Diseño de la Ruta de Datos y la Unidad de Control

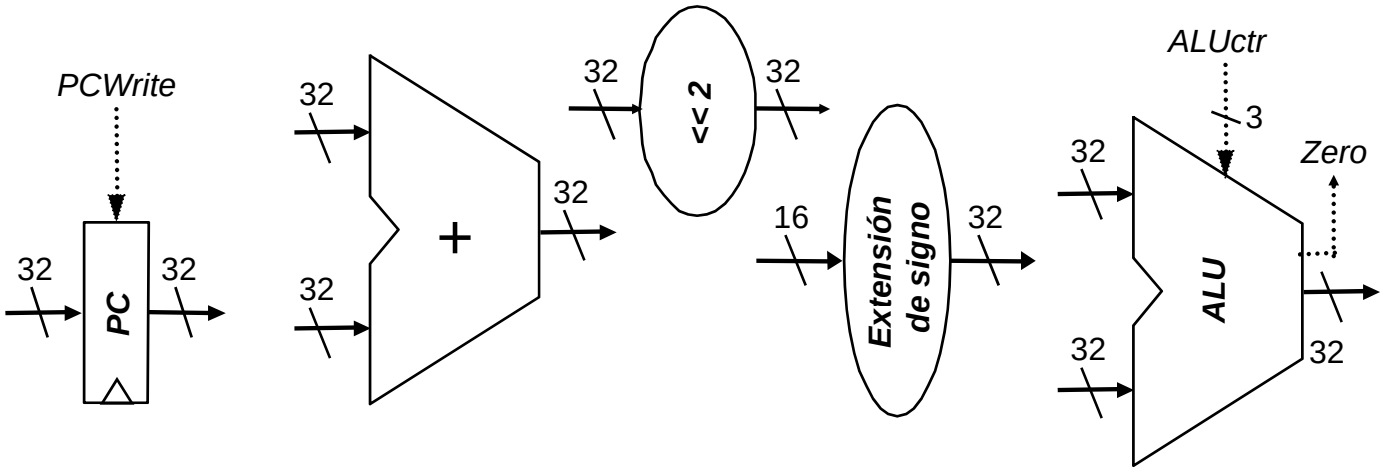
Ruta de datos monociclo

Contenidos

- Introducción
  - Importancia del diseño del procesador. Metodología de diseño de un procesador. Arquitectura MIPS: formato de la instrucción máquina y repertorio de instrucciones.
- Diseño de la ruta de datos monociclo
  - Componentes de la ruta de datos. Ensamblaje de la ruta de datos. Ruta de datos monociclo: puntos de control.
- Diseño del controlador monociclo
  - Determinación de los valores de los puntos de control. Control global vs. Control local. Ruta datos monociclo + controlador. Temporización monociclo.
- Diseño de la ruta de datos (multiciclo)
  - Ruta de datos multiciclo: con y sin buses.
- Diseño del controlador (multiciclo)
  - Diagrama de estados del controlador. El controlador como una FSM. Alternativas de implementación del controlador

Diseño de la ruta de datos monociclo

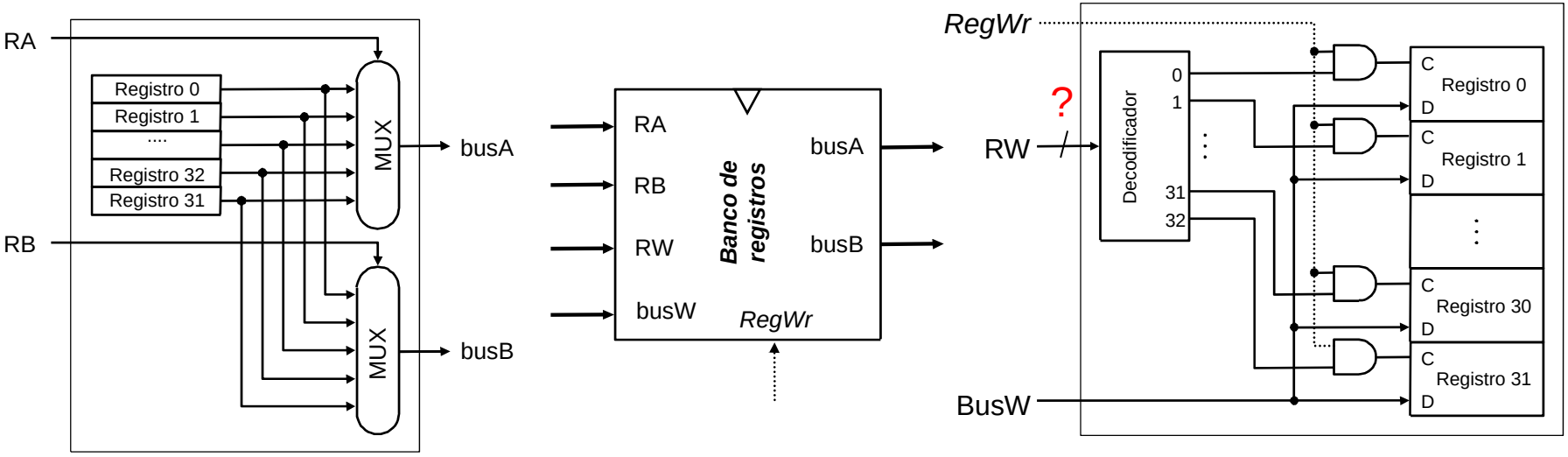
- Para implementar el subconjunto del repertorio del MIPS en una implementación monociclo se requieren (el tamaño de palabra es de 32 bits):
  - **Memoria de instrucciones**
  - **Memoria de datos**
  - **32 registros de datos:** visibles por el programador.
  - **Contador de programa**
  - **2 Sumadores:** 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.



