



Sistemas de Procesamiento de Datos – Unidad 7

- Unidad de Control
- Contador de Programa
- Pipelines
- RISC y CISC

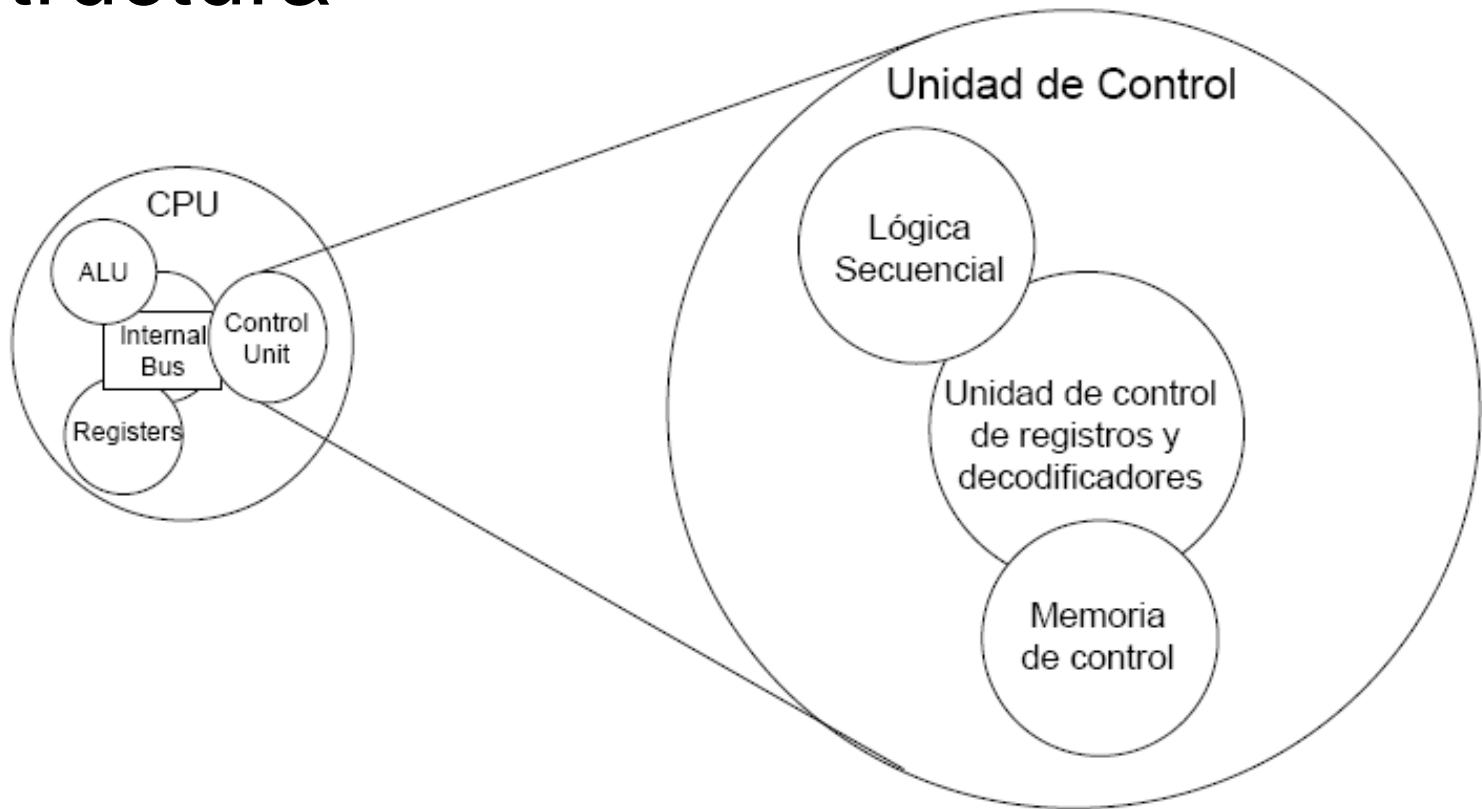
Profesor: Fabio Bruschetti

Ayudante: Pedro Iriso

Ver 2020

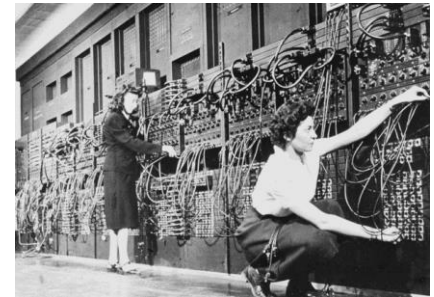
Unidad de control (UDC o CU)

■ Estructura



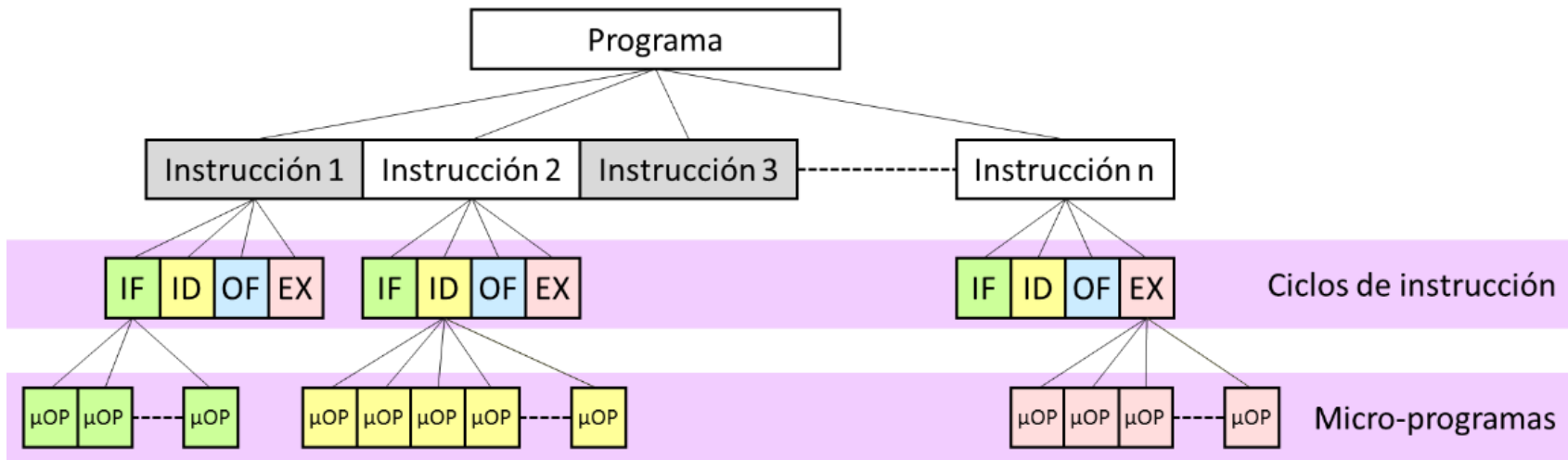
Unidad de control

- La **ALU** es la que hace los **cálculos y procesa**
- La **Unidad de Control** será la responsable de **ejecutar las instrucciones**, generar todas las señales eléctricas necesarias para su ejecución tanto dentro como fuera de la CPU
- En un principio, las UDC estaban internamente "**cableadas**"; es decir, se las construía con circuitos discretos fijos para decodificar y ejecutar instrucciones
- Esto tenía como ventaja una **alta velocidad** de ejecución pero el inconveniente de no poder incorporar nuevas instrucciones al set original; prácticamente había que realizar un nuevo diseño.
- Actualmente no son cableadas, son **microprogramadas**. Estas unidades poseen una Memoria de Control donde se encuentran almacenados los Microprogramas correspondientes a cada una de las instrucciones que es capaz de ejecutar la CPU.



Unidad de control

- Un Microprograma es una secuencia de Microinstrucciones, denominando así al conjunto de señales eléctricas que serán necesarias para dicha ejecución



