



Universidad Nacional de San Martín

Sistemas de Procesamiento de Datos

UNIDAD 7 = Unidad de control, pipelines, RISC y CISC

Tecnicatura en Programación Informática
Tecnicatura en Redes Informáticas

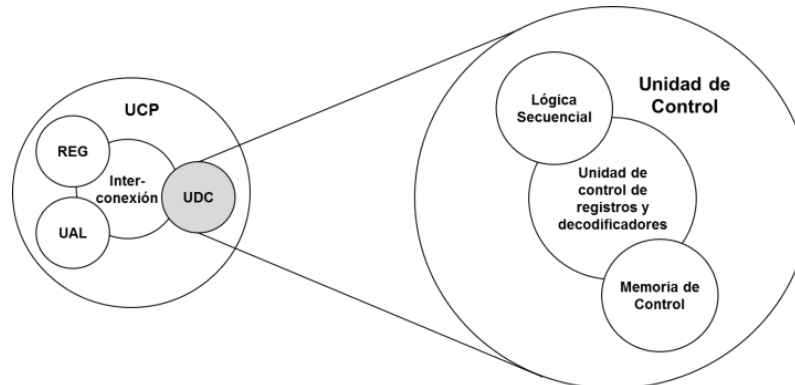
Profesor: Fabio Bruschetti

Ayudante: Pedro Iriso

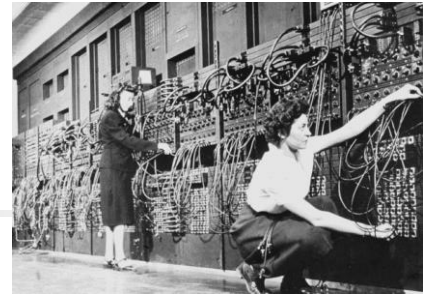
2024 – 2C

Función de la Unidad de Control

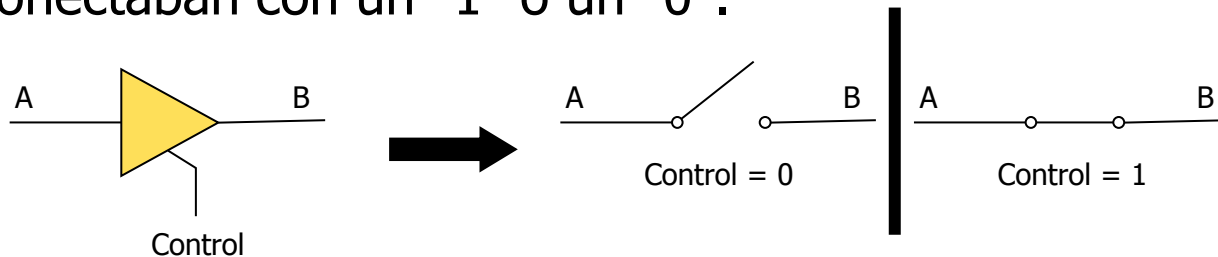
- La Unidad de Control (UDC o CU) Es la encargada de la coordinación y orquestación de todas las tareas para la **ejecución de las instrucciones en formato binario** (bajo nivel).
- Durante la ejecución de una instrucción, la UdC generará todas las señales eléctricas internas necesarias para la conexión y desconexión de las diferentes unidades estructurales internas de la CPU y las señales eléctricas externas (señales de lectura, escritura, etc.).
- La **ALU** realiza las **operaciones aritméticas y lógicas** ordenadas por la unidad de control → Procesa los datos.



UDC microprogramadas



- Inicialmente, eran cableadas
 - Las conexiones necesarias para la ejecución de cada instrucción se hacían con cables externos.
- Ahora, microprogramadas
 - Los cables fueron reemplazados por circuitos que se conectaban o desconectaban con un "1" o un "0".



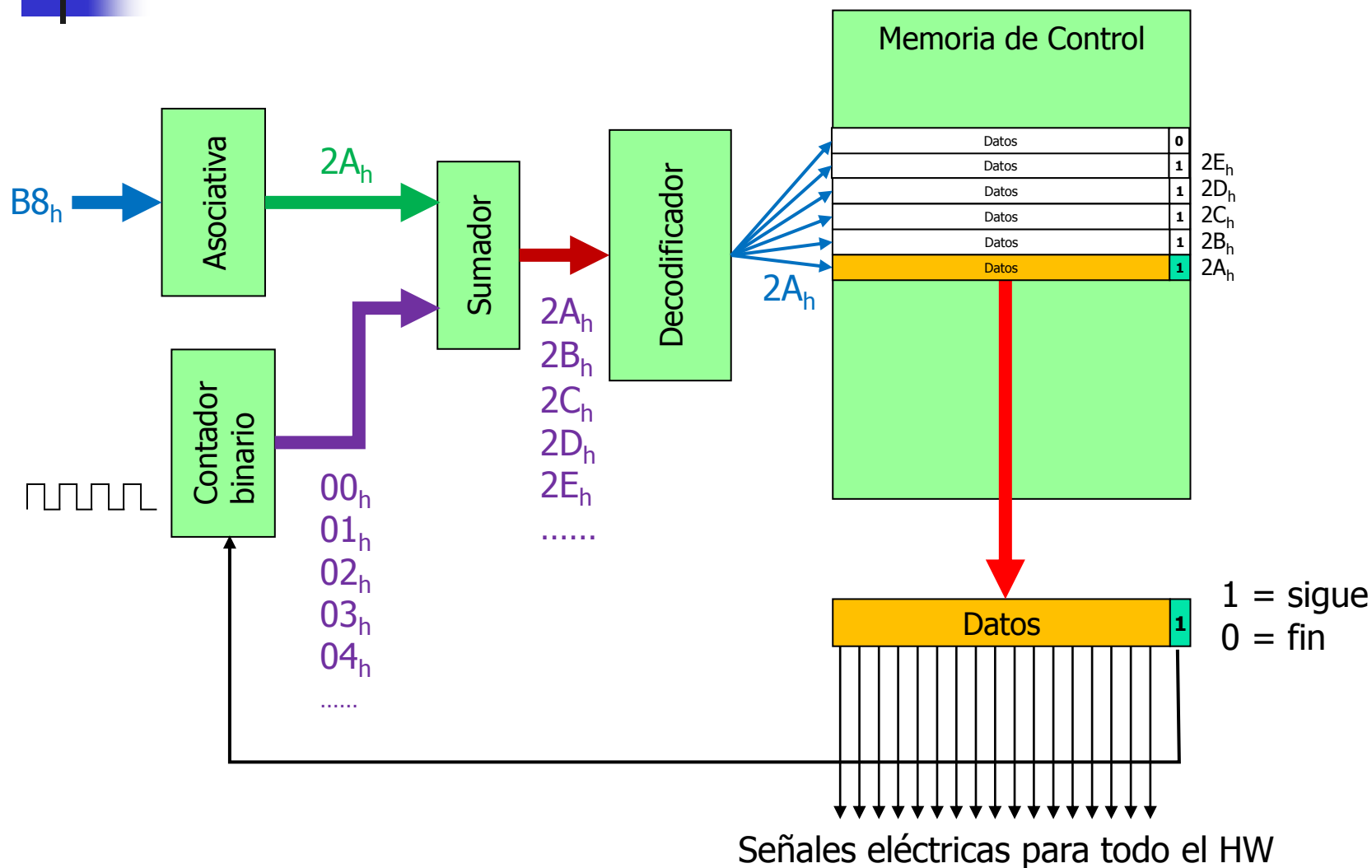
- Estos unos y ceros de Control están almacenados en la UDC.
- Cada instrucción se ejecuta en la UDC como un conjunto de microinstrucciones. A este conjunto se lo denomina microprograma. Cada microinstrucción ocupada un ciclo de CLK.



