Dept. de Teoría de la Señal y Comunicaciones y Sistemas Telemáticos y Computación
Área de Telemática (GSyC)

El procesador

Katia Leal Algara
katia.leal@urjc.es
http://gsyc.escet.urjc.es/~katia/

Objetivos

- Principios y técnicas utilizadas en la implementación de un procesador
  - Ruta de datos
  - Ruta de control
- Nos centraremos en el estudio de una arquitectura MIPS simplificada, el *nanoMIPS*

Ejecución de una instrucción

- **Fetch (F)**: buscar en memoria la instrucción apuntada por el PC. Actualización del PC
- **Decode (D)**: decodificación de la instrucción, separar los diferentes campos. Si es necesario, se leen 1 o 2 operandos de los registros
- **Execution (X)**: ejecución de la operación indicada en el opcode
- **Memory Access (M)**: si es necesario, acceder a memoria para leer o escribir
- **Writeback (W)**: si es necesario, se vuelca un resultado a un registro

Instrucciones tipo I: **Load/Store**

F	- Búsqueda en la MI de la instrucción apuntada por PC. - Actualización del PC	MI[PC] PC ← PC+4
D	- Decodificación y lectura de registros - Extensión de signo para inmediato	[RS] ext(Inmediato) Si es **Store**: [RD]
X	- Suma en la ALU del registro base + el desplazamiento	[RS]+ext(Inmediato)
M	- Acceso a la dirección de memoria calculada en la etapa anterior	**Load**: MD[[RS]+ext(Inmediato)] **Store**: MD[[RS]+ext(Inmediato)] ←[RD]
W	- En caso se Load, se escribe el contenido de memoria en el registro RD	[RD] ← MD[[RS]+ext(Inmediato)]

Instrucciones tipo I: **Aritmético-Lógicas**

F	- Búsqueda en la MI de la instrucción apuntada por PC. - Actualización del PC	MI[PC] PC ← PC+4
D	- Decodificación y lectura del registro RS - Extensión de signo para inmediato	[RS] ext(Inmediato)
X	- La ALU realiza la operación que indique el opcode	[RS] OP ext(Inmediato)
M		
W	- Se escribe el resultado de la operación en el registro RD	[RD] ← [RS] OP ext(Inmediato)

Instrucciones tipo I: **Saltos condicionales**

F	- Búsqueda en la MI de la instrucción apuntada por PC - Actualización del PC	MI[PC] PC ← PC+4
D	- Decodificación de la instrucción - Lectura de los registros RS y RD - Extensión de signo para inmediato	[RS] [RD] ext(Inmediato)
X	- Cálculo de la dirección de salto: PC + Inmediato - Evaluación condición de salto en RS y RD - Si evaluación positiva, se carga el PC con la dirección de salto	[PC] + ext(Inmediato) cond ([RS],[RD]) Si con = TRUE PC ← [PC] + ext(Inmediato)
M		
W		

Instrucciones tipo I: **Saltos incondicionales**

F	- Búsqueda en la MI de la instrucción apuntada por PC - Actualización del PC	MI[PC] PC ← PC+4
D	- Decodificación de la instrucción - Lectura del registro RS	 [RS]
X	- Se carga el PC con el valor del registro RS	PC ← [RS]
M		
W		

Instrucciones tipo R: **Aritmético-Lógicas**

F	- Búsqueda en la MI de la instrucción apuntada por PC. - Actualización del PC	MI[PC] PC ← PC+4
D	- Decodificación - Lectura de los registros RS y RD	[RS] [RD]
X	- La ALU realiza la operación indicada por la combinación del opcode y del campo Function	[RS] OP [RT]
M		
W	- Se escribe el resultado de la operación en el registro RD	[RD] ← [RS] OP [RT]

Instrucciones tipo J: **Saltos incondicionales** con direccionamiento relativo al PC

F	- Búsqueda en la MI de la instrucción apuntada por PC. - Actualización del PC	MI[PC] PC ← PC+4
D	- Decodificación - Lectura del registros RS - Extensión de signo para Offset	 [RS] ext(Offset)
X	- Se suma al PC el Offset para obtener la dirección del salto - Se carga el PC con el valor obtenido para la dirección de salto	[PC] + ext(Offset) PC ← [PC] + ext(Offset)
M		
W		

Diseño de un procesador

- Dos grandes módulos dentro del procesador
  - **Ruta de datos** → combinacional
  - **Unidad de control** → secuencial ó combinacional
- Un procesador es un circuito digital
  - Parte **combinacional** + Parte **secuencial**
  - Una parte opera sobre valores y la otra contiene estado
  - ALU Vs Memoria, registros
- **Sistema combinacional**
  - Sus salidas son función exclusiva del valor de sus entradas en un momento dado, sin que intervengan estados anteriores de las entradas o de las salidas
- **Sistema secuencial**
  - Los valores de las salidas, en un momento dado, no dependen exclusivamente de los valores de las entradas en dicho momento, sino también dependen del estado anterior o estado interno
  - **Gobernados por una señal de reloj!**

Procesador secuencial

- Hasta que no termina de ejecutar una instrucción no comienza a ejecutar la siguiente
- Según el método de temporización escogido:
  - **Procesador monociclo**
  - **Procesador multiciclo**
- **Procesador monociclo**
  - Cada instrucción se completa en un único ciclo de reloj
  - CPI = 1
  - La duración del ciclo de reloj viene fijada por la instrucción que más tarde en ejecutarse
- **Procesador multiciclo**
  - Cada instrucción puede tardar más de un ciclo en ejecutarse
  - CPI > 1
  - La duración del ciclo de reloj es menor que para monociclo
  - Se establece que la duración de un ciclo = duración de la etapa más larga

Caso de estudio: nanoMIPS

- Repertorio de instrucciones
  - **Acceso a memoria**: `LW, SW` (tipo I)
  - **Operaciones aritmético-lógicas**: `ADD, SUB, AND, OR, SLT` (tipo R)
  - **Control de flujo**: `BEQ` (tipo I)

