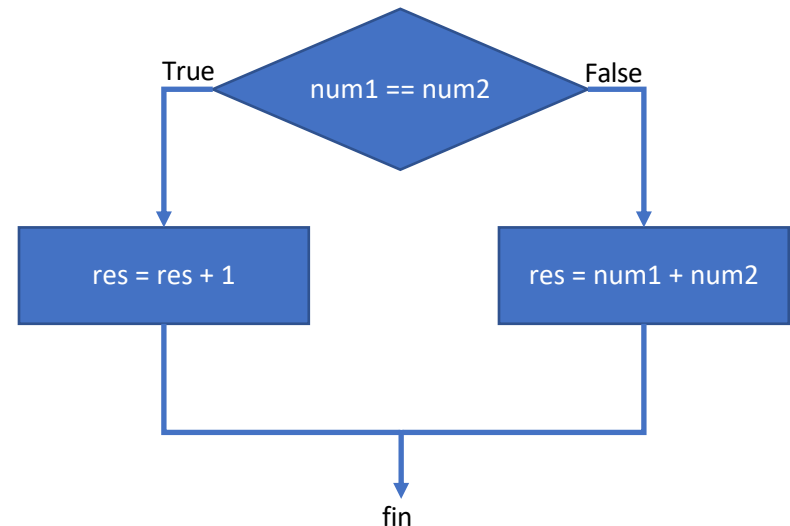
.text
lw    $s1,num1($zero)
lw    $s2,num2($zero)
beq   $s1,$s2,salto
      
j      fin

salto: 

fin:  sw      $s0,res($zero)
```

```
int num1 = 11;
int num2 = 12;
int res  = -1;

if (num1==num2)
    res = 1;
else
    res = num1 + num2;
```



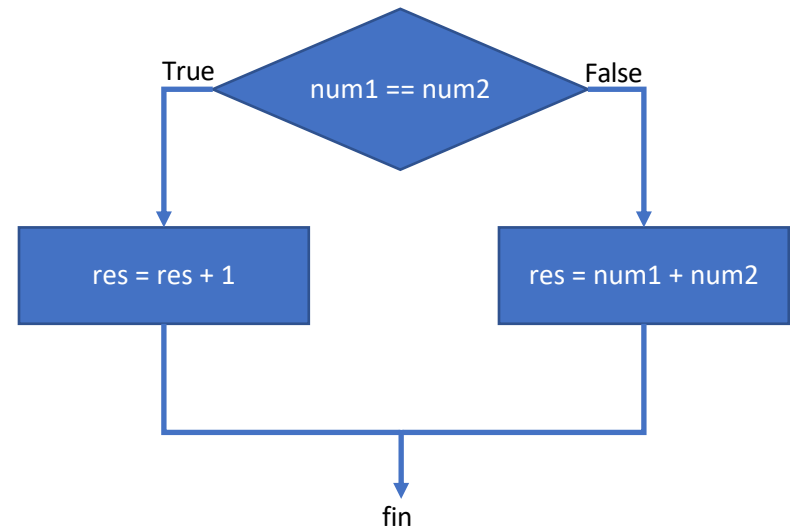
Tema 2: Completa las instrucciones

```
.data
num1: .word    11
num2: .word    12
res:  .word    -1

.text
lw    $s1,num1($zero)
lw    $s2,num2($zero)
beq   $s1,$s2,salto
¿?
j     fin
salto:¿?
fin:  sw      $s0,res($zero)
```

```
int num1 = 11;
int num2 = 12;
int res  = -1;

if (num1==num2)
    res = 1;
else
    res = num1 + num2;
```



Ejecutar código anterior en MARS

R

/Users/marmar/Library/Mobile Documents/com~apple~CloudDocs/#_EN_EJECUCION/01.DOCENCIA/ASIGNATURAS/2019-20_FOSQ/teoriaT3.1.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00003000	0x8c110000	lw \$17,0x00000000(\$0)	7: main: lw \$s1, num1(\$zero)
<input type="checkbox"/>	0x00003004	0x8c120004	lw \$18,0x00000004(\$0)	8: lw \$s2, num2(\$zero)
<input type="checkbox"/>	0x00003008	0x12320002	beq \$17,\$18,0x00000002	9: beq \$s1, \$s2, salto
<input type="checkbox"/>	0x0000300c	0x02328020	add \$16,\$17,\$18	10: add \$s0, \$s1, \$s2
<input type="checkbox"/>	0x00003010	0x08000c06	j 0x00003018	11: j fin
<input type="checkbox"/>	0x00003014	0x22100001	addi \$16,\$16,0x0000...	12: salto: addi \$s0, \$s0, 1
<input type="checkbox"/>	0x00003018	0xac100008	sw \$16,0x00000008(\$0)	13: fin: sw \$s0, res(\$zero)
<input type="checkbox"/>	0x0000301c	0x00000000	nop	14: nop

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00000000	0x0000000b	0x0000000c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000001a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000001c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x0000000b
\$s2	18	0x0000000c
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x00001800
\$sp	29	0x00002ffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00003008
hi		0x00000000
lo		0x00000000

0x00000000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Mars Messages Run I/O

Clear

Contenidos

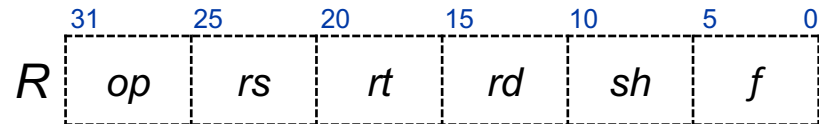
- Repaso de conocimientos previos
- **Instrucciones aritméticas y de carga/almacenamiento**
 - Codificación de las instrucciones
 - Construcción del camino de datos
- Instrucciones de salto condicional/no condicional
 - Codificación de las instrucciones
 - Construcción del camino de datos
- Señales de control según instrucción
- Conclusiones y bibliografía

Contenido de los campos de instrucción según su tipo

- **Tipo R:** todos los operandos en registros (**r**)

add
sub

rd, rs, rt
rd, rs, rt



Escribir los campos de instrucción

V

```

                                .data
00000  0x0000  num1: .word    11
                                num2: .word    12
                                res:  .word   -1
                                .text
12288  0x3000      lw      $s1,num1($zero)
                                lw      $s2,num2($zero)
                                beq     $s1,$s2,salto
                                add     $s0,$s1,$s2
                                j       fin
                                salto:addi $s0,$zero,1
                                fin:  sw     $s0,res($zero)
    
```

op	rs	rt	rd	sh	f
op	rs	rt	imm		
op	addr				
¿?	¿?	¿?	¿?	¿?	¿?
¿?	¿?	¿?	¿?	¿?	¿?

Contenido de los campos de instrucción según su tipo

- Rellenar campos de instrucción:
 - Correspondencia direcciones de registros

MIPS reference card

add	rd, rs, rt	Add	rd = rs + rt	R 0 / 20	registers \$0 \$zero \$1 \$at \$2-\$3 \$v0-\$v1 \$4-\$7 \$a0-\$a3 \$8-\$15 \$t0-\$t7 \$16-\$23 \$s0-\$s7 \$24-\$25 \$t8-\$t9 \$26-\$27 \$k0-\$k1 \$28 \$gp \$29 \$sp \$30 \$fp \$31 \$ra hi — lo — PC — c0 \$13 c0_cause c0 \$14 c0_epc
sub	rd, rs, rt	Subtract	rd = rs - rt	R 0 / 22	
addi	rt, rs, imm	Add Imm.	rt = rs + imm _±	I 8	
addu	rd, rs, rt	Add Unsigned	rd = rs + rt	R 0 / 21	
subu	rd, rs, rt	Subtract Unsigned	rd = rs - rt	R 0 / 23	
addiu	rt, rs, imm	Add Imm. Unsigned	rt = rs + imm _±	I 9	
mult	rs, rt	Multiply	{hi, lo} = rs * rt	R 0 / 18	
div	rs, rt	Divide	lo = rs / rt; hi = rs % rt	R 0 / 1a	
multu	rs, rt	Multiply Unsigned	{hi, lo} = rs * rt	R 0 / 19	
divu	rs, rt	Divide Unsigned	lo = rs / rt; hi = rs % rt	R 0 / 1b	
mfhi	rd	Move From Hi	rd = hi	R 0 / 10	
mflo	rd	Move From Lo	rd = lo	R 0 / 12	
and	rd, rs, rt	And	rd = rs & rt	R 0 / 24	
or	rd, rs, rt	Or	rd = rs rt	R 0 / 25	
nor	rd, rs, rt	Nor	rd = ~(rs rt)	R 0 / 27	
xor	rd, rs, rt	eXclusive Or	rd = rs ^ rt	R 0 / 26	
andi	rt, rs, imm	And Imm.	rt = rs & imm ₀	I c	
ori	rt, rs, imm	Or Imm.	rt = rs imm ₀	I d	
xori	rt, rs, imm	eXclusive Or Imm.	rt = rs ^ imm ₀	I e	

Contenido de los campos de instrucción según su tipo

- Rellenar campos de instrucción:
 - Correspondencia direcciones de registros
 - Códigos de operación y de función

MIPS reference card

	inst.	op.	funct.				
Tipo-R	add	0	32	add	rd, rs, rt	Add	rd = rs + rt
	sub	0	34	sub	rd, rs, rt	Subtract	rd = rs - rt
	sllt	0	42	addi	rt, rs, imm	Add Imm.	rt = rs + imm _±
	sll	0	0	addu	rd, rs, rt	Add Unsigned	rd = rs + rt
				subu	rd, rs, rt	Subtract Unsigned	rd = rs - rt
Tipo-I				addiu	rt, rs, imm	Add Imm. Unsigned	rt = rs + imm _±
	lw	35		mult	rs, rt	Multiply	{hi, lo} = rs * rt
	sw	43		div	rs, rt	Divide	lo = rs / rt; hi = rs % rt
	addi	8		multu	rs, rt	Multiply Unsigned	{hi, lo} = rs * rt
	sllti	10		divu	rs, rt	Divide Unsigned	lo = rs / rt; hi = rs % rt
	beq	4		mfhi	rd	Move From Hi	rd = hi
	bne	5		mflo	rd	Move From Lo	rd = lo
				and	rd, rs, rt	And	rd = rs & rt
				or	rd, rs, rt	Or	rd = rs rt
				nor	rd, rs, rt	Nor	rd = ~(rs rt)
Tipo-J	j	2		xor	rd, rs, rt	eXclusive Or	rd = rs ^ rt
	jal	3		andi	rt, rs, imm	And Imm.	rt = rs & imm ₀
				ori	rt, rs, imm	Or Imm.	rt = rs imm ₀
				xori	rt, rs, imm	eXclusive Or Imm.	rt = rs ^ imm ₀

R 0 / 20
 R 0 / 22
 I 8
 R 0 / 21
 R 0 / 23
 I 9
 R 0 / 18
 R 0 / 1a
 R 0 / 19
 R 0 / 1b
 R 0 / 10
 R 0 / 12
 R 0 / 24
 R 0 / 25
 R 0 / 27
 R 0 / 26
 I c
 I d
 I e

Escribir los campos de instrucción

```

                                .data
00000  0x0000  num1: .word    11
                                num2: .word    12
                                res:  .word   -1
                                .text
12288  0x3000          lw      $s1,num1($zero)
                                lw      $s2,num2($zero)
                                beq      $s1,$s2,salto
add    rd,rs,rt  → add      $s0,$s1,$s2
                                j         fin
                                salto:addi $s0,$zero,1
                                fin:  sw      $s0,res($zero)
    
```

<i>op</i>	<i>rs</i>	<i>rt</i>	<i>rd</i>	<i>sh</i>	<i>f</i>
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>imm</i>		
<i>op</i>	<i>addr</i>				
0	17	18	16	0	32

Escribir los campos de instrucción

V

```

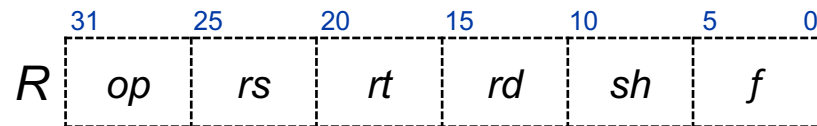
                                .data
00000  0x0000  num1: .word    11
                                num2: .word    12
                                res:  .word   -1
                                .text
12288  0x3000          lw      $s1,num1($zero)
                                lw      $s2,num2($zero)
                                beq      $s1,$s2,salto
                                add      $s0,$s1,$s2
                                j         fin
                                salto:addi $s0,$zero,1
                                fin:  sw      $s0,res($zero)
    