μOP Micro-operación



Unidad de control

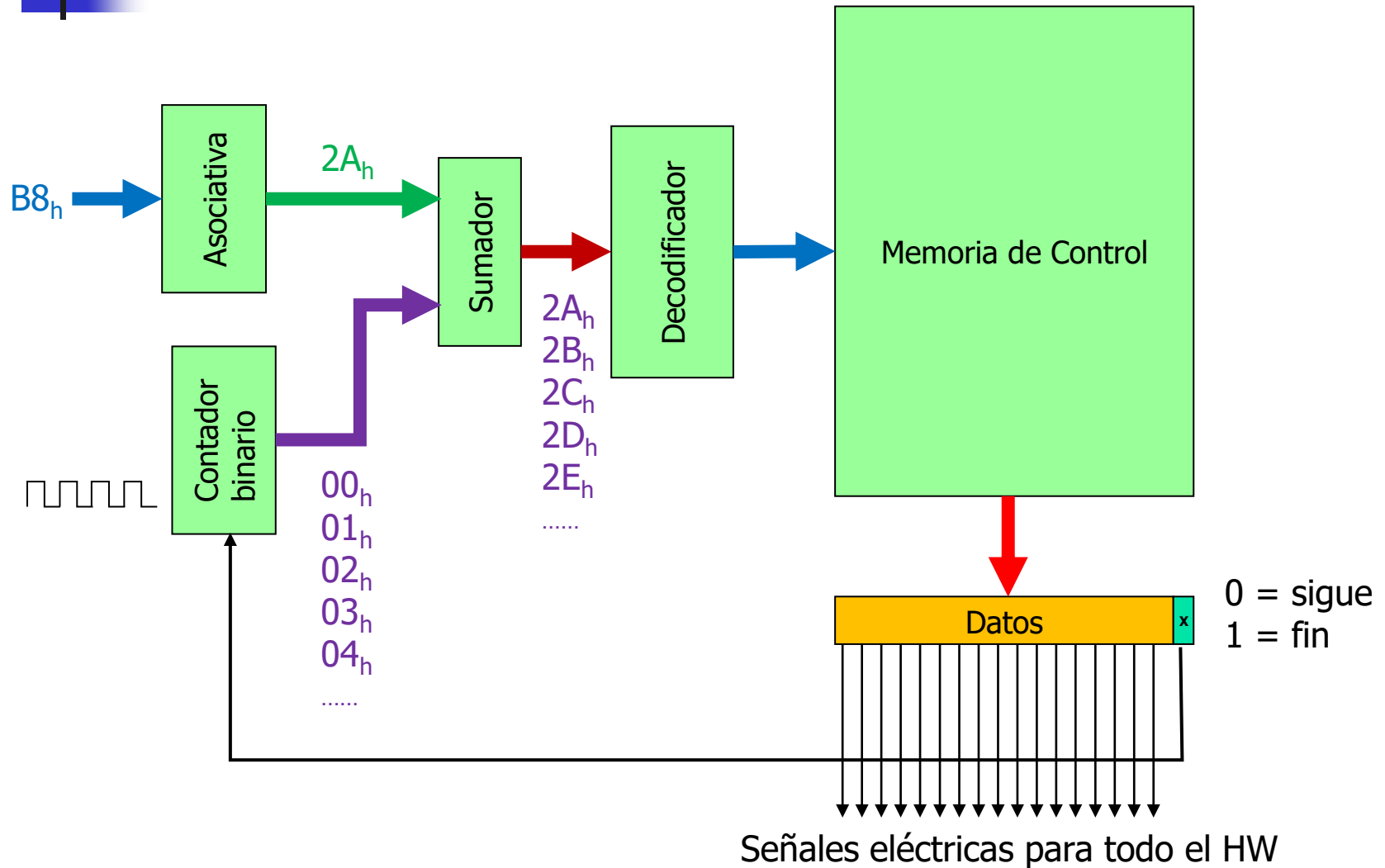
- La ejecución de dichas microinstrucciones tienen como base la señal de reloj que marca el paso de los acontecimientos
- Los “1” y “0” almacenados en la Memoria de Control se corresponderán con los valores eléctricos que tomarán las señales de control que emita la UDC
- La ventaja de las unidades de control microprogramadas es que es posible agregar nuevas instrucciones con solo almacenar los microprogramas correspondientes a la Memoria de Control. La Memoria de Control es inaccesible al usuario y es grabada por el fabricante del Microprocesador o por un proceso especial que el fabricante pone a disposición



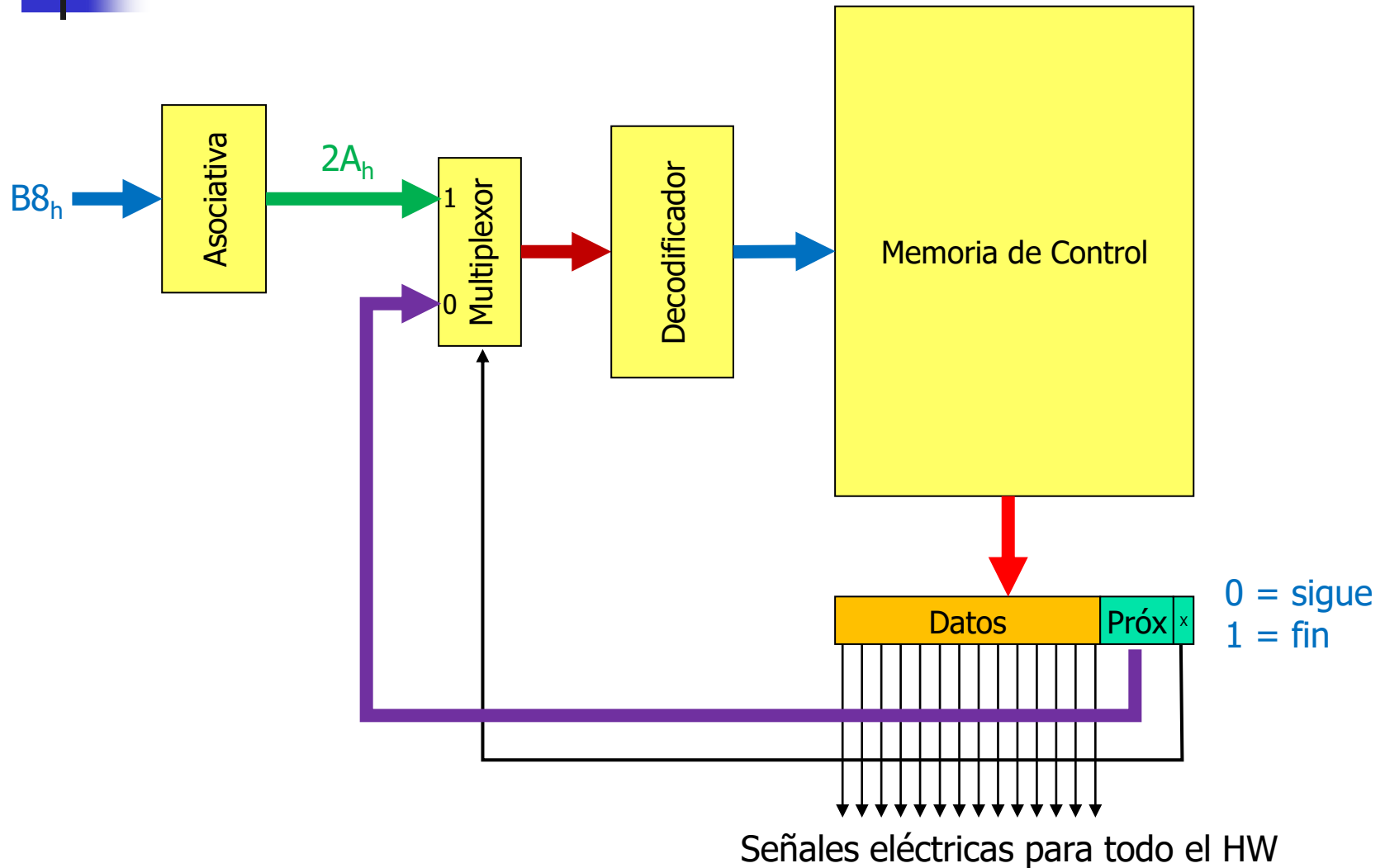
Unidad de control

- De acuerdo a como se ubiquen las microinstrucciones en la memoria de control se tendrán dos tipos de secuenciamiento de las mismas
- Secuenciamiento **explícito**: Las microinstrucciones no se encuentran ordenadas secuencialmente en la memoria de control, por lo tanto, cada microinstrucción incorpora la dirección de la siguiente instrucción
- Secuenciamiento **implícito**: Las microinstrucciones se encuentran ordenadas secuencialmente en la memoria física. Se emplea un incrementador, al cual ingresa la dirección de la microinstrucción en curso y de esa manera se accede a la microinstrucción siguiente

Secuenciamiento implícito



Secuenciamiento explícito





Unidad de control

- Set de instrucciones

- Es el conjunto de instrucciones disponible para el programador. Son códigos binarios que tienen una asociación a un lenguaje simbólico denominado assembler (o ensamblador)
- El lenguaje assembler que se utilice para programar atará la ejecución a un microprocesador determinado
- Cada instrucción está caracterizada por un código de operación
- Mover datos desde la CPU a la memoria
 - `MOV AX, → "B8h"`



Unidad de control

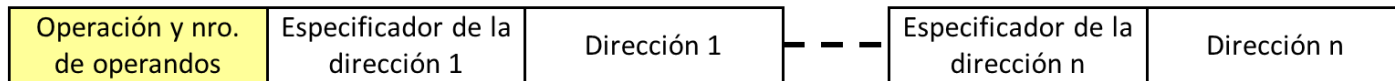
- Set de instrucciones

- El código de operación de una instrucción provee en forma directa o indirecta (a través de un decodificador) el lugar en donde se encuentra de la primera microinstrucción del microprograma a ejecutar
- Cada instrucción del “set” indica los registros que lee o modifica
- La longitud de cada instrucción depende del tipo de instrucción
 - Registro/Registro o Registro/Memoria
 - Comparación o Manejo de Bits
 - Salto Condicional o Salto Incondicional
- Cada instrucción puede demandar más o menos de ciclos de reloj para su ejecución completa

Unidad de control

■ Instrucciones

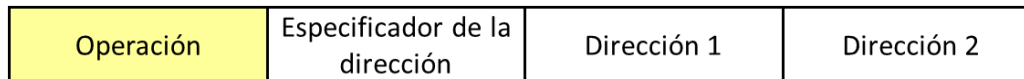
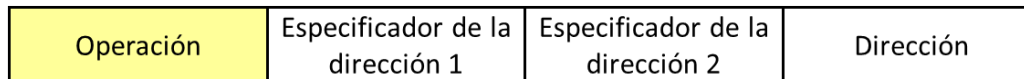
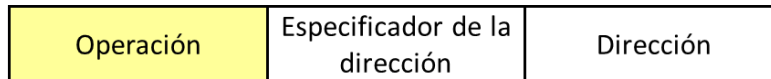
Formato de instrucciones de longitud variable (VAX)



Formato de instrucciones de longitud fija (DLX, MIPS, PowerPC, HP-PA, SPARC)