Instrucción	Pseudocódigo	Opcode	Funct
LW	LW RT,inmediato(RS)	100011	-
SW	SW RT,inmediato(RS)	101011	-
ADD	ADD RD,RS,RT	000000	100000
SUB	SUB RD,RS,RT	000000	100010
AND	AND RD,RS,RT	000000	100100
OR	OR RD,RS,RT	000000	100101
SLT	SLT RD,RS,RT	000000	101010
BEQ	BEQ RS,RT,destino	000100	-

Funcionamiento y diseño de la ruta de datos y la unidad de control

1. Análisis del repertorio de instrucciones a ejecutar
2. Establecer la **metodología de temporización**
3. Seleccionar los **módulos** necesarios para operar sobre y almacenar datos, y que formarán la ruta de datos teniendo en cuenta el repertorio de instrucciones y la metodología de temporización
4. **Ensamblar la ruta de datos** conectando los módulos escogidos e identificando los puntos de control
5. Determinar los valores de los **puntos de control** para cada instrucción del repertorio
6. Diseñar la unidad de control
7. Optimizar el diseño obtenido: **segmentación**

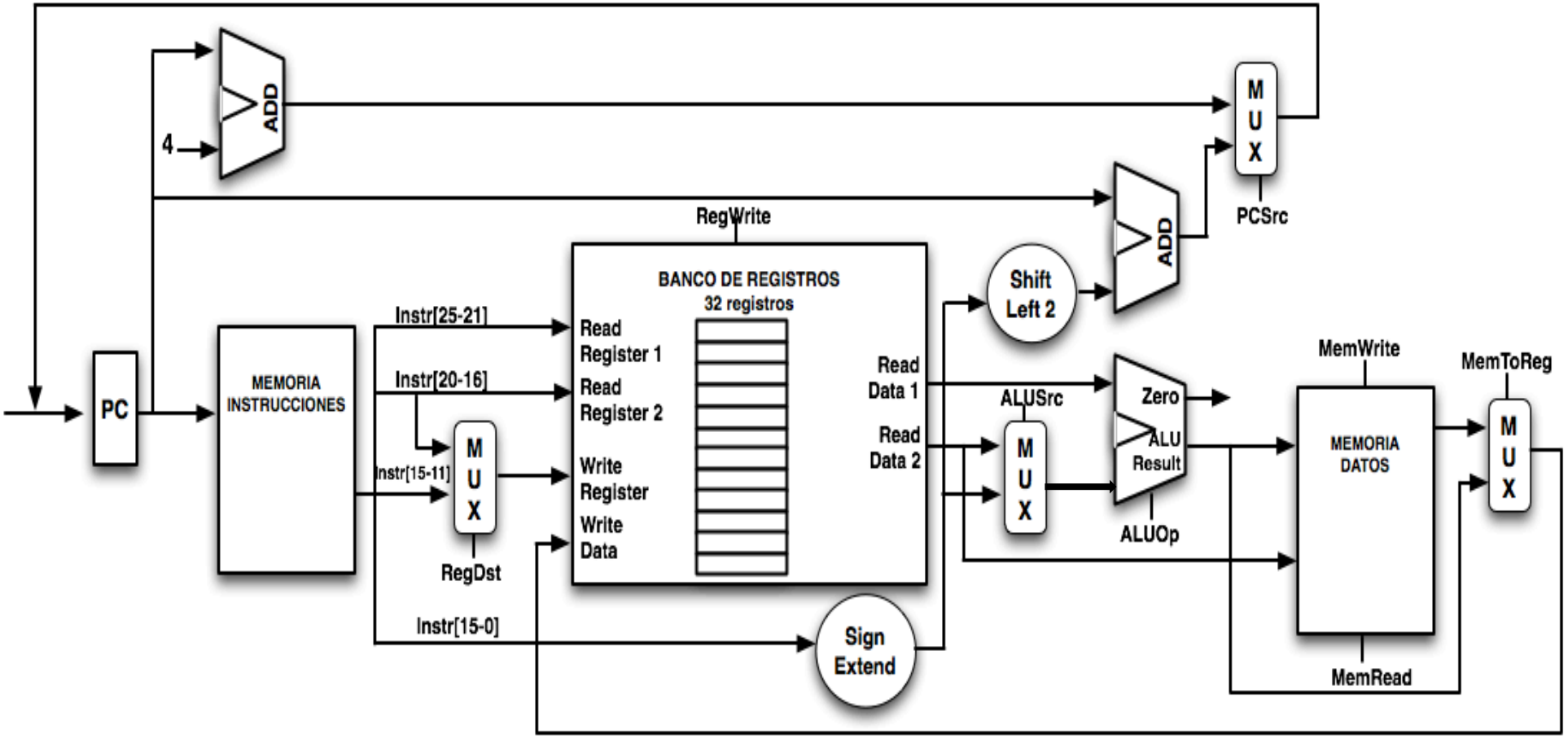
Ruta de datos del nanoMIPS monociclo

- **Contador de programa**: PC
- **Memorias separadas de instrucciones y de datos**: nunca se accede al mismo recurso más de una vez si la implementación es monociclo
- Un único **banco de 32 registros**: instrucciones tipo R, acceso simultáneo a dos registros
  - **4 entradas**: 1 entrada de datos de 32 bits y 3 entradas de 5 bits para la identificación de registros
  - **2 salidas**: dos salidas de datos de 32 bits
- **Dos sumadores**
  - Para sumar 4 al PC
  - Para sumar el desplazamiento relativo al PC
- Una **ALU** de enteros
  - BEQ, hacer una resta y comprobar si el resultado es cero
- Un **extensor de signo** para extender el signo de los datos inmediatos
- Un **desplazador a la izquierda** de 2 posiciones para recuperar los dos ceros al final del desplazamiento

Ruta de datos del nanoMIPS monociclo

- ¿Elementos necesarios para acceder y almacenar instrucciones? Memoria de instrucciones, contador de programa, sumador
- ¿Elementos necesarios para implementar operaciones de ALU tipo R? Banco de registros y ALU
- ¿Componentes necesarios para implementar carga y almacenamiento? Banco de registros, ALU, memoria de datos, extensor de signo para datos inmediatos
- ¿Elementos necesarios para implementar instrucciones de salto (*branch*)? ALU para evaluar la condición, un sumador separado para calcular la dirección de salto, un extensor de signo y un desplazador a la izquierda de 2 bits