```

<i>op</i>	<i>rs</i>	<i>rt</i>	<i>rd</i>	<i>sh</i>	<i>f</i>
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>imm</i>		
<i>op</i>	<i>addr</i>				
0	17	18	16	0	32
¿?	¿?	¿?	¿?		

Contenido de los campos de instrucción según su tipo

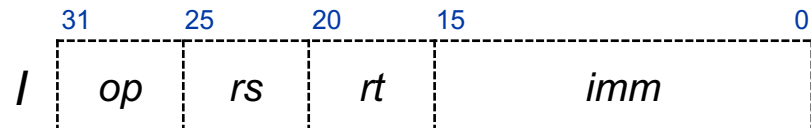
- **Tipo R:** todos los operandos en registros (**r**)

add rd, rs, rt
sub rd, rs, rt



- **Tipo I:** incluyen algún dato inmediato (**immediate**)

addi rt, rs, imm
subi rt, rs, imm



Escribir los campos de instrucción

```

                                .data
00000  0x0000  num1: .word    11
                                num2: .word    12
                                res:  .word   -1
                                .text
12288  0x3000          lw      $s1,num1($zero)
                                lw      $s2,num2($zero)
                                beq      $s1,$s2,salto
                                add      $s0,$s1,$s2
                                j         fin
addi    rt,rs,imm  addi    $s0,$zero,1
                                fin:    sw      $s0,res($zero)

```

<i>op</i>	<i>rs</i>	<i>rt</i>	<i>rd</i>	<i>sh</i>	<i>f</i>
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>imm</i>		
<i>op</i>	<i>addr</i>				
0	17	18	16	0	32
¿?	¿?	¿?	¿?		

Escribir: (1) las direcciones de datos e instrucciones (2) los campos de instrucción

v

```

                                .data
00000  0x0000  num1: .word    11
      ¿?   ¿?   num2: .word    12
      ¿?   ¿?   res:  .word   -1

                                .text
12288  0x3000          lw      $s1,num1($zero)
      ¿?   ¿?          lw      $s2,num2($zero)
      ¿?   ¿?          beq     $s1,$s2,salto
      ¿?   ¿?          add     $s0,$s1,$s2
      ¿?   ¿?          j       fin
      ¿?   ¿?  salto:addi    $s0,$zero,1
      ¿?   ¿?  fin:  sw      $s0,res($zero)
  
```

op	rs	rt	rd	sh	f
op	rs	rt	imm		
op	addr				
¿?	¿?	¿?	¿?		
¿?	¿?	¿?	¿?		
¿?	¿?	¿?	¿?		

Escribir: (1) las direcciones de datos e instrucciones

```
                .data
00000  0x0000  num1: .word    11
00004  0x0004  num2: .word    12
00008  0x0008  res:  .word   -1

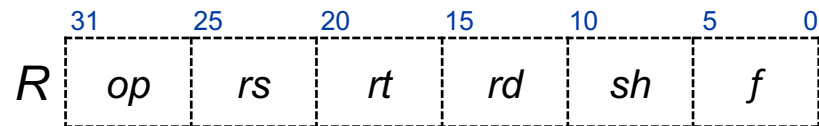
                .text
12288  0x3000          lw      $s1,num1($zero)
12292  0x3004          lw      $s2,num2($zero)
12296  0x3008          beq     $s1,$s2,salto
12300  0x300C          add     $s0,$s1,$s2
12304  0x3010          j       fin
12308  0x3014  salto: addi     $s0,$zero,1
12312  0x3018  fin:   sw      $s0,res($zero)
```

<i>op</i>	<i>rs</i>	<i>rt</i>	<i>rd</i>	<i>sh</i>	<i>f</i>
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>imm</i>		
<i>op</i>	<i>addr</i>				

Contenido de los campos de instrucción según su tipo

- **Tipo R:** todos los operandos en registros (**r**)

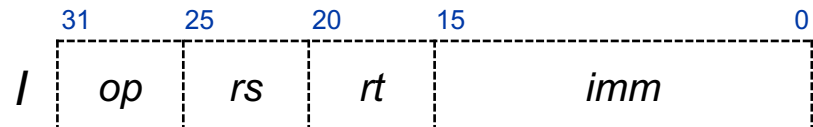
add rd, rs, rt
sub rd, rs, rt



- **Tipo I:** incluyen algún dato inmediato (**immediate**)

addi rt, rs, imm
subi rt, rs, imm

lw rt, imm(rs)
sw rt, imm(rs)



Escribir: (2) los campos de instrucción

```

                                .data
00000 0x0000 num1: .word 11
                                num2: .word 12
                                res:  .word -1
                                .text
lw      rt,imm(rs)  lw      $s1,num1($zero)
                                lw      $s2,num2($zero)
                                beq      $s1,$s2,salto
                                add      $s0,$s1,$s2
                                j         fin
                                salto:addi $s0,$zero,1
                                fin:  sw   $s0,res($zero)

```

<i>op</i>	<i>rs</i>	<i>rt</i>	<i>rd</i>	<i>sh</i>	<i>f</i>
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>imm</i>		
<i>op</i>	<i>addr</i>				
35	0	17	0		

Escribir los campos de instrucción

V

```

                                .data
00000  0x0000  num1: .word    11
                                num2: .word    12
                                res:  .word   -1
                                .text
12288  0x3000      lw      $s1,num1($zero)
                                lw      $s2,num2($zero)
                                beq      $s1,$s2,salto
                                add      $s0,$s1,$s2
                                j         fin
                                salto:addi $s0,$zero,1
                                fin:  sw      $s0,res($zero)
    
```

<i>op</i>	<i>rs</i>	<i>rt</i>	<i>rd</i>	<i>sh</i>	<i>f</i>
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>imm</i>		
<i>op</i>	<i>addr</i>				
35	0	17	0		
¿?	¿?	¿?	¿?		
¿?	¿?	¿?	¿?		

Escribir los campos de instrucción

```

                                .data
00000 0x0000 num1: .word 11
00004 0x0004 num2: .word 12
00008 0x0008 res:  .word -1

