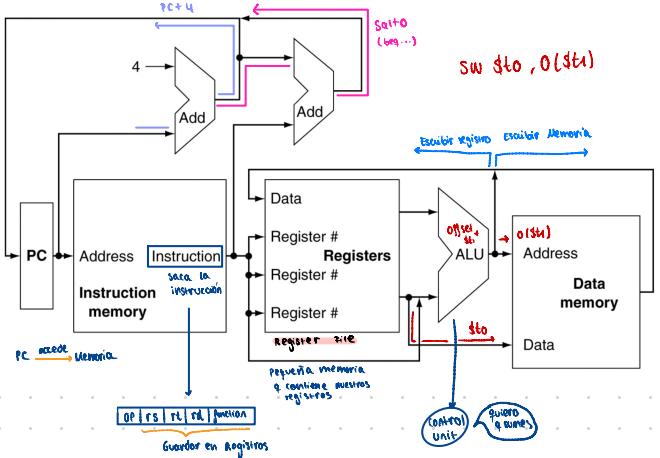


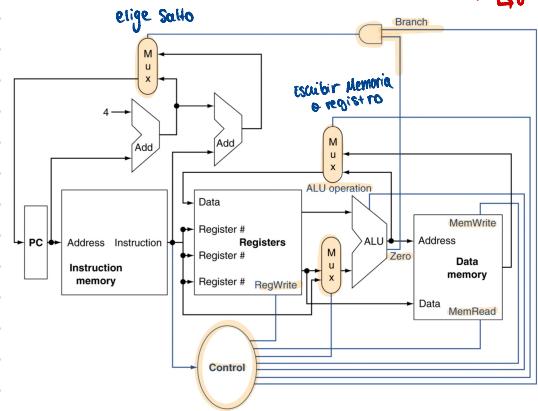


Tema 5: El Procesador

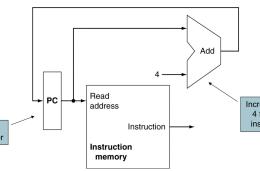
Sistema Básico de la CPU



Control

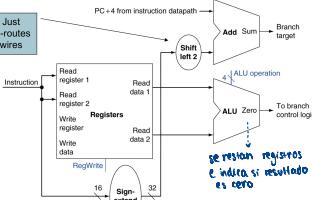


Coger Instrucción

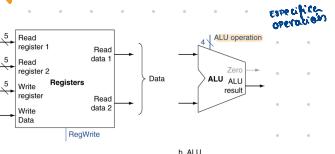


• Instrucciones Básicas

- 1) Lee registros operando
 - 2) Compara operandos [ALU]
 - 3) Calcula direcc. destino



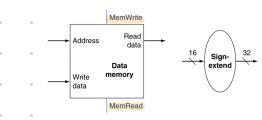
Instrucciones R-Type



- 1) Lee 2 Registros de operando
 - 2) opera \rightarrow ALU
 - 3) escribe resultado en el Registro

Instrucciones Load/Store

- 1) Lee Registros de los operando
2) Calcula dirección con offset 16bits
3) SUS: Lee Memoria → actualiza reg
 Din: escribe valor registro → Memoria

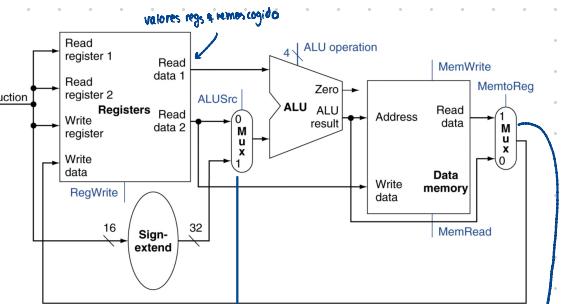


señal read, no todo el rato

read como en los R-type

¶ no podemos leer a la vez

R-Type / Load / store . DataPath



En tu por ejemplo solo cojo un registro

**INSTRUCC. CAMBIA
O Sobreexhiben datos
del registro.**

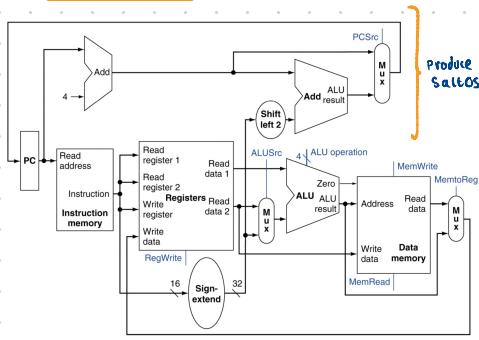
¿ A donde va MIU result?

