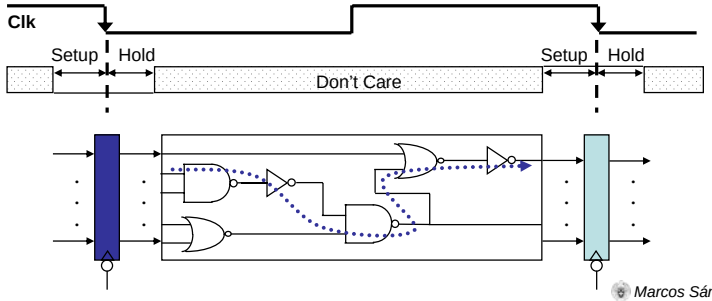
Procesador Monociclo

Diseño de la ruta de datos monociclo

- **Ejecución típica (de una instrucción)**
  - Todos los registros se cargan simultáneamente (de modo selectivo)
  - Todos los valores se propagan a través de las redes combinacionales hasta estabilizarse en las entradas de los registros
  - Todos los elementos de almacenamiento están sincronizados al mismo flanco de reloj:
    - Tiempo de ciclo = Hold + Camino con retardo máximo + Setup + Clock Skew



Diseño de la ruta de datos monociclo

1. El ciclo de instrucción comienza buscando la instrucción en memoria (***fetch***)
   - instrucción ← Memoria(PC)
2. En función del tipo de instrucción se realiza una de las siguientes operaciones
3. Se vuelve a comenzar

- **Instrucciones con referencia a memoria** (formato tipo I):
  - lw rt, inmed(rs) rt ← Memoria(rs + SignExt(inmed)) , PC ← PC + 4
  - sw rt, inmed(rs) Memoria(rs + SignExt(inmed)) ← rt , PC ← PC + 4
- **Instrucciones aritmético-lógicas con operandos en registros** (formato tipo R)
  - add rd, rs, rt rd ← rs + rt, PC ← PC + 4
  - sub rd, rs, rt rd ← rs - rt , PC ← PC + 4
  - and rd, rs, rt rd ← rs and rt , PC ← PC + 4
  - or rd, rs, rt rd ← rs or rt , PC ← PC + 4
  - slt rd, rs, rt (si (rs < rt) entonces (rd ← 1)
    en otro caso (rd ← 0)), PC ← PC+4
- **Instrucciones de salto condicional** (formato tipo I)
  - beq rs, rt, inmed si (rs = rt) entonces (PC ← PC + 4 + 4·SignExp(inmed))
    en otro caso PC ← PC + 4

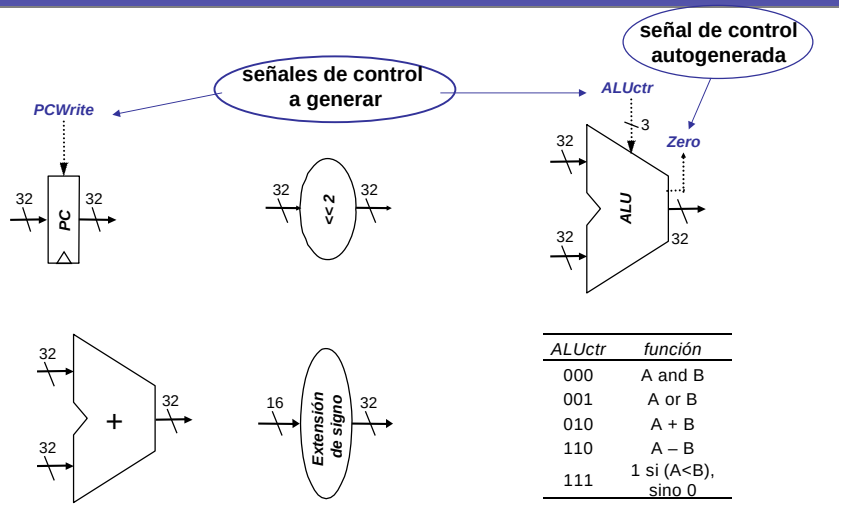
Diseño de la ruta de datos monociclo

```
lw r1, 0(r0)
lw r2, 1(r0)
add r3, r1, r2
beq r3, r5, 1
sub r3, r3, r5
sw r3, 2(r0)
```

Diseño de la ruta de datos monociclo

- Para implementar el subconjunto del repertorio del MIPS en una implementación monociclo se requieren (el tamaño de palabra es de 32 bits):
  - **Memoria de instrucciones**
  - **Memoria de datos**
  - **32 registros de datos:** visibles por el programador.
  - **Contador de programa**
  - **2 Sumadores:** para sumar 4 al PC, y para sumar al PC el valor inmediato de salto.
  - **ALU:** capaz de realizar suma, resta, and, or, comparación de mayoría e indicación de que el resultado es cero (para realizar la comparación de igualdad mediante resta)
  - **Extensor de signo:** para adaptar el operando inmediato de 16 bits al tamaño de palabra.
  - **Desplazador a la izquierda:** para implementar la multiplicación por 4 (necesaria en el salto).