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 ___ bits
  - 1 entradas de datos de ___ bits
  - ___ entradas de ___ 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)

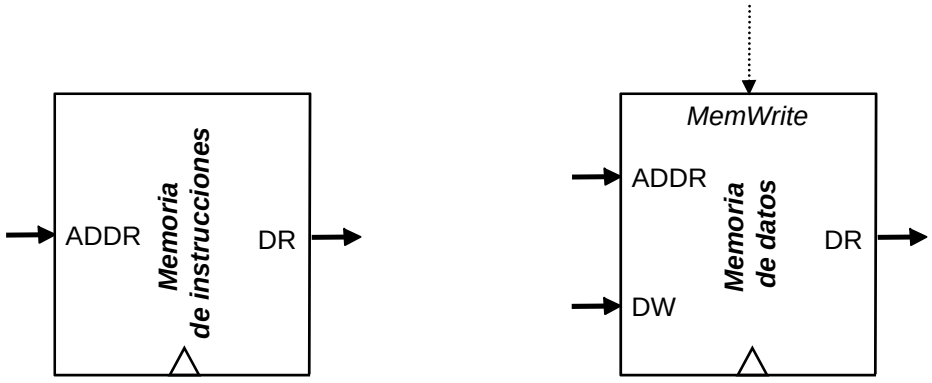


Mecanismo de lectura

Mecanismo de escritura

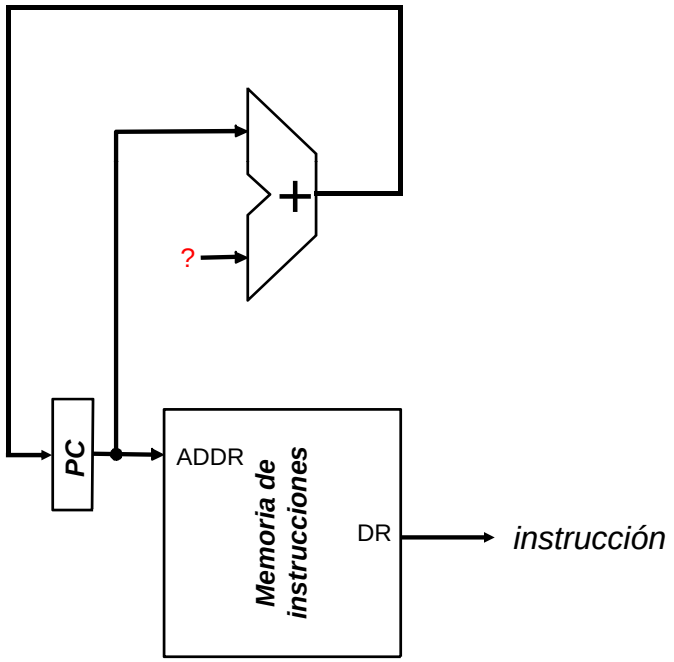
Diseño de la ruta de datos monociclo

- 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)
    - 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:
    - Memoria de instrucciones
    - Memoria de datos