                                .text
12288 0x3000      lw      $s1,num1($zero)
12292 0x3004      lw      $s2,num2($zero)
12296 0x3008      beq     $s1,$s2,salto
12300 0x300C      add     $s0,$s1,$s2
12304 0x3010      j       fin
12308 0x3014      salto:addi $s0,$zero,1
12312 0x3018      fin:   sw      $s0,res($zero)

```

<i>op</i>	<i>rs</i>	<i>rt</i>	<i>rd</i>	<i>sh</i>	<i>f</i>
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>imm</i>		
<i>op</i>	<i>addr</i>				
35	0	17	0		

Convertir a código máquina

V

add \$s0,\$s1,\$s2

tipo-R	op(6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
Decim	0	17	18	16	0	32
Bin	0000 00	10 001	1 0010	1000 0	000 00	10 0000
Hex	0x02328020					

lw \$s2,num2(\$zero)

tipo-I	op (6)	rs (5)	rt (5)	imm/offset (16)
Decim				
Bin				
Hex				

addi \$s0,\$zero,1

tipo-I	op (6)	rs (5)	rt (5)	imm/offset (16)
Decim				
Bin				
Hex				

Tema 1: Representación hexadecimal

- Expresar de forma más compacta **datos y direcciones** de memoria
- Símbolos con valores de 0 a 15
 - 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- Muy fácil pasar de binario a hexadecimal
 - Grupo de 4 bits → Un símbolo hexadecimal
 - Palabra de 32 bits → 8 símbolos hexadecimal

0000	0010	0011	1101	1000	0000	1111	0010
↓	↓	↓	↓	↓	↓	↓	↓
0	2	3	D	8	0	F	2

0x023D80F2

Convertir a código máquina

add \$s0,\$s1,\$s2

tipo-R	op(6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
Decim	0	17	18	16	0	32
Bin	0000 00	10 001	1 0010	1000 0	000 00	10 0000
Hex	0x02328020					

lw \$s2,num2(\$zero)

tipo-I	op (6)	rs (5)	rt (5)	imm/offset (16)
Decim				
Bin				
Hex				

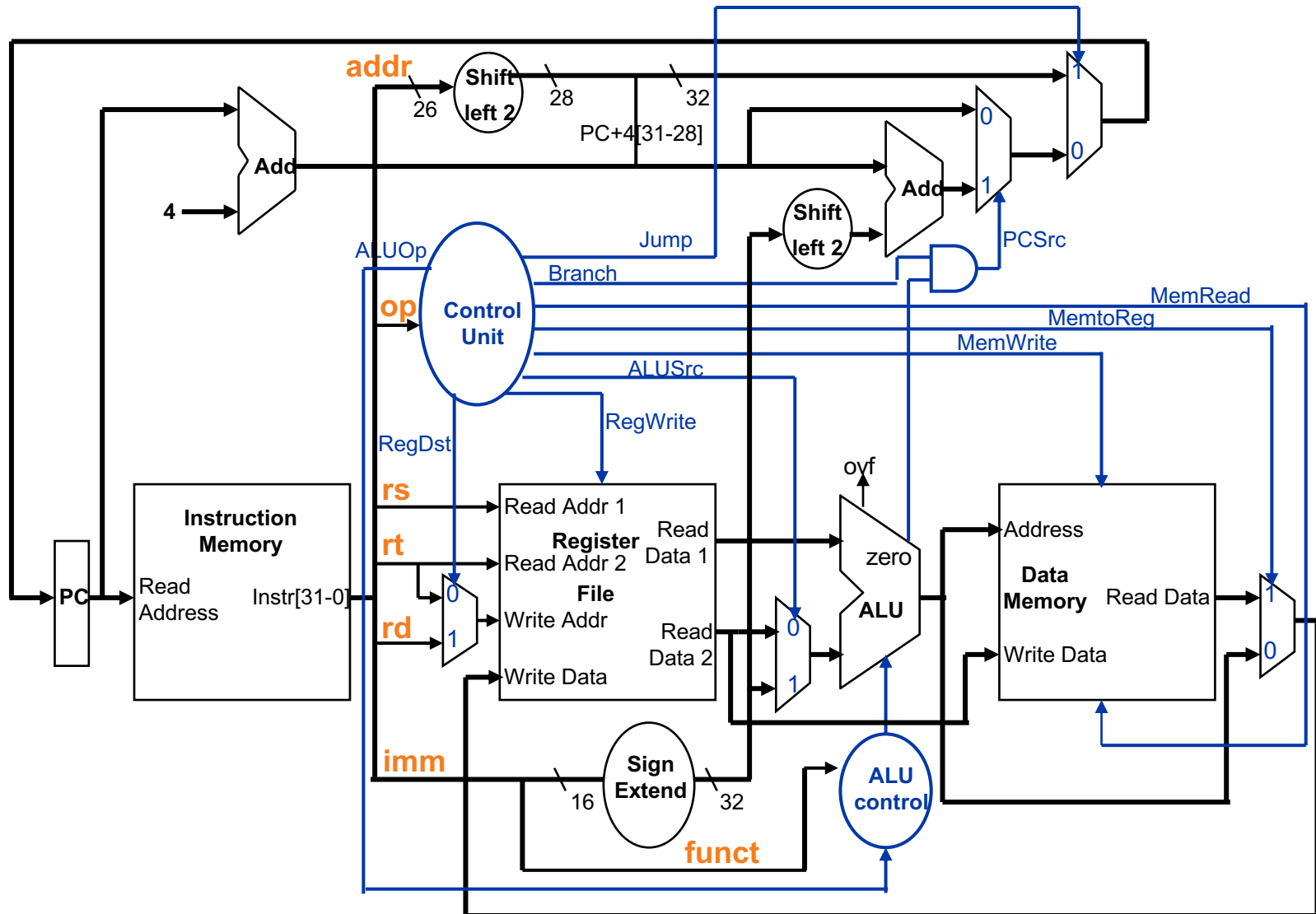
addi \$s0,\$zero,1

tipo-I	op (6)	rs (5)	rt (5)	imm/offset (16)
Decim				
Bin				
Hex				

Contenidos

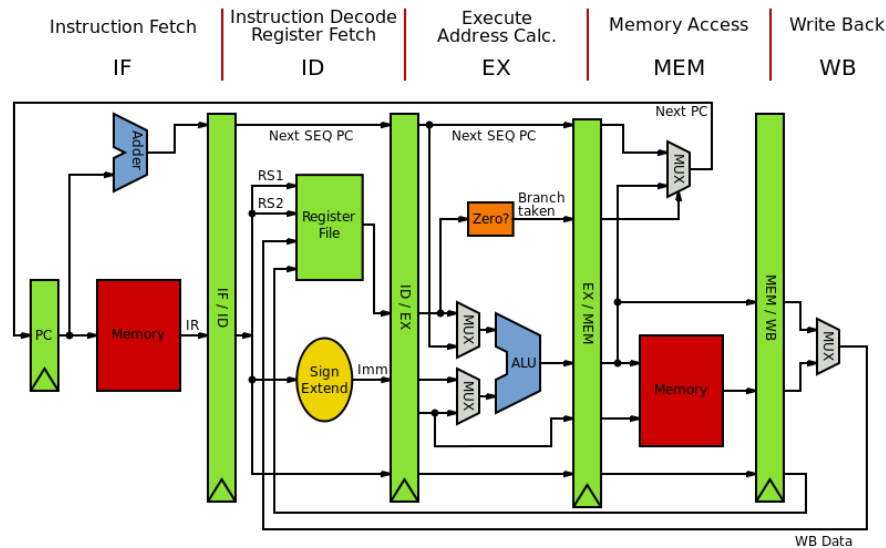
- Repaso de conocimientos previos
- **Instrucciones aritméticas y de carga/almacenamiento**
 - Codificación de las instrucciones
 - Construcción del camino de datos
- Instrucciones de salto condicional/no condicional
 - Codificación de las instrucciones
 - Construcción del camino de datos
- Señales de control según instrucción
- Conclusiones y bibliografía

Camino de Datos y Unidad de Control



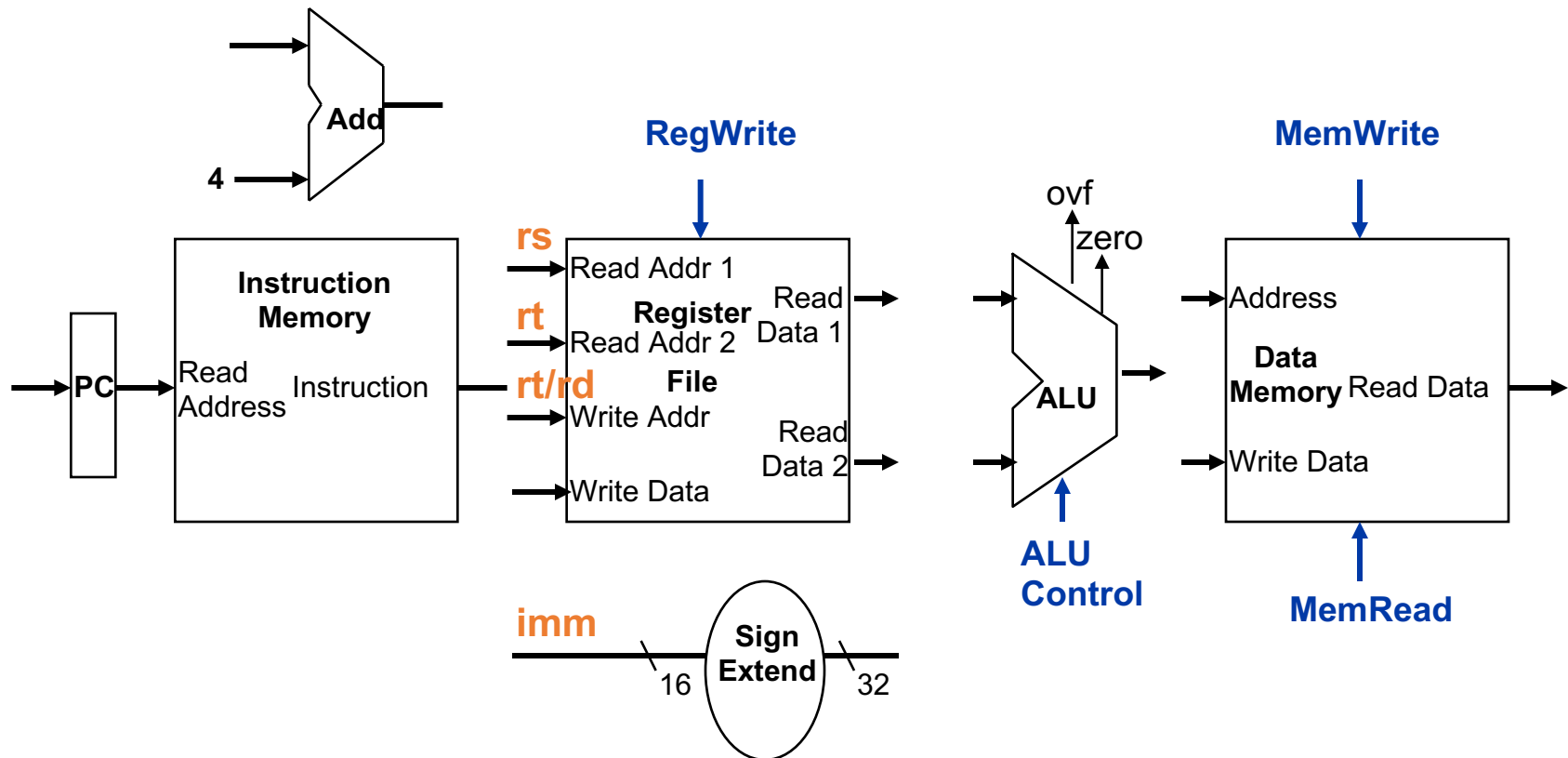
Camino de Datos y Unidad de Control

- Etapas para ejecutar una instrucción
 - (1) Leer la instrucción
 - (2) Decodificar la instrucción y acceder registros
 - (3) Ejecutar la instrucción o calcular dirección
 - (4) Leer/escribir dirección de memoria
 - (5) Escribir en fichero de registros



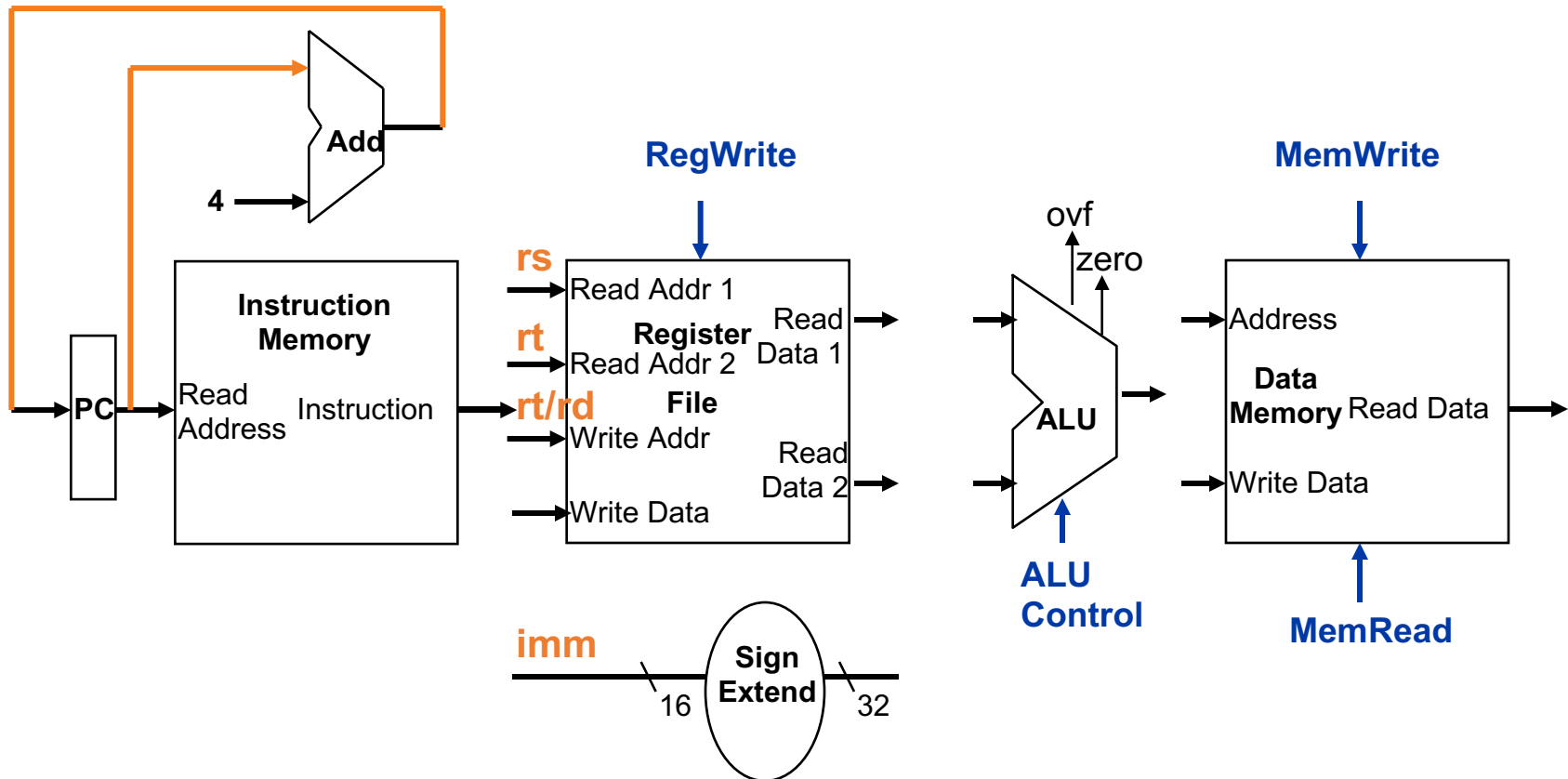
Camino de Datos y Unidad de Control

- Interconectaremos los elementos del camino de datos mediante **buses de datos**
- Y los gobernaremos con **señales de control**



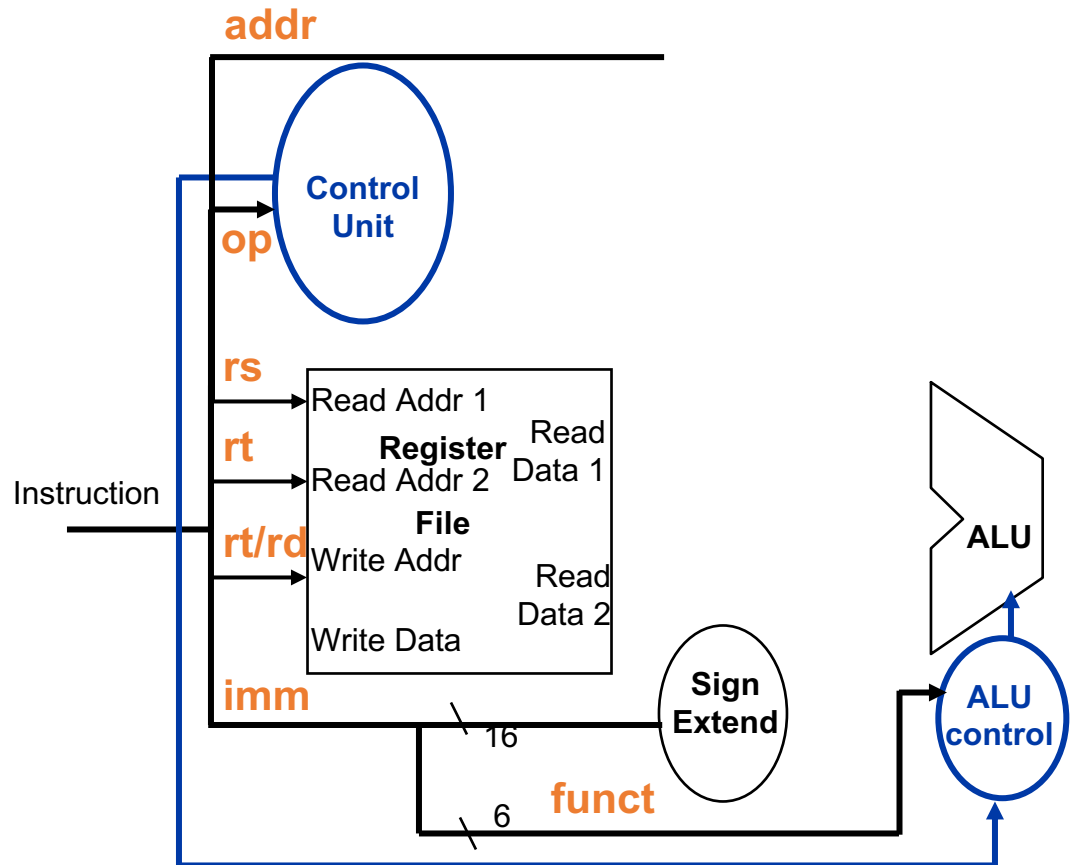
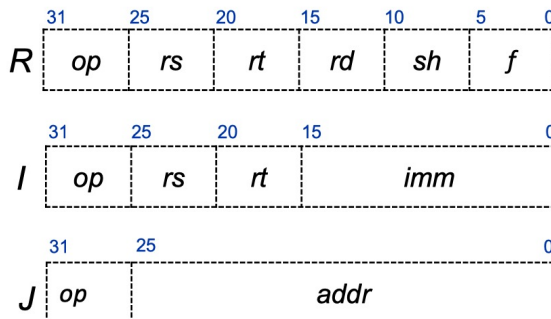
(1) Leer la instrucción

- PC: registro contador de programa (\$pc)
 - Acceso a la memoria de instrucciones
 - Siguiete instrucción: PC + 4 si ejecución secuencial



(2) Decodificar la instrucción y acceder registros

- Interpretar campos **opcode** y **funct** (**Unidad de Control**)
- Leer valores almacenados en registros **rs**, **rt** y enviar por el bus valores en **imm** o **addr**.



(2) Decodificar la instrucción y acceder registros: add