Formato de instrucciones combinados (IBM 360, IBM 370, Intel 80x86)



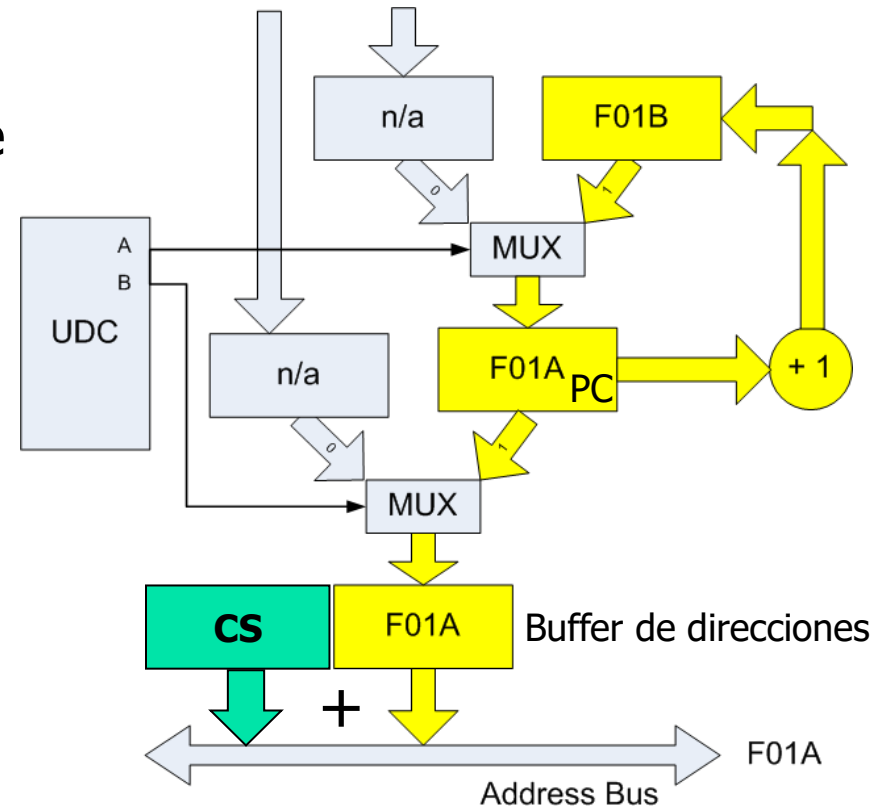


Unidad de control

- Contador de programa (PC = **P**rogram **C**ounter)
 - Contiene la dirección de la próxima instrucción a ejecutarse
 - Es parte de la arquitectura del computador (es accesible por el programador)
 - Cuando se inicia la ejecución de un programa nuevo, se carga este registro con la dirección de la primera instrucción del programa en cuestión
 - Dependiendo del tipo de instrucción a ejecutar y de la lógica del programa, el PC se comportará de diferentes maneras

Contador de Programa (PC)

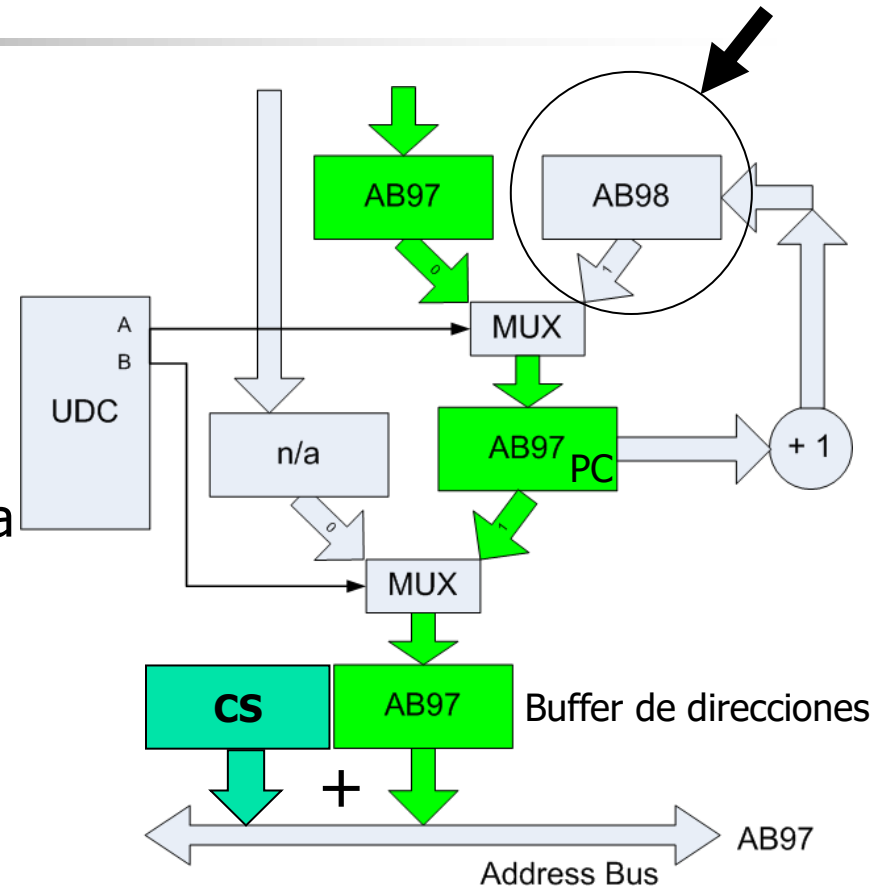
- Funcionamiento habitual
 - El PC se carga con la dirección de la primera instrucción del programa
 - Se lee y se incrementa en 1 para proseguir leyendo la próxima posición de memoria
 - A=1 y B=1
 - Ejemplo
 - MOV AX, 3421_h
 - INC AX
 - ...



Contador de Programa (PC)

■ Salto (Jump)

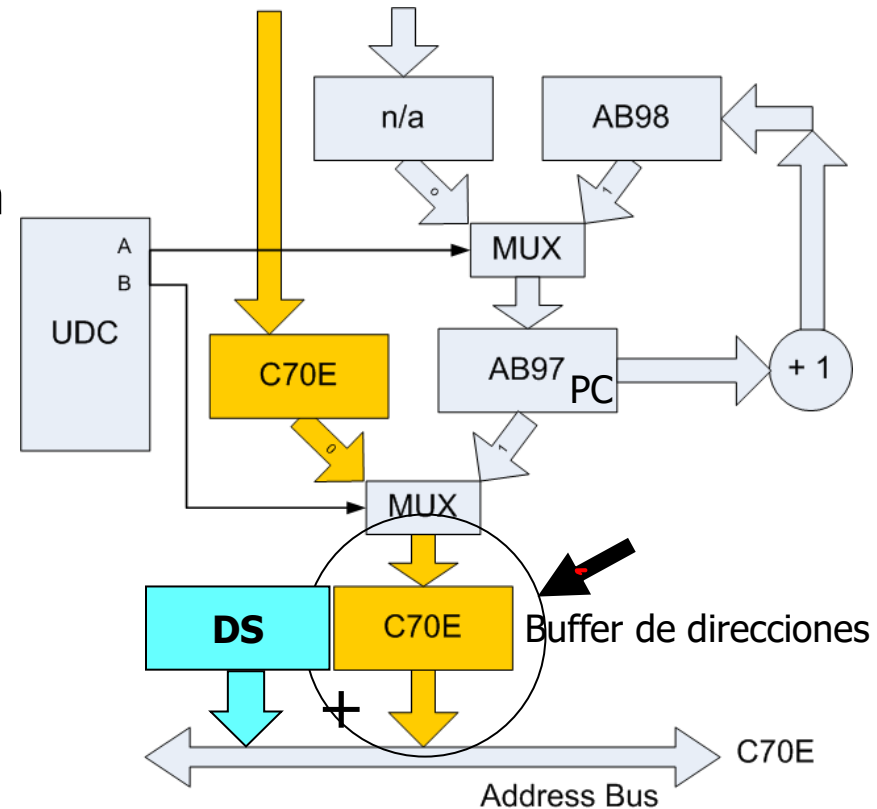
- Se ha procesado un salto en la ejecución del programa. Se deberá continuar la ejecución en una posición distinta a la próxima (funcionamiento habitual)
- Se carga el contador de programa en forma directa
- Se continúa luego la ejecución en forma habitual
- A=0 y B=1
- Ejemplo
 - MOV AX, 3421_h
 - JMP AB97_h
 - ...



Contador de Programa (PC)

■ Indirecciones

- El objeto requerido no se encuentra ubicado a continuación
- Se resuelve la indirección y se la carga en el buffer de direccionamiento
- La ejecución del programa no se altera → El PC no cambia
- $A=X$ y $B=0$
- Ejemplo
 - `MOV AX, [C70E]`
 - `INC AX`
 - ...



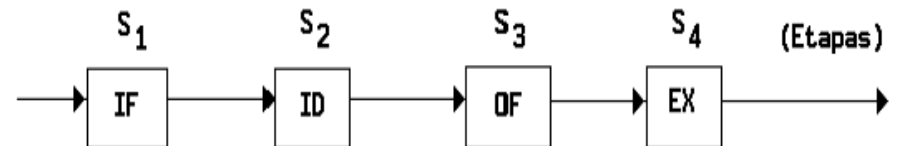
Paralelismo

- Paralelismo Temporal y Espacial

- Espacial: Existen varias unidades funcionales (simultaneidad)
- Temporal: Se solapan tiempos.