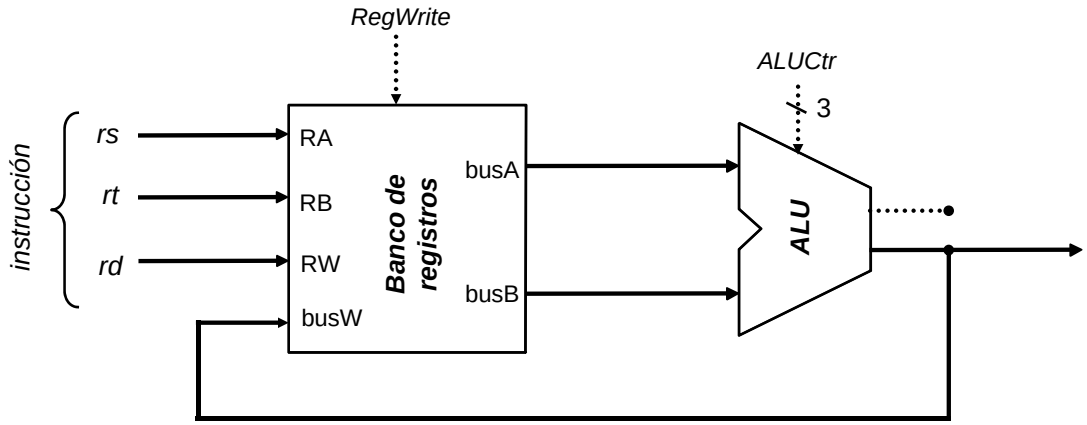
Diseño de la ruta de datos monociclo

- 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 el ***secuenciamiento implícito***:
  - Actualizar el **contador de programa** para que apunte a la siguiente instrucción (sumando ? por ser una memoria direccionable por bytes y una arquitectura con tamaño de palabra de ? bits

