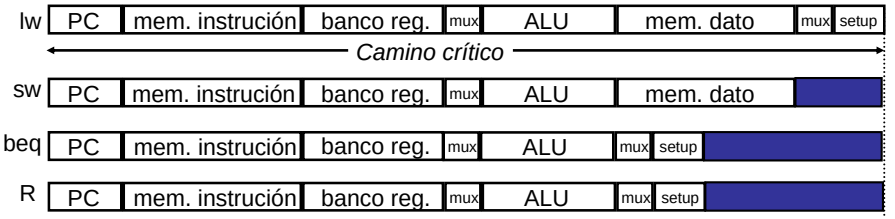
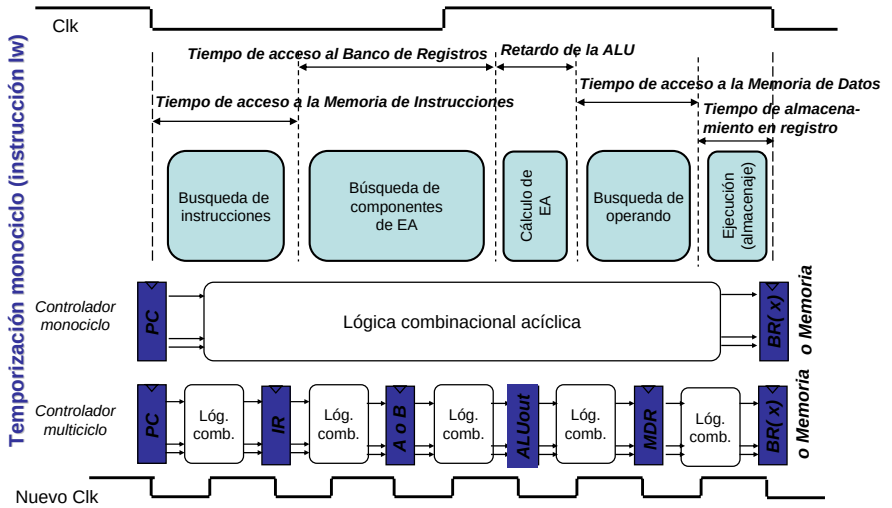
Procesador Multiciclo

Diseño de la ruta de datos multiciclo

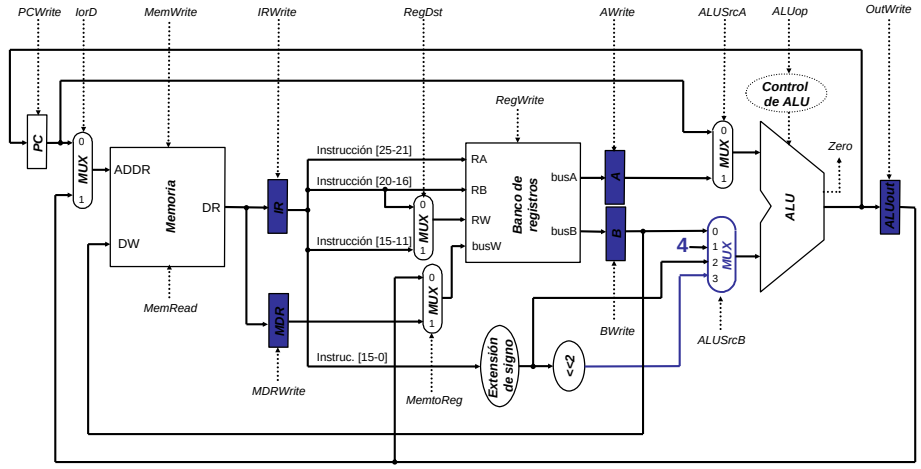


- **Problema**: en un controlador monociclo:
  - El reloj debe tener igual periodo que la instrucción más lenta
  - No se puede reusar hardware
- **Solución**: dividir la ejecución de la instrucción en varios ciclos más pequeños:
  - Cada instrucción usará el número de ciclos que necesite
  - Un mismo elemento hardware se puede ser utilizado varias veces en la ejecución de una instrucción si se hace en ciclos diferentes
  - Se requieren elementos adicionales para almacenar valores desde el ciclo en que se calculan hasta el ciclo en que se usan.