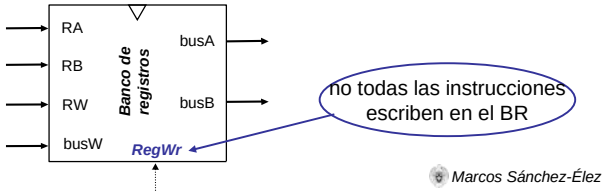
Diseño de la ruta de datos monociclo



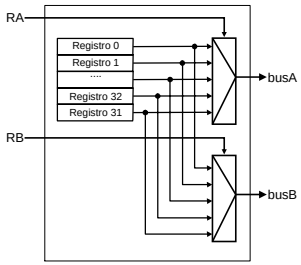
ALUctr	*función*
000	A and B
001	A or B
010	A + B
110	A – B
111	1 si (A<B), sino 0

Diseño de la ruta de datos monociclo

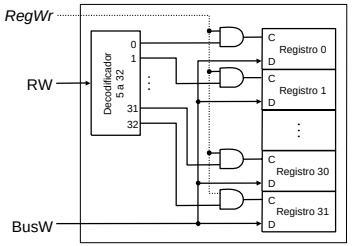
- Los 32 registros se almacenan en un **banco de registros**. Dado que en las instrucciones de tipo R, se requiere un acceso simultáneo a 3 registros:
  - 2 salidas de datos de 32 bits
  - 1 entradas de datos de 32 bits
  - 3 entradas de 5 bits para la identificación de los registros
  - 1 entrada de control para habilitar la escritura sobre uno de los registros
  - 1 puerto de reloj (sólo determinante durante las operaciones de escritura, las de lectura son combinacionales)



Diseño de la ruta de datos monociclo



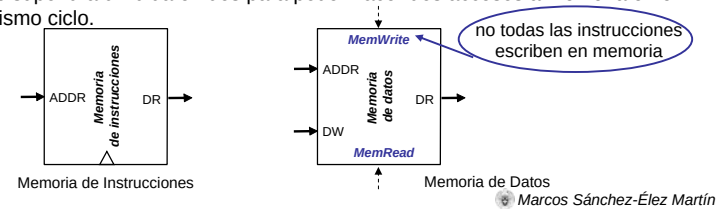
Mecanismo de Lectura



Mecanismo de Escritura

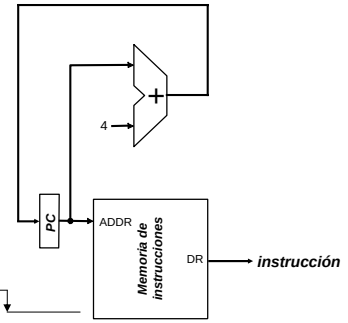
Diseño de la ruta de datos monociclo

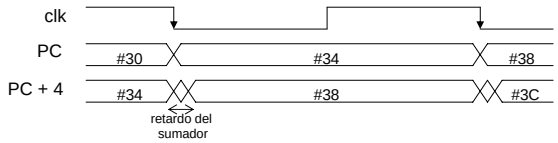
- La memoria tendrá un comportamiento idealizado.
  - "Integrada" dentro de la CPU.
  - Direccionable por bytes, pero capaz de aceptar/ofrecer 4 bytes por acceso
    - 1 entrada de dirección
    - 1 salida de datos de 32 bits
    - 1 entrada de datos de 32 bits (sólo en la de datos)
  - 1 entrada de control para seleccionar el tipo de operación (lectura/escritura), sólo en la de datos
  - Se supondrá que se comporta temporalmente como el banco de registros (síncronamente) y que tienen un tiempo de acceso menor que el tiempo de ciclo
  - Se supondrá dividida en dos para poder hacer dos accesos a memoria en el mismo ciclo.



Diseño de la ruta de datos monociclo

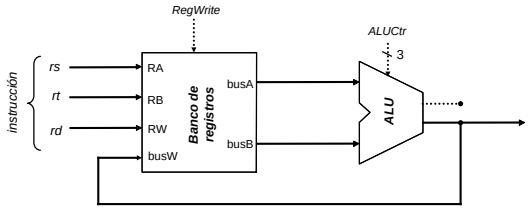
- La **búsqueda de instrucciones** implica:
  - Leer la instrucción ubicada en la dirección de la **memoria de instrucciones** indicada por el **contador de programa**:
- La **ejecución secuencial** de programas implica:
  - Actualizar el **contador de programa** para que apunte a la siguiente instrucción (sumando 4 por ser una memoria direccionable por bytes y una arquitectura con tamaño de palabra de 32 bits