```
                                .data
00000  0x0000  num1: .word    11
                                num2: .word    12
                                res:  .word   -1
                                .text
12288  0x3000          lw      $s1,num1($zero)
                                lw      $s2,num2($zero)
                                beq     $s1,$s2,salto
add    rd,rs,rt  add      $s0,$s1,$s2
                                j        fin
                                salto:addi $s0,$zero,1
                                fin:  sw      $s0,res($zero)
```

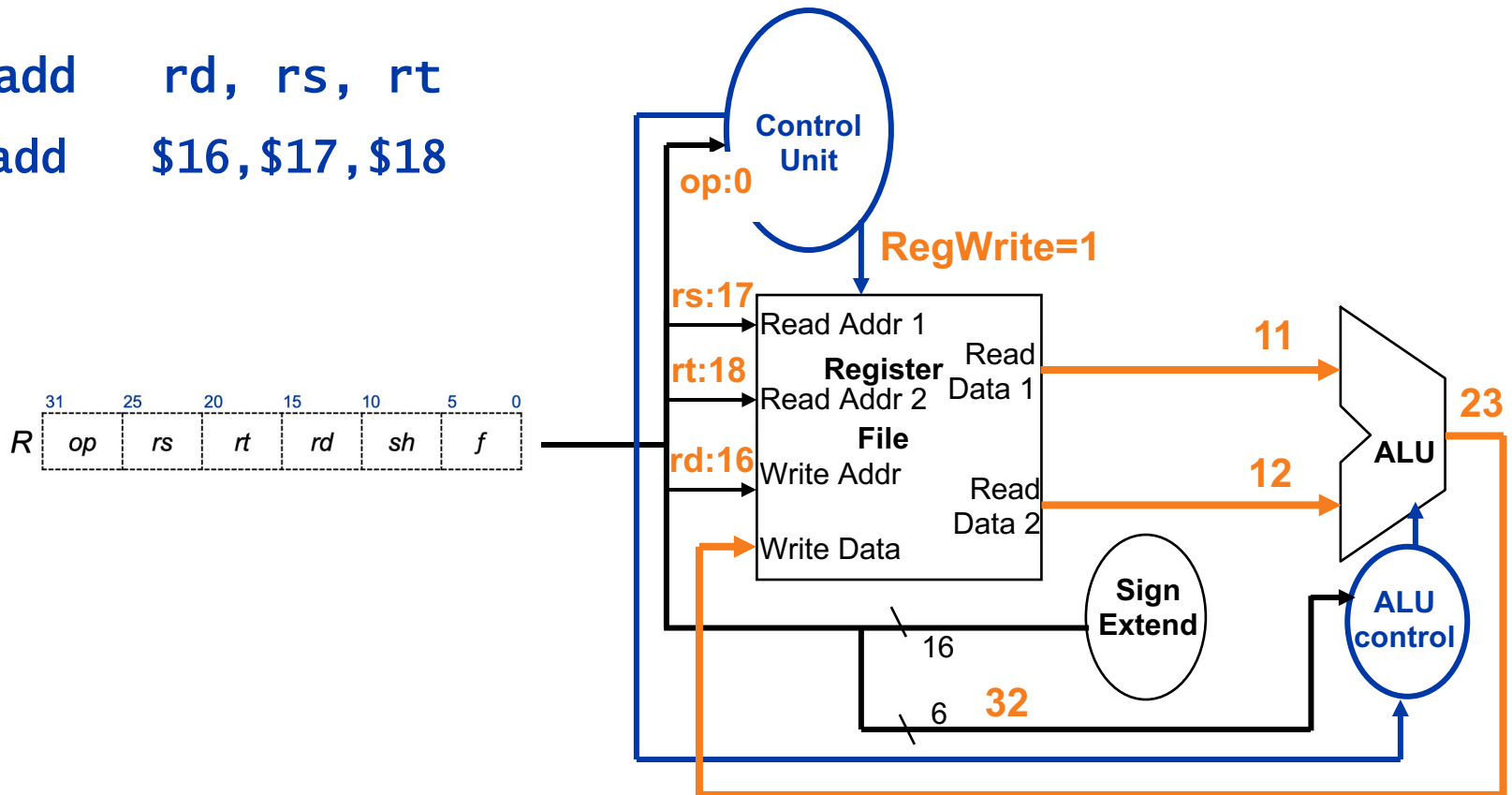
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>rd</i>	<i>sh</i>	<i>f</i>
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>imm</i>		
<i>op</i>	<i>addr</i>				
0	17	18	16	0	32

(3) Ejecutar la instrucción + (4) Escribir en registro: add

- Una vez decodificada la instrucción, ¿cómo conectamos los buses? ¿Qué valores viajan por ellos? ¿Qué valor debe de tomar la señal de control **RegWrite**?

add rd, rs, rt

add \$16,\$17,\$18



(3) Ejecutar la instrucción + (4) Escribir en registro: addi

v

```

                                .data
00000  0x0000  num1: .word    11
                                num2: .word    12
                                res:  .word   -1
                                .text
12288  0x3000          lw      $s1,num1($zero)
                                lw      $s2,num2($zero)
                                beq      $s1,$s2,salto
                                add      $s0,$s1,$s2
                                j         fin
addi    rt,rs,imm → addi    $s0,$zero,1
                                fin:    sw      $s0,res($zero)

```

<i>op</i>	<i>rs</i>	<i>rt</i>	<i>rd</i>	<i>sh</i>	<i>f</i>
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>imm</i>		
<i>op</i>	<i>addr</i>				

Escribir los campos de instrucción

```

                                .data
00000  0x0000  num1: .word    11
                                num2: .word    12
                                res:  .word   -1
                                .text
12288  0x3000          lw      $s1,num1($zero)
                                lw      $s2,num2($zero)
                                beq      $s1,$s2,salto
                                add      $s0,$s1,$s2
                                j         fin
addi    rt,rs,imm  addi      $s0,$zero,1
                                fin:    sw      $s0,res($zero)

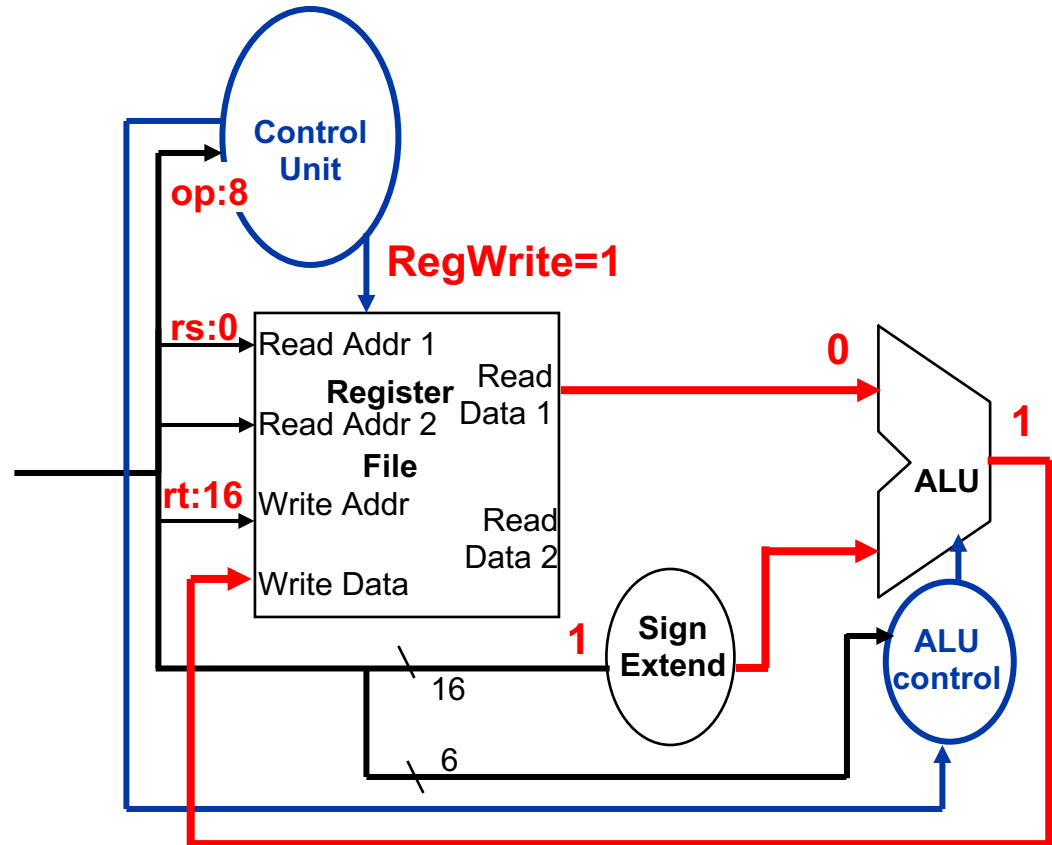
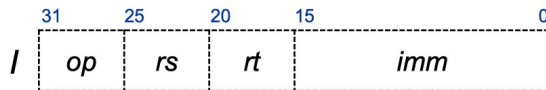
```

op	rs	rt	rd	sh	f
op	rs	rt	imm		
op	addr				
0	17	18	16	0	32
¿?	¿?	¿?	¿?		

(3) Ejecutar la instrucción + (4) Escribir en registro: addi

- Para estas instrucciones la ALU opera con el dato inmediato

`addi rt, rs, imm`
`addi $16, $0, 1`



Camino de los datos según instrucción

v

- ¿Cuál es el registro que se direcciona a la entrada de **WriteAddr** en cada uno de los casos anteriores?

`add rd, rs, rt`

`addi rt, rs, imm`

- ¿Cuál es el **segundo operando** que entra a la ALU en cada caso?
- ¿Cómo podemos **contemplar todos los casos** en el conexionado?

Camino de los datos según instrucción

- ¿Cuál es el registro que se direcciona a la entrada de **WriteAddr** en cada uno de los casos anteriores?

`add rd, rs, rt`
`addi rt, rs, imm`

- ¿Cuál es el **segundo operando** que entra a la ALU en cada caso?

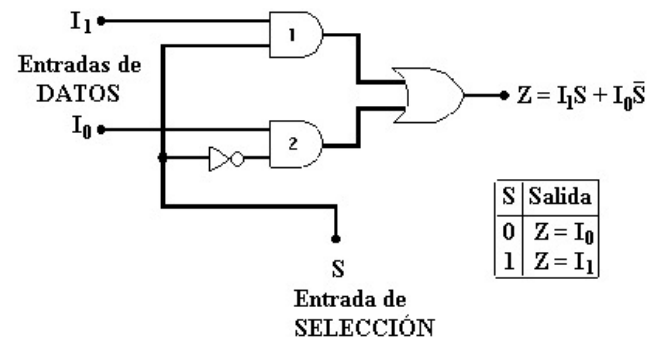
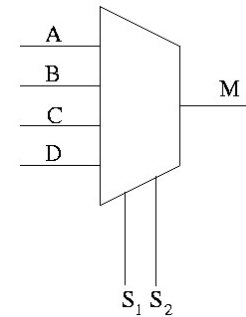
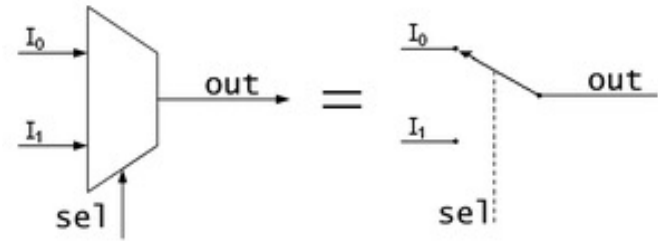
`add rd, rs, rt`
`addi rt, rs, imm`

- ¿Cómo podemos contemplar todos los casos en el conexionado?

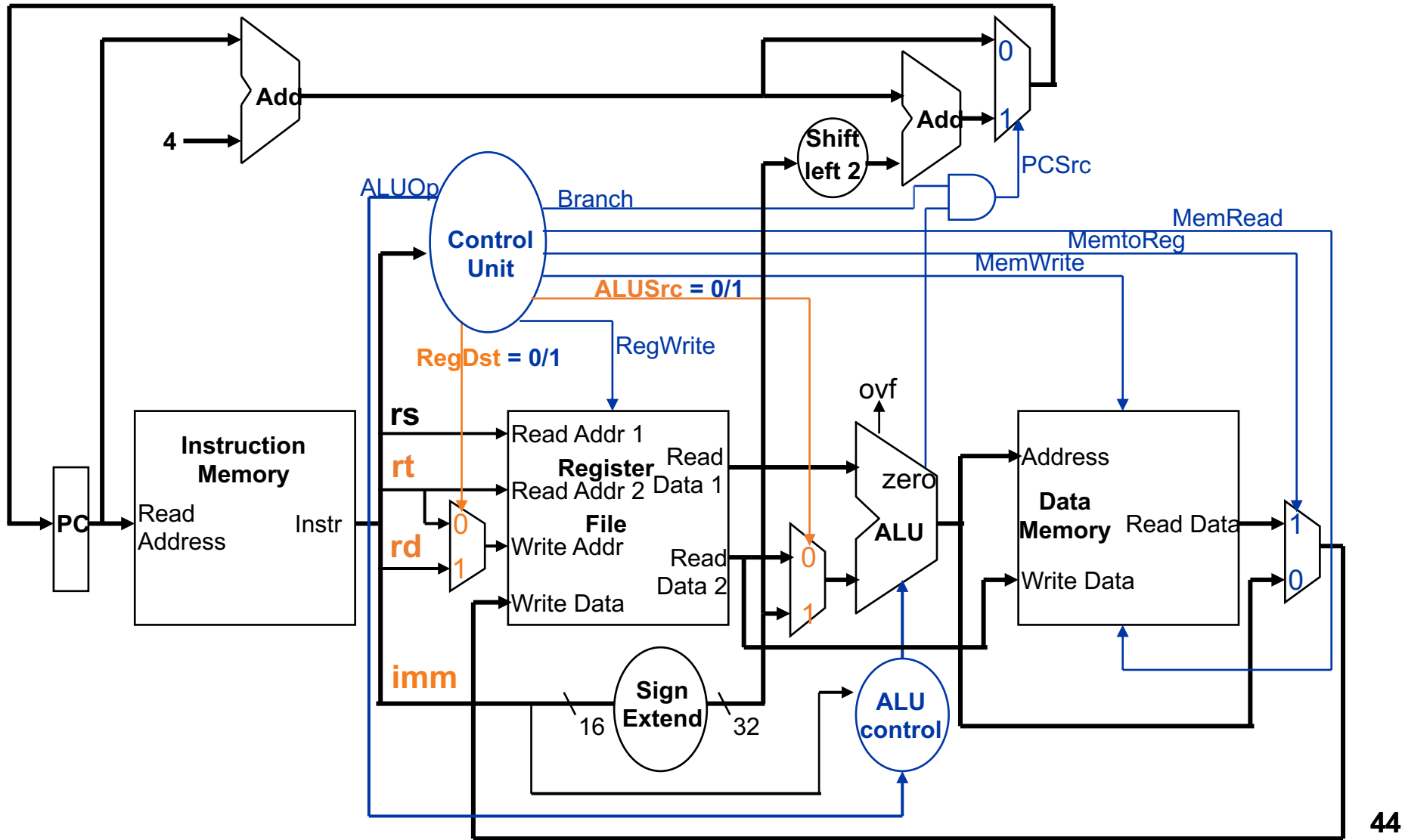
Usando **multiplexores**

Utilización de multiplexores

- ¿Cuántos bits de selección se necesitan según el número de entradas?
 - En esta asignatura nos interesa como bloque funcional.
 - En otras asignaturas veréis funcionamiento con puertas lógicas.



Utilización de multiplexores



Escribir: (2) los campos de instrucción