Diseño de la ruta de datos multiciclo



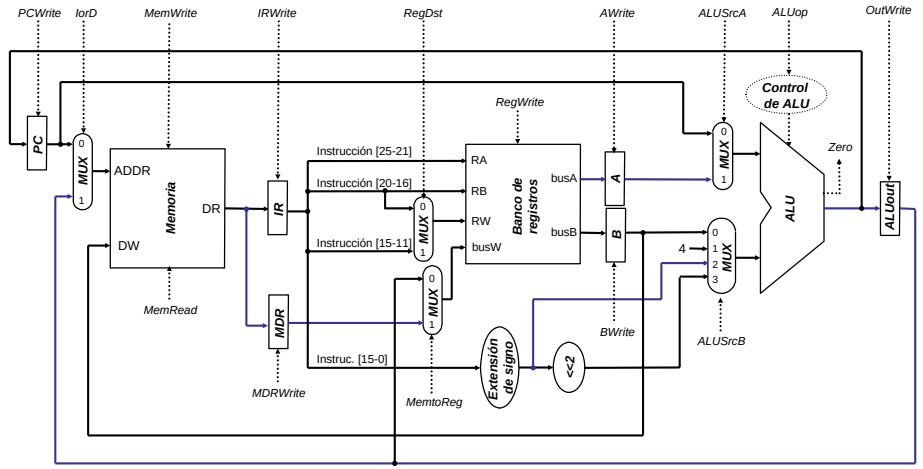
Diseño de la ruta de datos multiciclo



Diseño de la ruta de datos multiciclo

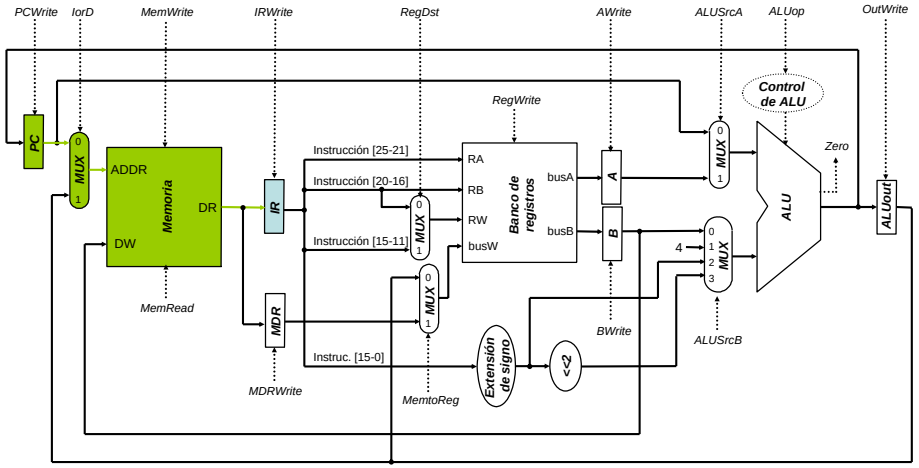
- Cambios realizados, respecto a monociclo:
  - Existencia de registros para guardar cálculos intermedios
  - Desaparece el sumador para el cálculo de la dirección de la siguiente instrucción
  - Desaparece la necesidad de tener dos memorias diferenciadas
    - memoria con dos puertos de lectura

Diseño de la ruta de datos multiciclo



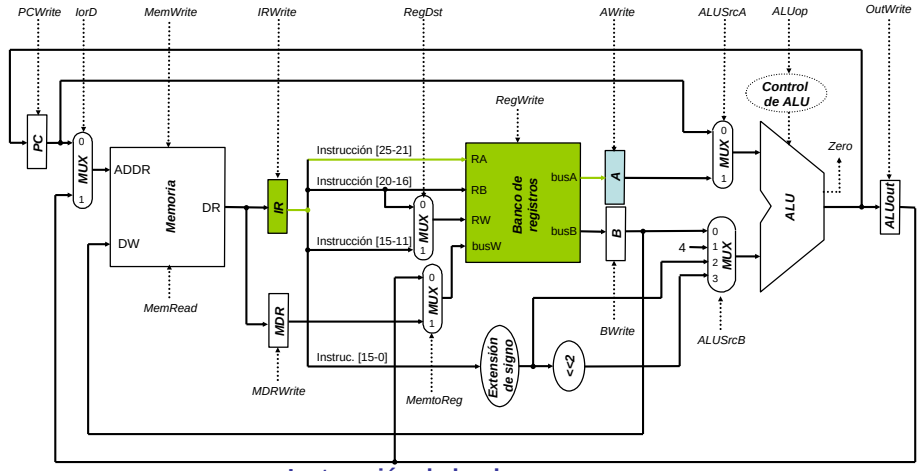
Instrucción de load

Diseño de la ruta de datos multiciclo



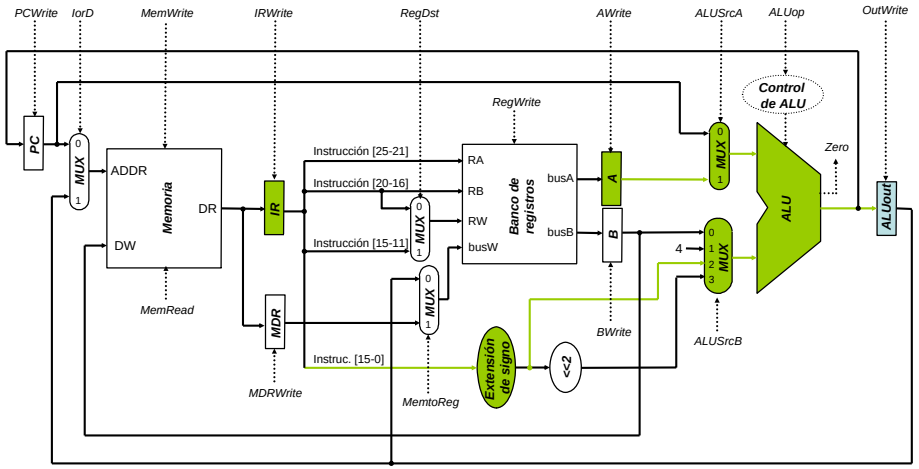
Instrucción de load
Etapa de carga de la instrucción – instruction fetch

Diseño de la ruta de datos multiciclo



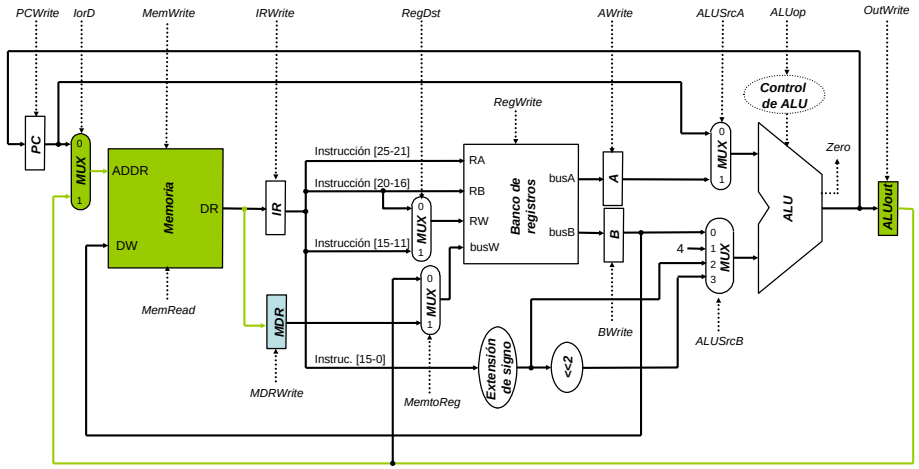
Instrucción de load
Etapa de decodificación de la instrucción y búsqueda de los registros – instruction deco