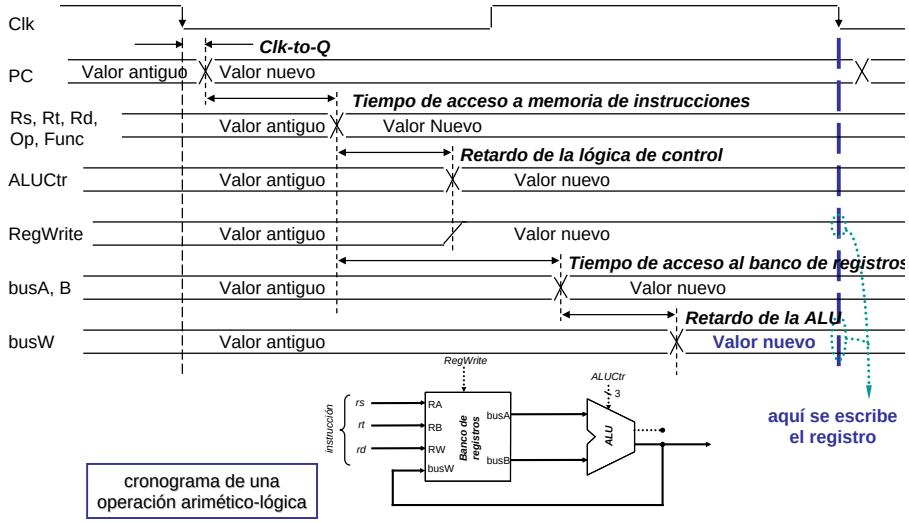


Diseño de la ruta de datos monociclo

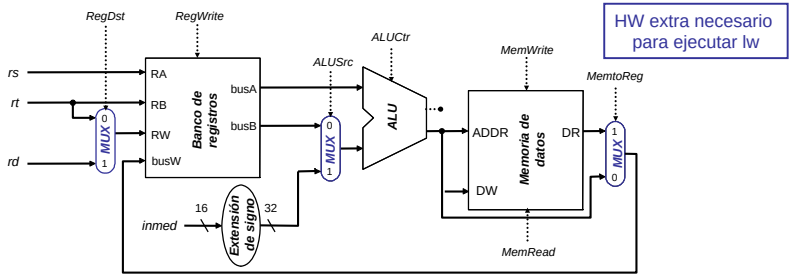
- Las instrucciones **aritmético lógicas** (tipo-R) implican:
  - **BR(rd) ← BR(rs) funct BR(rt)**
  - Leer dos registros cuyos identificadores se ubican en los campos **rs** y **rt** de la instrucción:
  - Operar sobre ellos según el contenido del campo de código de operación aritmética (**funct**) de la instrucción
  - Almacenar el resultado en otro registro cuyo identificador se localiza en el campo **rd** de la instrucción

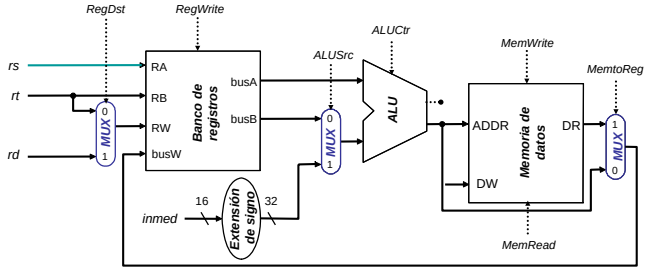


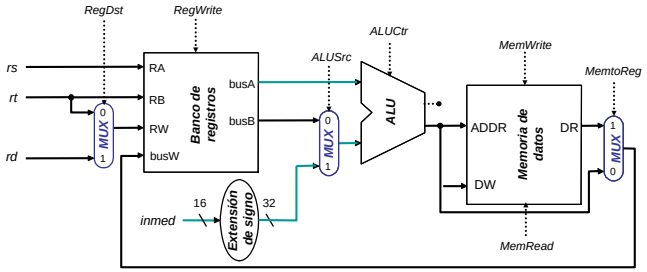
Diseño de la ruta de datos monociclo

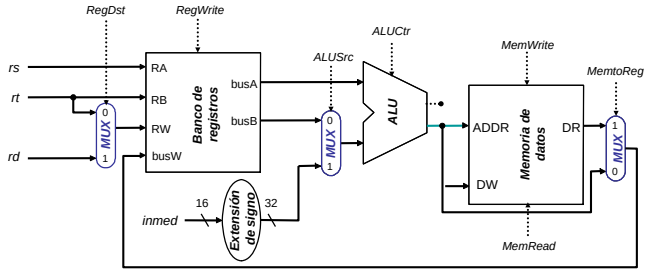


cronograma de una operación aritmético-lógica









Diseño de la ruta de datos monociclo

- La **instrucción de carga** (***lw***) implica:
  - **BR(rt) ← Memoria(BR(rs) + SignExt(inmed))**
  - Calcular la dirección efectiva de memoria:
    - Leyendo el *registro base* cuyo identificador se ubica en el campo **rs** de la instrucción
    - Obteniendo un *desplazamiento* de 32 bits a partir de la **extensión** del campo de operando inmediato (**inmed**) de la instrucción
    - **Sumando** base y desplazamiento.
  - Leer dato ubicado en la **memoria de datos** cuya dirección es la anteriormente calculada
  - Almacenar el dato leído de memoria en el registro cuyo identificador se especifica en el campo **rt** de la instrucción