L direcc [address]

Llegar a escribir datos de registro

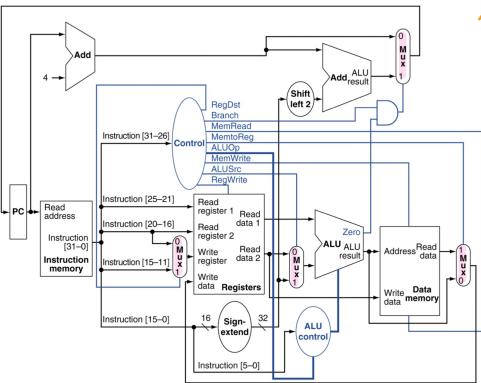
TEMA 5: El Procesador

Full DataPath



juce
itos

datapath for control



Ejercicio:

Dada la microarquitectura mostrada en la Figura, explica detalladamente cómo funcionan las siguientes instrucciones siguiendo su flujo a lo largo del datapath. Recuerda explicar el formato de cada instrucción como parte de la explicación.

```
Lw $s0, 0($s1)
Sub $t0, $t2, $t3
Beq $t4, $s4, Exit
```

Para cada operación indicar los valores de todas las señales de control.

instrucción lw: OP rs rt offset

6 bits 3 bits 5 bits 16 bits

.....

CP nos devuelve : BB SP \Rightarrow "es un BBS, entra"

CP nos devuelve : 00 0P \Rightarrow "es un Pco, entra en la unidad de control"

↳ "rs → reg1"

$\hookrightarrow "t \rightarrow \text{reg}^2"$

L "offset + sign extension

Control de la ALU

ALU control	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set-on-less-than
1100	NOR

opcode	ALUOp	Operation	funct	ALU function	ALU control
lw	00	load word	XXXXXX	add	0010
sw	00	store word	XXXXXX	add	0010
beq	01	branch equal	XXXXXX	subtract	0110
R-type					
	10	add	100000	add	0010
		subtract	100100	subtract	0110
		AND	001000	AND	0000
		OR	100100	OR	0001
		set-on-less-than	101010	set-on-less-than	0111

Unidad Central de Control

R-type	0 31:26	rs 25:21	rt 20:16	rd 15:11	shamt 10:6	funct 5:0
Load/ Store	35 or 43 31:26	rs 25:21	rt 20:16	address 15:0		
Branch	4 31:26	rs 25:21	rt 20:16	address 15:0		
	opcode	always read	read, except for load	write for R-type and load	sign-extend and add	