Ruta de datos del nanoMIPS monociclo



Ejemplo

- Cálculo de la frecuencia máxima de funcionamiento de un procesador nanoMIPS monociclo
- Datos:
  - Lectura de la memoria de instrucciones: 0,3 ns
  - Lectura de la memoria de datos: 0,35 ns
  - Escritura en la memoria de datos: 0,5 ns
  - Lectura y escritura en el banco de registros: 0,05 ns
  - Operación aritmético-lógica en la ALU: 0,25 ns
  - Suma para preparar el siguiente PC: 0,1 ns
  - Suma del PC y el desplazamiento del salto: 0,1 ns

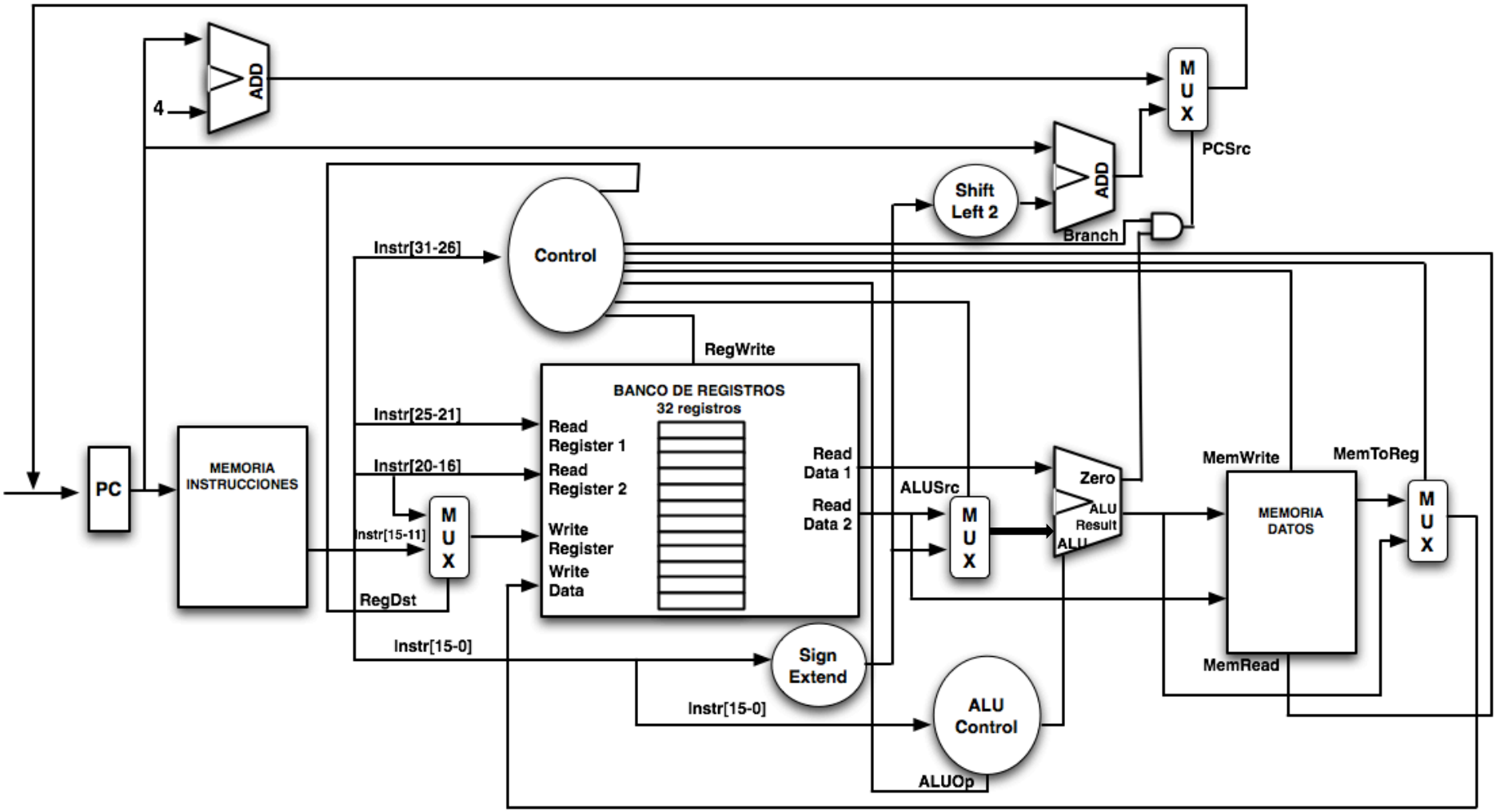
¿Cuál será el periodo de reloj del procesador? ¿Cuál será su frecuencia máxima?

$$f_{max} = 1 / T_{min} = 0{,}833 \text{ GHz}$$

Unidades de control del nanoMIPS monociclo

- Unidad de control **global**
  - Decodifica el campo *Opcode*
  - Configuración global de la ruta de datos
- Unidad de control **local** para la ALU
  - Decodifica el campo *Funct*
  - Genera la señal *ALU Control* dependiendo de la operación concreta a realizar
- **Decodificación multinivel**
  1. La U.C global decodifica la instrucción leyendo su opcode
  2. Si se trata de una instrucción de tipo R, el control local de la ALU realiza una segunda decodificación leyendo el campo Funct

Ruta de datos y Unidad de Control del nanoMIPS monociclo



Unidad de control global del nanoMIPS monociclo

- **Entrada** ← *Opcode*
- **Salidas** → Genera los valores adecuados para las diferentes señales de control
- *PCSrc* no se genera directamente
  - Cuando la instrucción es un salto, *Branch* = 1
  - *Branch* AND *resultado evaluación condición*
- *Las señales de control permanecen activas hasta que finaliza el ciclo*. Cuando llega una nueva subida del flanco de reloj, se vuelve a comenzar el proceso
- **Se suele diseñar mediante decodificadores y puertas OR**, lo cual facilita en gran medida la decodificación en el repertorio

Unidad de control local del nanoMIPS monociclo

- **Entradas** ← *ALUOp, Funct*
- **Salida** → Genera la señal ALUControl
- Se puede implementar con cualquiera de las metodologías de diseño típicas de circuitos combinacionales