HW extra necesario para ejecutar lw

Diseño de la ruta de datos monociclo

- La **instrucción de carga** (***lw***) implica:
  - **BR(rt) ← Memoria(BR(rs) + SignExt(inmed))**
  - Calcular la dirección efectiva de memoria:
    - Leyendo el *registro base* cuyo identificador se ubica en el campo **rs** de la instrucción
    - Obteniendo un *desplazamiento* de 32 bits a partir de la **extensión** del campo de operando inmediato (**inmed**) de la instrucción
    - **Sumando** base y desplazamiento.
  - Leer dato ubicado en la **memoria de datos** cuya dirección es la anteriormente calculada
  - Almacenar el dato leído de memoria en el registro cuyo identificador se especifica en el campo **rt** de la instrucción

---

Diseño de la ruta de datos monociclo

- La **instrucción de carga** (***lw***) implica:
  - **BR(rt) ← Memoria(BR(rs) + SignExt(inmed))**
  - Calcular la dirección efectiva de memoria:
    - Leyendo el *registro base* cuyo identificador se ubica en el campo **rs** de la instrucción
    - Obteniendo un *desplazamiento* de 32 bits a partir de la **extensión** del campo de operando inmediato (**inmed**) de la instrucción
    - **Sumando** base y desplazamiento.
  - Leer dato ubicado en la **memoria de datos** cuya dirección es la anteriormente calculada
  - Almacenar el dato leído de memoria en el registro cuyo identificador se especifica en el campo **rt** de la instrucción

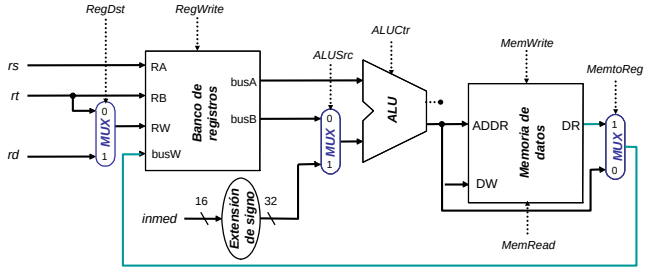
---

Diseño de la ruta de datos monociclo

- La **instrucción de carga** (***lw***) implica:
  - **BR(rt) ← Memoria(BR(rs) + SignExt(inmed))**
  - Calcular la dirección efectiva de memoria:
    - Leyendo el *registro base* cuyo identificador se ubica en el campo **rs** de la instrucción
    - Obteniendo un *desplazamiento* de 32 bits a partir de la **extensión** del campo de operando inmediato (**inmed**) de la instrucción
    - **Sumando** base y desplazamiento.
  - Leer dato ubicado en la **memoria de datos** cuya dirección es la anteriormente calculada
  - Almacenar el dato leído de memoria en el registro cuyo identificador se especifica en el campo **rt** de la instrucción

Diseño de la ruta de datos monociclo

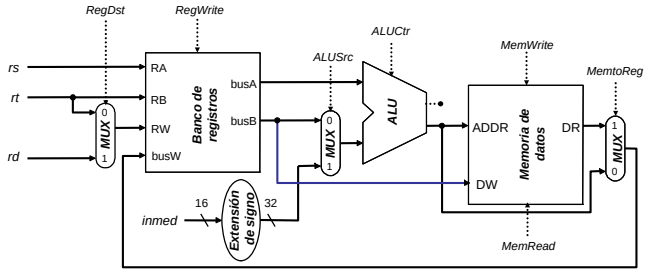
- La **instrucción de carga** (*lw*) implica:
  - **BR(rt) ← Memoria(BR(rs) + SignExt(inmed))**
  - Calcular la dirección efectiva de memoria:
    - Leyendo el *registro base* cuyo identificador se ubica en el campo **rs** de la instrucción
    - Obteniendo un *desplazamiento* de 32 bits a partir de la **extensión** del campo de operando inmediato (**inmed**) de la instrucción
    - **Sumando** base y desplazamiento.
  - Leer dato ubicado en la **memoria de datos** cuya dirección es la anteriormente calculada
  - Almacenar el dato leído de memoria en el registro cuyo identificador se especifica en el campo **rt** de la instrucción



Diseño de la ruta de datos monociclo

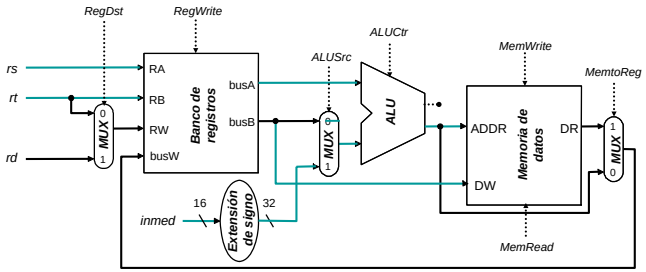
Interconexión necesaria para ejecutar sw