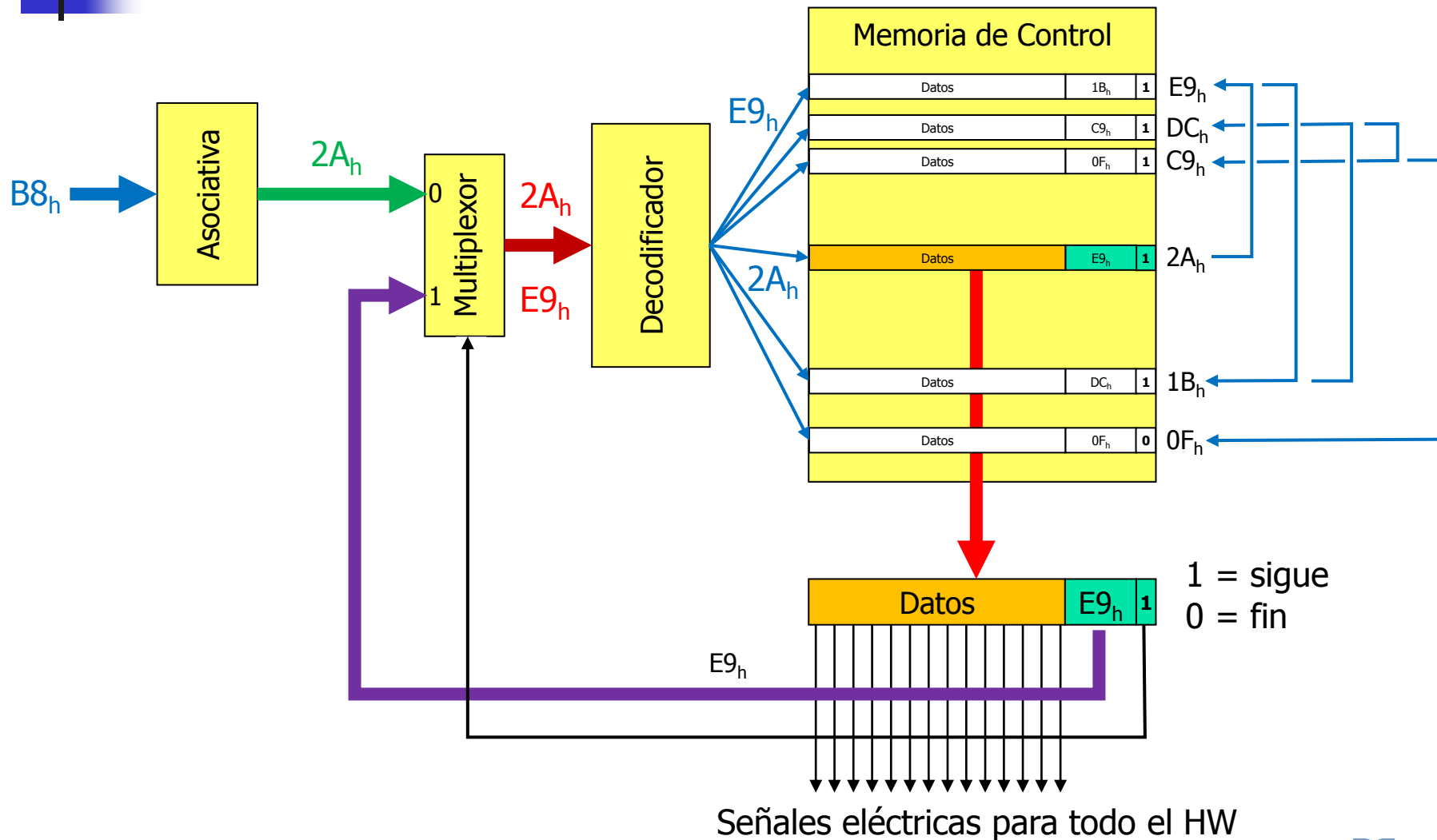
Secuenciamiento interno de microinstrucciones

- De acuerdo a como se ubiquen las microinstrucciones en la memoria de control se tendrán dos tipos de secuenciamiento de las mismas.
- Secuenciamiento implícito: Las microinstrucciones se encuentran ordenadas secuencialmente en la memoria física en direcciones contiguas. Se emplea un incrementador al cual ingresa la dirección de la microinstrucción en curso y de esa manera se accede a las subsiguientes microinstrucciones.
- 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 microinstrucción.

Secuenciamiento: Implícito



Secuenciamiento: Explícito

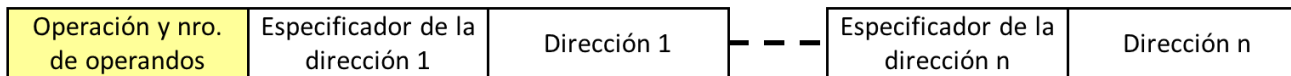


Formato de instrucciones

- El formato consiste en un primer paquete de bits que define la operación (OpCode o COP = Código de Operación) y luego una serie de paquetes de bits (Objects) con la información complementaria para completar la operación



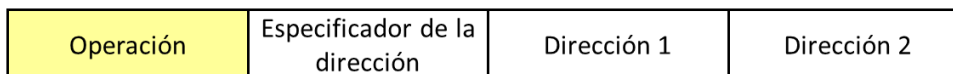
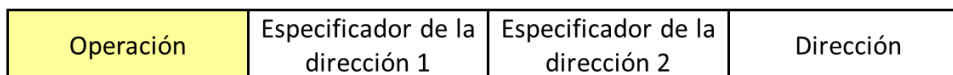
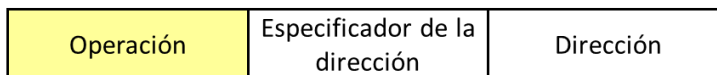
Formato de instrucciones de longitud variable (VAX)



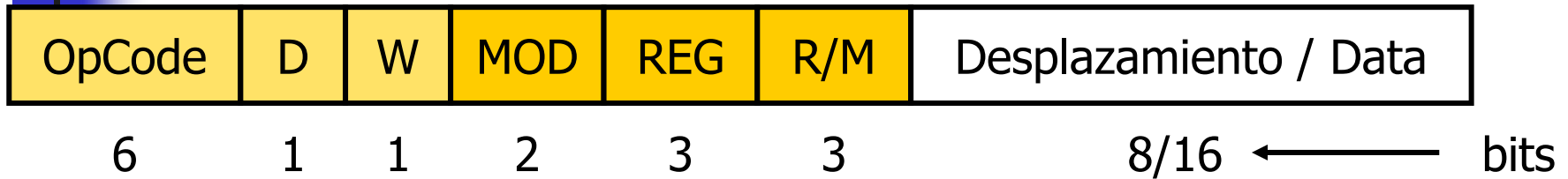
Formato de instrucciones de longitud fija (DLX, MIPS, PowerPC, HP-PA, SPARC)