```

                                .data
00000 0x0000 num1: .word 11
                                num2: .word 12
                                res:  .word -1
                                .text
lw      rt,imm(rs)  lw      $s1,num1($zero)
                                lw      $s2,num2($zero)
                                beq      $s1,$s2,salto
                                add      $s0,$s1,$s2
                                j         fin
                                salto:addi $s0,$zero,1
                                fin:  sw   $s0,res($zero)

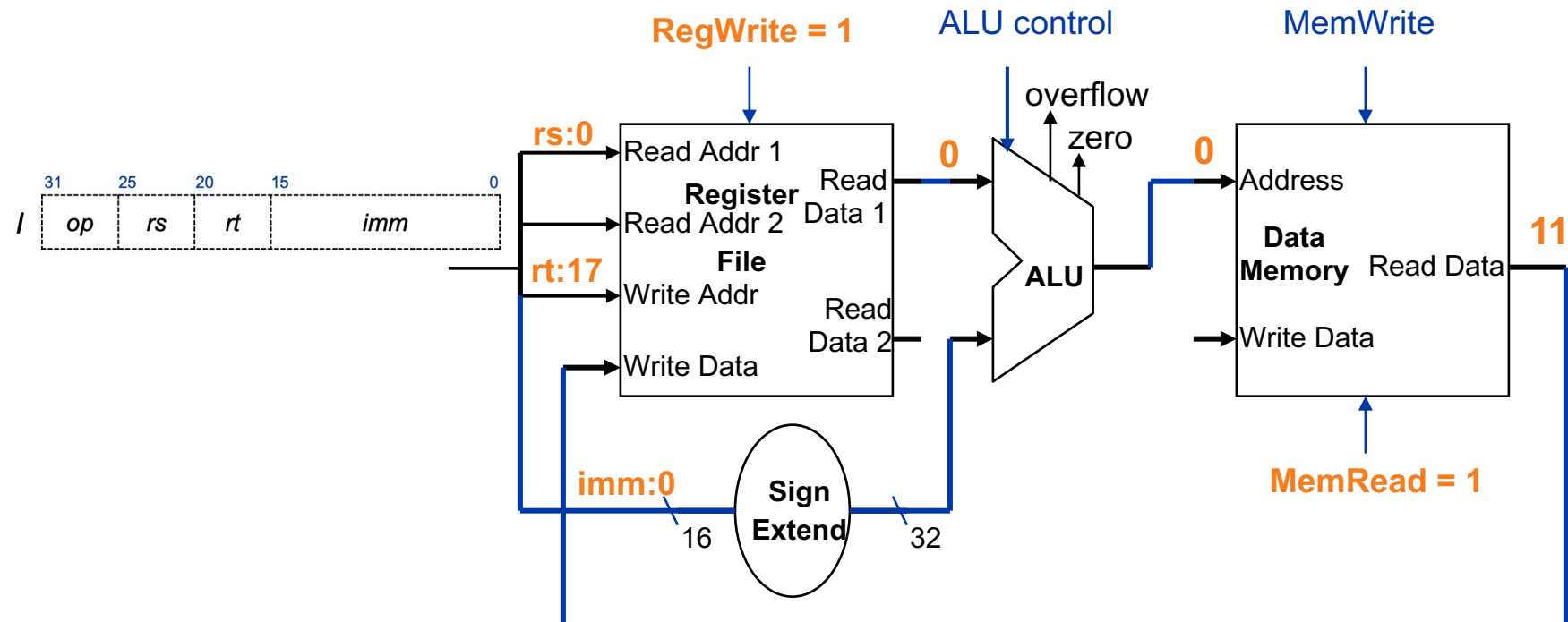
```

<i>op</i>	<i>rs</i>	<i>rt</i>	<i>rd</i>	<i>sh</i>	<i>f</i>
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>imm</i>		
<i>op</i>	<i>addr</i>				
35	0	17	0		

(3) Calcular dirección + (4) Leer memoria + (5) Escribir en registro: lw

- La ALU calcula la dirección de memoria a acceder

```
lw    rt, imm(rs)
lw    $17, 0($0)
```



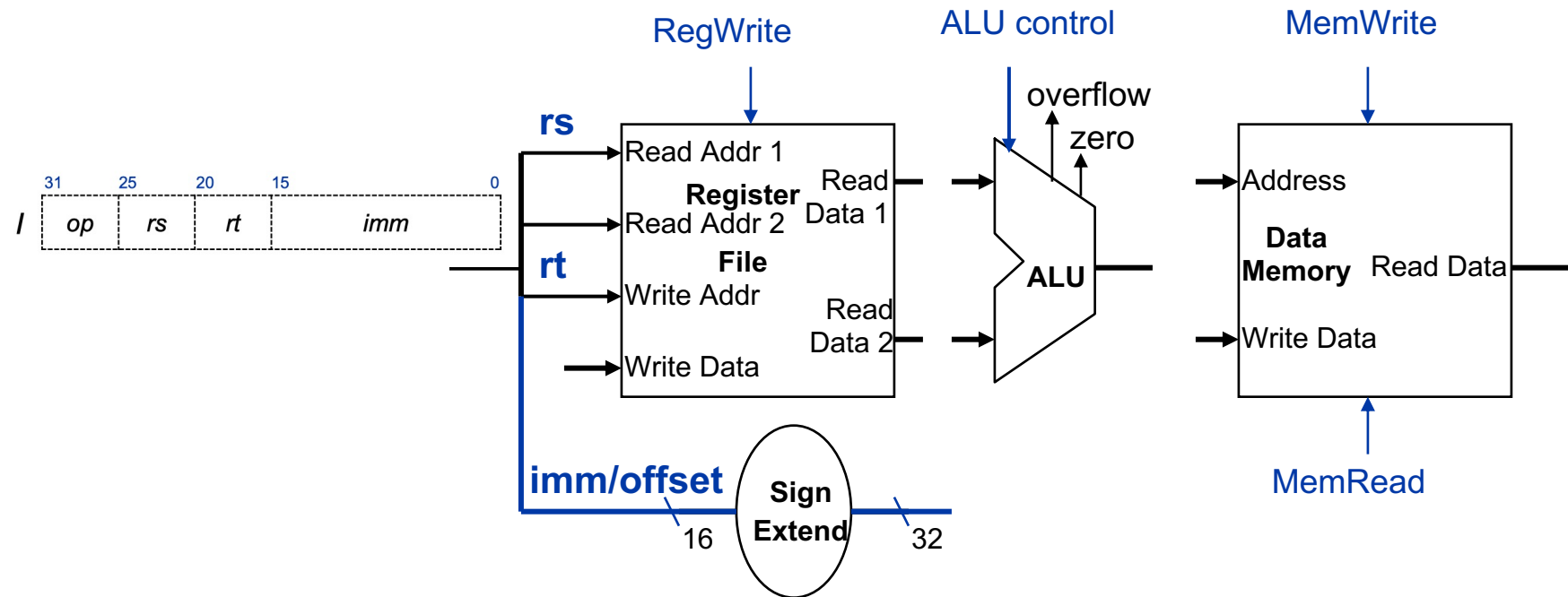
(3) Calcular dirección + (4) Escribir memoria: sw

v

- ¿Cómo queda el conexionado para la instrucción sw?

sw rt, imm(rs)

sw \$s0, res(\$zero)



Escribir los campos de instrucción