va al Mux q
eige en a reg
en el 100 %
tiene q ir al
offset luego
0

aux: q dato
varios a EY
en register file
de
L1a Memoria
L1(b) datos

- ALU: Suma 2 n³
- 1 reg + 1 offset
- max C₀: register
- C₁: offset

Sub	1	0	0	0	10	0	0	1
beg.	X	1	0	x	01	0	0	0

↓
no va a
en el
registro

↓
queremos
meter en la
ALU los dos
registros

TEMA 5: El Procesador

Implementación jumps

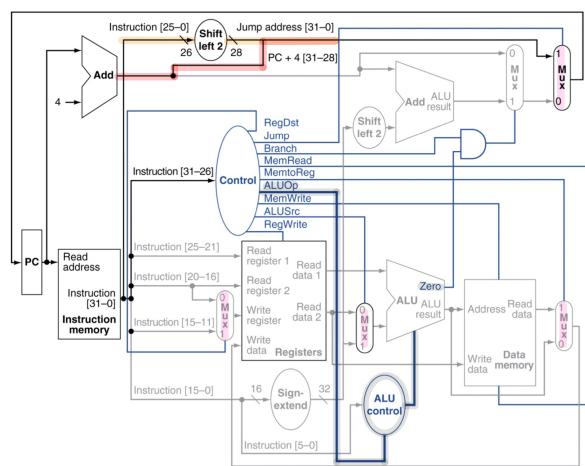


ACTUALIZA CP : concatenando

- 4 bits iniciales CP antiguo
- 26 bits direct jump
- 00

Se necesita un control extra decodificado en opcode

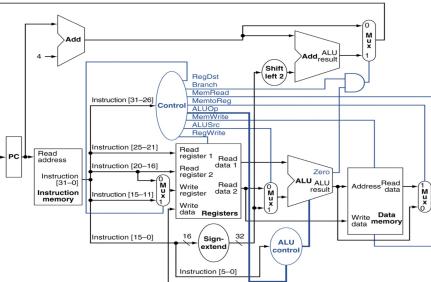
26 bits → 26 bits (SLU) + 3 bits (PCtu)



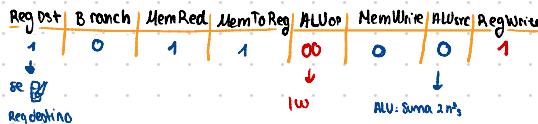
Ejercicio :

Usando como referencia la implementación mostrada en la Figura, explica detalladamente todos los cambios y señales de control necesarias para implementar la instrucción:

LWI Rd, Rm(Rn) (Interpretación: Reg[Rd]=Mem[Reg[Rm]+Reg[Rn]])



Rd rega → direct Rega = Reg Rm + Reg Rn



En Caso de salto incondicional:



Preguntar!!

Procesamiento en Paralelo

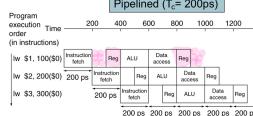
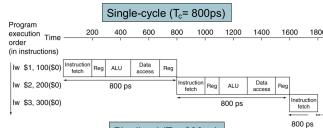
5 Fases:

- 1) IF : instruction fetch
- 2) ID : instruction decode + register read
- 3) EX : Execute operation or calculate address
- 4) MEM : Access memory operand
- 5) WB : Write result Back to register

Ejercicio: compara dos (con y sin Pipeline)

- 400 ps : instrucciones read o write
- 200 ps : para el resto

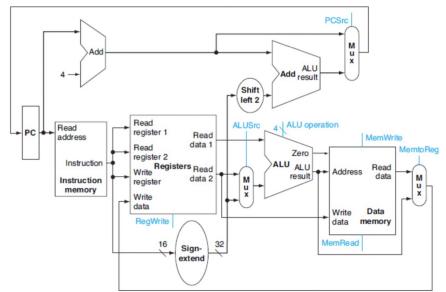
Instr	Instr fetch	Register read	ALU op	Memory access	Register write	Total time
lw	200ps	100 ps	200ps	200ps	100 ps	800ps
sw	200ps	100 ps	200ps	200ps	100 ps	700ps
R-format	200ps	100 ps	200ps	200ps	100 ps	600ps
beq	200ps	100 ps	200ps			500ps



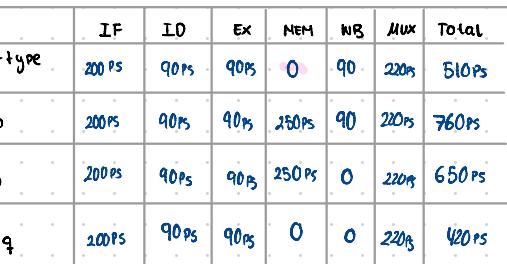
Tema 5: El Procesador

Procesamiento en Paralelo

Para la siguiente ruta de datos y las latencias de los bloques, ¿cuál es el tiempo de ciclo, si hay que soportar instrucciones aritméticas, saltos condicionales e instrucciones de carga y almacenamiento?



I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-ext	Shift-left 2
200 ps	70 ps	20 ps	90 ps	90 ps	250 ps	15 ps	10 ps



¿Pq no utiliza Mem?

Mem: guardar dato en la memoria

add \$t0,\$t1,\$t2 \Rightarrow b guarda reg!!

NO en memoria!