Tabla de verdad: puntos de control en la ruta de datos

Instrucción	RegDest	RegWrite	ALUSrc	ALUOp
Load	0	1	1	00
Store	-	0	1	00
Artimético-lógicas	1	1	0	10
BEQ	-	0	0	01

Instrucción	MemRead	MemWrite	PCSrc	MemtoReg
Load	1	0	0	0
Store	0	1	0	-
Artimético-lógicas	0	0	0	1
BEQ	0	0	0/1 (según evaluación de la condición de salto)	-

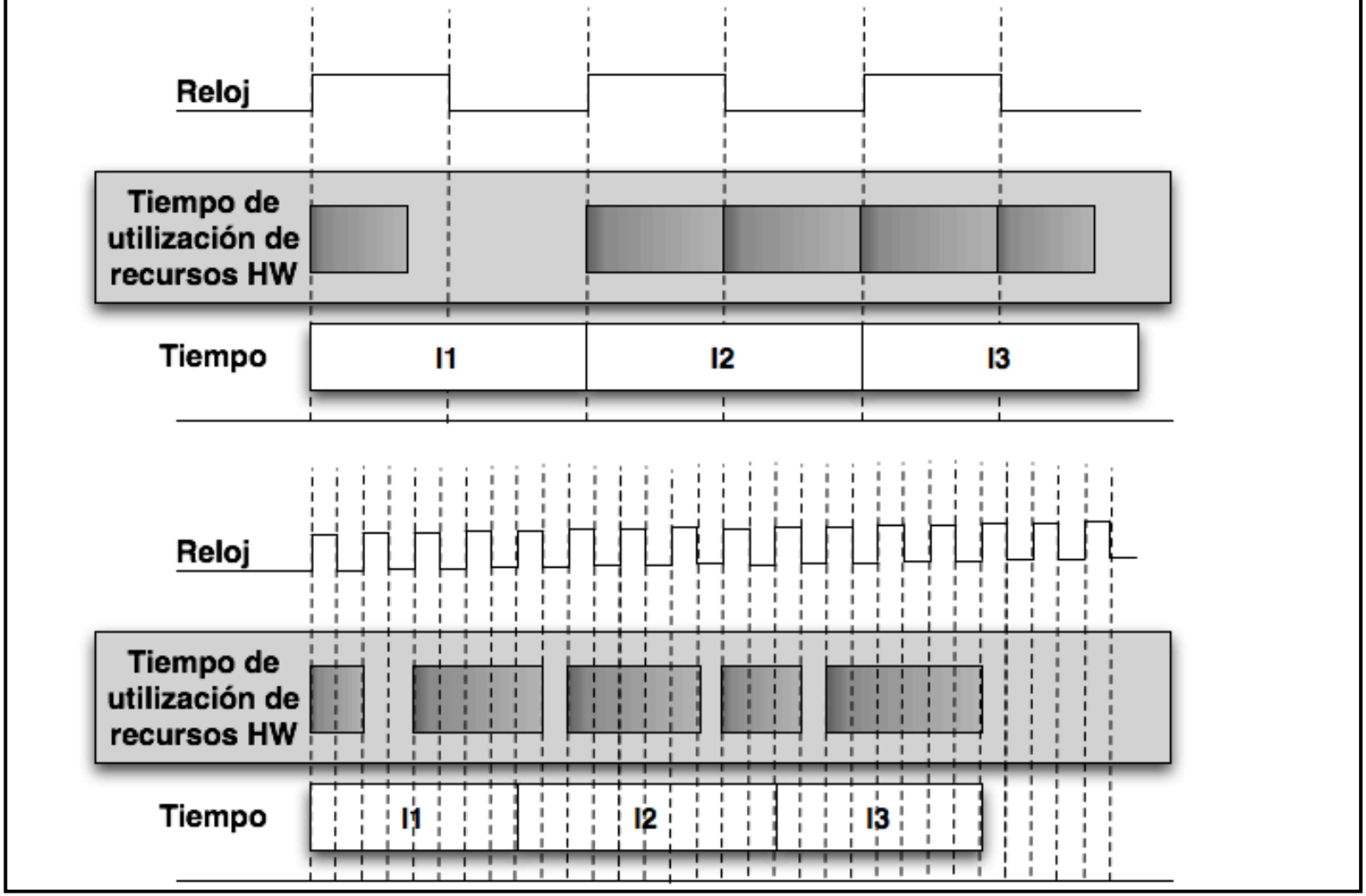
Procesador multiciclo

- Los diseños monociclo **no son eficientes**, por lo que no se emplean en la actualidad
  - Adaptación del ciclo de reloj para ejecutar la instrucción más larga → Impensable para instrucciones de coma flotante, cuya duración es mucho mayor
  - Es casi imposible optimizar la ruta de datos
  - La mayor parte del tiempo los recursos del procesador están desaprovechados