```

                                .data
00000  0x0000  num1: .word    11
                                num2: .word    12
                                res:  .word   -1
                                .text
12288  0x3000      lw      $s1,num1($zero)
                                lw      $s2,num2($zero)
                                beq     $s1,$s2,salto
                                add     $s0,$s1,$s2
                                j        fin
                                salto:addi $s0,$zero,1
                                fin:  sw     $s0,res($zero)

```

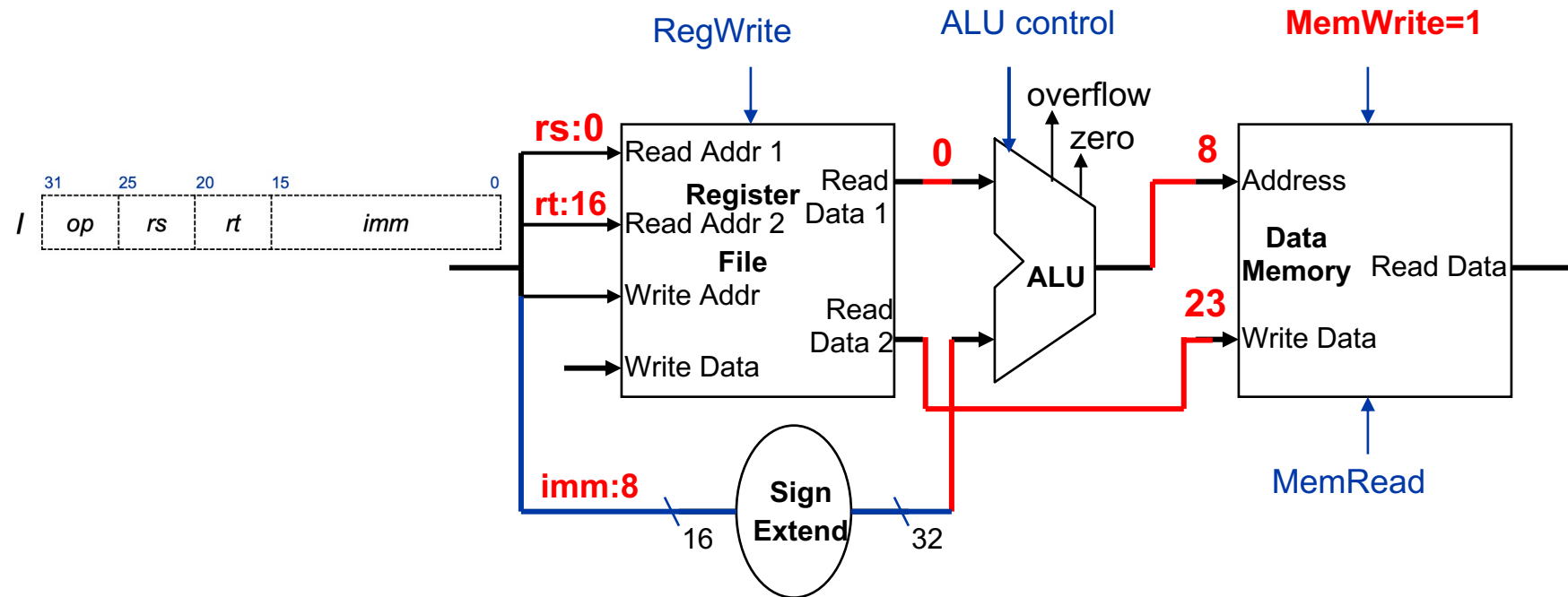
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>rd</i>	<i>sh</i>	<i>f</i>
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>imm</i>		
<i>op</i>	<i>addr</i>				
<i>¿?</i>	<i>¿?</i>	<i>¿?</i>	<i>¿?</i>		

(3) Calcular dirección + (4) Escribir memoria: sw

- ¿Cómo queda el conexionado para la instrucción sw?

sw rt, imm(rs)

sw \$16,8(\$0)



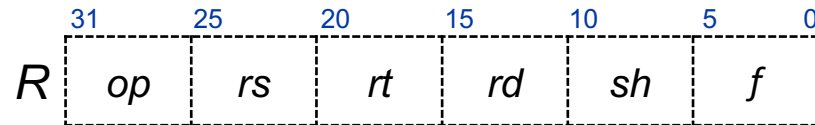
Contenidos

- Repaso de conocimientos previos
- Instrucciones aritméticas y de carga/almacenamiento
 - Codificación de las instrucciones
 - Construcción del camino de datos
- **Instrucciones de salto condicional/no condicional**
 - **Codificación de las instrucciones**
 - Construcción del camino de datos
- Señales de control según instrucción
- Conclusiones y bibliografía

Contenido de los campos de instrucción según su tipo

- **Tipo R:** todos los operandos en registros (**r**)

add rd, rs, rt
sub rd, rs, rt

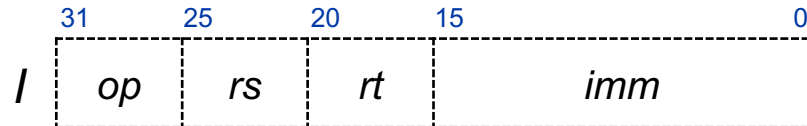


- **Tipo I:** incluyen algún dato inmediato (**immediate**)

addi rt, rs, imm
subi rt, rs, imm

lw rt, imm(rs)
sw rt, imm(rs)

beq rs, rt, imm
bne rs, rt, imm



Escribir los campos de instrucción

V

```

                                .data
00000  0x0000  num1: .word    11
                                num2: .word    12
                                res:  .word    0
                                .text
12288  0x3000          lw      $s1,num1($zero)
                                lw      $s2,num2($zero)
                                beq      $s1,$s2,salto
                                add      $s0,$s1,$s2
                                j         fin
                                salto:addi $s0,$s0,1
                                fin:  sw      $s0,res($zero)
    
```

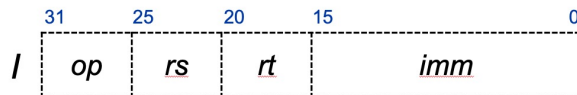
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>rd</i>	<i>sh</i>	<i>f</i>
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>imm</i>		
<i>op</i>	<i>addr</i>				
¿?	¿?	¿?	¿?		

Campos de instrucción para instrucciones de salto tipo I

- Instrucción de salto tipo I

beq *rs,rt,imm*

bne *rs,rt,imm*



- Dato **inmediato** indica el desplazamiento en memoria de instrucciones necesario hasta la etiqueta objetivo.
 - Expresado como **número de instrucciones**, a partir de la siguiente instrucción
 - Es un **salto relativo** al contador de programa

Escribir los campos de instrucción

```

                                .data
00000 0x0000 num1: .word 11
                                num2: .word 12
                                res:  .word 0
                                .text
                                lw      $s1,num1($zero)
                                lw      $s2,num2($zero)
12296 0x3008 beq      $s1,$s2,salto
12300 0x300C add      $s0,$s1,$s2
                                j      fin
12308 0x3014 salto:addi $s0,$s0,1
                                fin:  sw      $s0,res($zero)
    
```

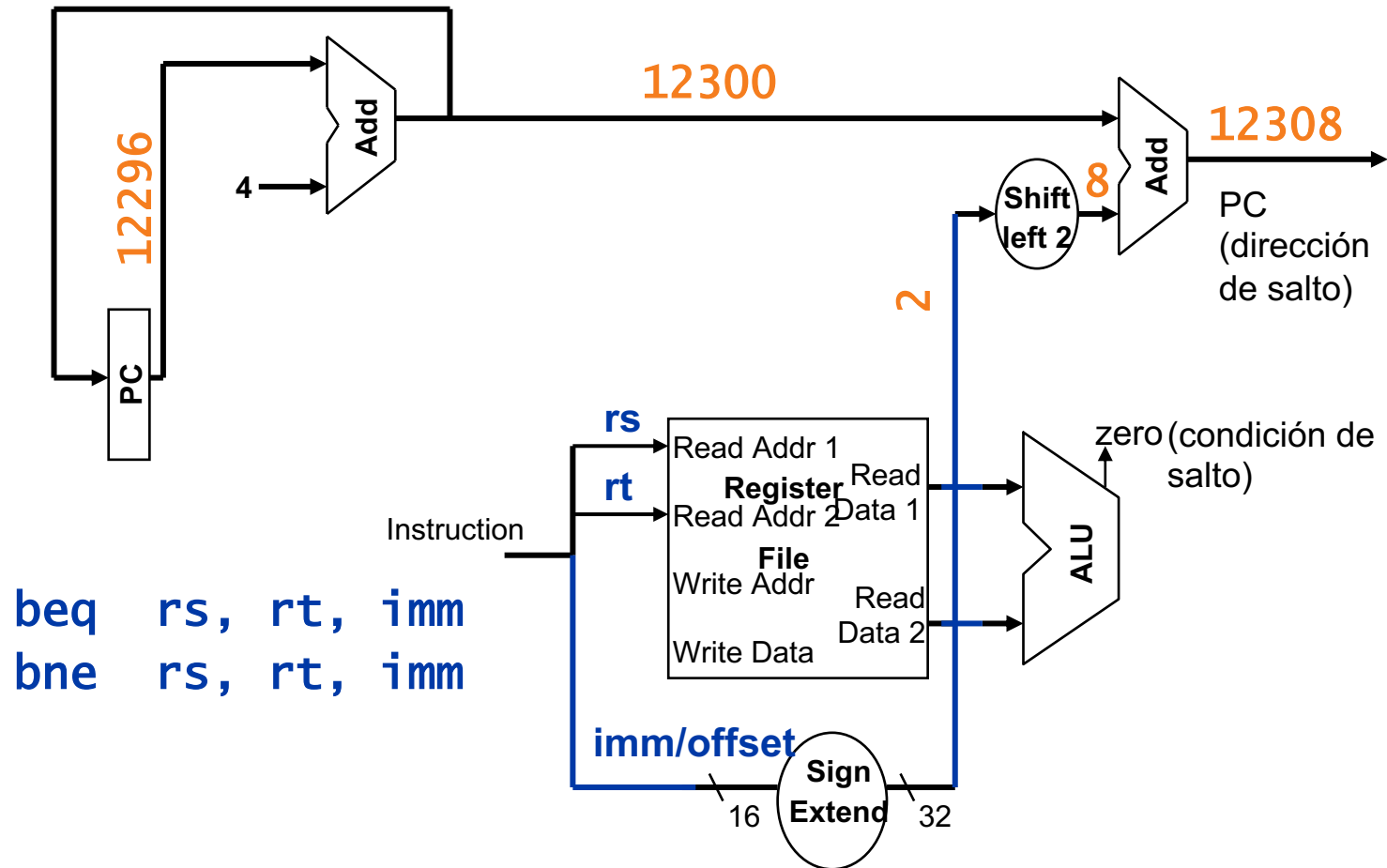
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>rd</i>	<i>sh</i>	<i>f</i>
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>imm</i>		
<i>op</i>	<i>addr</i>				
4	17	18	2		

Contenidos

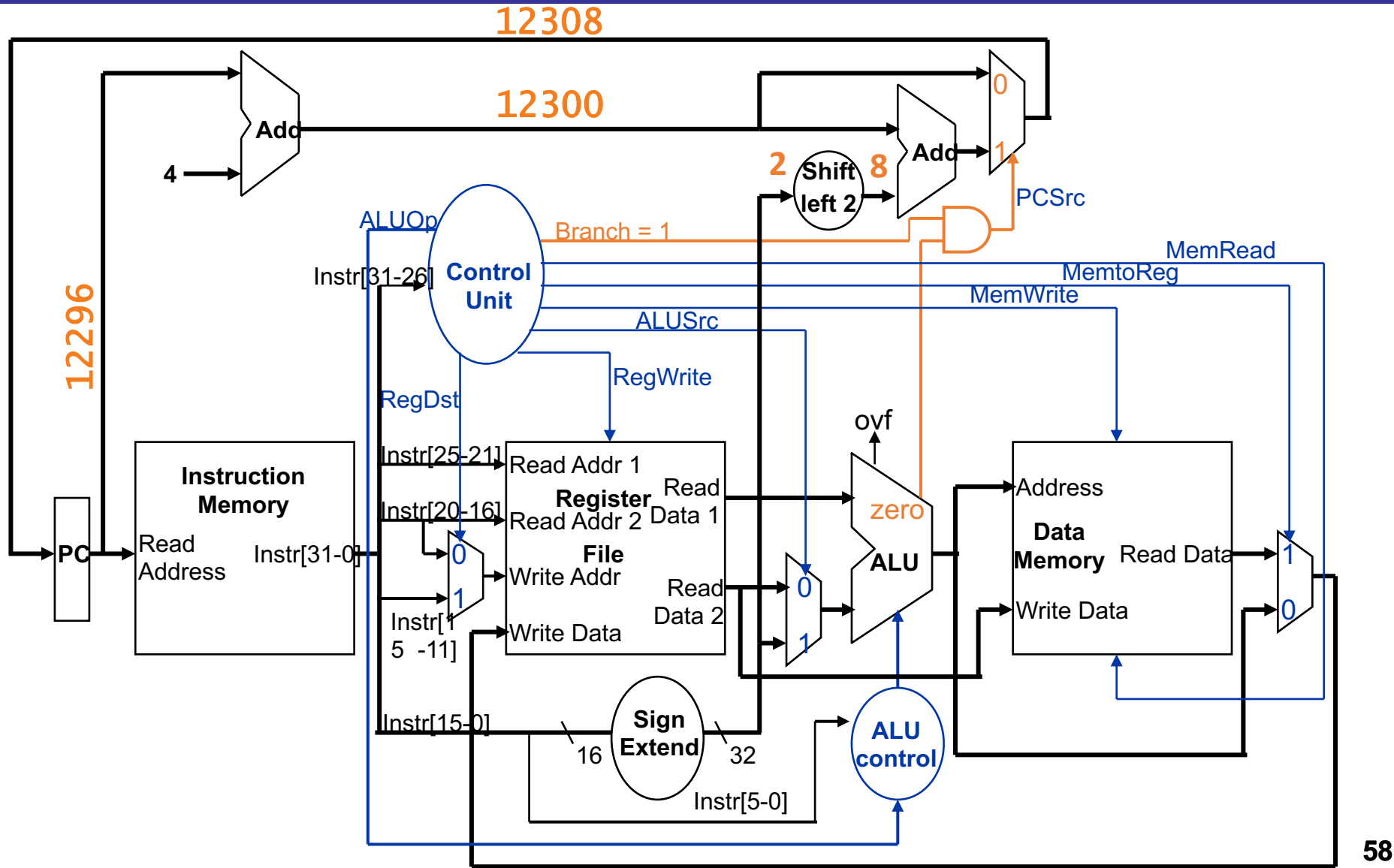
- Repaso de conocimientos previos
- **Instrucciones aritméticas y de carga/almacenamiento**
 - Codificación de las instrucciones
 - Construcción del camino de datos
- **Instrucciones de salto condicional/no condicional**
 - Codificación de las instrucciones
 - Construcción del camino de datos
- Señales de control según instrucción
- Conclusiones y bibliografía

(3) Ejecutar la instrucción: beq

- Actualizar PC con la dirección de salto
- La ALU compara valores



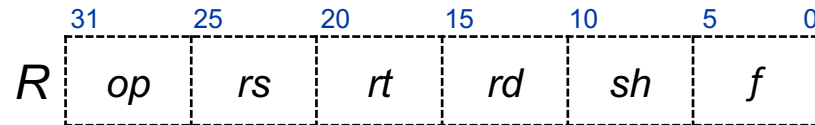
(3) Ejecutar la instrucción: beq



Contenido de los campos de instrucción según su tipo

- **Tipo R:** todos los operandos en registros (**r**)

add rd, rs, rt
sub rd, rs, rt

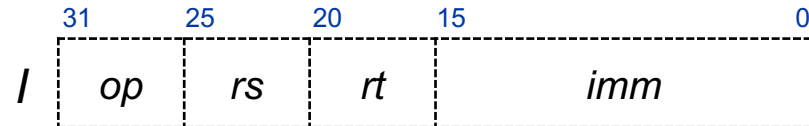


- **Tipo I:** incluyen algún dato inmediato (**immediate**)

addi rt, rs, imm
subi rt, rs, imm

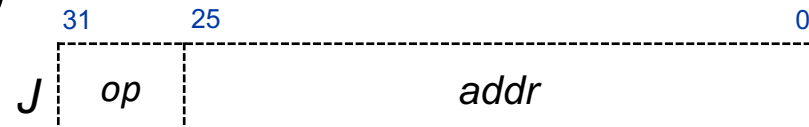
lw rt, imm(rs)
sw rt, imm(rs)

beq rs, rt, imm
bne rs, rt, imm



- **Tipo J:** para salto (**jump**)

j addr



Escribir los campos de instrucción

V

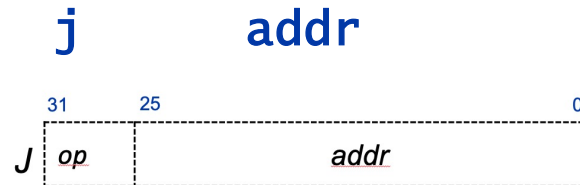
```