- Ciclo de ejecución

- IF = Instruction Fetch
- ID = Instruction Decode
- OF = Operand Fetch
- EX = Instruction Execution



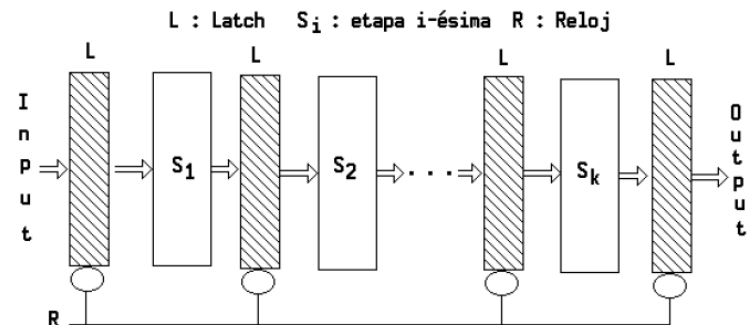
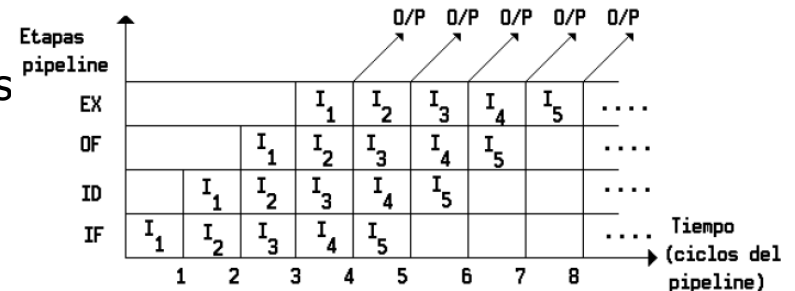
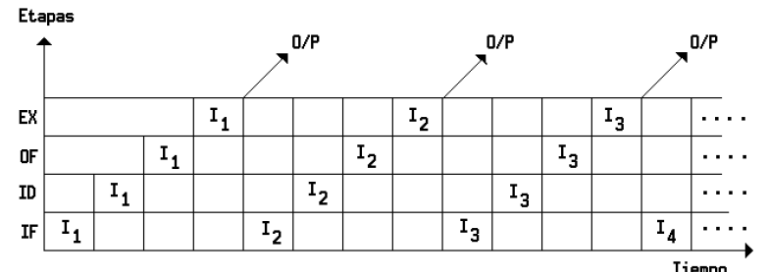
Pipelines

■ Pipeline

- En la segmentación de instrucciones, cada etapa o segmento de la cadena está especializada en una tarea específica de la "línea de ejecución" y lleva a cabo siempre la misma actividad
- Esta tecnología es propia de procesadores eficientes

■ Tipos

- Aritméticos
- de Instrucción
- de Procesador
- Uni/Multifuncionales
- Estáticos/Dinámicos
- Escalares/Vectoriales



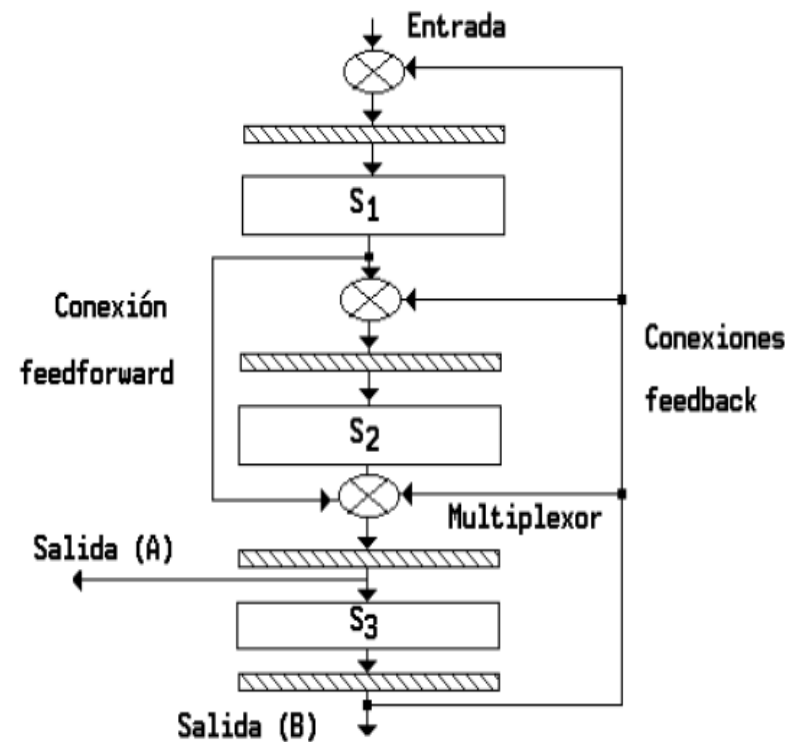
Pipelines

- Pipelines Generales
 - Tablas de Reservas

Tiempo

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
S_1	A			A			A	
S_2		A						A
S_3			A		A	A		

	t_0	t_1	t_2	t_3	t_4	t_5	t_6
S_1	B				B		
S_2			B			B	
S_3		B		B			B



Pipelines: tablas de ocupación

Instrucción A

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
S_1	A			A			A	
S_2		A						A
S_3			A		A	A		

Instrucción B

S_1	B		B					
S_2				B	B		B	
S_3		B				B		B

Instrucción C

S_1	C				C		C	
S_2		C	C	C				
S_3						C		C

Pipelines: tablas de ocupación

Instrucción A

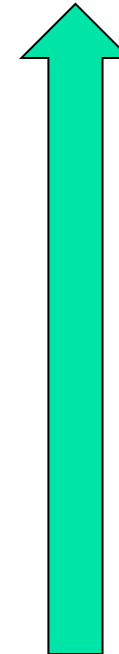
	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
S_1	A			A			A	
S_2		A						A
S_3			A		A	A		

Instrucción B

S_1	B		B					
S_2				B	B		B	
S_3		B				B		B

Instrucción C

S_1	C				C		C	
S_2		C	C	C				
S_3						C		C



Pipelines: tablas de ocupación

Instrucción A

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
S_1	B		B	A			A	
S_2		A		B	B		B	A
S_3		B	A		A	B		B

Instrucción B

S_1
 S_2
 S_3

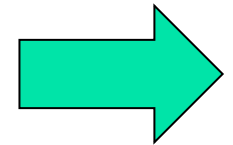
Instrucción C

S_1	C				C		C	
S_2		C	C	C				
S_3						C		C

Pipelines: tablas de ocupación

Instrucción A

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
S_1	A	B		B			A		
S_2		A			B	B		B	
S_3			B		A	A	B		B



Instrucción B

S_1
 S_2
 S_3

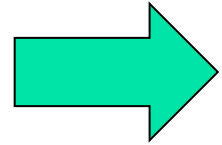
Instrucción C

S_1	C				C		C	
S_2		C	C	C				
S_3						C		C

Pipelines: tablas de ocupación

Instrucción A

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
S_1	A		B	A	B		A			
S_2		A				B	B	A	B	
S_3			A	B	A	A		B		B



Instrucción B

S_1
 S_2
 S_3

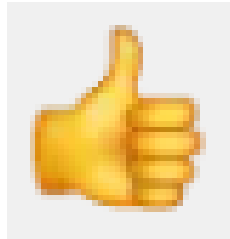
Instrucción C

S_1	C				C		C	
S_2		C	C	C				
S_3						C		C

Pipelines: tablas de ocupación

Instrucción A

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
S_1	A		B	A	B		A			
S_2		A				B	B	A	B	
S_3			A	B	A	A		B		B



Instrucción B

S_1
 S_2
 S_3

Instrucción C

S_1	C				C		C	
S_2		C	C	C				
S_3						C		C

Pipelines: tablas de ocupación

Instrucción A

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
S_1	A		B	A	B		A			
S_2		A				B	B	A	B	
S_3			A	B	A	A		B		B

Instrucción B

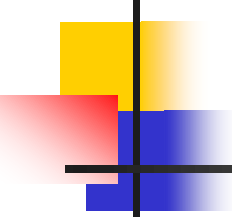
S_1
 S_2
 S_3

Instrucción C

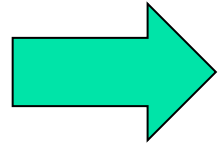
S_1	C				C		C	
S_2		C	C	C				
S_3						C		C



Pipelines: tablas de ocupación



	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
Instrucción A	S_1	A		B	A	B		A		
	S_2		A	C	C		B	B	A	B
	S_3			A	B	A	A		B	



Instrucción B

S_1
 S_2
 S_3

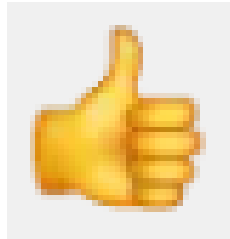
Instrucción C

S_1
 S_2
 S_3

Pipelines

Instrucción A

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
S_1	A	C	B	A	B	C	A	C		
S_2		A	C	C	C	B	B	A	B	
S_3			A	B	A	A	C	B	C	B