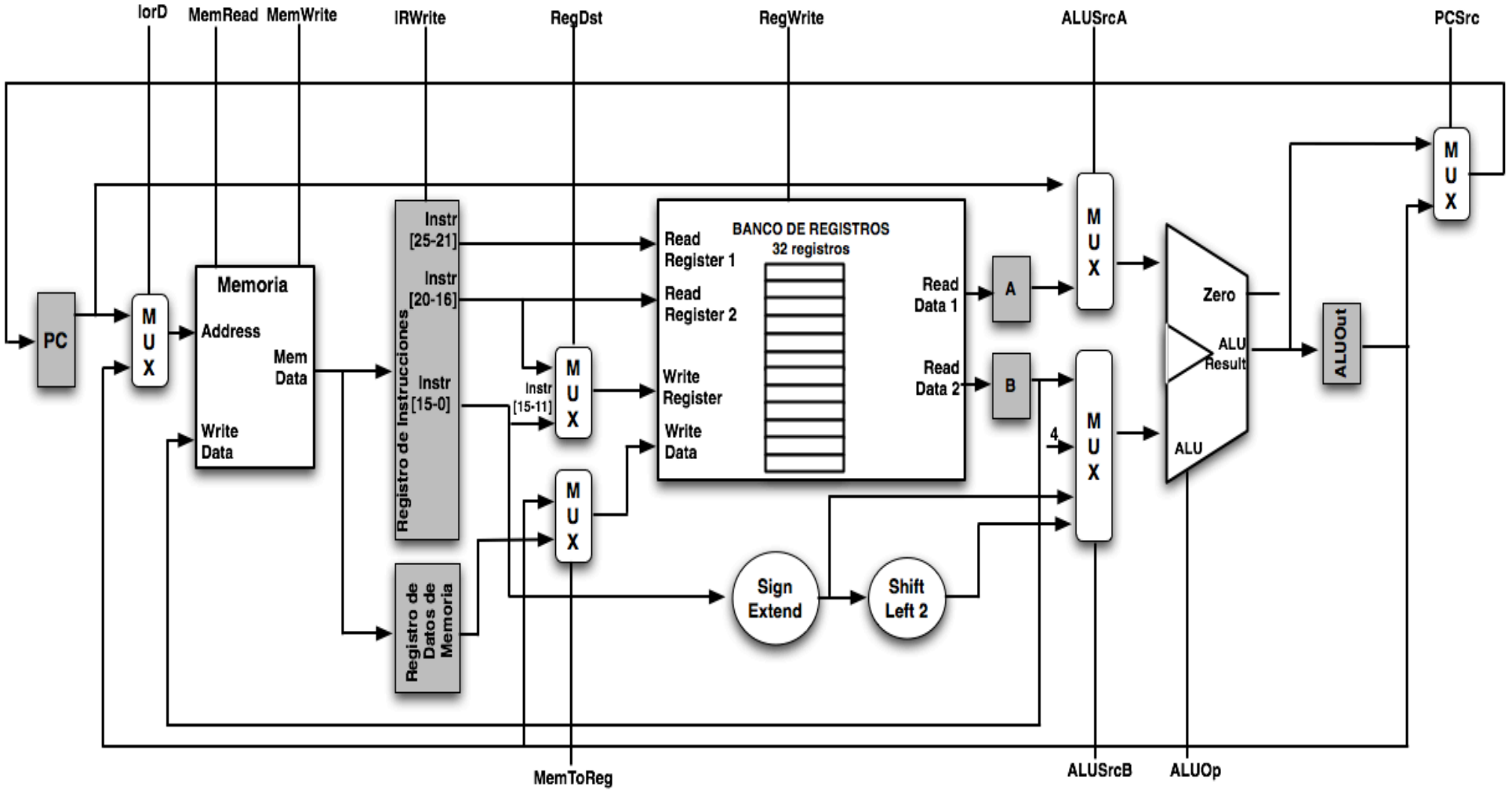
Diseño de la ruta de datos de un procesador multiciclo

- Se utiliza una **división del trabajo en etapas** típica de los procesadores con repertorio RISC
- Cada etapa está relacionada con el hardware de la ruta de datos que se utiliza
- Cada etapa debe completarse en 1 ciclo de reloj
- Se reduce el periodo del procesador
- El **CPI** medio será mayor que 1
- Cada instrucción tarda en ejecutarse tantos ciclos como sea necesario

Utilización de recursos en nanoMIPS mono y multiciclo



Ruta de datos del nanoMIPS multiciclo



Ruta de datos del nanoMIPS multiciclo

- Etapa **F** → Registro de instrucción *IR*
- Etapa **D** → Lectura de operandos, *A* y *B*
- Etapa **X** → Operandos fuente ALU en *A* y *B*
- Etapa **M** → De memoria a *MDR*
- Etapa **W** → De *ALUOut* a registro
- No son necesarios sumadores extra
- No son necesarias dos memorias separadas
- Un mismo recurso puede usarse en diferentes etapas de la ejecución de una instrucción
- ¿Cuántas veces se utiliza la ALU en la ejecución de la instrucción BEQ?

Ejemplo

- Comparación de una versión monociclo del nanoMIPS con una versión multiciclo
- Datos:
  - Lectura de la memoria: 0,3 ns
  - Escritura en la memoria: 0,45 ns
  - Lectura y escritura en el banco de registros: 0,05 ns
  - Operación aritmético-lógica en la ALU: 0,25 ns
  - Suma para preparar el siguiente PC: 0,1 ns

¿Cuál será el periodo de reloj del procesador? ¿Cuál será su frecuencia máxima?