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



Formato de instrucciones en 8086/88



- D = Desde/hacia registros
 - W = Transferencia de 1 o 2 bytes
 - MOD = Modo de direccionamiento
 - REG = Registros involucrados
 - R/M = Lugar del segundo operando registro/memoria
 - Desplazamiento/Data = Dependiendo de la instrucción (Objetos)
- Ejemplo: MOV AX,1234 → B8 34 12

Immediate to Register

1 0 1 1 w reg	data	data if w = 1
---------------	------	---------------

REG Code	Register Selected
0 0 0	AL AX



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,` → "B8_h"



Set de instrucciones

■ 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

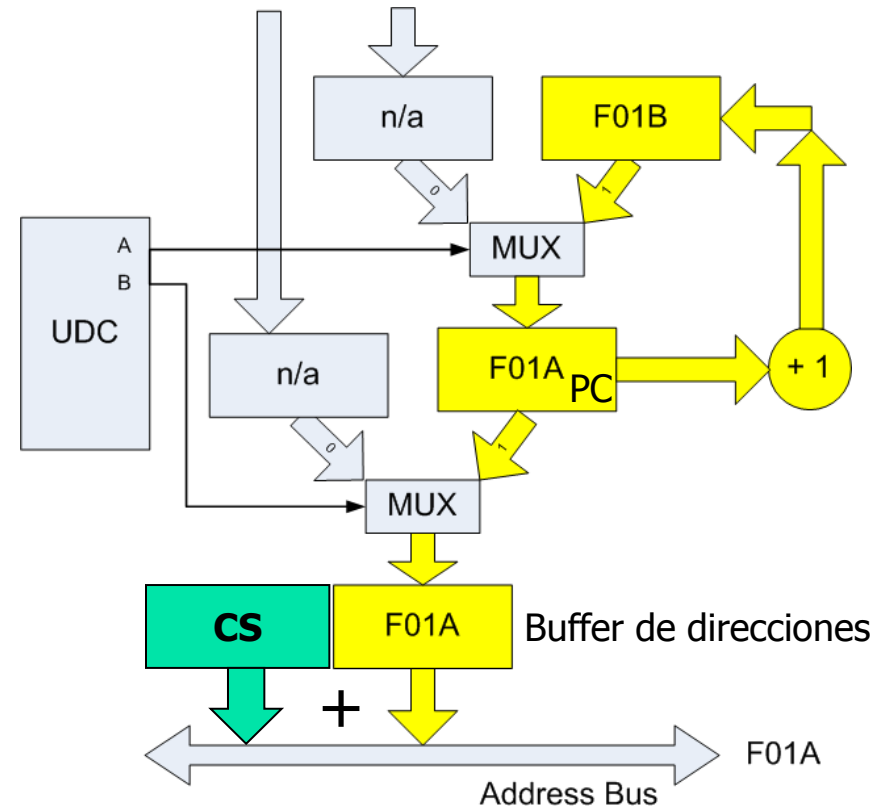


Contador de Programa (PC)

- Llamado "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

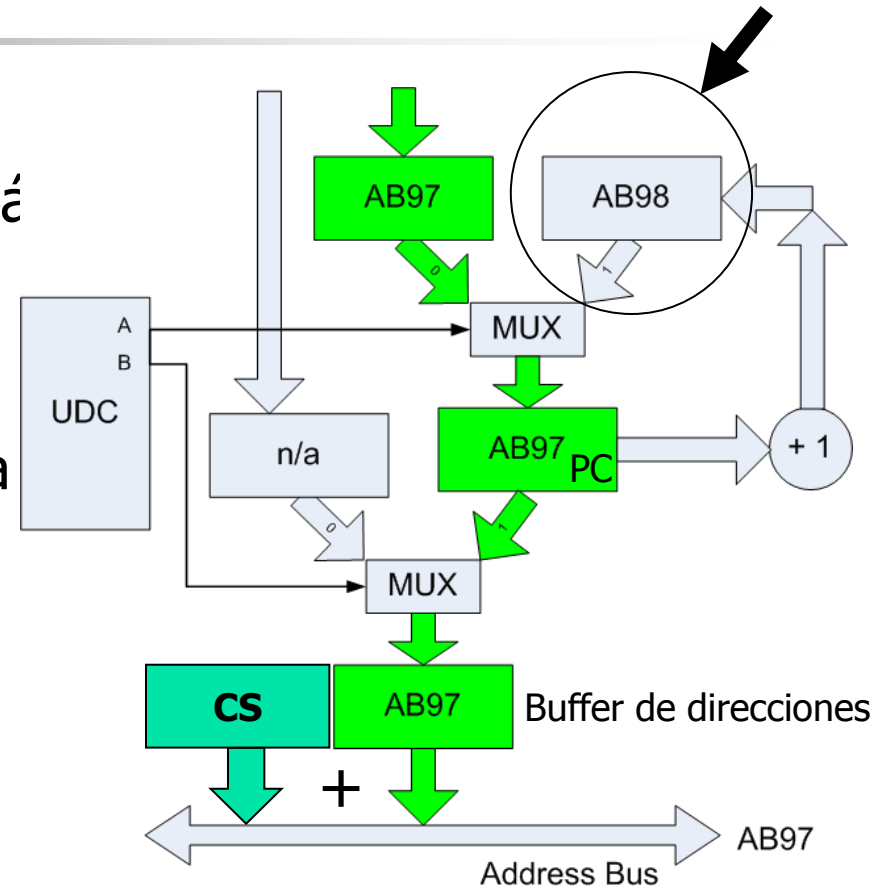
Contador de Programa: Modo Habitual

- 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, 1234_h
 - INC AX
 - ...



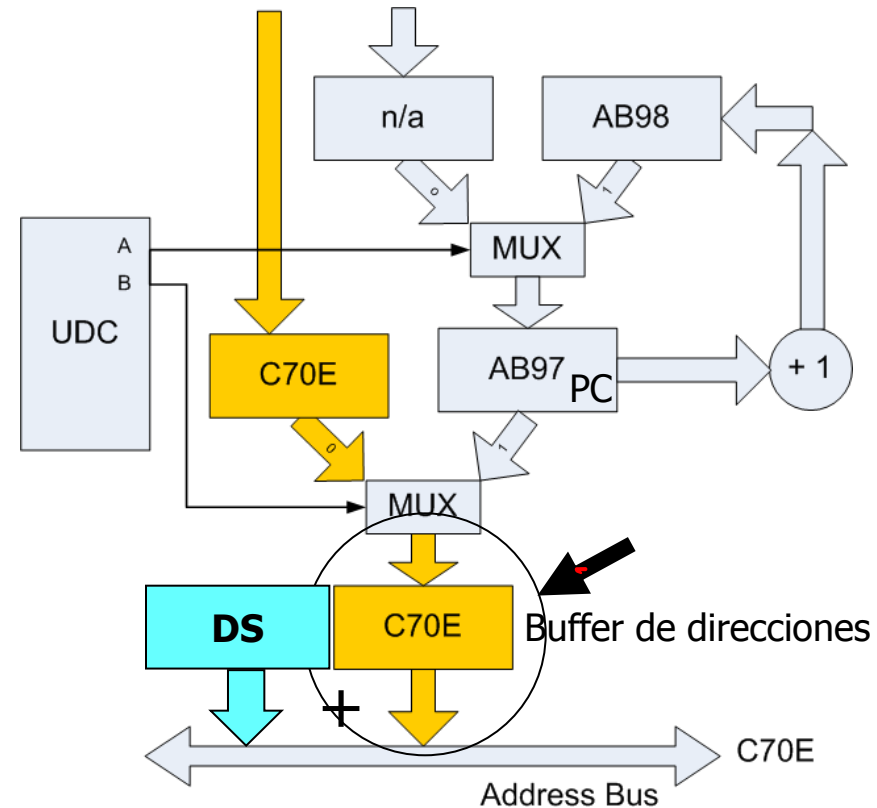
Contador de Programa: Modo Salto

- 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 continua luego la ejecución en forma habitual
- A=0 y B=1
- Ejemplo
 - MOV AX, 1234_h
 - JMP AB97_h
 - ...