Instrucción B

S_1
 S_2
 S_3

Instrucción C

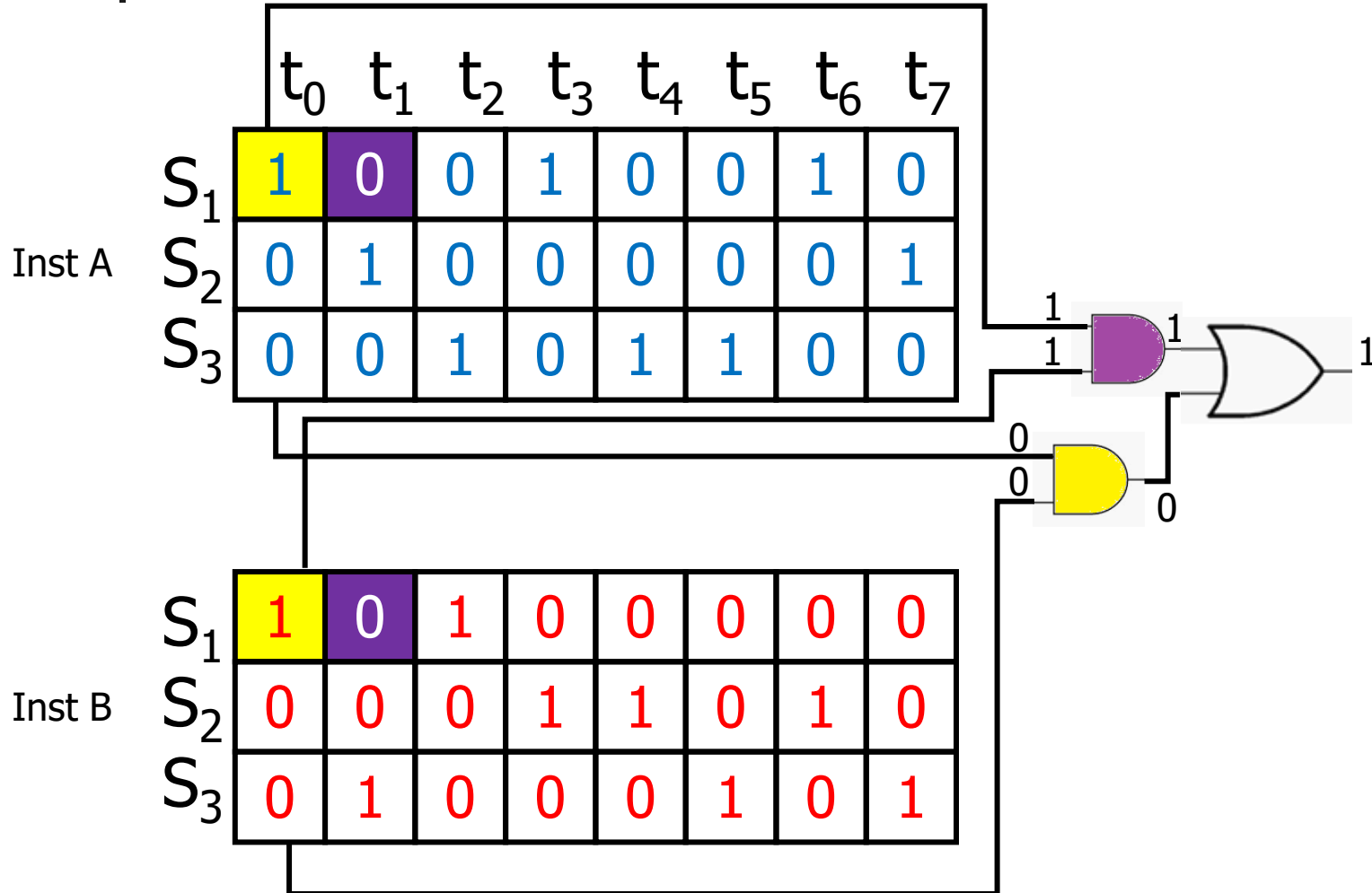
S_1
 S_2
 S_3



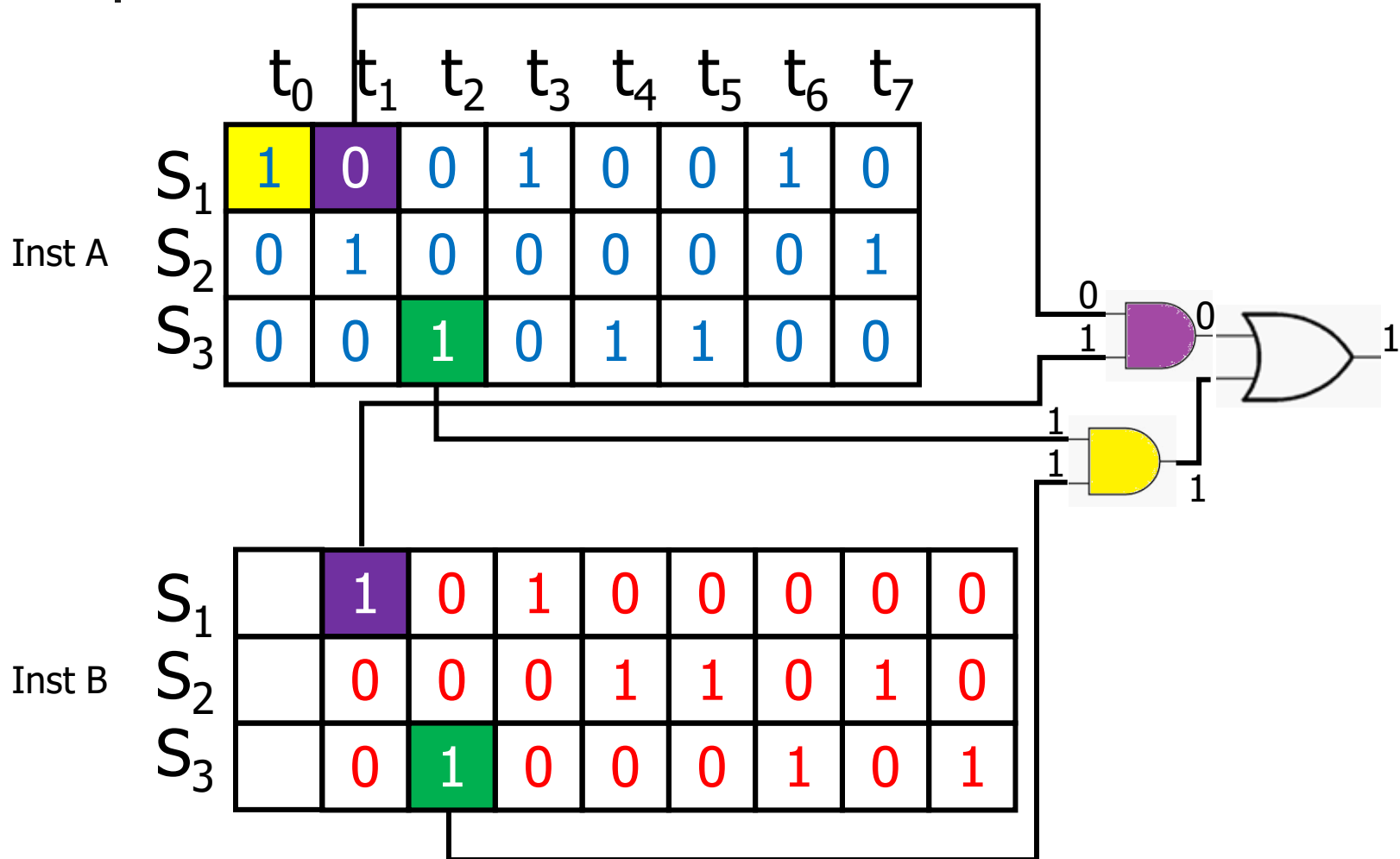
Sin pipelines

	t_0																							t_{23}
S_1	A			A			A		B		B						C				C		C	
S_2		A					A				B	B		B			C	C	C					
S_3			A		A	A				B				B		B						C		C