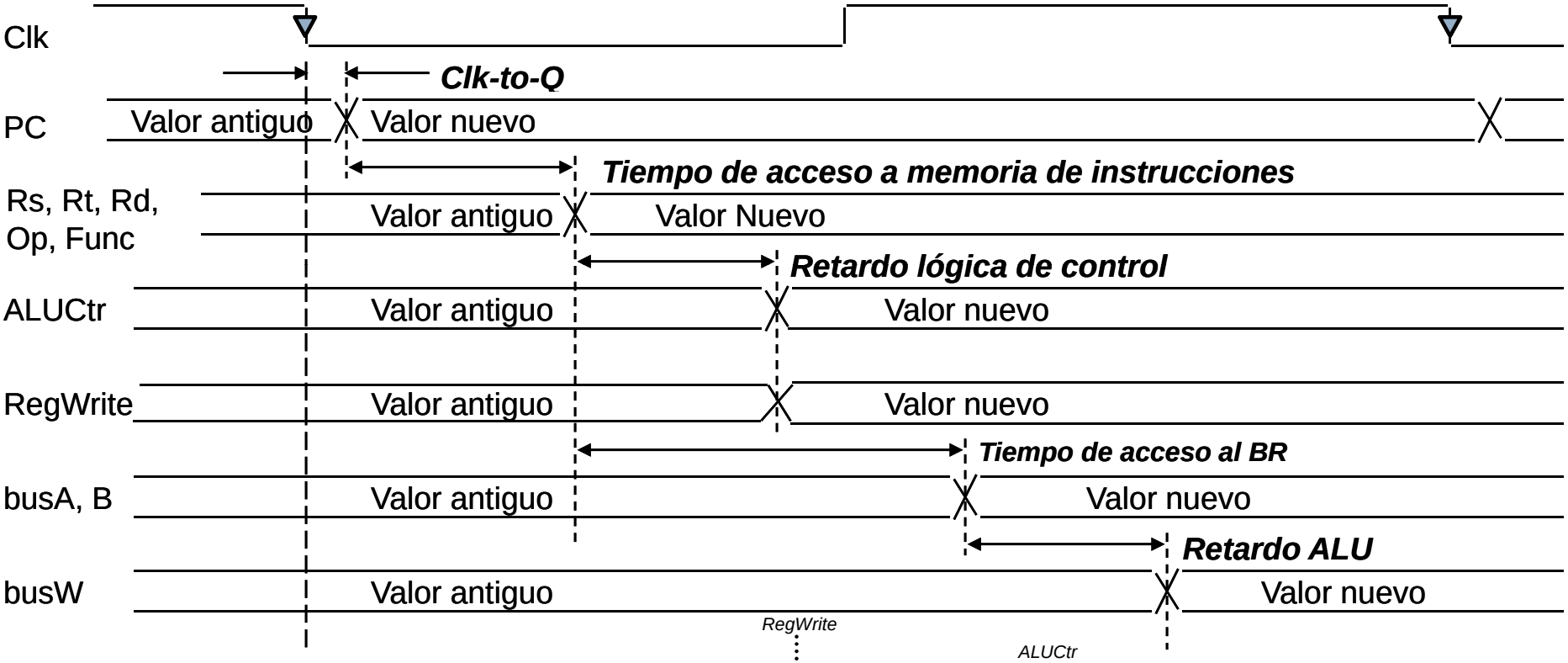


Diseño de la ruta de datos monociclo

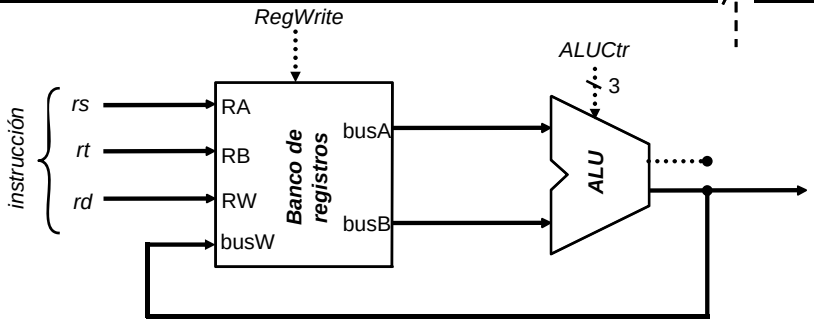
- Las instrucciones **aritmético lógicas** (tipo-R) implican:
  - **BR(rd) ← BR(rs) funct BR(rt)**
  - Leer dos registros cuyos identificadores se ubican en los campos **rs** y **rt** de la instrucción:
  - Operar sobre ellos según el contenido del campo de código de operación aritmética (**funct**)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

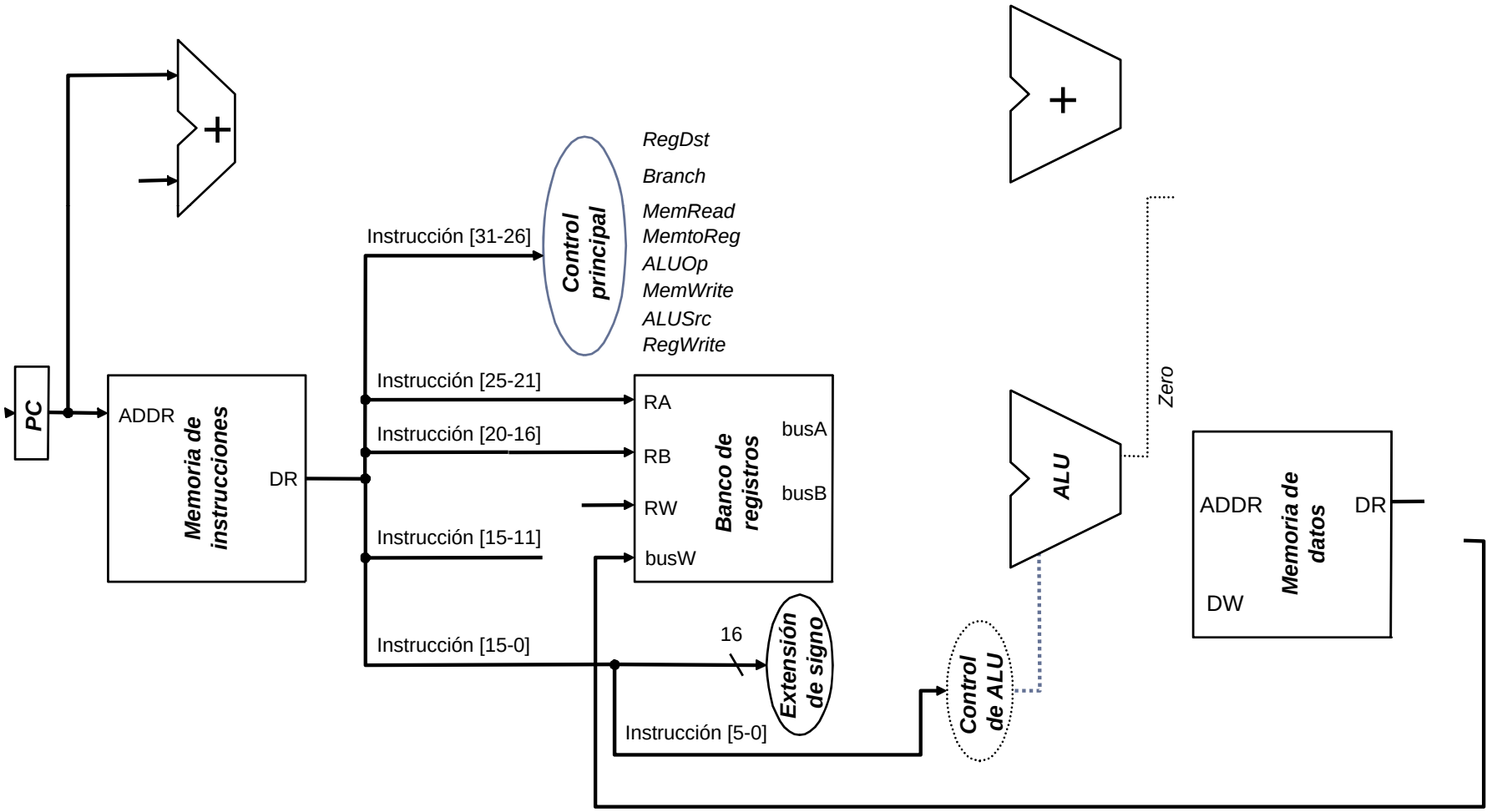


cronograma de una operación arimético-lógica



Señala la escritura

Diseño de la ruta de datos monociclo - Completar



Diseño de la ruta de datos monociclo – ld/st

- 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
- 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 - beq