Contador de Programa: Modo Indirección

- 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`
 - ...



Ciclo de instrucción

- La UDC ejecutará instrucciones siguiendo un ciclo repetitivo al que llamaremos Ciclo de Instrucción
- El ciclo de instrucción de nuestro procesador es el siguiente:

Ciclo de instrucción del procesador en estudio



IF = Instruction Fetch (Búsqueda del Código de Operación)

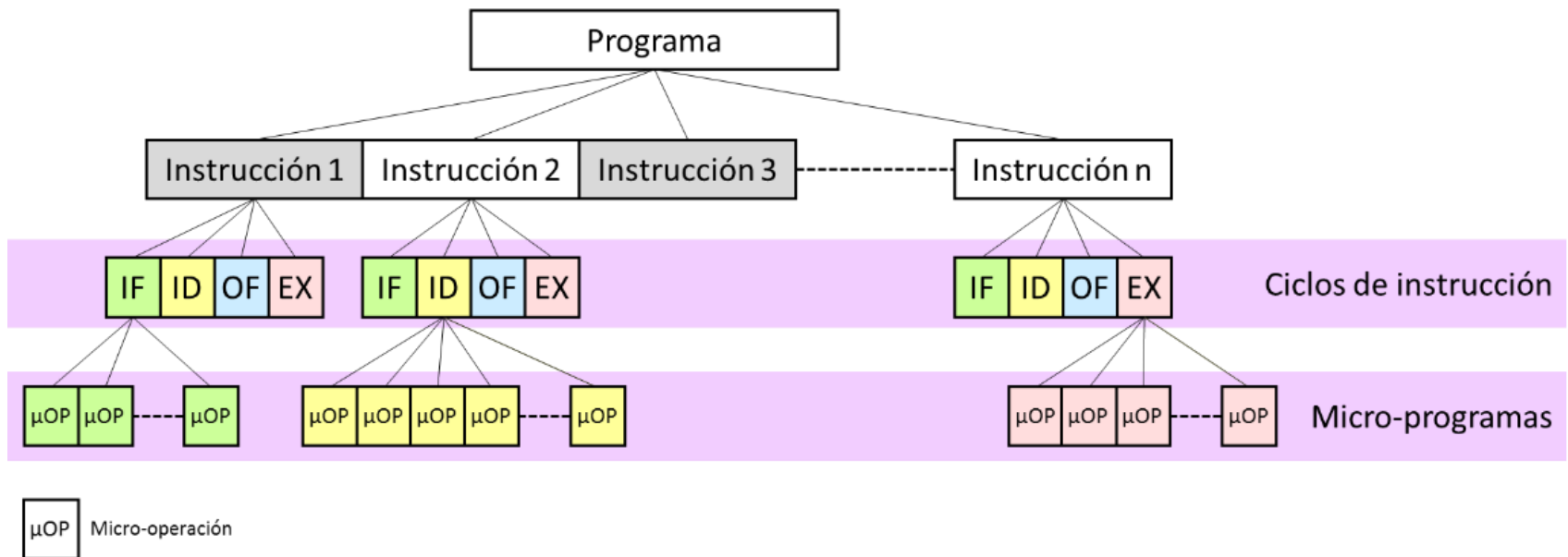
ID = Instrucion Decode (Decodificación del Código de Operación)

OF = Operand Fetch (Búsqueda de los operandos u objetos restantes)

EX = Execution (Ejecución de la operación)

Ciclo de instrucción

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



Ciclo de instrucción

- Búsqueda del Código de Operación (Instruction Fetch)
 - La UDC procederá a acceder a la MP a la dirección donde apunta el PC y leerá el COP. Este COP vendrá por el bus de datos y será alojado en un registro interno llamado Instruction Register (Registro de Instrucción).

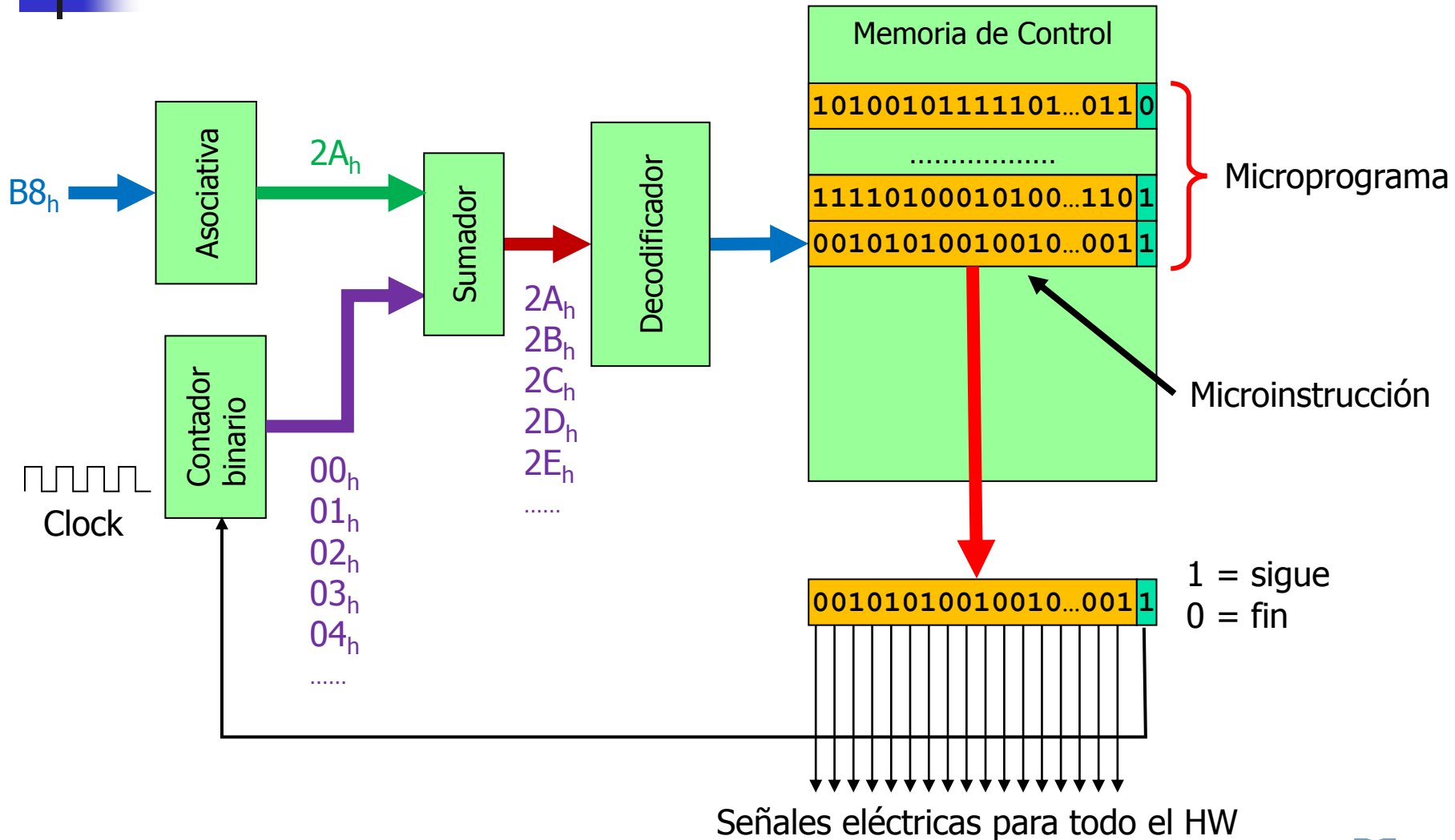
Operación	Especificador de la dirección	Dirección
-----------	-------------------------------	-----------