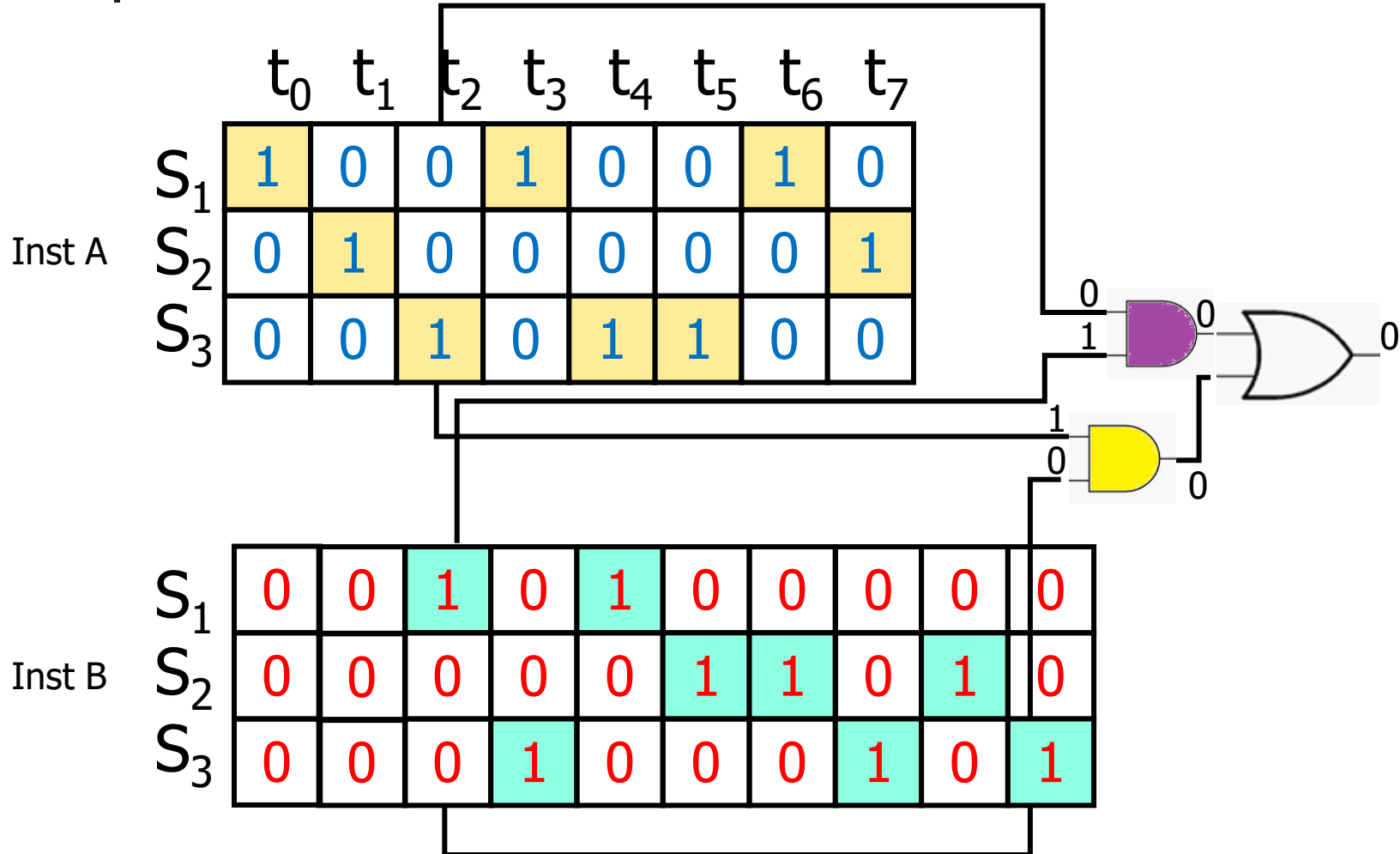
Pipelines: tablas de ocupación



Pipelines: tablas de ocupación



Pipelines: tablas de ocupación





Problemas en los Pipelines

- Pueden aparecer cuando la CPU trata de ejecutar instrucciones en forma simultánea y entre ellas hay dependencia de datos (hazards)
- Hay tres tipos de problemas
 - De datos, de control y de estructura
- Problemas de datos
 - RAW (Read-After-Write)
 - Un dato que es modificado, es leído inmediatamente luego. Como la modificación puede no haber finalizado, la lectura puede ser errónea
 - $i1.R2 = R1 + R3$
 - $i2.R4 = R2 + R3$



Problemas en los Pipelines

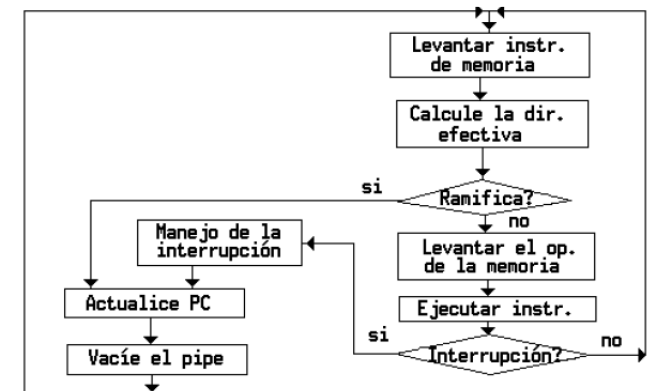
- Problemas de datos (Cont.)
 - WAR (Write-After-Read)
 - Idem anterior pero cambiando lecturas por modificaciones y viceversa
 - $i1.R4 = R1 + R3$
 - $i2.R3 = R1 + R2$
 - WAW (Write-After-Write)
 - Se ejecutan dos instrucciones que escriben el mismo operando. La primera puede terminar luego de la segunda
 - $i1.R2 = R1 + R2$
 - $i2.R2 = R4 \times R7$

Problemas en los Pipelines

■ Problemas de control

- Ocurren cuando una instrucción de salto deberá ejecutarse pero no se puede saber si la condición de salto será satisfactoria con lo cual la próxima instrucción a ejecutar no es la que físicamente está a continuación del salto

- i1. $A = B - C$
- i2. $\text{JMP } A = 0 \text{ to } i.n$
- i3. $D = A \times B$



■ Problemas de estructura

- Ocurren cuando dos instrucciones requieren una misma unidad estructural (por ejemplo la ALU)
 - i1. $\text{JMP } A = 0 \leftarrow$ Requiere la ALU para comparar
 - i2. $\text{CMP } A, B \leftarrow$ Requiere la ALU para comparar



Problemas en los Pipelines

- Problemas de datos

- ¿Cuánto vale R2?

- $i1.R1 = 6$
 - $i2.R1 = 3$
 - $i3.R2 = R1 + 7$

- ¿Cómo se soluciona? → Forwarding

- No se espera a finalizar la $i2$, se envía el dato directamente a la $i3$



Problemas en los Pipelines

- Predicción de saltos

- Se basa en poder determinar si una condición de salto (o bifurcación) en el flujo de instrucciones se ejecutará o no
- Esto permitirá a la CPU encontrar y ejecutar instrucciones sin tener que esperar que se resuelva la condición de salto
- Estos predictores se utilizan en arquitecturas con Pipeline, con lo cual evitan que el pipeline de instrucciones se vacíe



Problemas en los Pipelines

- Predictores de saltos - Tipos

- Estáticos

- No se basan en la dinámica del código, solo de la instrucción aislada. Simplemente predicen que el salto no va a ocurrir, con lo cual continúan con la instrucción siguiente. Cuando condición de salto es evaluada, si se da, continua la ejecución en la dirección de salto
 - Otros asumen que los saltos "hacia atrás" ocurrirán, mientras que los saltos "hacia delante" no lo harán. Efectivos para LOOPS

- Dinámicos

- Se basa en una larga lista de historia de saltos. Es más preciso.
 - Se guardan bits indicando si en el pasado se ejecutó el salto o no
 - Se basa en conceptos neurales

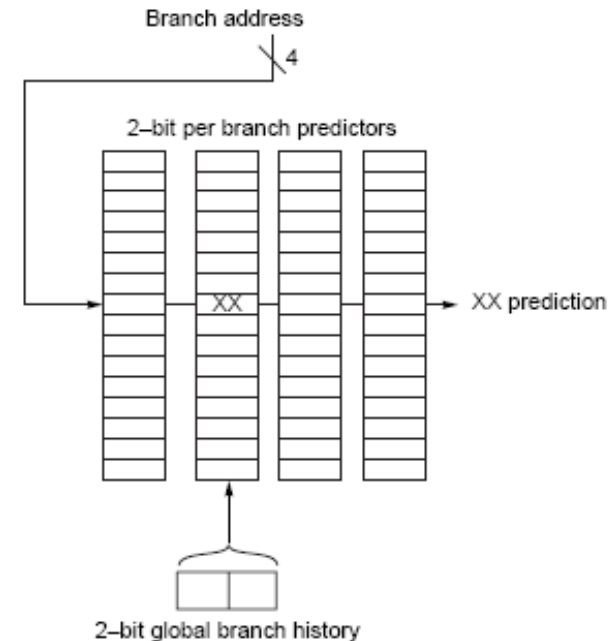
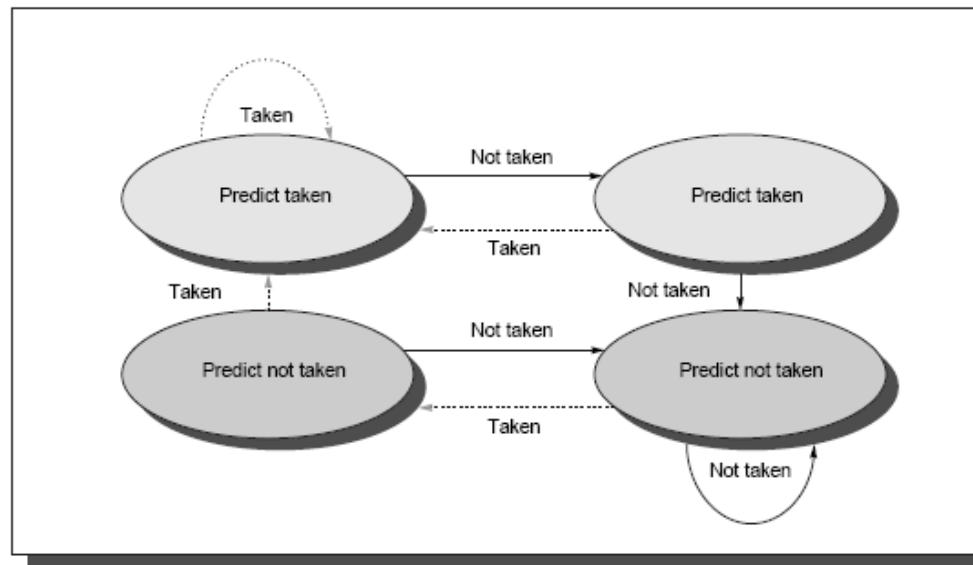
- Otros