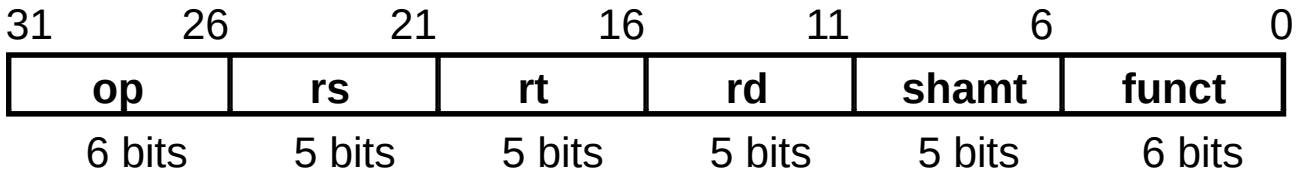
- La **instrucción de salto condicional** (beq) implica
  - si (BR(rs) = BR(rt)) entonces (PC ← PC + 4·SignExp(inmed))
  - Leer dos registros cuyos identificadores se ubican en los campos **rs** y **rt** de la instrucción:
  - Comparar la igualdad de sus contenidos y en función del resultado:
    - No 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

Señales de Control – arit / br

Aritméticas:

rd ← rs and rt, PC ← PC + 4

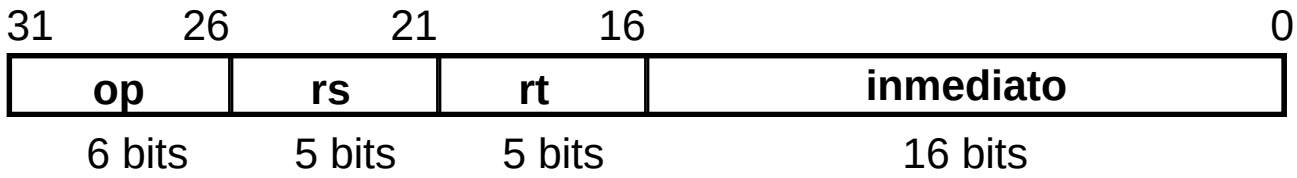
Tipo R:
aritmético-lógicas



Salto:

beq rs, rt, inmed si (rs = rt) entonces (PC ← PC + 4 + 4·SignExp(inmed))
en otro caso PC ← PC + 4