Operación	Especificador de la dirección 1	Especificador de la dirección 2	Dirección
-----------	---------------------------------	---------------------------------	-----------

Operación	Especificador de la dirección	Dirección 1	Dirección 2
-----------	-------------------------------	-------------	-------------

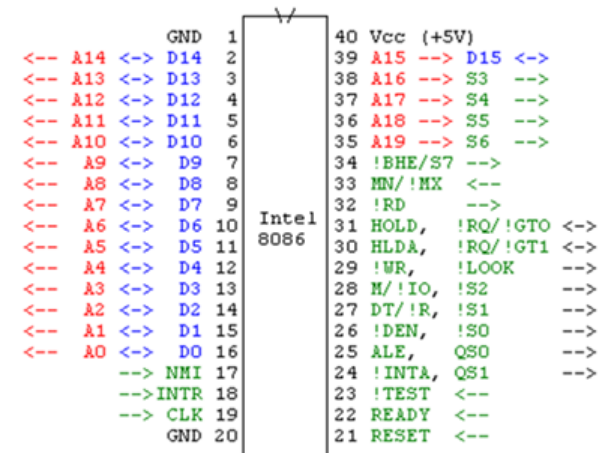
- Decodificación del Código de Operación (Instruction Decode)
 - El contenido del registro de instrucción será decodificado (mediante un Decoder o Decodificador) entregando la posición de la Memoria de Control en donde se encuentra la primera microinstrucción a ejecutar. A partir de allí, la UDC por si sola recorrerá todas las microinstrucciones del microprograma de ejecución de la instrucción

Ejecución de microprogramas



Capacidad de direccionamiento

- La capacidad de direccionamiento de un procesador se referirá a varias características. Indicará cuántas direcciones (en binario) es capaz de generar para:
 - Acceder a la memoria física (MP)
 - Acceder a los puertos de entrada/salida
 - Acceder a la memoria virtual (si posee esta característica)
- La capacidad de direccionamiento para "n" bits es 2^n
- La cantidad de bits de cada dirección estará dada por la cantidad de pines que tenga el microprocesador en el bus de direcciones
- El Intel 8086/8088 tiene 20 bits ($A_0...A_{19}$)
- Para el acceso a los periféricos solo se utilizarán los primeros 16 bits ($A_0...A_{15}$) es decir, 65.536 direcciones (desde la 0000_h hasta la $FFFF_h$).





Paralelismo

- Técnica para reducción de tiempos y costos de recursos
→ Mejorar la performance
- Paralelismo temporal
 - Solapar tiempos en un mismo dispositivo
- Paralelismo espacial
 - Agregar más dispositivos para trabajar al mismo tiempo

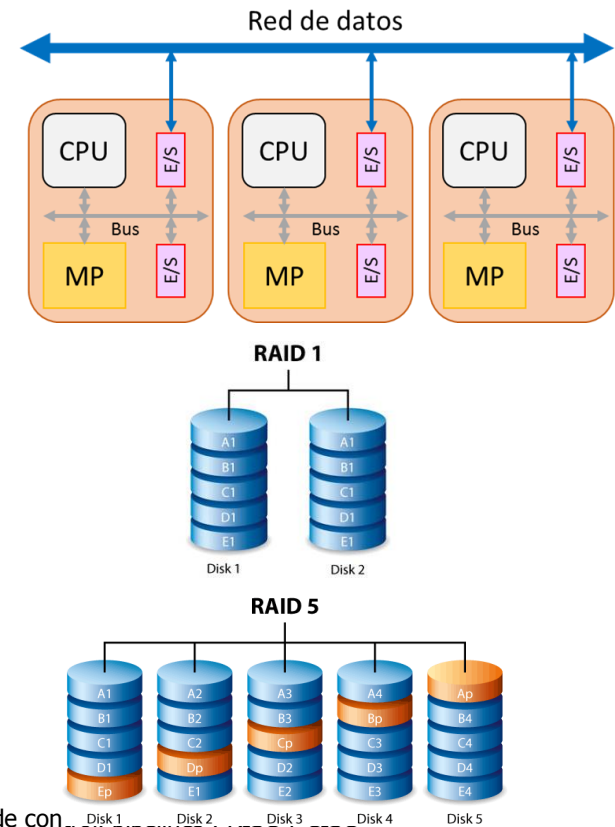


Paralelismo: Multiprocesamiento simétrico (SMP)

- Todos los procesadores (2 o más):
 - Son independientes entre sí
 - Son de similares características
 - Comparten el mismo espacio de memoria y de Entrada/Salida
 - Poseen similares tiempos de acceso a la misma Memoria Principal
 - Pueden ejecutar los mismos algoritmos o funciones
 - Son controlados por un único Sistema Operativo integrado
- Multicores
- Es transparente para el usuario
- Brinda un incremento sensible en la performance

Paralelismo: Multiprocesamiento en Clústeres

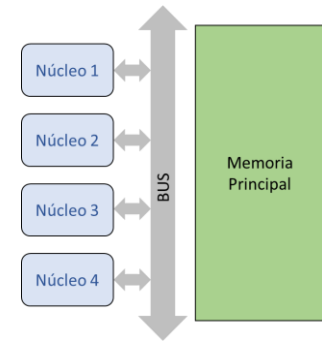
- Compiten con los SMP
- Conjunto de computadores que trabajan como si fueran una sola super-computadora
- Cada computadora puede funcionar por sí sola como un ordenador con todas las funciones
- Nodo → Cada computador del clúster
- Se enlazan o comunican a través de una red:
 - De Área Local (LAN – Local Area Network)
 - Infiniband EDR: cerca de 195 Gbps (giga bits por seg.)
 - Fiber Channel: 128 Gbps
 - De Área Ampliada (WAN – Wide Area Network)
- Poseen discos compartidos por todos los computadores con configuración redundante de discos (RAID – Redundant Array of Inexpensive Disks)



Acceso Uniforme y Heterogéneo a Memoria (UMA y NUMA)

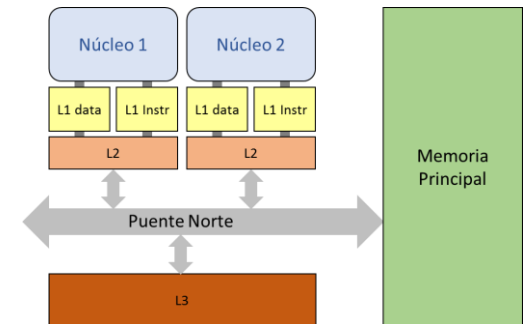
■ Acceso Uniforme (Homogéneo)

- Todos los procesadores comparten el mismo espacio de memoria
- Los tiempos de acceso a la memoria son independientes del procesador que acceda