                                .data
00000  0x0000  num1: .word    11
                                num2: .word    12
                                res:  .word    0
                                .text
12288  0x3000          lw      $s1,num1($zero)
                                lw      $s2,num2($zero)
                                beq      $s1,$s2,salto
                                add      $s0,$s1,$s2
                                j         fin
                                salto:addi $s0,$s0,1
                                fin:  sw      $s0,res($zero)
    
```

<i>op</i>	<i>rs</i>	<i>rt</i>	<i>rd</i>	<i>sh</i>	<i>f</i>
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>imm</i>		
<i>op</i>	<i>addr</i>				
¿?	¿?				

Campos de instrucción para instrucciones de salto tipo J

- Instrucción de salto tipo J



- Campo **address** indica la dirección de la etiqueta objetivo dividida entre 4.
 - Es un **salto pseudoabsoluto** en memoria.

Escribir los campos de instrucción

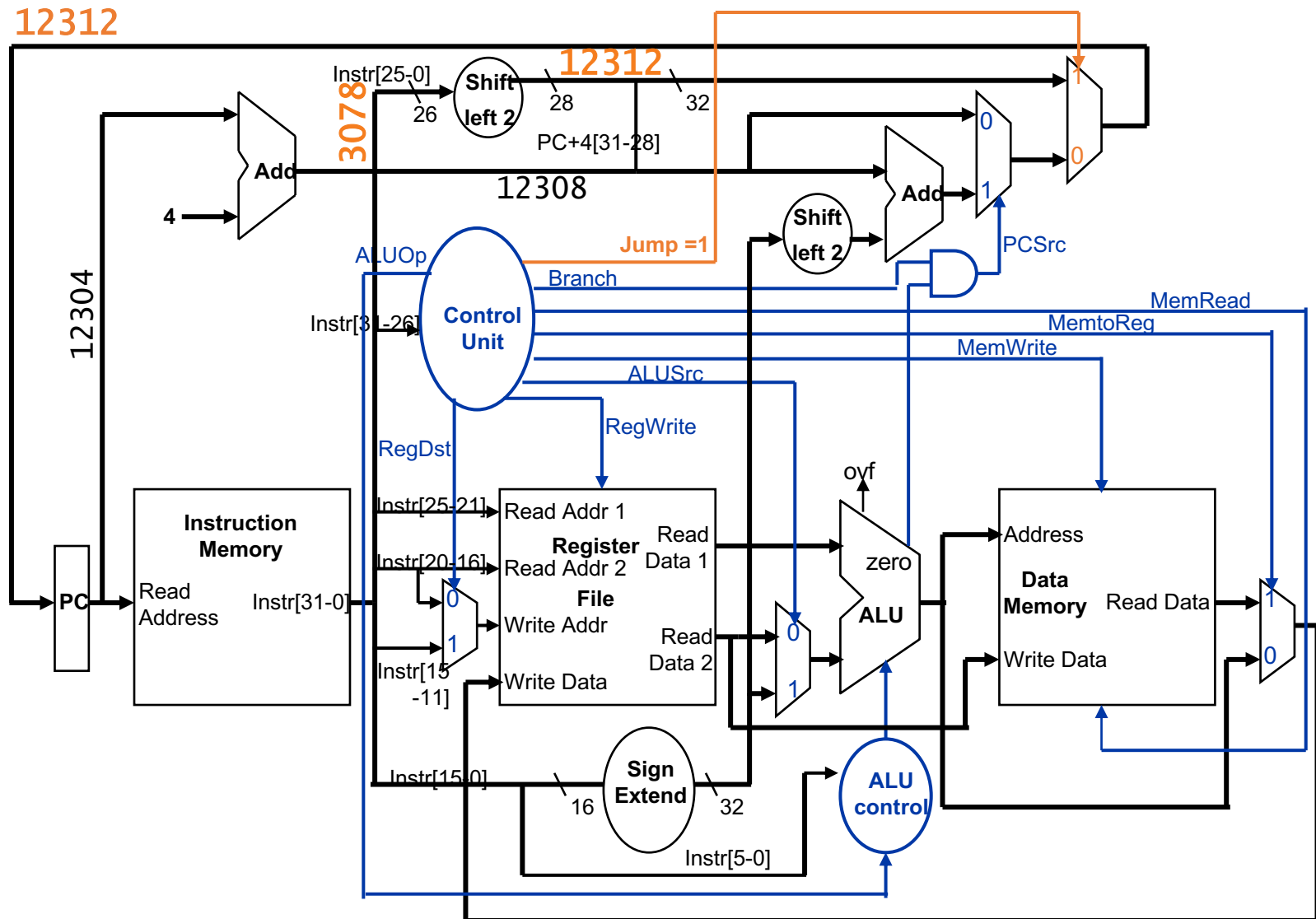
```

                                .data
00000 0x0000 num1: .word 11
00004 0x0004 num2: .word 12
00008 0x0008 res:  .word 0