Tipo I:
con memoria
salto condicional

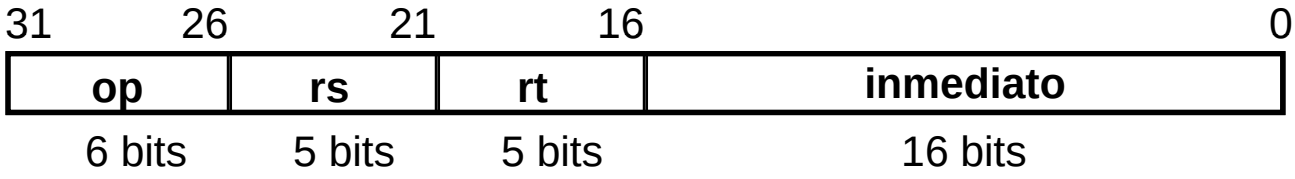


Señales de Control – ld/st

Load

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

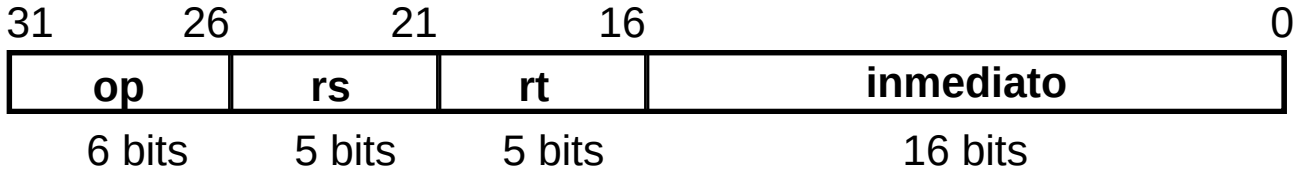
Tipo I:
con memoria
salto condicional



Store

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

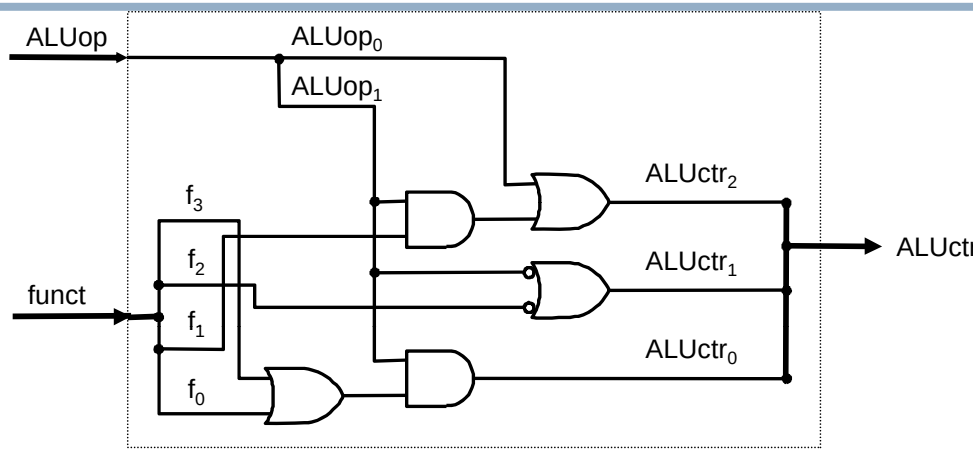
Tipo I:
con memoria
salto condicional



diseño del controlador (monociclo)