■ Acceso No Uniforme (Heterogéneo)

- Cada procesador tiene un espacio propio de memoria que no comparte con otros procesadores
- Se utiliza el caché L3 para compartir entre todos los procesadores y los niveles L2 y L1 son propios
- Esto evita que los datos compartidos circulen por los buses ya que están en el L3
- Coherencia de caché entre todos los niveles



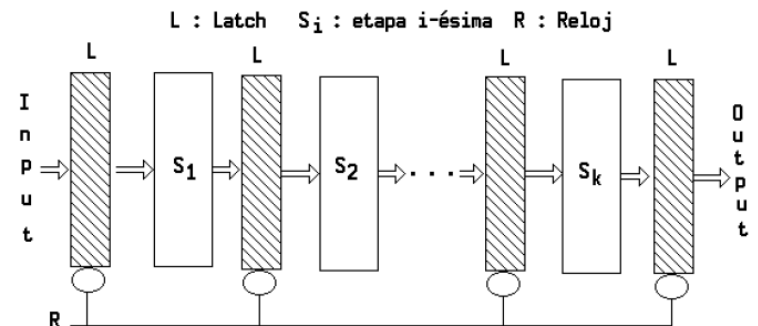
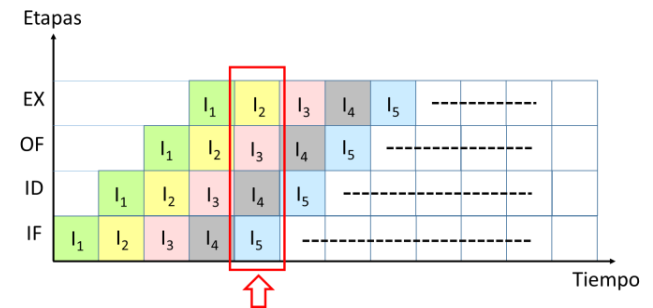
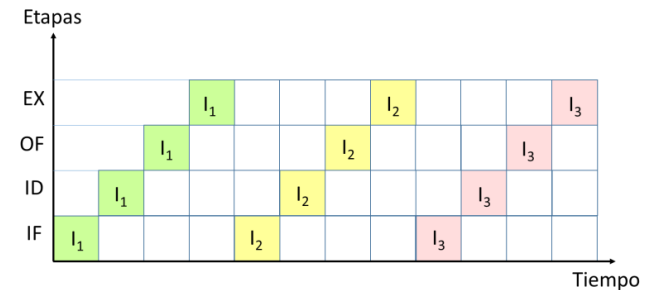
Pipelines de instrucciones

■ 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

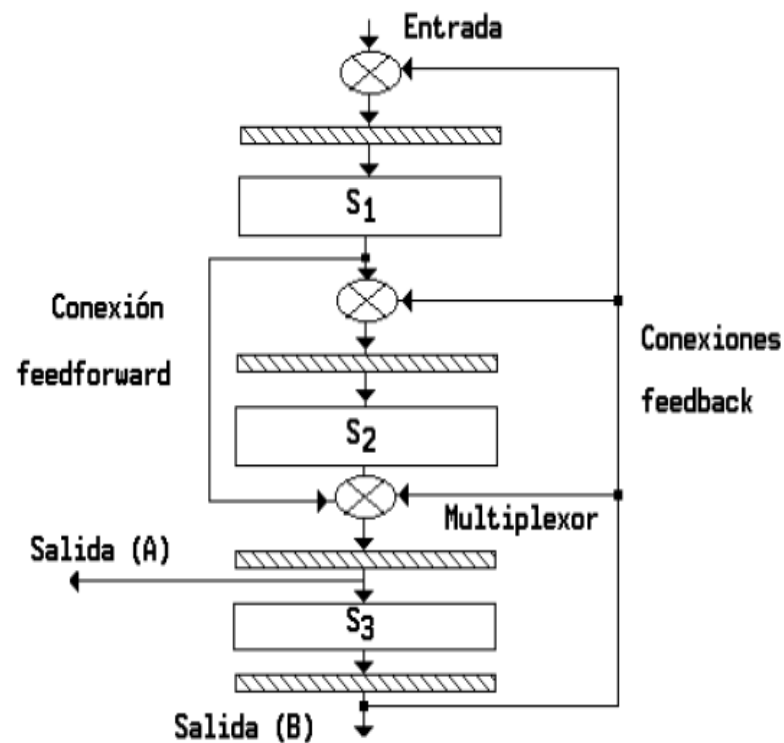


Pipelines de instrucciones: Tablas de reservas

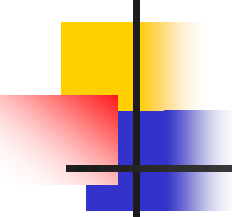
Tiempo

	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



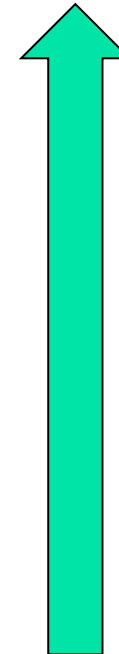
Pipelines de instrucciones: Tablas de reservas



	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
Instrucción A	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	

Instrucción C	S_1	C				C		C
	S_2		C	C	C			
	S_3						C	



Pipelines de instrucciones: Tablas de reservas

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								

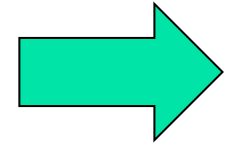
Instrucción C

S_1	C				C		C	
S_2		C	C	C				
S_3						C		C

Pipelines de instrucciones: Tablas de reservas

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

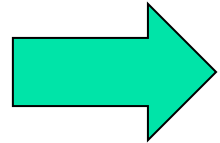
Instrucción C

S_1	C				C		C	
S_2		C	C	C				
S_3						C		C

Pipelines de instrucciones: Tablas de reservas

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

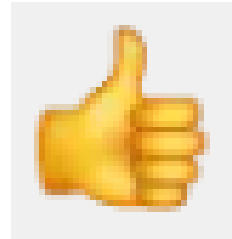
Instrucción C

S_1	C				C		C	
S_2		C	C	C				
S_3						C		C

Pipelines de instrucciones: Tablas de reservas

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 de instrucciones: Tablas de reservas

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

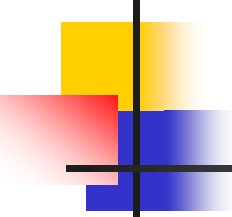
S_1
 S_2
 S_3

Instrucción C

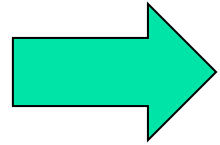
S_1	C				C		C	
S_2		C	C	C				
S_3						C		C



Pipelines de instrucciones: Tablas de reservas



		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		B



Instrucción B

S_1

S_2

S_3

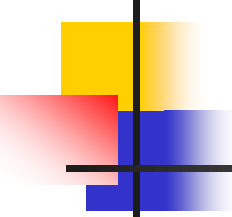
Instrucción C

S_1

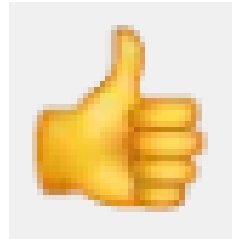
S_2

S_3

Pipelines de instrucciones: Tablas de reservas



		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	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