Diseño de la ruta de datos multiciclo



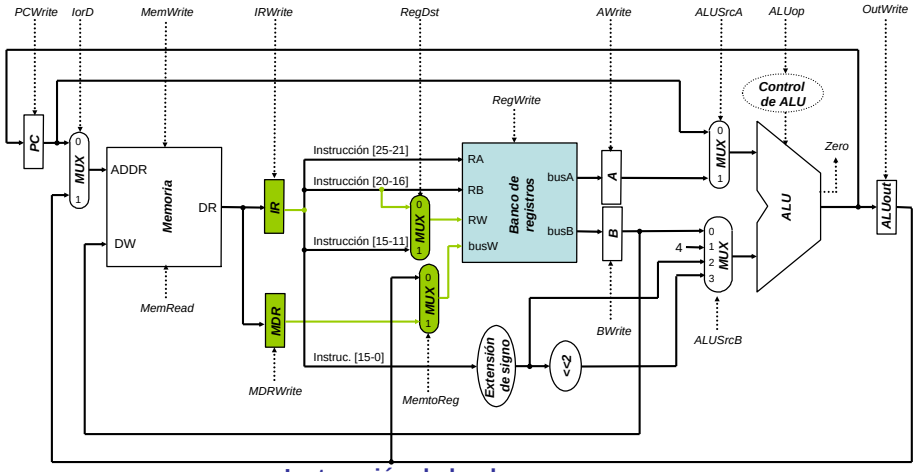
Instrucción de load
Cálculo de la dirección – execution

Diseño de la ruta de datos multiciclo



Instrucción de load
Etapa de acceso a memoria – memory access

Diseño de la ruta de datos multiciclo

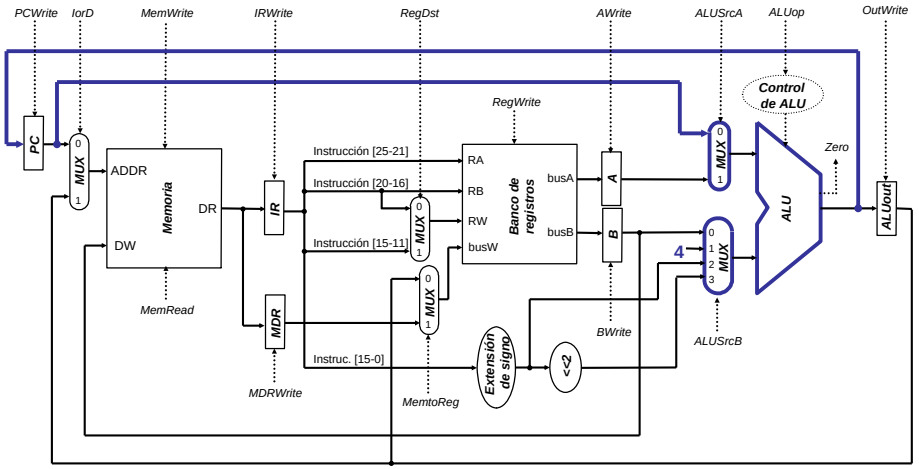


Instrucción de load
Etapa de finalización de la lectura de memoria
- write back

Diseño de la ruta de datos multiciclo

- ¿Y el cálculo de la siguiente instrucción?
- ¿Cómo y cuándo conviene realizarlo?

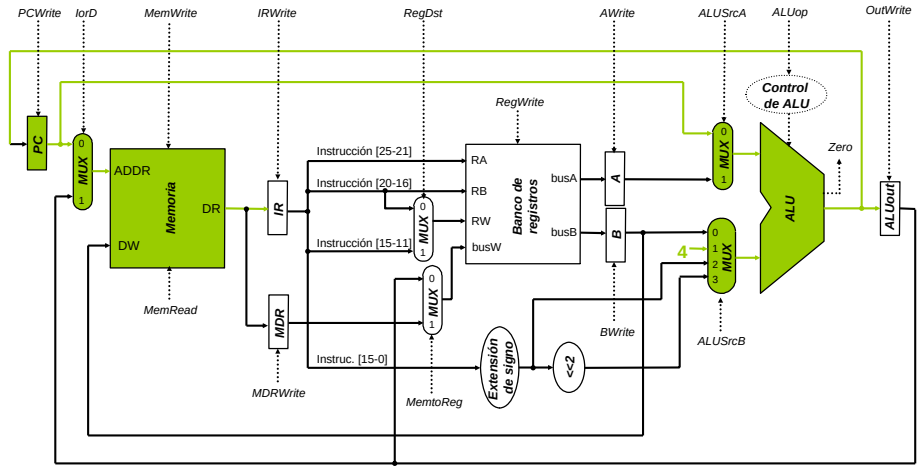
Diseño de la ruta de datos multiciclo



Cálculo de la dirección de la siguiente instrucción (cómo)

Marcos Sánchez-Élez Martín

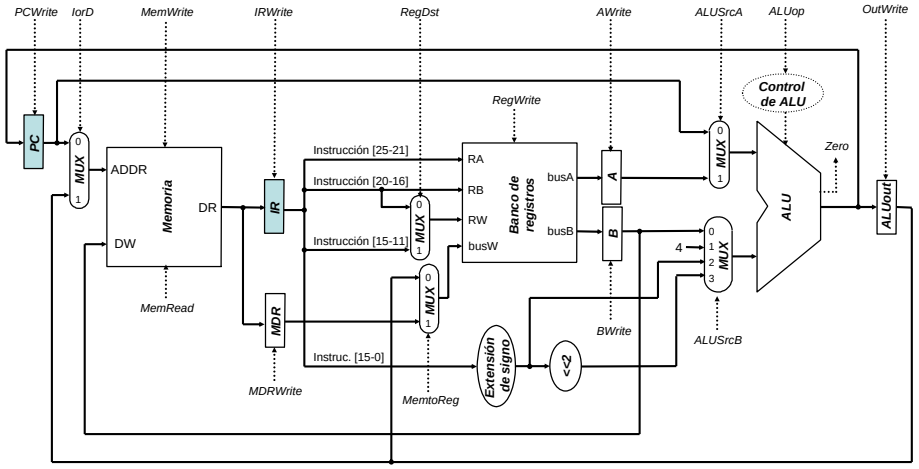
Diseño de la ruta de datos multiciclo



Cálculo de la dirección de la siguiente instrucción (cuándo)