- La **instrucción de almacenaje** (**sw**) implica:
  - Memoria(BR(rs) + SignExt(inmed)) ← BR(rt)
  - Leer el dato almacenado en el registro cuyo identificador se especifica en el campo **rt** de la instrucción
  - Calcular la dirección efectiva de memoria:
    - Leyendo el *registro base* cuyo identificador se ubica en el campo **rs** de la instrucción
    - Obteniendo un *desplazamiento* de 32 bits a partir de la **extensión** del campo de operando inmediato (**inmed**) de la instrucción
    - **Sumando** base y desplazamiento.
  - Almacenar el dato leído en la **memoria de datos** en la dirección anteriormente calculada



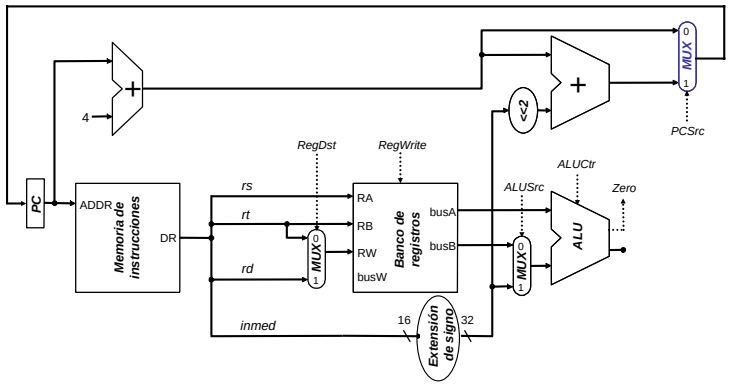
Diseño de la ruta de datos monociclo

- La **instrucción de almacenaje** (**sw**) implica:
  - Memoria(BR(rs) + SignExt(inmed)) ← BR(rt)
  - Leer el dato almacenado en el registro cuyo identificador se especifica en el campo **rt** de la instrucción
  - Calcular la dirección efectiva de memoria:
    - Leyendo el *registro base* cuyo identificador se ubica en el campo **rs** de la instrucción
    - Obteniendo un *desplazamiento* de 32 bits a partir de la **extensión** del campo de operando inmediato (**inmed**) de la instrucción
    - **Sumando** base y desplazamiento.
  - Almacenar el dato leído en la **memoria de datos** en la dirección anteriormente calculada



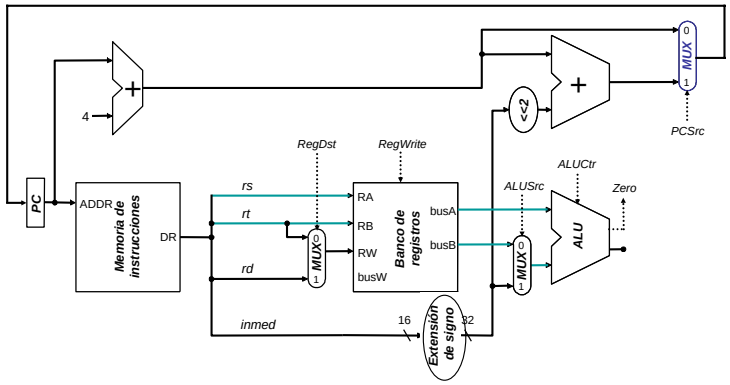
Diseño de la ruta de datos monociclo

- La **instrucción de salto condicional** (beq) implica
  - si (BR(rs) = BR(rt)) entonces (PC ← PC + 4·SignExp(inmed))



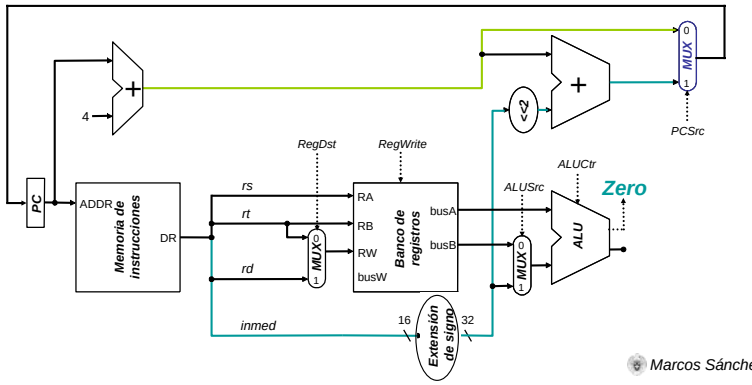
Diseño de la ruta de datos monociclo

- La **instrucción de salto condicional** (beq) implica:
  - Leer dos registros cuyos identificadores se ubican en los campos **rs** y **rt** de la instrucción:

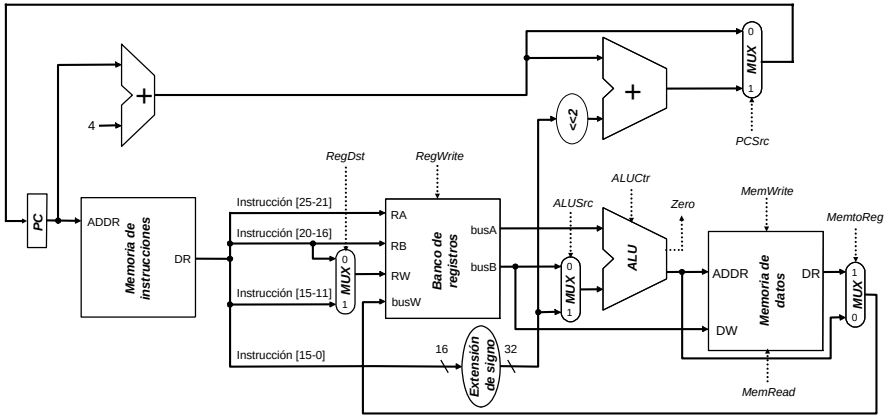


Diseño de la ruta de datos monociclo

- La **instrucción de salto condicional** (beq) implica
  - Comparar la igualdad de sus contenidos y en función del resultado:
    - **No hacer nada** o
    - **Sumar** al contador del programa un *desplazamiento* de 32 bits obtenido a partir de la **extensión** del campo de operando inmediato (**inmed**) de la instrucción



Diseño de la ruta de datos monociclo



Diseño del controlador monociclo

- Para poder realizar la máquina de control es importante saber los valores de codificación de *op* y *funct*

Diseño del controlador monociclo

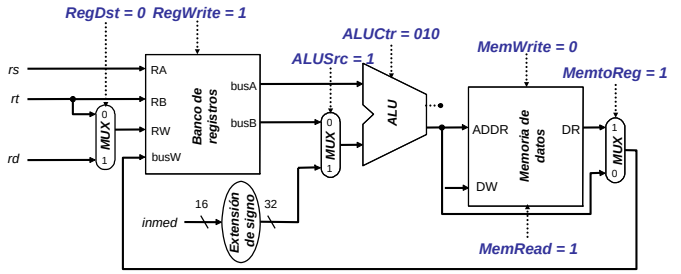
- La tarea del **controlador** es:
  - Los valores de las señales de control dependen de la instrucción que se está ejecutando
  - Seleccionar las operaciones a realizar por los módulos multifunción (ALU, read/write, ...)
  - Controlar el flujo de datos, controlando la entrada de selección de los multiplexores y la señal de carga de los registros

Diseño del controlador monociclo

Instrucción de carga (lw)

rt ← Memoria(rs + SignExt(inmed)), PC ← PC + 4

RegDest ← 0, RegWrite ← 1, ALUsrc ← 1, ALUctr ← 010, MemWrite ← 0, MemRead ← 1, MemtoReg ← 1 , PCSrc ← 0

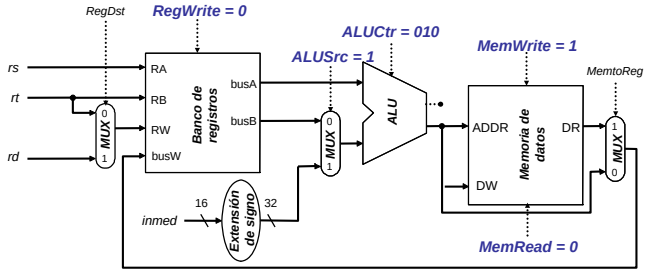


Diseño del controlador monociclo

Instrucción de almacenaje (sw)

Memoria(rs + SignExt(inmed)) ← rt, PC ← PC + 4

RegDest ←X, RegWrite ← 0, ALUsrc ← 1, ALUctr ← 010, MemWrite ← 1, MemRead ← 0, MemtoReg ← X , PCSrc ← 0

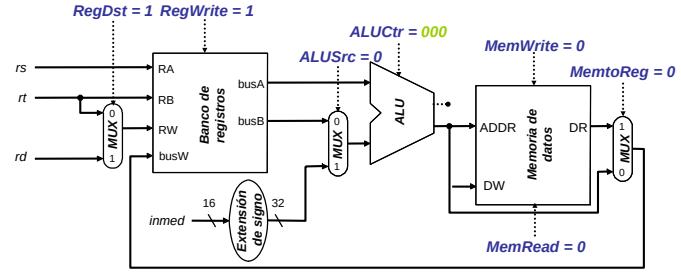


Diseño del controlador monociclo

Instrucción aritmético-lógicas

rd ← rs and rt, PC ← PC + 4

RegDest ← 1, RegWrite ← 1, ALUsrc ← 0, ALUctr ← 000, MemWrite ← 0, MemRead ← 0, MemtoReg ← 0 , PCSrc ← 0