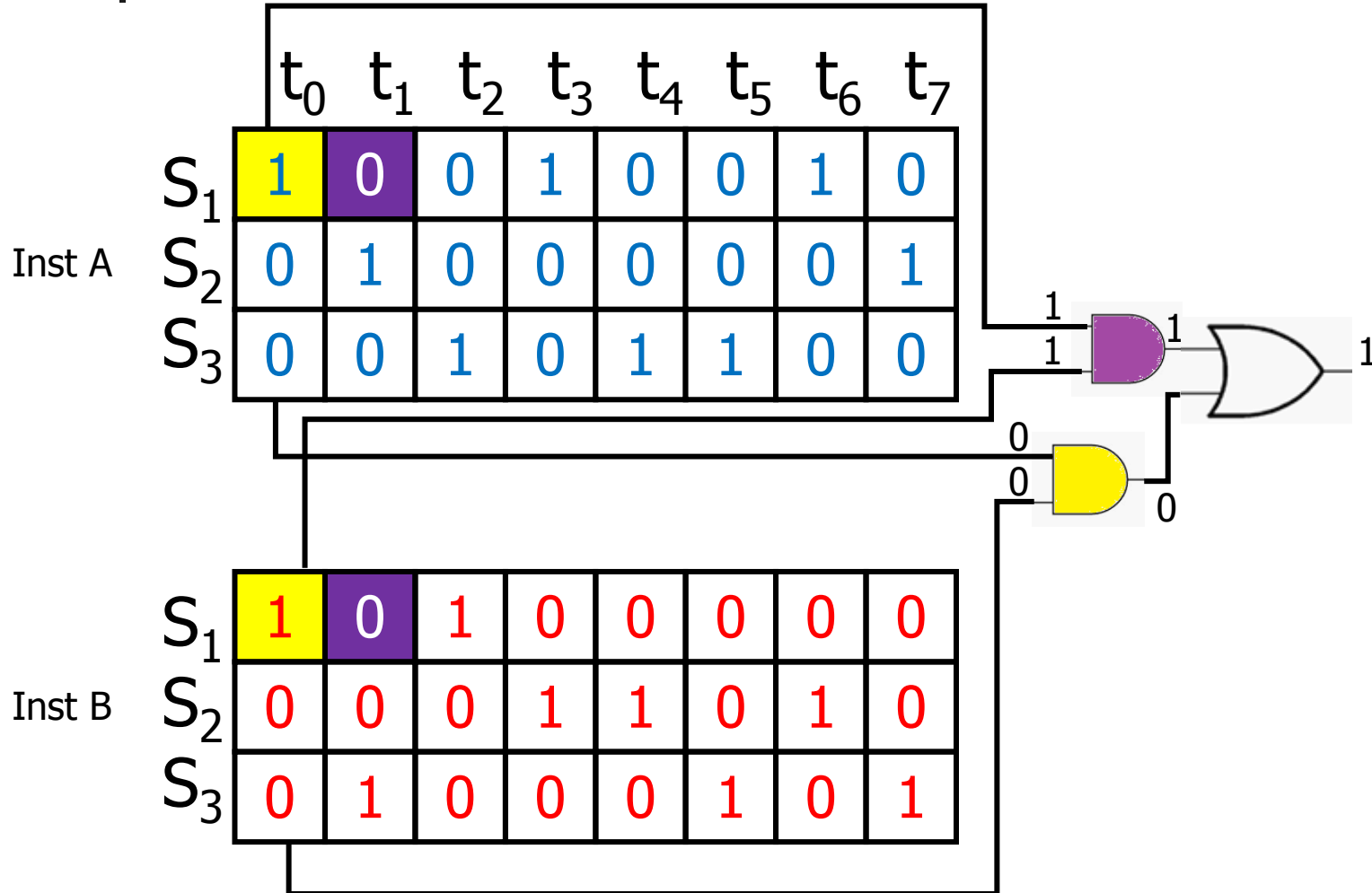
S_3

Pipelines de instrucciones: Tablas de reservas

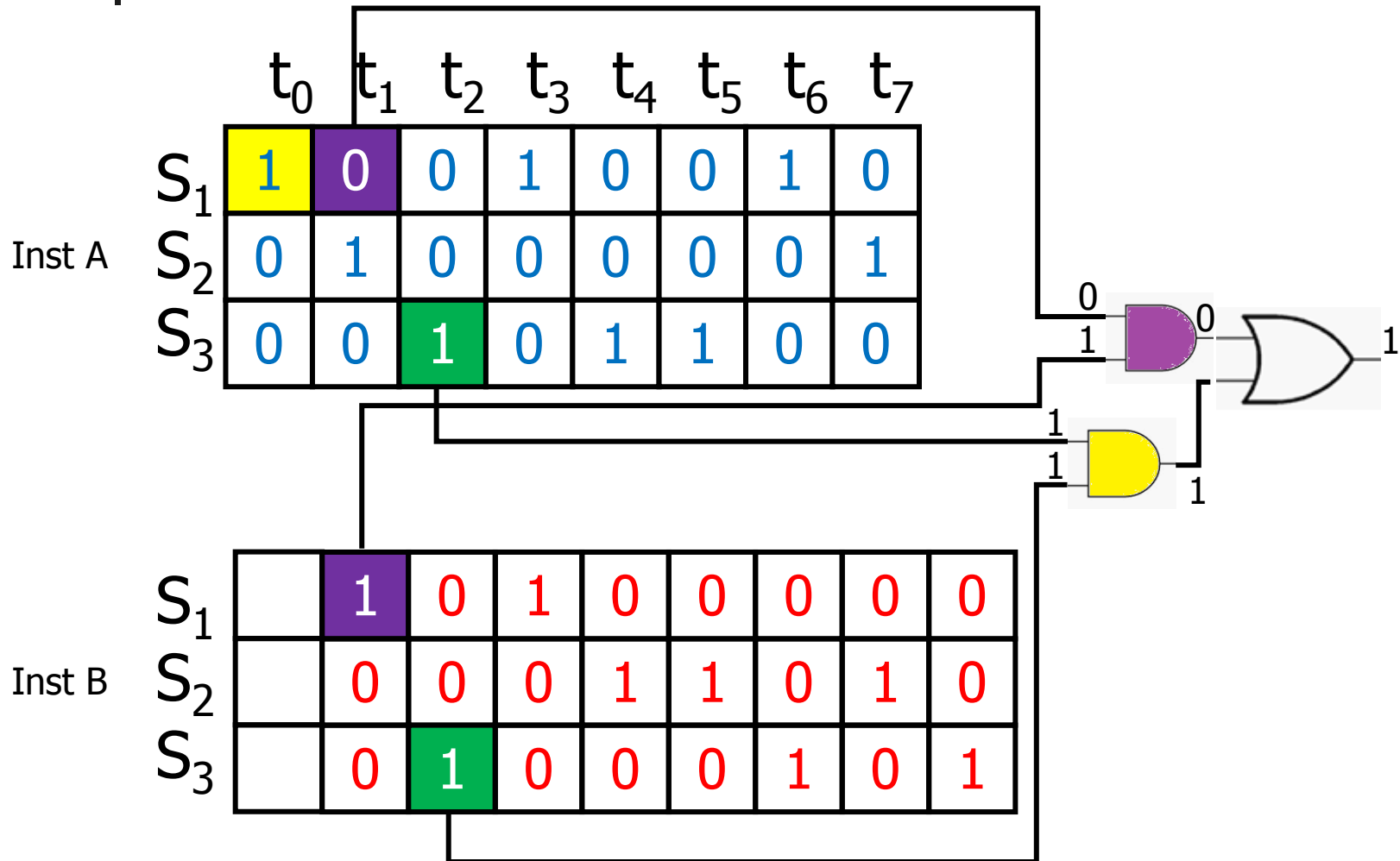
	t ₀	t ₁	t ₂	t ₃	t ₄	t ₅	t ₆	t ₇	t ₈	t ₉	t ₁₀	t ₁₁	t ₁₂	t ₁₃	t ₁₄	t ₁₅	t ₁₆	t ₁₇	t ₁₈	t ₁₉	t ₂₀	t ₂₁	t ₂₂	t ₂₃
S ₁	A			A			A		B		B						C				C		C	
S ₂		A						A				B	B		B			C	C	C				
S ₃			A		A	A				B				B		B						C		C