Marcos Sánchez-Élez Martín

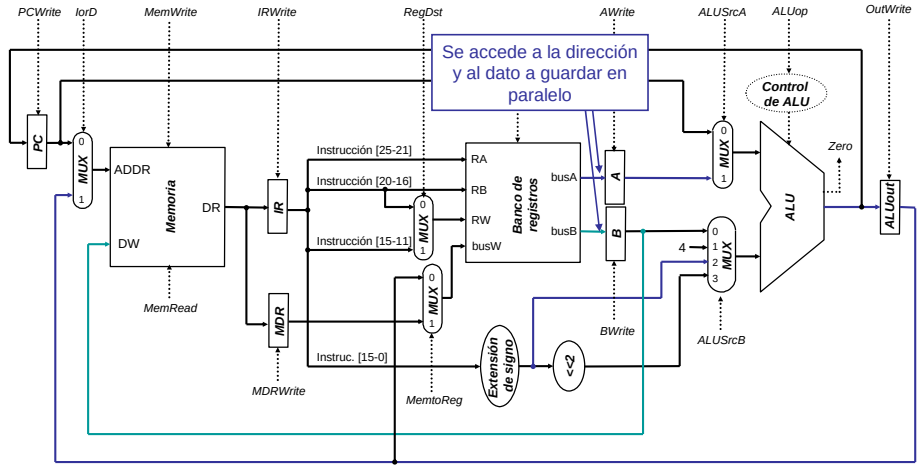
Diseño de la ruta de datos multiciclo



Cálculo de la dirección de la siguiente instrucción (cuando)