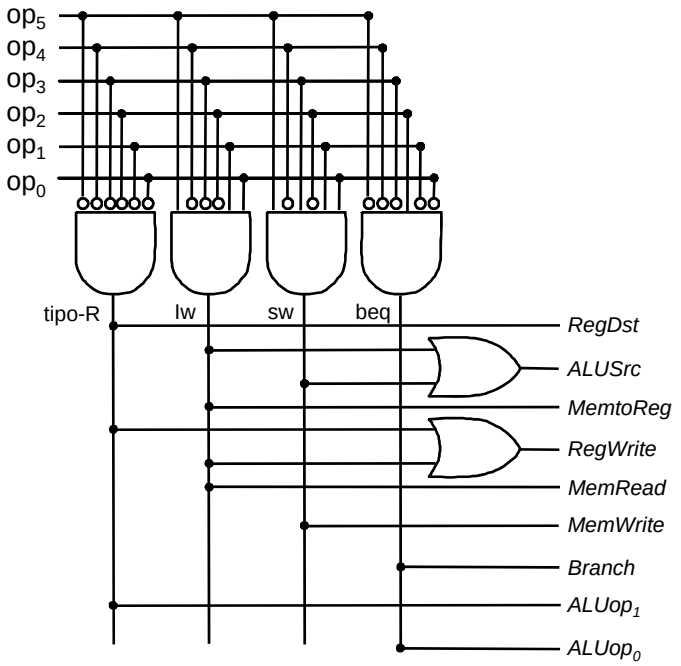
Control de la ALU

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



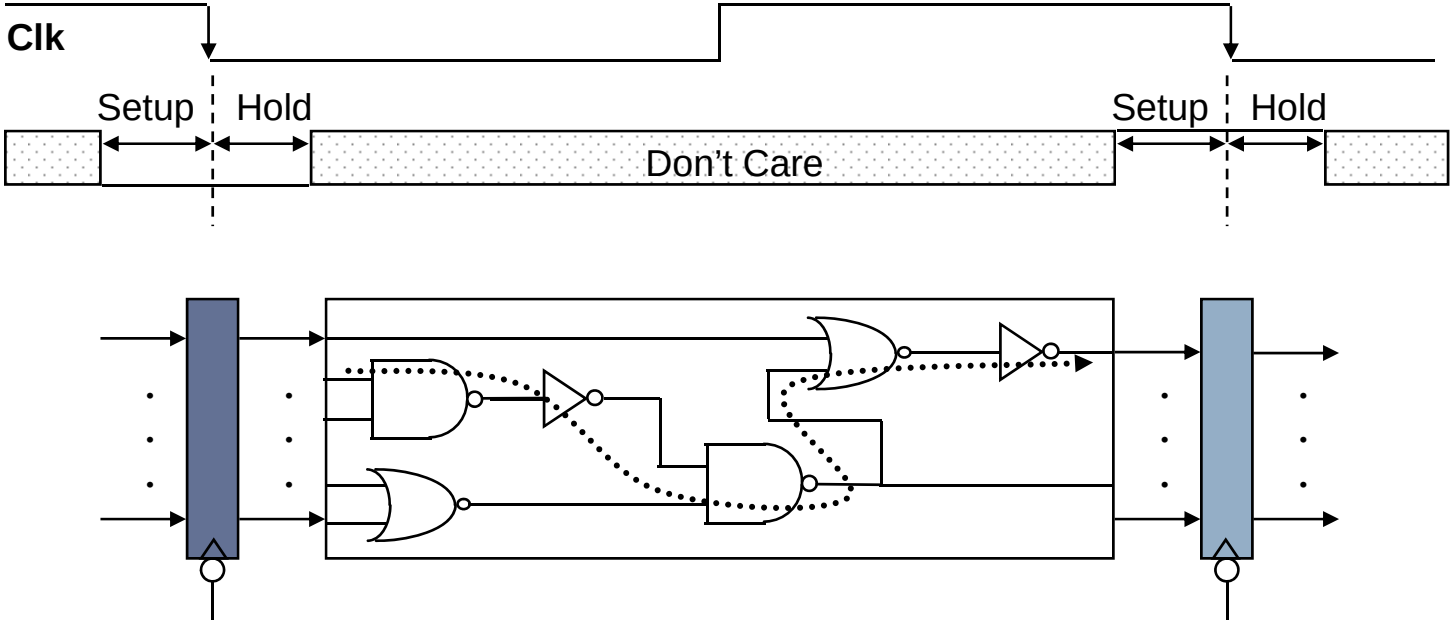
Control principal

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	11

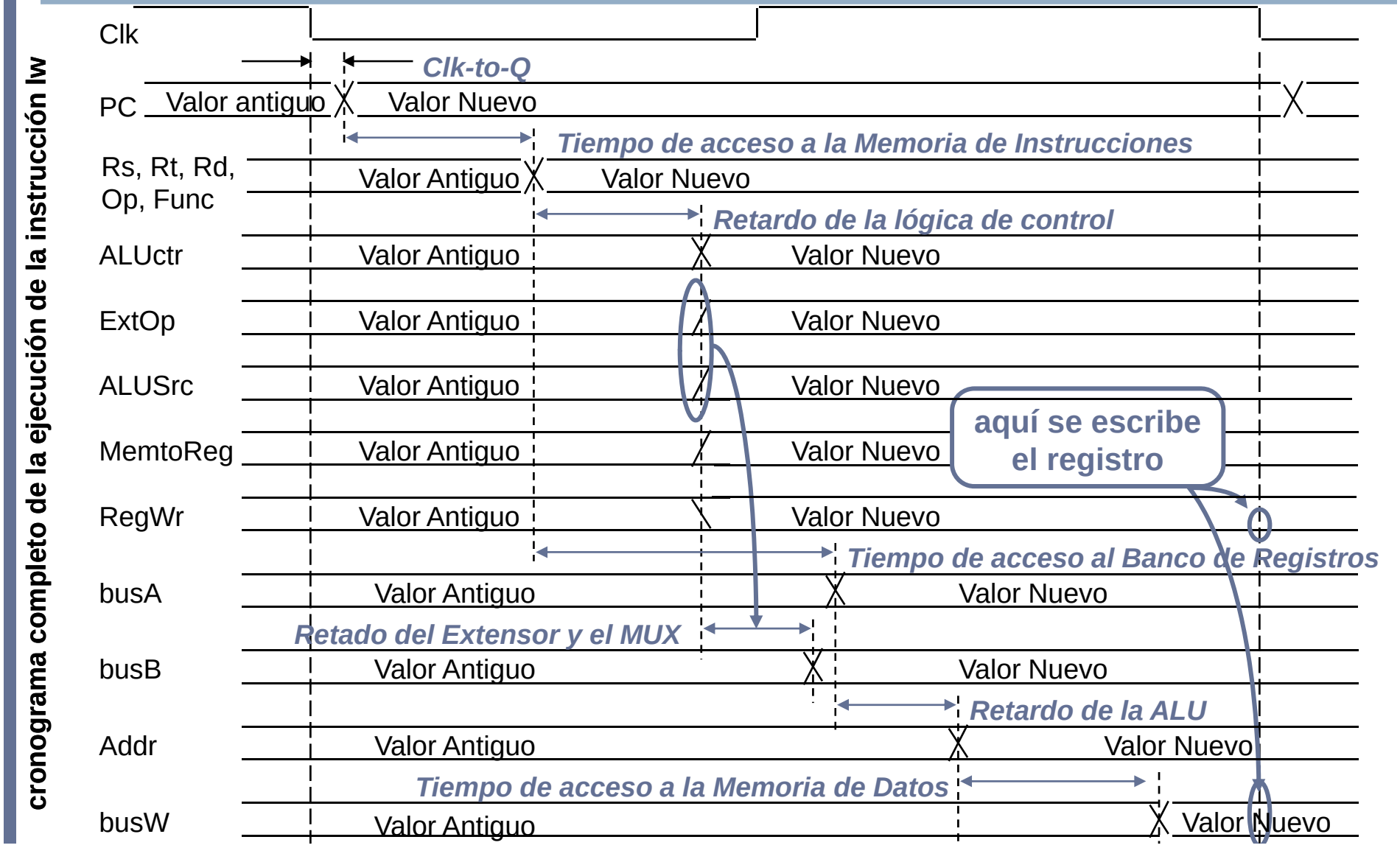


Temporización (monociclo)

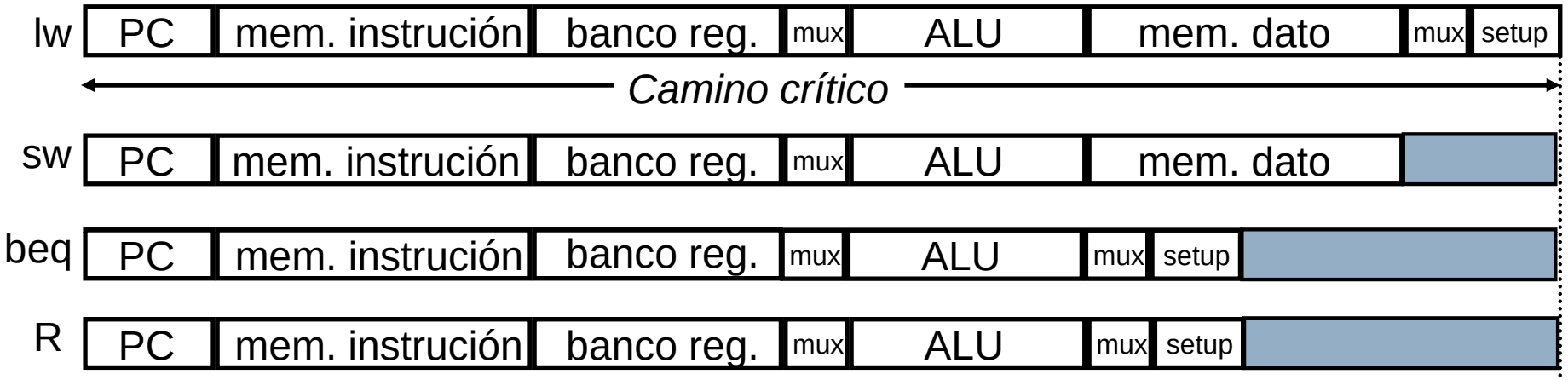
- **Ejecución típica (de una instrucción)**
  - Todos los registros se cargan simultáneamente (de modo selectivo)
  - Todos los valores se propagan a través de las redes combinacionales hasta estabilizarse en las entradas de los registros
  - Se repite indefinidamente el proceso
- Todos los elementos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



Cronograma de la ejecución de una instrucción lw



Limitaciones del MIPS monociclo



- **Problema**: en un controlador monociclo:
  - El reloj debe tener igual periodo que la instrucción más lenta
  - No se puede reutilizar 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.

Ejemplo de diseño de ruta de datos

- Problema:
  - Crear un procesador capaz de multiplicar y trasponer matrices (DSP: Digital Signal Processing)
  - El procesador debe poseer siguiente juego de instrucciones:
    - Carga de un valor en un registro **LOAD #inmediato, Ri**
    - Lectura de una posición de memoria **READ Rf, Rd** (lee el valor de la memoria que se encuentra en Rf y lo escribe en Rd, además suma uno al valor almacenado en Rf).
    - Lectura especial de memoria **READE Rf, Rd** (lee el valor de la memoria que se encuentra en Rf y lo escribe en Rd, además suma 'n' al valor almacenado en Rf).
    - Escritura en una posición de memoria **WRITE Rf, Rd** (escribe e valor almacenado en Rf en la posición de memoria Rd, además suma uno al valor almacenado en Rd).
    - Escritura especial de memoria **WRITEE Rf, Rd** (escribe e valor almacenado en Rf en la posición de memoria Rd, además suma 'n' al valor almacenado en Rd).
    - Suma **ADD Rf1, Rf2, Rd** => Rd = Rf1+Rf2
    - Multiplicación **MUL Rf1, Rf2, Rd** => Rd = Rf1*Rf2
  - Desarrollar el hardware necesario para que las instrucciones sean decodificadas y realicen su función.