$$f_{max} = 1 / T_{min} = 2{,}2 \text{ GHz}$$

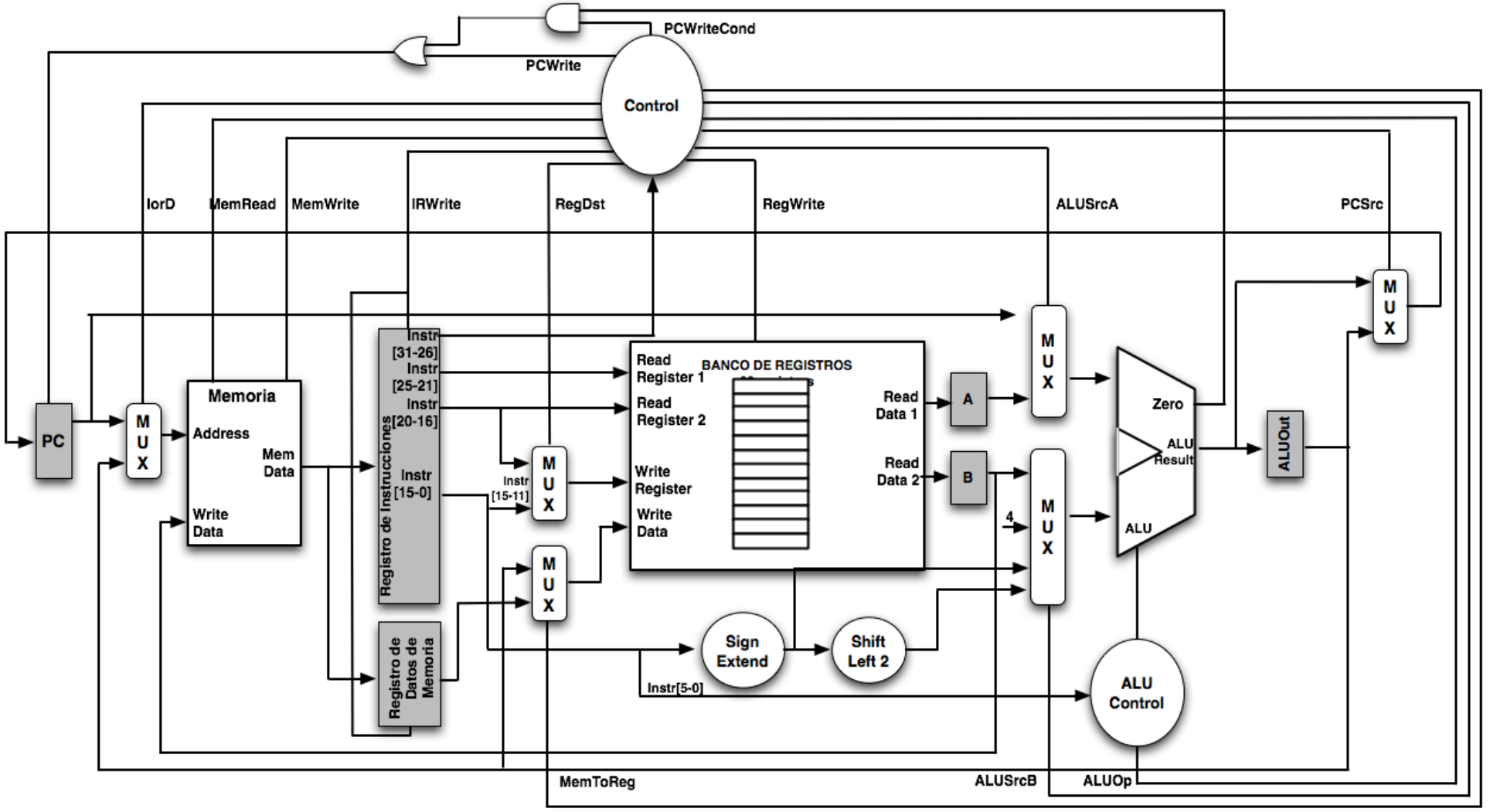
¿CPI multiciclo?

- Load: 20%
- Store: 10%
- Aritmético-lógicas: 35%
- BEQ: 35%

Puntos de control del procesador multiciclo

- Los puntos de control **no se pueden dar como una tabla de verdad**, ya que no se mantienen constantes a lo largo de toda la ejecución
- Los valores de las señales se van modificando en los diferentes ciclos de reloj y dependen de la etapa en la que se encuentra la instrucción
- Señales a generar: *IorD, MemRead, MemWrite, MemToReg, RegDst, RegWrite, ALUSrcA, ALUSrcB, ALUOp, PCSrc, PCWrite, PCWriteCond e IRWrite*

Ruta de datos y unidad de control del nanoMIPS multiciclo



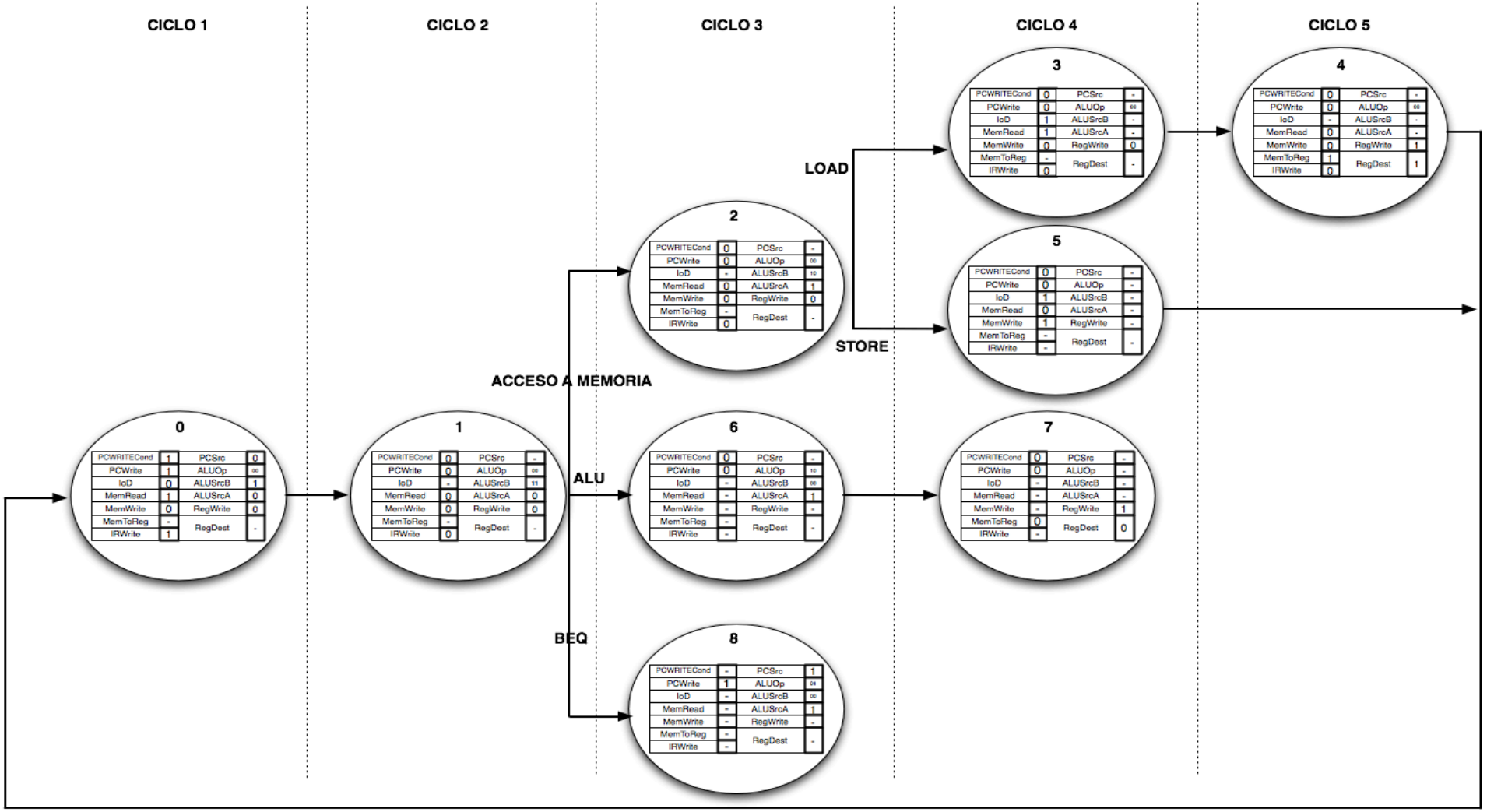
Unidades de control del nanoMIPS multiciclo