Marcos Sánchez-Élez Martín

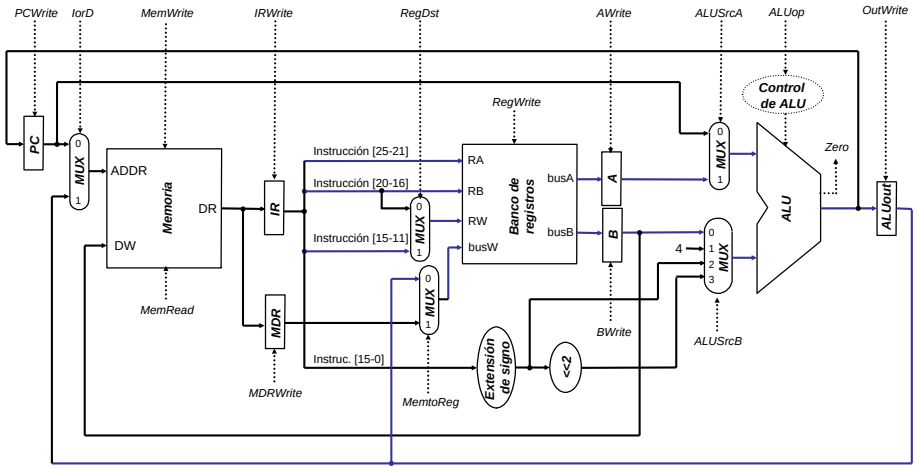
Diseño de la ruta de datos multiciclo



Instrucción de store

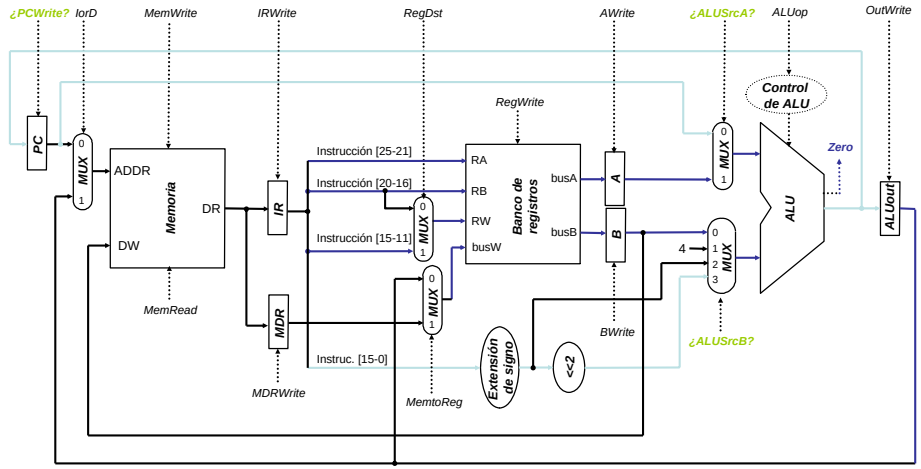
Marcos Sánchez-Élez Martín

Diseño de la ruta de datos multiciclo



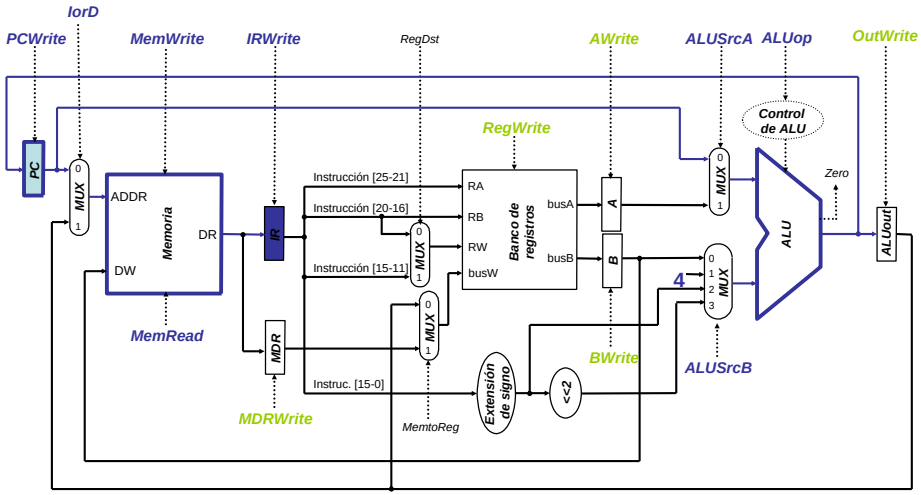
Instrucción aritmeticológica

Diseño de la ruta de datos multiciclo



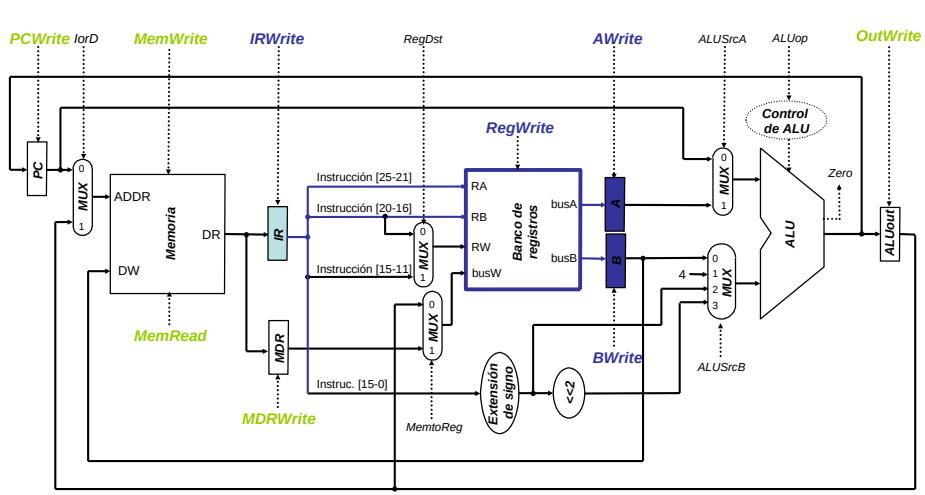
Instrucción de salto

Diseño del controlador multiciclo



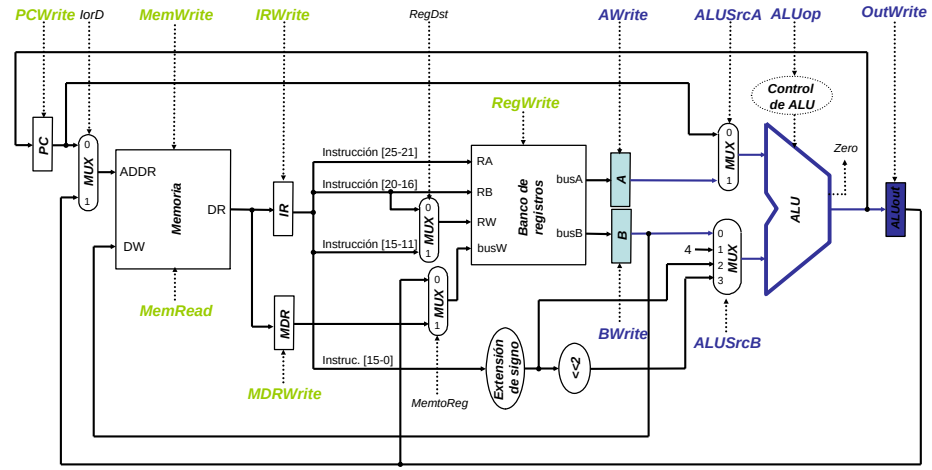
Ejemplo completo para i. aritmeticológicas

Diseño del controlador multiciclo



Ejemplo completo para i. aritmeticológicas

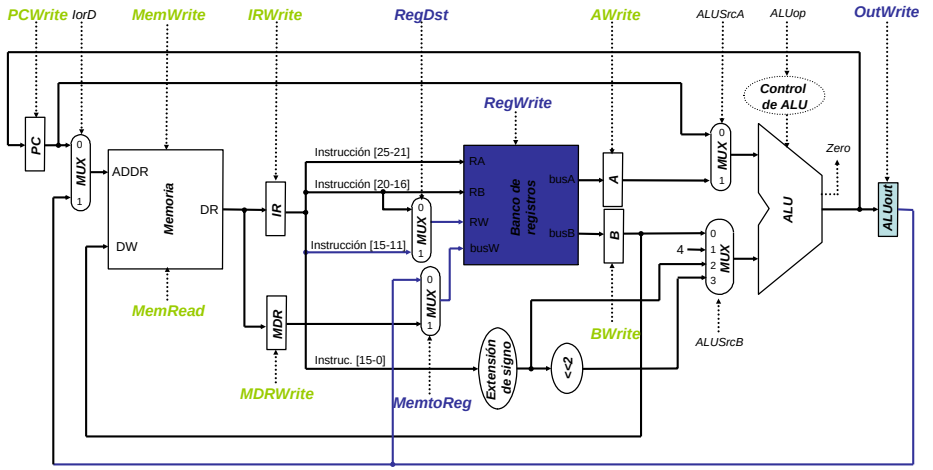
Diseño del controlador multiciclo



Ejemplo completo para i. aritmeticológicas

Procesador Multiciclo

Diseño del controlador multiciclo



Ejemplo completo para i. aritmeticológicas

Marcos Sánchez-Élez Martín

Procesador Multiciclo

Diseño del controlador multiciclo

- I. Aritmeticológicas
  - Todas las señales de carga de registros deberían estar a **cero** cuando no se estén cargando dichos registros.
  - La señal de MemWrite debería estar a **cero** siempre
  - La señal RegWrite debería estar a **cero** en el resto de estados
  - La señal MemRead debería estar a **cero** en el resto de estados