Diseño del controlador monociclo

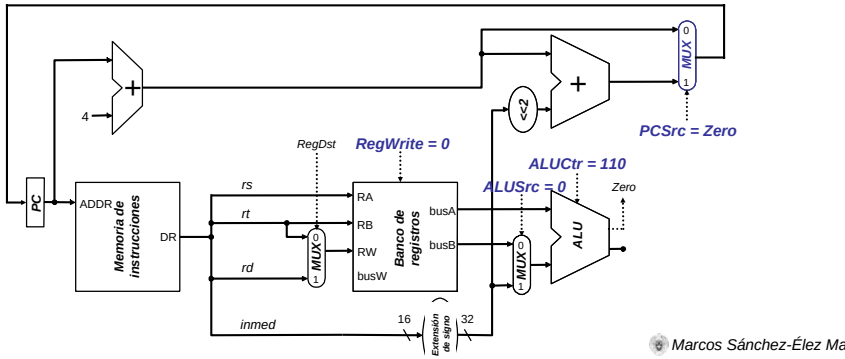
Instrucción de salto condicional (beq)

si (rs = rt) entonces (PC ← PC + 4 + 4·SignExp(inmed))

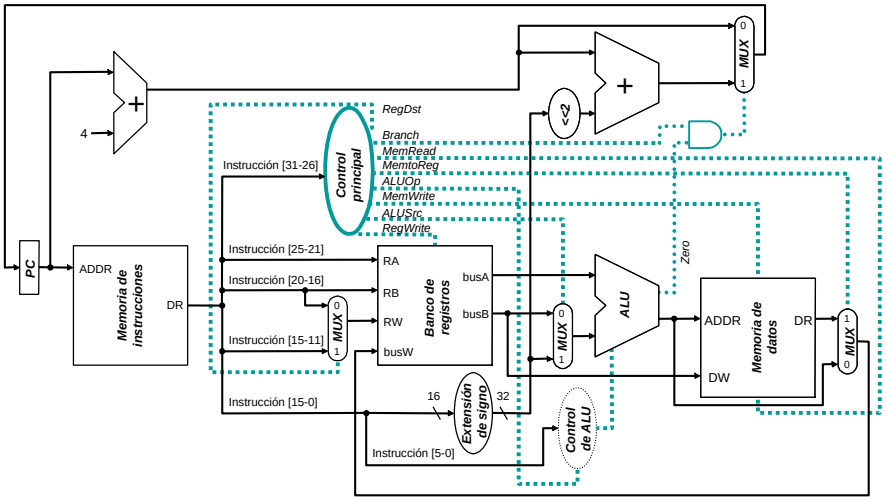
en otro caso PC ← PC + 4

RegDest ←X, RegWrite ← 0, ALUsrc ← 0, ALUctr ← 110, PCSrc ← Zero,

MemWrite ← 0, MemRead ← 0, MemtoReg ← X



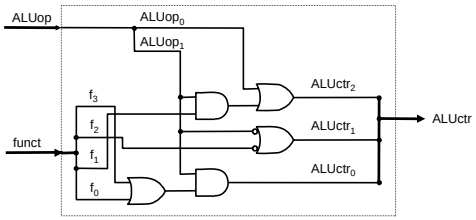
Diseño del controlador monociclo



Diseño del controlador monociclo

Control de la ALU

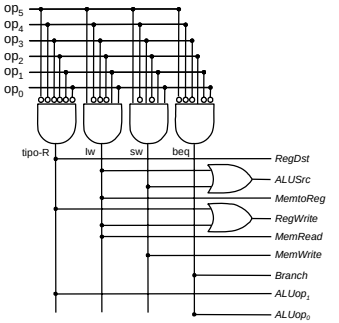
op	funct	ALUop	ALUctr
100011 (lw)	XXXXXX	00	010
101011 (sw)		00	010
000100 (beq)		01	110
000000 (tipo-R)	100000 (add)	11	010
	100010 (sub)	11	110
	100100 (and)	11	000
	100101 (or)	11	001
	101010 (slt)	11	111