- Unidad de control **global**
  - Circuito secuencial
  - Entrada ← *opcode*
  - Se puede diseñar como **máquina de estados** o mediante **microprograma**
- Unidad de control **local** de la ALU

Unidades de control como máquina de estados

- También llamada cableada
- 9 estados que se ejecutan en un máximo de cinco ciclos
- Los ciclos F y D se ejecutan para todas las instrucciones por igual
- Cada tipo de instrucción evoluciona por unos estados diferentes
- Problemas cuando se modifica una única instrucción, **hay que rediseñar toda la unidad de control**

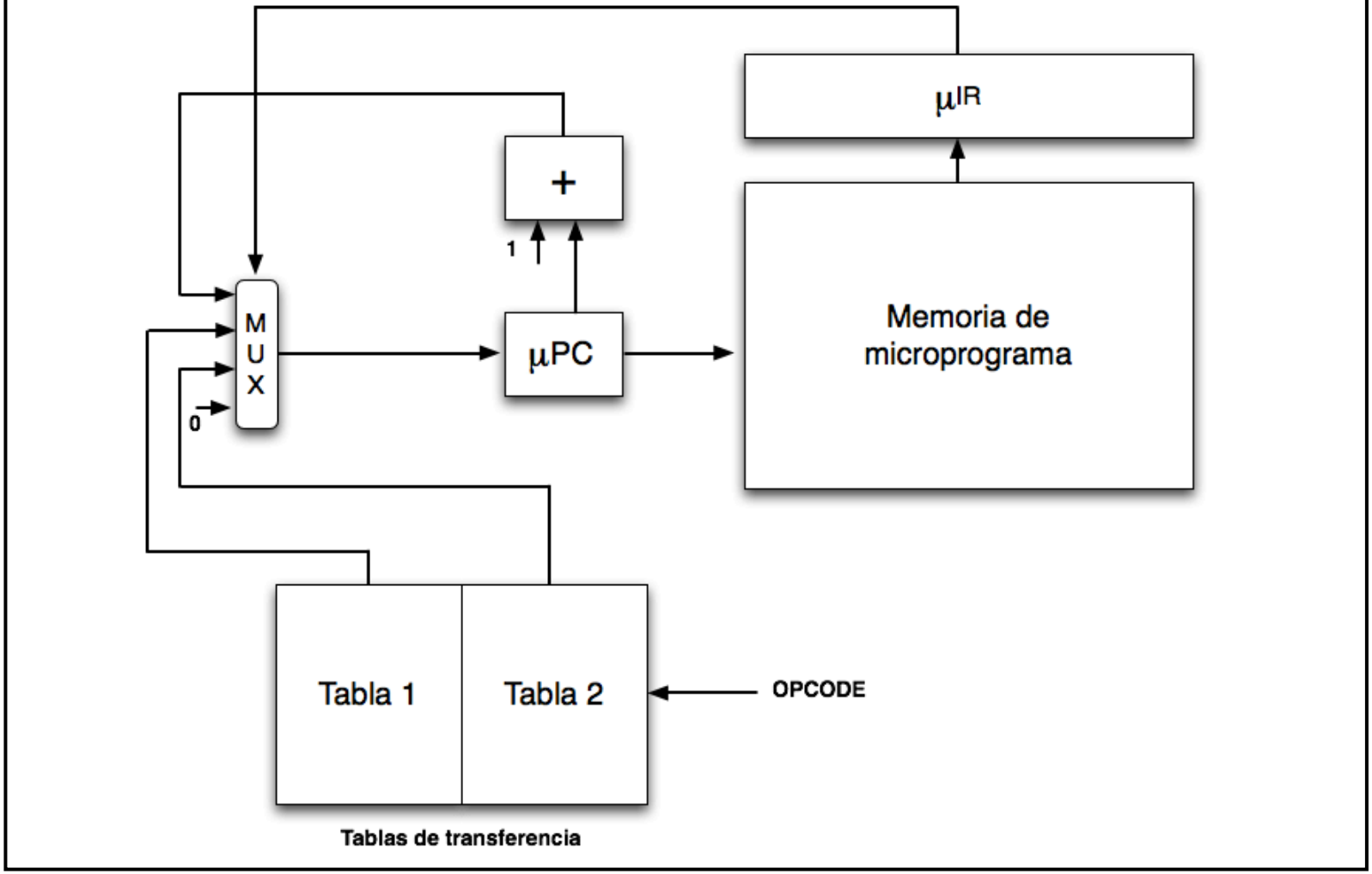
Unidades de control como máquina de estados



Unidades de control microprogramada

- **Memoria ROM** que almacena vectores con el valor de las señales de control que se deben generar en cada momento
- Ejecutar una instrucción es equivalente a leer palabras de esta memoria en un orden determinado
- Cada palabra es una **microinstrucción**
- El conjunto de micorinstrucciones que permiten ejecutar una instrucción es un **microprograma**
- **Ventajas**
  - Más flexible
  - Ocupa menos área
- **Desventaja**
  - Más lenta que la cableada
- Recuperar 3 o 4 microinstrucciones de la memoria, llevarlas al registro de microinstrucción (μIR), dejar tiempo para que se estabilicen las señales de control generadas, etc