Marcos Sánchez-Élez Martín

Procesador Multiciclo

Diseño del controlador multiciclo

Transferencias entre registros "lógicas"

BR(rd) ← BR(rs) funct BR(rt), PC ← PC + 4

Instrucción Aritmeticológica

Transferencias entre registros "físicas"

1. IR ← Memoria(PC), PC ← PC + 4
2. A ← BR(rs), B ← BR(rt)
3. ALUout ← A funct B
4. BR(rd) ← ALUout

PCWrite = 1
IorD = 0
MemWrite = 0
MemRead = 1
IRWrite = 1

Marcos Sánchez-Élez Martín

Diseño del controlador multiciclo

Transferencias entre registros "lógicas"

BR(rd) ← BR(rs) funct BR(rt), PC ← PC + 4

Instrucción Aritmeticológica

Transferencias entre registros "físicas"

1. IR ← Memoria(PC), PC ← PC + 4
2. A ← BR(rs), B ← BR(rt)
3. ALUout ← A funct B
4. BR(rd) ← ALUout

AWrite = 1
BWrite = 1
RegWrite = 0

Diseño del controlador multiciclo

Transferencias entre registros "lógicas"

BR(rd) ← BR(rs) funct BR(rt), PC ← PC + 4

Instrucción Aritmeticológica

Transferencias entre registros "físicas"

1. IR ← Memoria(PC), PC ← PC + 4
2. A ← BR(rs), B ← BR(rt)
3. ALUout ← A funct B
4. BR(rd) ← ALUout

ALUSrcA = 0
ALUSrcB = 0
ALUOpt = "*function*"
OutWrite = 1

Diseño del controlador multiciclo

Transferencias entre registros "lógicas"

BR(rd) ← BR(rs) funct BR(rt), PC ← PC + 4

Instrucción Aritmeticológica

Transferencias entre registros "físicas"

1. IR ← Memoria(PC), PC ← PC + 4
2. A ← BR(rs), B ← BR(rt)
3. ALUout ← A funct B
4. BR(rd) ← ALUout

MemtoReg = 0
RegDst = 1
RegWrite = 1

Diseño del controlador multiciclo

Instrucción de carga (lw)

Transferencias entre registros "lógicas"
BR(rt) ← Memoria(BR(rs) + SignExt(inmed)), PC ← PC + 4
Transferencias entre registros "físicas"
1. IR ← Memoria(PC), PC ← PC + 4
2. A ← BR(rs)
3. ALUout ← A + SignExt(inmed)
4. MDR ← Memoria(ALUout)
5. BR(rt) ← MDR

Instrucción de almacenaje (sw)

Transferencias entre registros "lógicas"
Memoria(BR(rs) + SignExt(inmed)) ← BR(rt), PC ← PC + 4
Transferencias entre registros "físicas"
1. IR ← Memoria(PC), PC ← PC + 4
2. A ← BR(rs), B ← BR(rt)
3. ALUout ← A + SignExt(inmed)
4. Memoria(ALUout) ← B

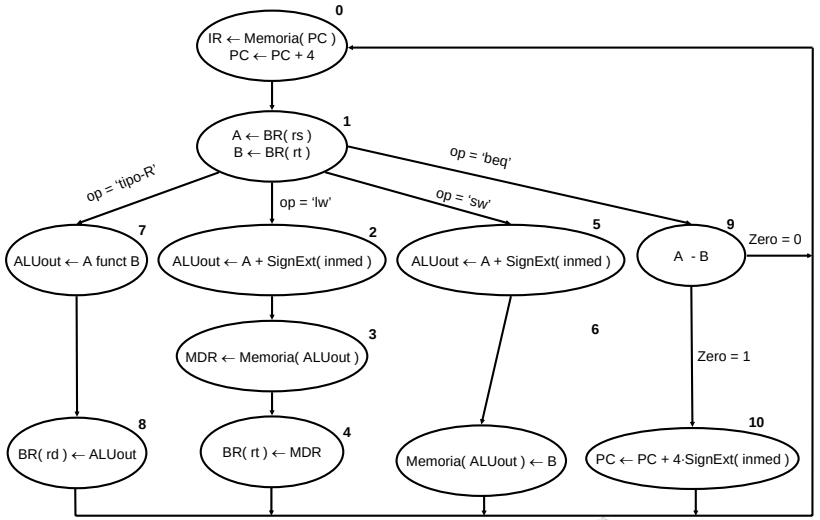
Diseño del controlador multiciclo

Instrucción de salto condicional (beq)

Transferencias entre registros "lógicas"
si (BR(rs) = BR(rt))
entonces PC ← PC + 4 + 4·SignExt(inmed)
sino PC ← PC + 4
Transferencias entre registros "físicas"
1. IR ← Memoria(PC), PC ← PC + 4
2. A ← BR(rs), B ← BR(rt),
3. A - B
4. si Zero entonces PC ← PC + 4·SignExt(inmed)

Observaciones: en todas las instrucciones las acciones 1. y 2. son iguales (*excepto en lw, pero no habría problema en modificarla*)