- Next line, Bimodal, Local, Global, Combined

Problemas en los Pipelines

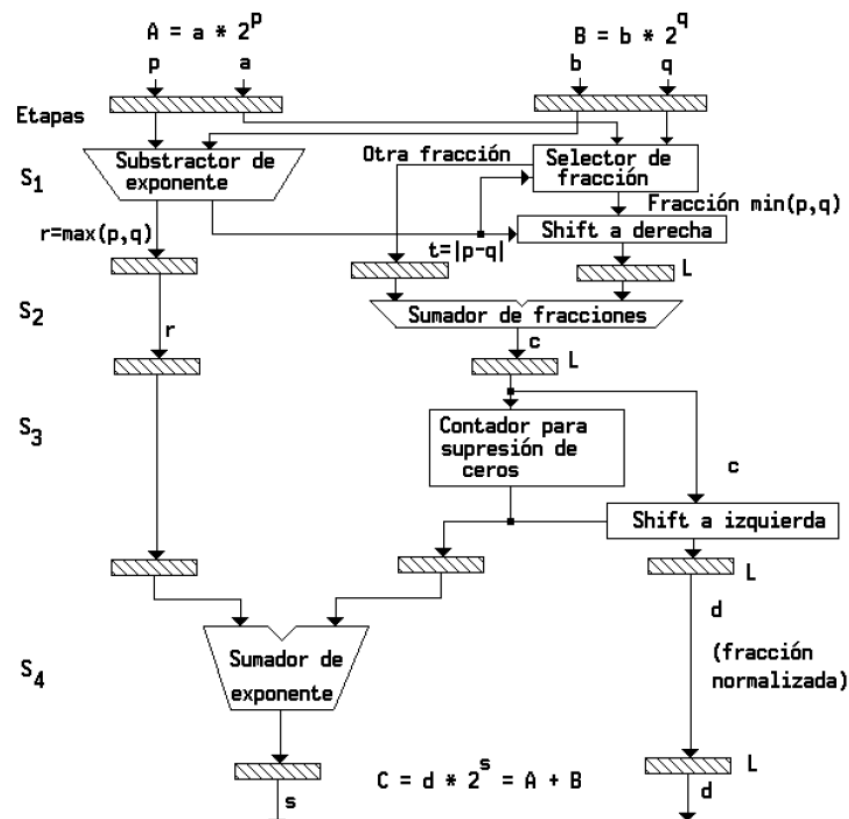
■ Predictores de saltos

- Utilización de 2 bits para historia de saltos
- Si una predicción se cumple, se mantiene la predicción
- Si una predicción no se cumple por primera vez, lo registra pero mantiene la predicción
- Si una predicción no se cumple por segunda vez consecutiva, se cambia la predicción



Ejemplos de Pipelines

- Pipeline Sumador de Punto Flotante en 4 etapas



Un sumador pipeline de punto flotante de cuatro etapas de procesamiento.

Ejemplos de Pipelines

■ Pipeline Vectorial

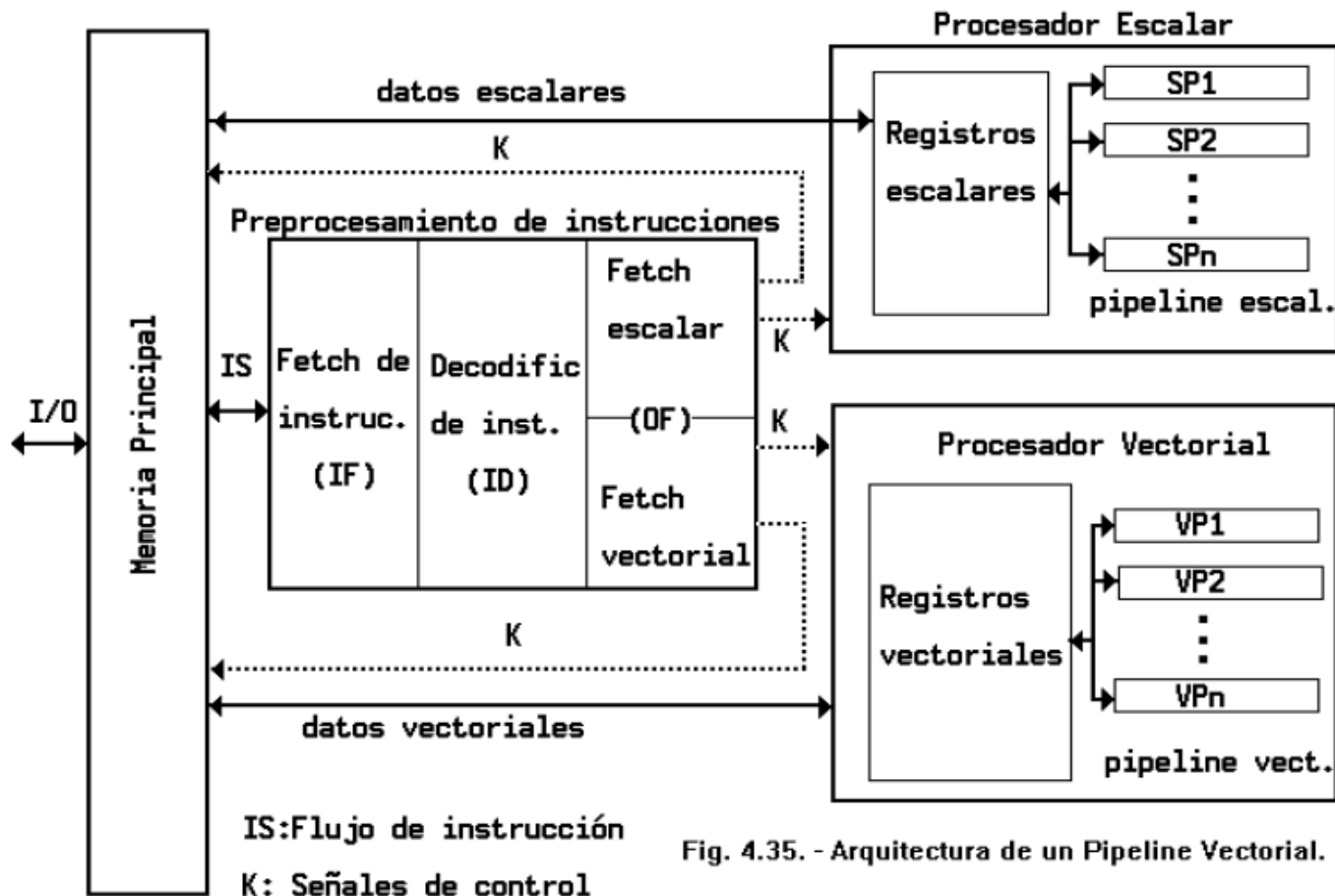


Fig. 4.35. - Arquitectura de un Pipeline Vectorial.



Arquitecturas RISC y CISC

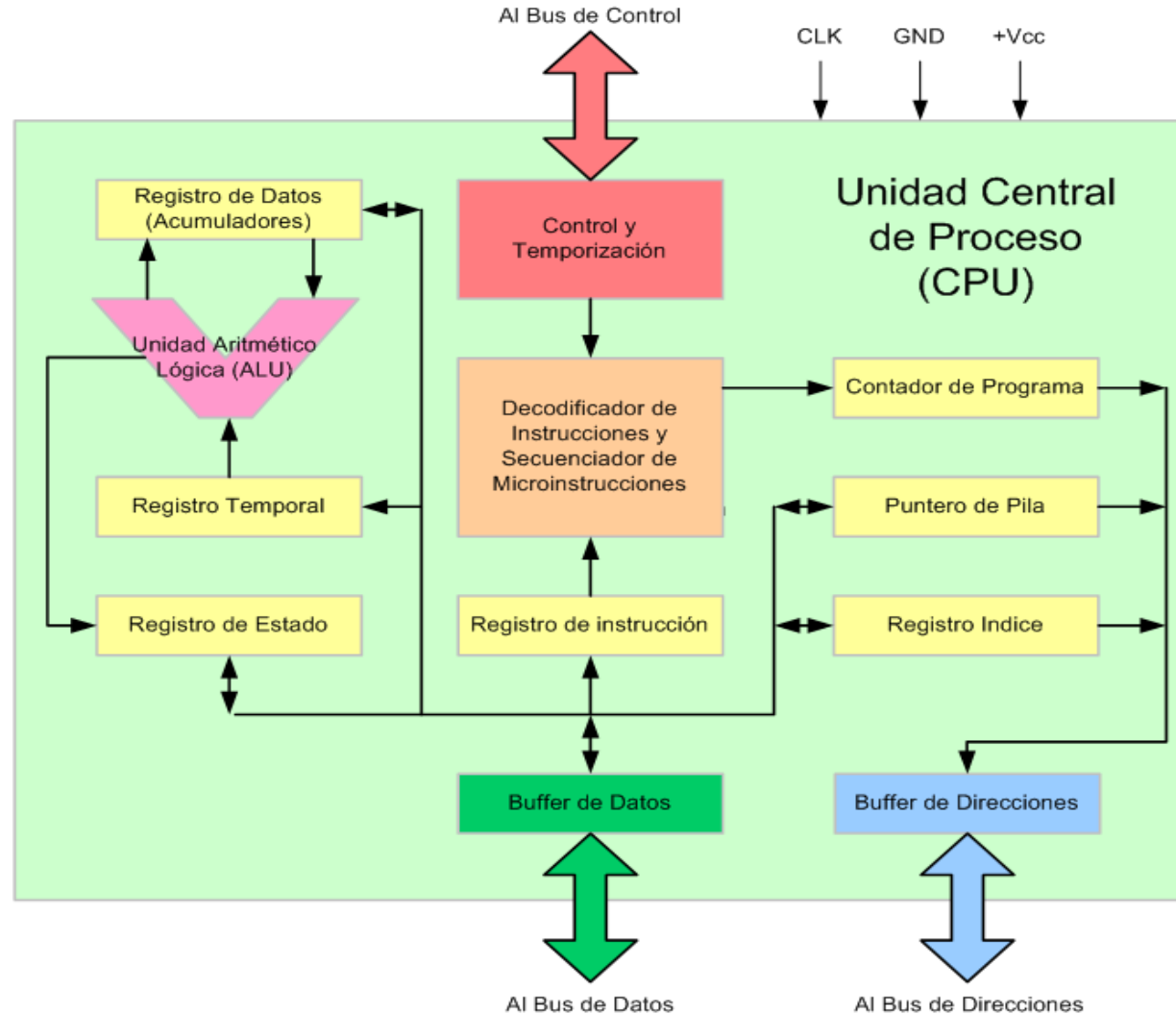
- RISC (**Reduced Instruction Set Computing**)
 - Conjunto de instrucciones simples que hacen menos cosas en poco tiempo
 - Arquitectura LOAD-STORE
 - Alpha, ARC, MIPS, PA-RISC, PIC, Power Architecture (que incluye el PowerPC), SuperH, SPARC
 - Basados en el modelo de Harvard
 - Intel Core 2 y AMD K6 tienen unidades de ejecución internas del tipo RISC
- CISC (**Complex Instruction Set Computing**)
 - Cada instrucción puede requerir ejecutar una o varias instrucciones de bajo nivel (load, calculate y store en una misma instrucción)
 - IBM System/360, PDP-11, VAX, 68000, y familia x86
 - Otorga muy buena productividad en Assembler, Fortran, Algol