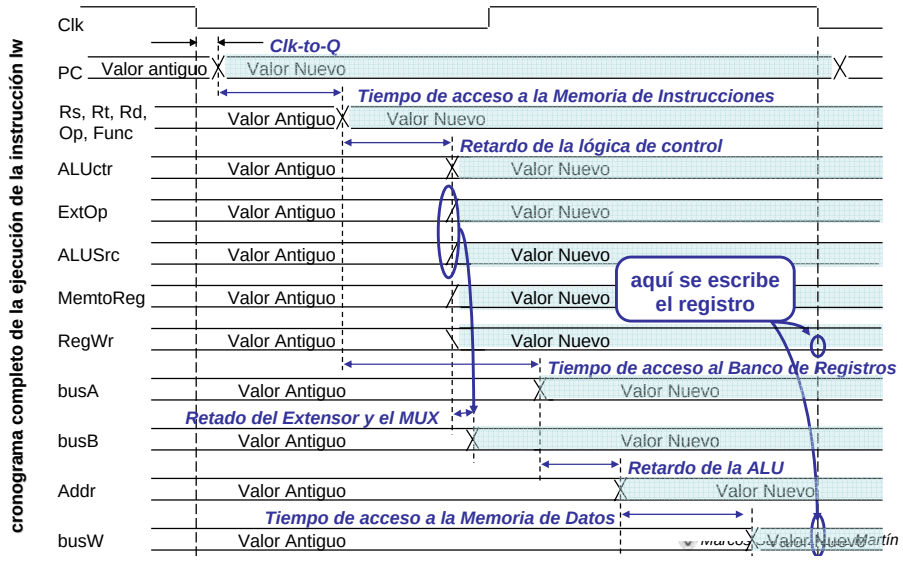
Diseño del controlador monociclo

Control principal

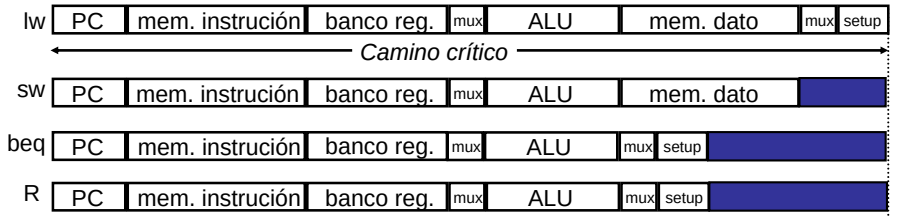
op	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Brach	ALUop
100011 (lw)	0	1	1	1	1	0	0	00
101011 (sw)	X	1	X	0	0	1	0	00
000100 (beq)	X	0	X	0	0	0	1	01
000000 (tipo-R)	1	0	0	1	0	0	0	10



Procesador monociclo



Procesador monociclo

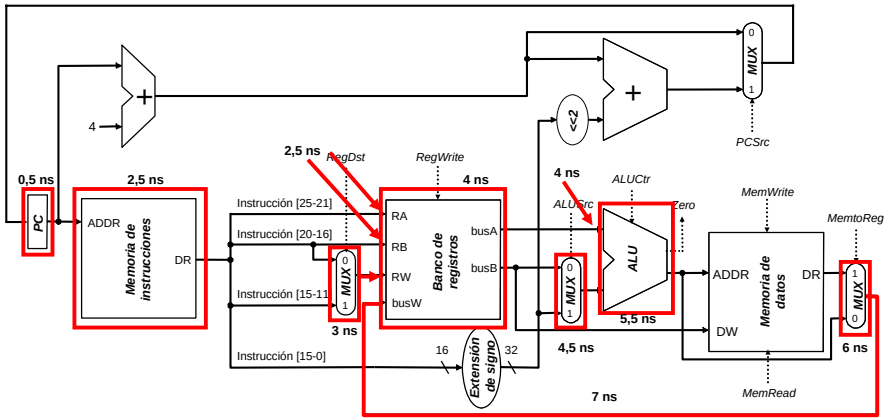


- **Problema**: en un controlador monociclo:
  - El reloj debe tener igual periodo que la instrucción más lenta
  - No se puede reusar hardware
- **Solución**: dividir la ejecución de la instrucción en varios ciclos más pequeños:
  - Cada instrucción usará el número de ciclos que necesite
  - Un mismo 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.

Procesador monociclo

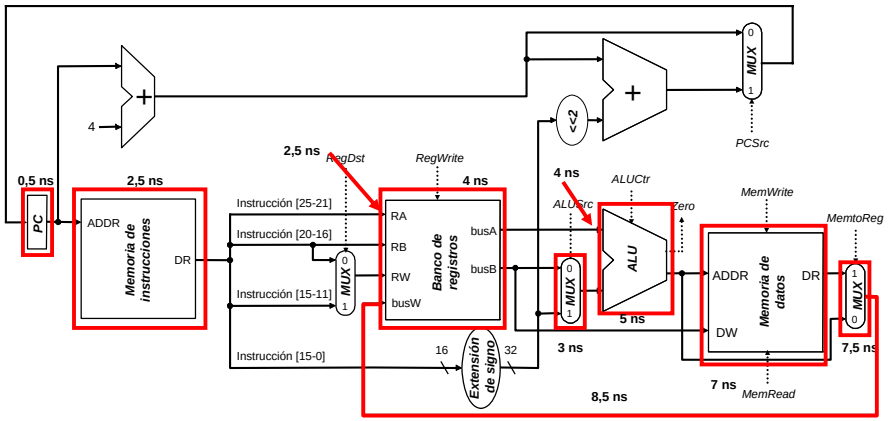
- Ejemplo:
  - Multiplexores: 0,5 ns
  - Memorias (lectura): 2 ns
  - Banco de Registros (lectura): 1,5 ns
  - ALU: 1 ns
  - Cables, extensor de signo … : 0ns
  - La señal debe permanecer estable durante 1 ns antes de la escritura

Procesador monociclo



ARITMETICOLÓGICA

Procesador monociclo



LOAD

Solución: Procesador multiciclo