Diseño del controlador multiciclo



Diseño del controlador multiciclo

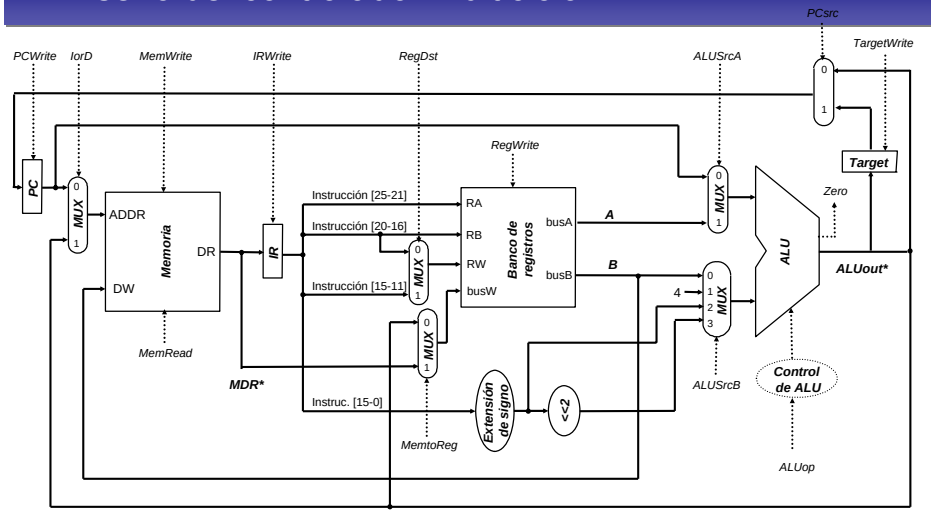
lw r1, 0(r0)
lw r2, 4(r0)
add r3, r1, r2
beq r3, r5, 1
sub r3, r3, r5
sw r3, 8(r0)

¿Tiempo de ejecución (ciclos)?

Diseño del controlador multiciclo

- ¿Podría disminuirse el tiempo de ejecución?

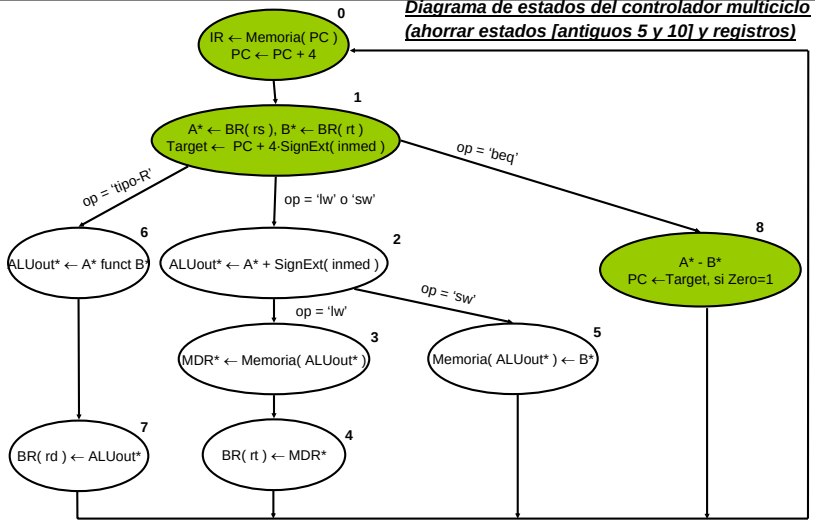
Diseño del contralador multiciclo



Ruta de datos multiciclo (ahorro de registros temporales y tiempo de ejecución)

Diseño del controlador multiciclo

Diagrama de estados del controlador multiciclo (ahorrar estados [antiguos 5 y 10] y registros)

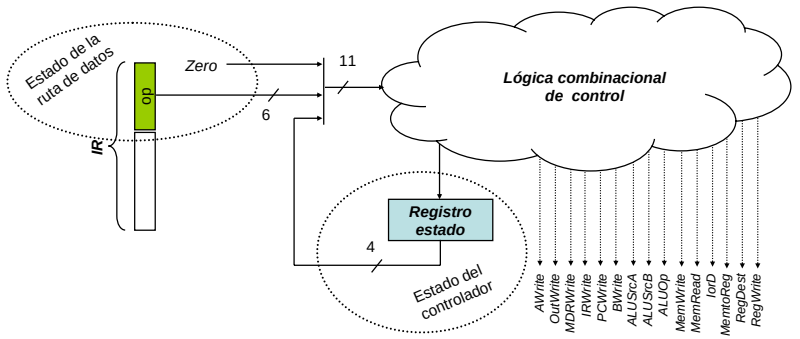


Diseño del controlador multiciclo

El controlador como FSM (finite state machine)

- 1. Se **traducen** las transferencias entre registros como **conjuntos de activaciones** de los puntos de control de la ruta de datos
- 2. Se **codifican** los estados
- 3. Mediante **tablas de verdad** se describen:
  - las **transiciones de estado** en función del código de operación y del estado de la ruta de datos
  - el **valor de las señales** de control en función del estado (controlador tipo Moore) y adicionalmente en función del estado de la ruta de datos (controlador tipo Mealy).
- 4. La **estructura del controlador** estará formada por:
  - **Registro de estado**
  - Conjunto de **lógica combinacional de control** que implementa las anteriores tablas

Diseño del controlador multiciclo



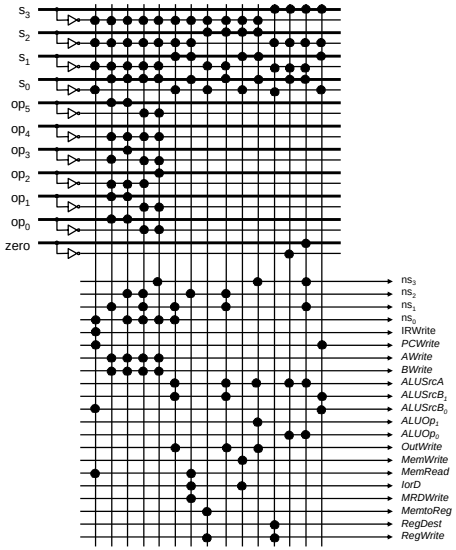
Diseño del controlador multiciclo

Tabla de verdad del controlador

Estado actual	op	Zero	Estado siguiente	IRWrite	PCWrite	AWrite	BWrite	ALUSrcA	ALUScrB	ALUOp	OutWrite	MemWrite	MemRead	IorD	MDRWrite	MemtoReg	RegDest	RegWrite
0000	XXXXXX	X	0001	1	1			0	01	00 (add)		0	1	0				0
0001	100011 (lw)	X	0010															
0001	101011 (sw)	X	0101	0	0	1	1					0	0					0
0001	000000 (tipo-R)	X	0111															
0001	000100 (beq)	X	1001															
0010	XXXXXX	X	0011	0	0			1	10	00 (add)	1	0	0					0
0011	XXXXXX	X	0100	0	0							0	1	1	1			0
0100	XXXXXX	X	0000	0	0							0	0			1	0	1
0101	XXXXXX	X	0110	0	0		0	1	10	00 (add)	1	0	0					0
0110	XXXXXX	X	0000	0	0							1	0	1				0
0111	XXXXXX	X	1000	0	0			1	00	10 (funct)	1	0	0					0
1000	XXXXXX	X	0000	0	0							0	0			0	1	1
1001	XXXXXX	0	0000	0	0			1	00	01 (sub)		0	0					0
1001	XXXXXX	1	1010															
1010	XXXXXX	X	0000	0	1			0	11	00 (add)		0	0					0