Arquitecturas RISC y CISC

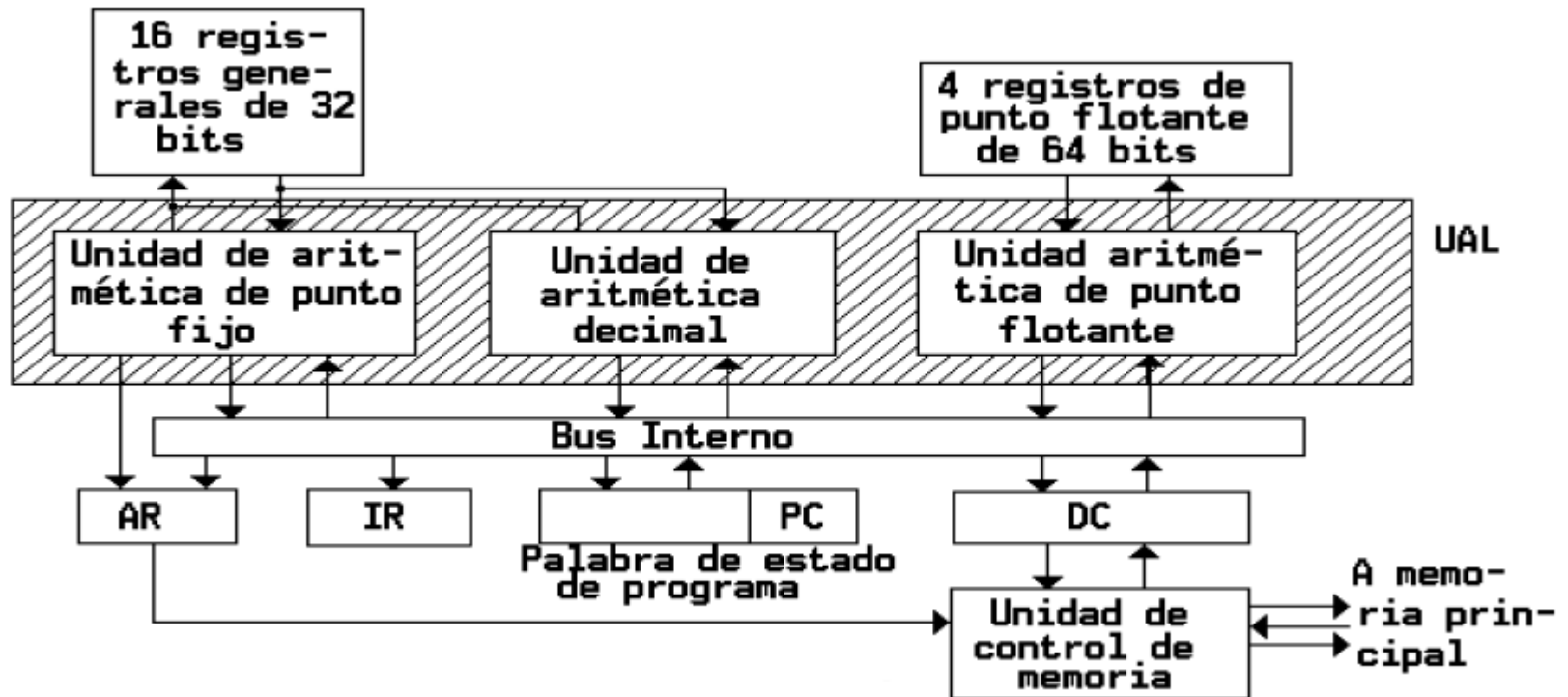
Característica	CISC	RISC
Cantidad de instrucciones en lenguaje máquina	Muchas	Pocas
Cantidad de modos de direccionamiento	Muchos	Pocos
Cantidad de formatos de instrucción	Varios	Unico
Cantidad de ciclos de reloj necesarios para ejecutar cada instrucción	Muchas, más de uno	Uno en todas
Instrucciones para acceder a la memoria	Muchas	2, Load y Store
Registros de propósito esécífico	Si	No
Control microprogramado	Si	No

Diagrama en Bloques de una CPU



CPU

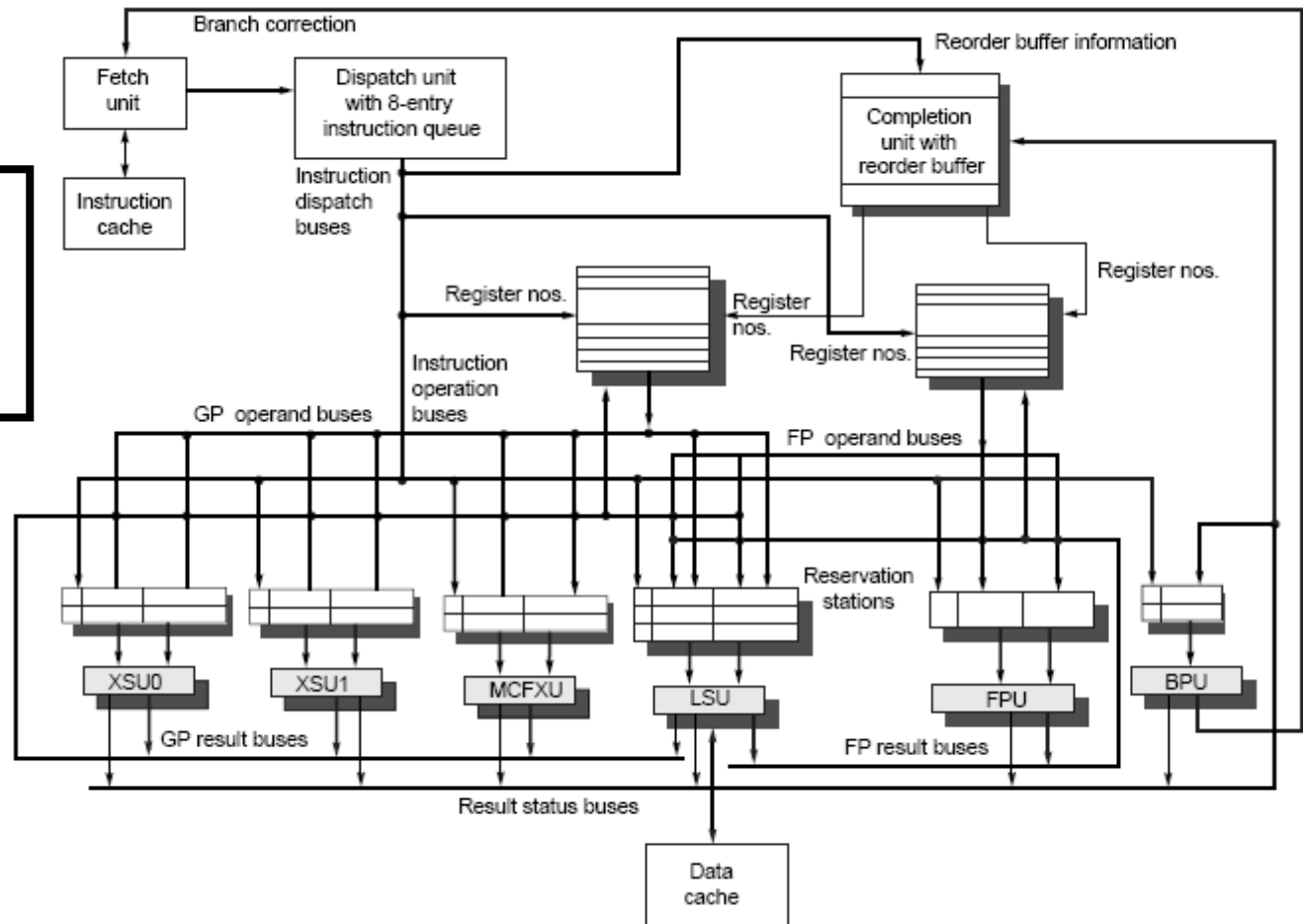
- Diagrama CPU S/360 y S/370



CPU

■ Diagrama PowerPC 620 (64-bits)

Simple Integer Units: XSU0 XSU1
Complex Integer Units: MCFXU
Load-Store Unit: LSU
Floating Point Unit: FPU
Branch Unit: BU



CPU

■ Diagrama PowerPC G5