                                .text
12288 0x3000      lw      $s1,num1($zero)
12292 0x3004      lw      $s2,num2($zero)
12296 0x3008      beq     $s1,$s2,salto
12300 0x300C      add     $s0,$s1,$s2
12304 0x3010      j       fin
12308 0x3014      salto:addi $s0,$s0,1
12312 0x3018      fin:    sw      $s0,res($zero)
  
```

op	rs	rt	rd	sh	f
op	rs	rt	imm/addr		
op	addr				
¿?	¿?				

(3) Ejecutar la instrucción: j



Contenidos

- Repaso de conocimientos previos
- Instrucciones aritméticas y de carga/almacenamiento
 - Codificación de las instrucciones
 - Construcción del camino de datos
- Instrucciones de salto condicional/no condicional
 - Codificación de las instrucciones
 - Construcción del camino de datos
- Señales de control según instrucción
- Conclusiones y bibliografía

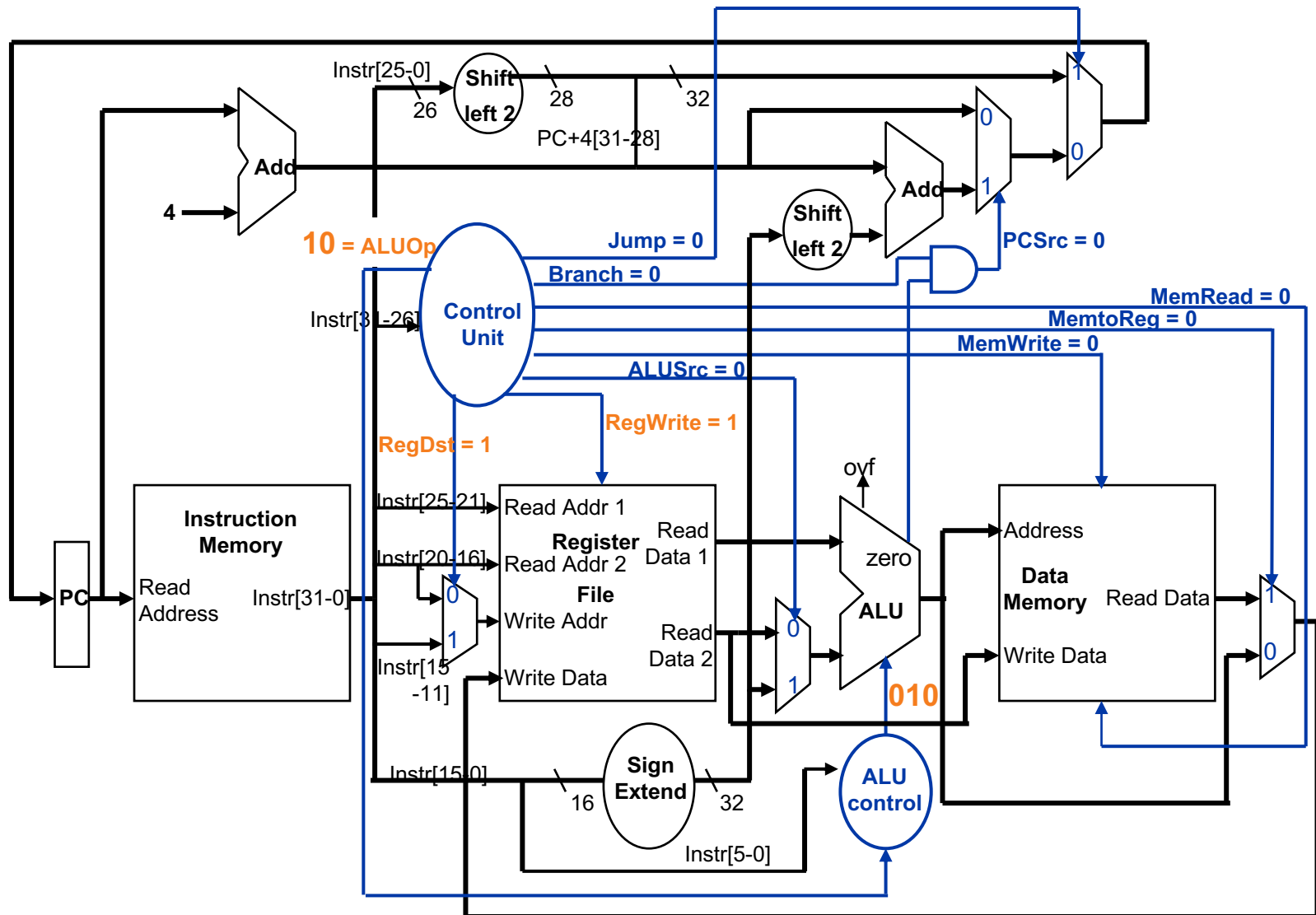
Piensa en el valor necesario para las señales de la Unidad de Control

R

- Pensar en **valor necesario en líneas de control (0/1)**, para las siguientes instrucciones
 - Instrucciones tipo-R
 - Ej. Instrucción **add rd, rs, rt**
 - Instrucción de carga/almacenamiento tipo-I
 - Ej. Instrucción **lw rt, imm(rs)**
 - Instrucción de salto condicional tipo-I
 - Ej. Instrucción **beq rs, rt, imm**
 - Instrucción de salto no condicional tipo-J
 - Ej. Instrucción **j addr**

Unidad de control: add rd, rs, rt

V

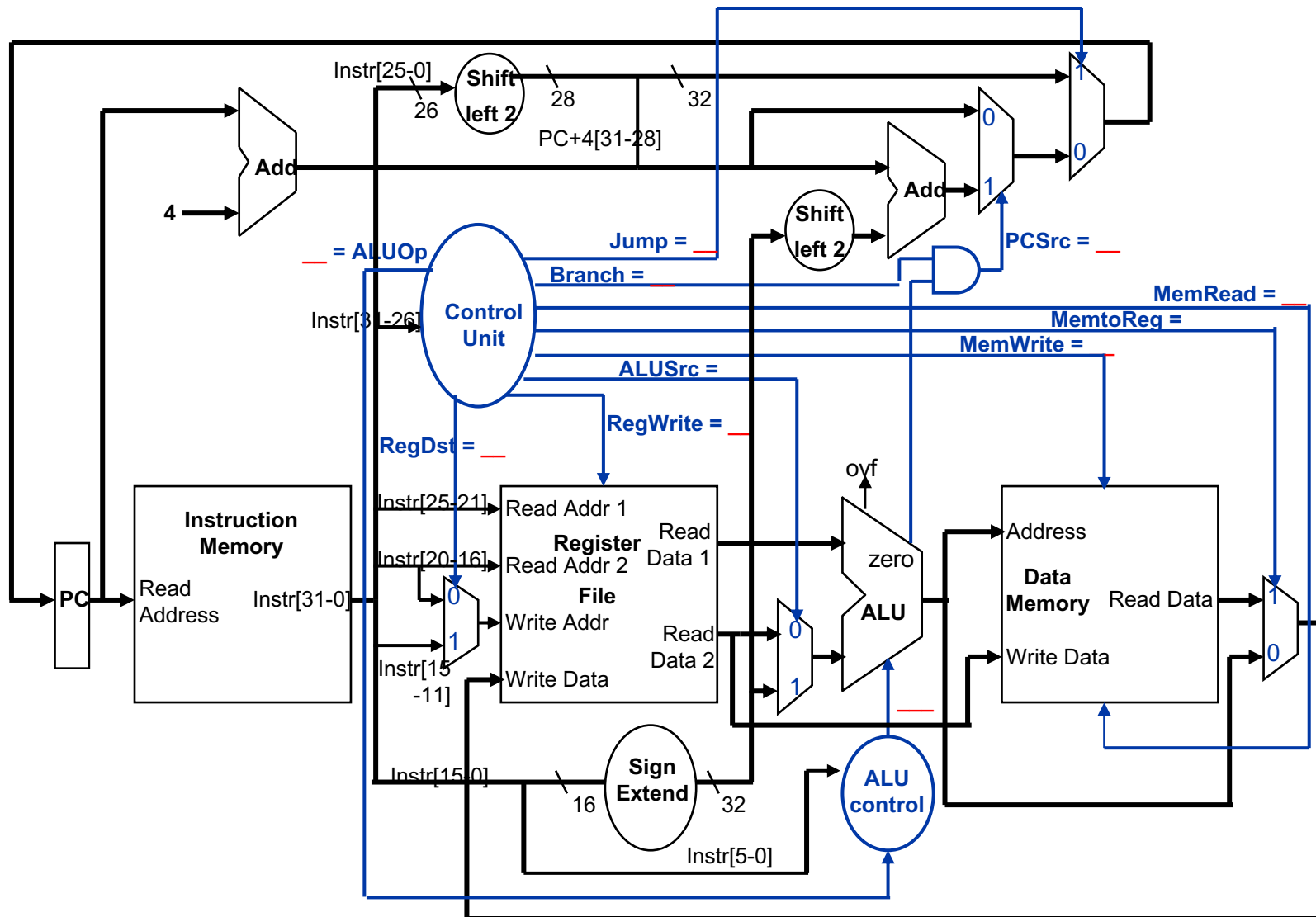


Valores ALU Op y ALU Control

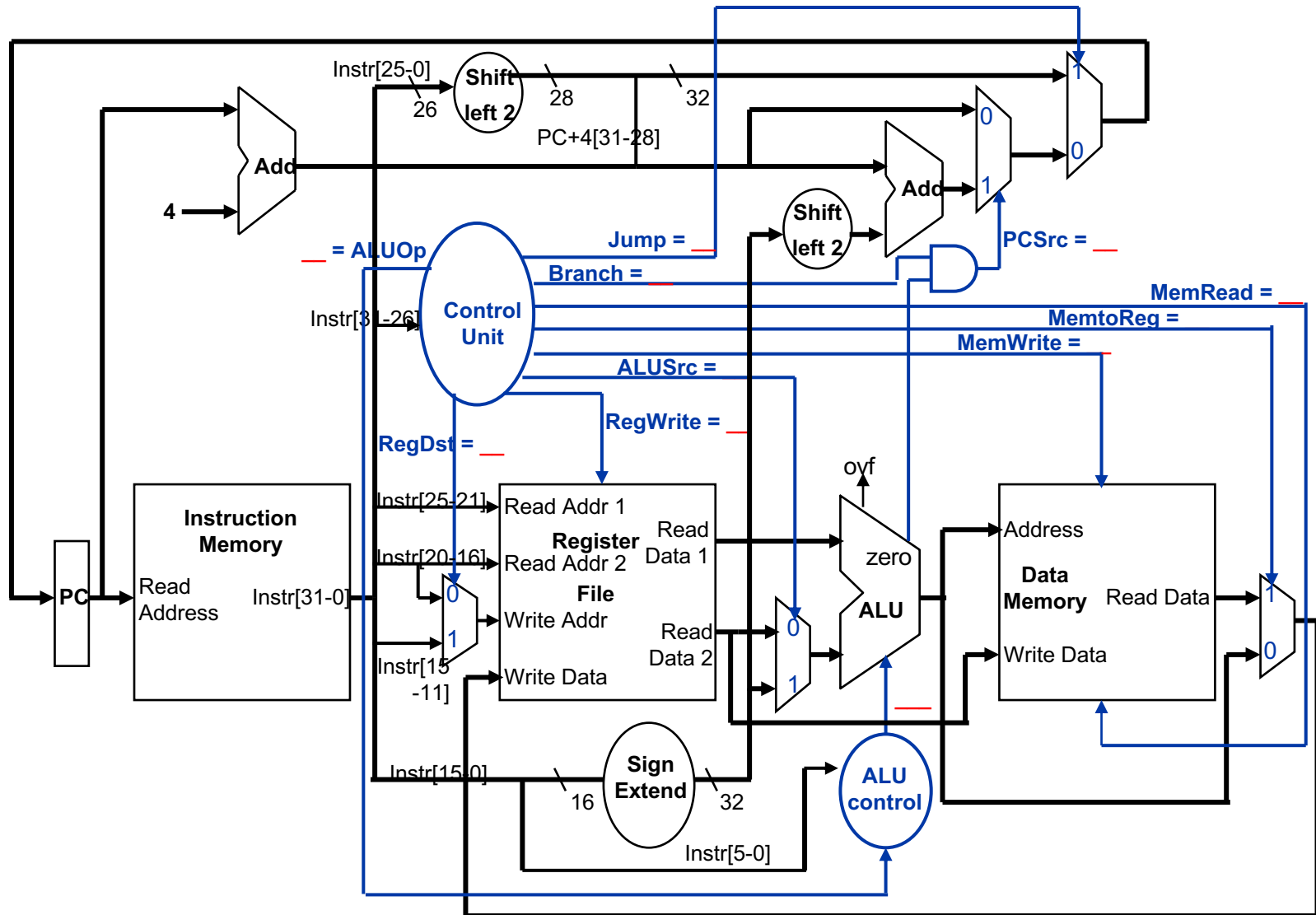
- Señales ALU Op y ALU Control según instrucción

Instrucc.	ALU Op	ALU control	Función ALU
lw sw	00	010	suma
beq bne	01	110	resta
add	10	010	suma
sub	10	110	resta
and	10	000	multiplicación lógica
or	10	001	suma lógica
slt	10	111	menor que

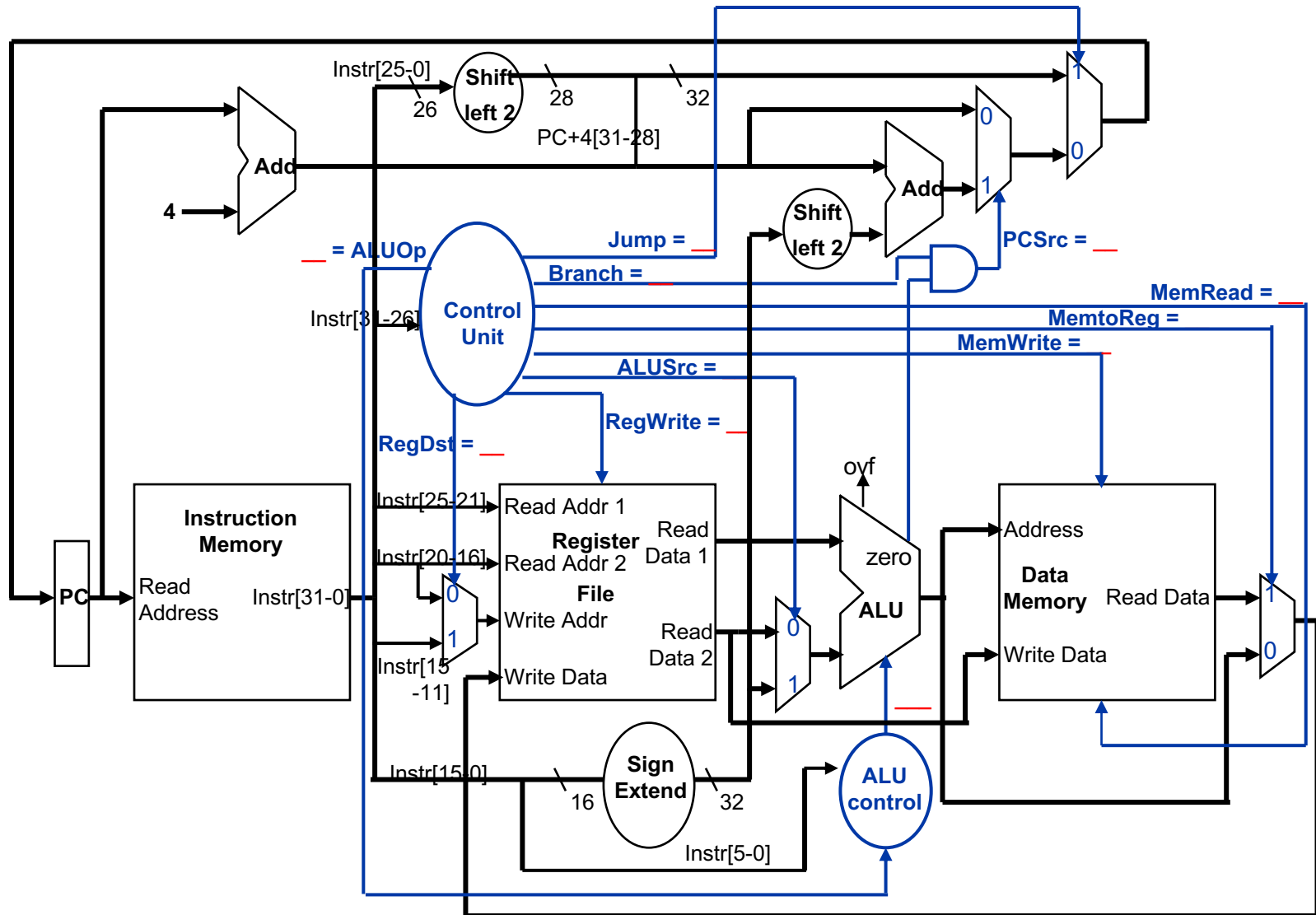
Unidad de control: *lw rt, imm(rs)*



Unidad de control: beq rt, rs, imm



Unidad de control: j addr



Contenidos

- Repaso de conocimientos previos
- Instrucciones aritméticas y de carga/almacenamiento
 - Codificación de las instrucciones
 - Construcción del camino de datos
- Instrucciones de salto condicional/no condicional
 - Codificación de las instrucciones
 - Construcción del camino de datos
- Señales de control según instrucción
- Conclusiones y bibliografía

Conclusiones

¿Cómo se convierten las instrucciones a **código máquina**?

- Campos de instrucción según tipo
- Codificar campos en binario (hexadecimal)

¿**Cómo se ejecuta** el código máquina en el microprocesador?

- Diseño del camino de datos y de la unidad de control

Bibliografía

- Lectura básica:
 - **Capítulo 4 (Patterson y Hennessy, 2011)**
 - **4.1 Introducción**
 - **4.3 Construcción de un camino de datos**
 - **4.4 Esquema de una implementación simple**
- Ampliar conocimientos:
 - Capítulo 4 (Patterson y Hennessy, 2011)
 - 4.5 Descripción general de la segmentación
 - 4.6 Camino de datos segmentados y control de la segmentación
 - Apéndice C
 - Conceptos clásicos de diseño lógico