Diseño del controlador multiciclo

- Lógica discreta:
  - 21 funciones de conmutación
  - 11 variables diferentes
- 1 PLA
  - 11 entradas
  - 21 salidas
  - 15 términos producto
- 1 ROM (~42 Kbits):
  - 11 bits de dirección (2^{11} palabras)
  - palabras de 21 bits
- 2 ROM (~10 Kbits)
  - ROM de control:
    - 4 bits de dirección (2^4 palabras)
    - palabra de 17 bits
  - ROM de siguiente estado:
    - 11 bits de dirección (2^{11} palabras)
    - palabras de 4 bits
- Ventajas de la lógica discreta:
  - velocidad y coste
- Ventajas de la lógica almacenada:
  - facilidad de diseño
  - adaptabilidad



Diseño del controlador multiciclo

Estado Actual	op	Zero	Estado Siguiente	IR Write	PC Write	A Write	B Write	ALUSrcA	ALUSrcB	ALUop	OutWrite	MemWrite	MemRead	IorD	MDRWrite	MemtoReg	RegDest	RegWrite
fetch	xxxxxx	x	deco	1	1	0	0	0	01	+	0	0	1	0	0	x	x	0
deco	lw	x	ex	0	0	1	0	x	xx	x	0	0	0	x	0	x	x	0
	otras						1											
ex	lw	x	mem	0	0	0	0	1	10	+	1	0	0	x	0	x	x	0
	sw		wb						10	+	1							
	tipo-R		wb						00	--	1							
	beq	0	fetch						00	-	0							
		1	wb															
mem	xxxxxx	x	wb	0	0	0	0	x	x	x	0	0	1	1	1	x	x	0
wb	lw	x	fetch	0	0	0	0	x	x	x	0	0	0	x	0	1	0	1
	sw	x			0							1		1		x	0	0
	tipo-R	x			0							0		x		0	1	1
	beq	x			1							0		x		x	x	0

Comparación: monociclo vs. multiciclo

Asumir que el tiempo de ciclo en el procesador muticiclo es 5 veces menor que el tiempo de ciclo en el procesador monociclo

10^6 instrucciones tardan en ejecutarse:

- $t_{multi} = 10^6 \cdot CPI_{multi} \cdot t_{multi} = 10^6 \cdot 4.03 \cdot t_{multi}$
- $t_{mono} = 10^6 \cdot CPI_{mono} \cdot t_{mono} = 10^6 \cdot 1 \cdot 5 \cdot t_{multi}$

$t_{multi} / t_{mono} = 4.03 / 5 = 0.8$
los programas tardan un **20% menos** en ejecutarse en el computador multiciclo

Operación	*Frecuencia*	*Ciclos*	*CPI*
tipo-R	50%	4	2.0
lw	20%	5	1.0
st	10%	4	.4
beq (salta)	2.5%	4	.1
beq (no salta)	17.5%	3	0.53
			4.03

Comparación monociclo vs multiciclo

Ejemplo:

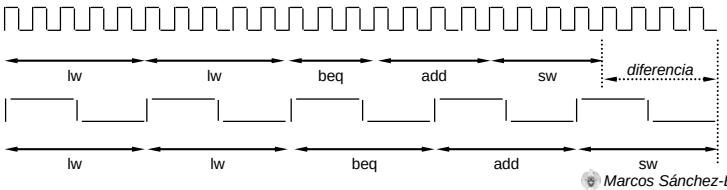
lw $2, 0($3)

lw $3, 4($3)

beq $2, $3, Label #asumir que no se salta

add $5, $2, $3

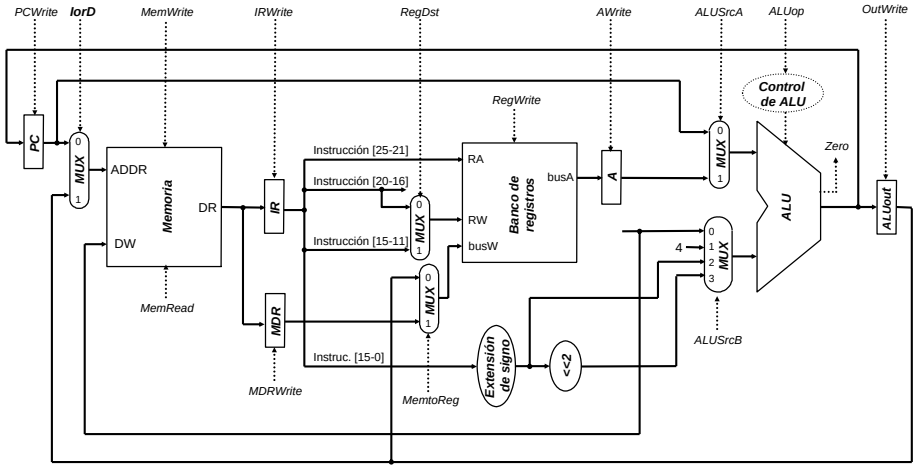
sw $5, 8($3)



Ejemplo

- ¿Qué ocurriría si el banco de registros sólo tuviese un puerto de lectura?

Diseño del controlador multiciclo



¿?