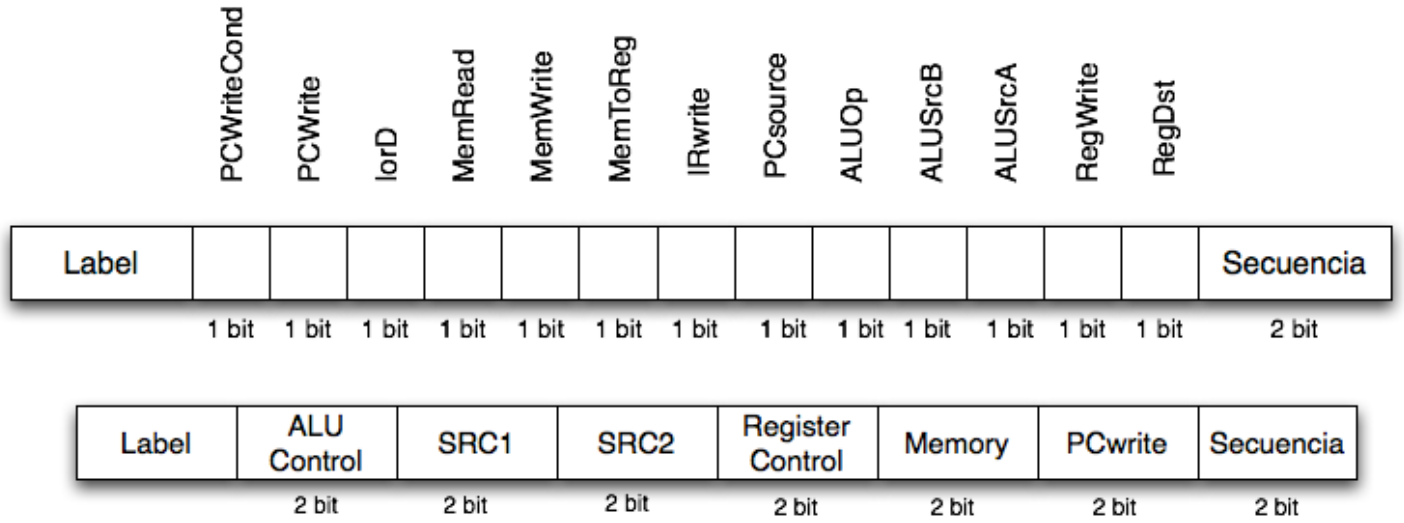
Unidades de control microprogramada



Unidades de control microprogramada

- **Codificación horizontal**
  - Cada microinstrucción incorpora directamente los valores que deben tomar las señales de control
- **Codificación vertical**
  - Para evitar que sean demasiado largas, las microinstrucciones se codifican
  - Se necesita un paso previo de decodificación
  - 6 campos, cada uno codifica un grupo de señales de control relacionadas entre sí: control de la ALU, selección de operandos 1 y 2, control del banco de registros y control del PC
  - A estos campos se les añade la etiqueta de la microistrucción y la información de secuencia
  - En el caso de nanoMIPS se emplea *secuenciamiento explícito*

Unidades de control microprogramada



¿Cómo cargar el PC del microprograma?

- Opción 0 → siguiente instrucción
- Opción 1 → traduce el opcode de la instrucción máquina a una dirección de microprograma para instrucciones ALU, BEQ o acceso a memoria
- Opción 2 → traduce el opcode de la instrucción máquina a una dirección de microprograma para instrucciones load o store
- Opción 3 → permite ir a la microinstrucción 0

Unidades de control microprogramada

- Hoy día la microprogramación ha desaparecido casi por completo
- Existen herramientas avanzadas para diseñar complejas unidades de control con millones de transistores. Estas herramientas garantizan la ausencia de errores de diseño
- Las unidades de control cableadas tienen un rendimiento significativamente mayor que cualquier unidad microprogramada, resultando más competitivas

Tratamiento de excepciones

- Hasta ahora nos hemos centrado en un funcionamiento correcto del procesador en todo momento
- El tratamiento de excepciones consiste en transferir el control a otro programa qué:
  - Salve el estado del procesador cuando se produzca la excepción
  - Corrija la causa de la excepción
  - Restaure el estado del procesador
  - Repita la ejecución de la instrucción causante de la excepción para poder continuar con la ejecución por el punto en el que se encontraba
- A este programa se le denomina *Rutina de Tratamiento de Excepción* o **RTE**

Excepción Vs Interrupción

- **Excepción**: evento no planificado que interrumpe la ejecución de un programa
- **Interrupción**: una excepción que proviene de fuera del microprocesador

Tipo de evento	Origen?	Terminología MIPS
Petición de E/S	Externo	Interrupción
Llamada al SSOO desde un programa de usuario	Interno	Excepción
Desbordamiento aritmético	Interno	Excepción
Instrucción no definida	Interno	Excepción
Mal funcionamiento hardware	Ambos	Excepción o interrupción

Tipos de excepciones

- Interrupciones de E/S
- Llamadas al Sistema Operativo
- Puntos de ruptura
- **Códigos de operación inválidos**
- **Overflow o desbordamiento en la ALU**
- Fallos de página
- Accesos a memoria no alineados
- Violación de zonas protegidas de memoria
- Fallos de hardware
- Fallos de alimentación

Componentes hardware necesarios

- **Registro Exception**: almacena el código del tipo de excepción para que la RTE pueda leerlo
- Contador de programa de excepción, **EPC**: almacena PC - 4
- **Restador** para realizar PC -4
- Se debe cargar en el PC la dirección de memoria donde comienza la RTE

Señales de control

- **ALU_overflow** e **llegal_opcode**
- **Exception**: para escribir el código de las excepciones en el registro Exception
- **ExceptionWrite** y **EPCWrite**: para controlar la escritura en los dos nuevos registros
- PCWrite: para controlar la carga del PC

Unidad de control en el nanoMIPS monociclo

- Sólo hay que añadir las nuevas señales al circuito combinacional

Unidad de control en el nanoMIPS multiciclo

- **Modificar la máquina de estados** con la que se diseña la unidad de control
- Añadir **nuevos microprogramas** a la unidad de control microprogramada
  - Nuevas bifurcaciones hacia estos estados en las tablas de transferencia
  - Nuevas microinstrucciones que activen las nuevas señales de control en la memoria de microprograma

Nuevos estados nanoMIPS multiciclo