	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

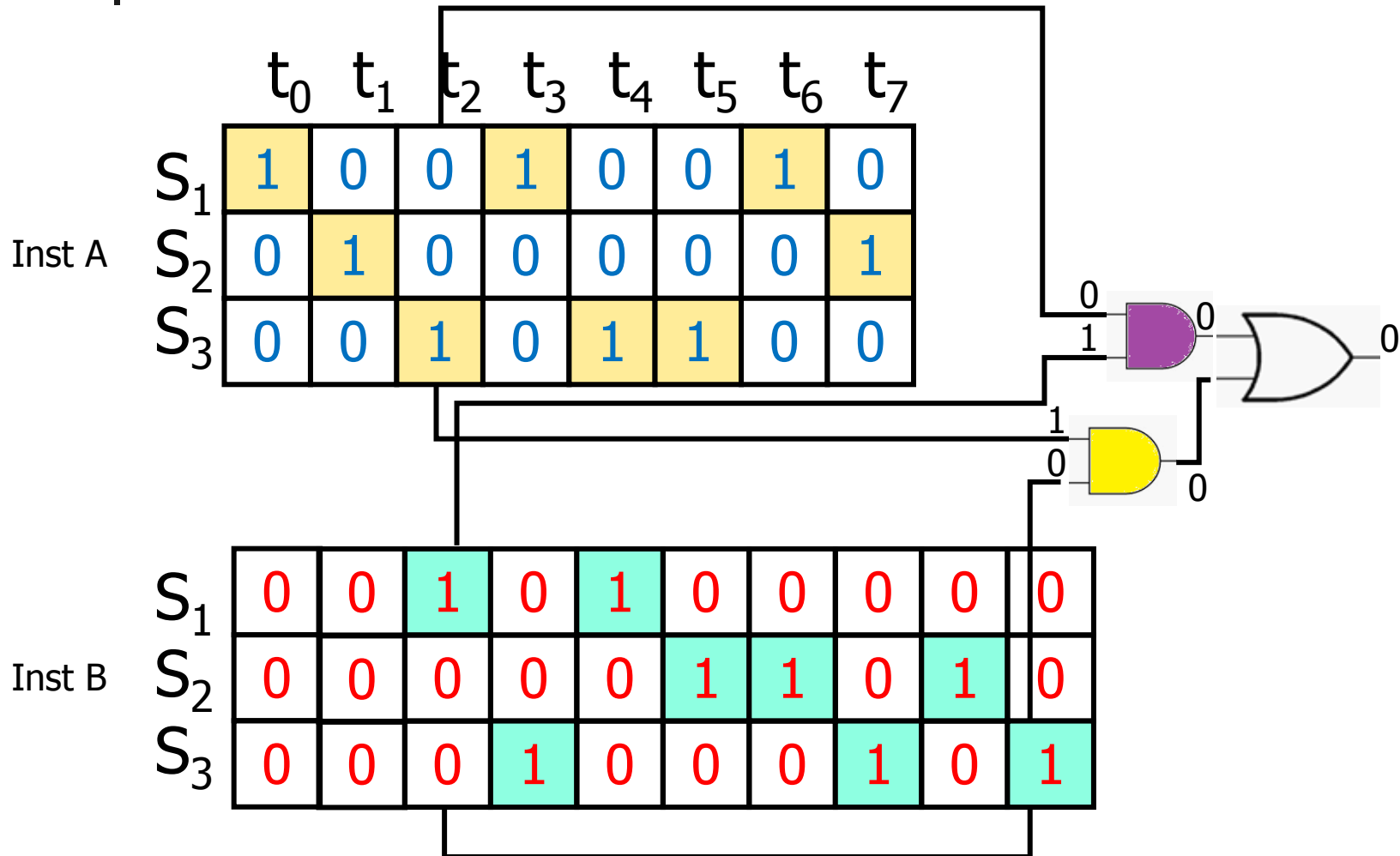
Pipelines de instrucciones: Tablas de reservas



Pipelines de instrucciones: Tablas de reservas



Pipelines de instrucciones: Tablas de reservas



Problemas en los Pipelines: de Datos

- 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 registro modificado, es leído inmediatamente luego. Como la modificación puede no haber finalizado, la lectura puede ser errónea
 - $i1.R2 = R1 + R3$
 - $i2.R4 = R2 + R3$
 - En nuestro procesador, un ejemplo sería:
 - `MOV AX, BX`
 - `MOV [DS:0302], AX`

Problemas en los Pipelines: De Datos

- Problemas de datos: **WAR (Write-After-Read)**
 - Similar al anterior pero alternando la lectura por modificación y viceversa
 - $i1.R4 = R1 + R3$
 - $i2.R3 = R1 + R2$
 - En nuestro procesador, un ejemplo sería:
 - `MOV SI, AX`
 - `MOV AX, [BX + 20h]`



Problemas en los Pipelines: De Datos

- Problemas de datos: **WAR (Write-After-Write)**
 - Se ejecutan dos instrucciones que escriben el mismo operando. La primera puede terminar luego de la segunda
 - $i1.R2 = R1 + R2$
 - $i2.R2 = R4 \times R7$
 - En nuestro procesador, un ejemplo sería:
 - `MOV DX, 200h`
 - `MOV DX, offset DATONUEVO`

Problemas en los Pipelines: De Control

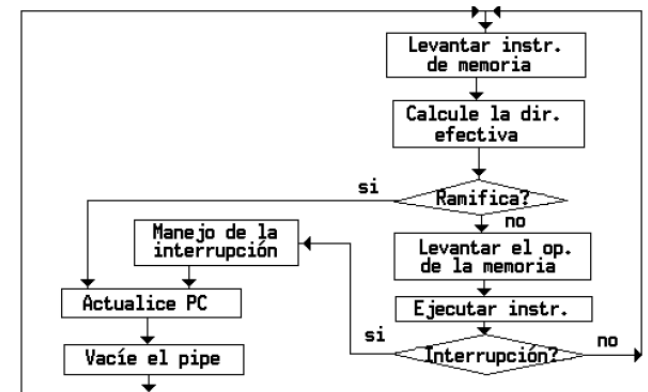
■ Problemas de control: Salto condicionados

- 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.JMP A = 0 to i.n
- i3.D = A x B

- En nuestro procesador sería:

```
073F:0100 B401      MOV     AH,01
073F:0102 CD21      INT      21
073F:0104 3C0D      CMP     AL,0D
073F:0106 7478      JZ      0180
073F:0108 EBF6      JMP     0100
-- --
073F:0180 B8004C     MOV     AX,4C00
073F:0183 CD21      INT      21
```



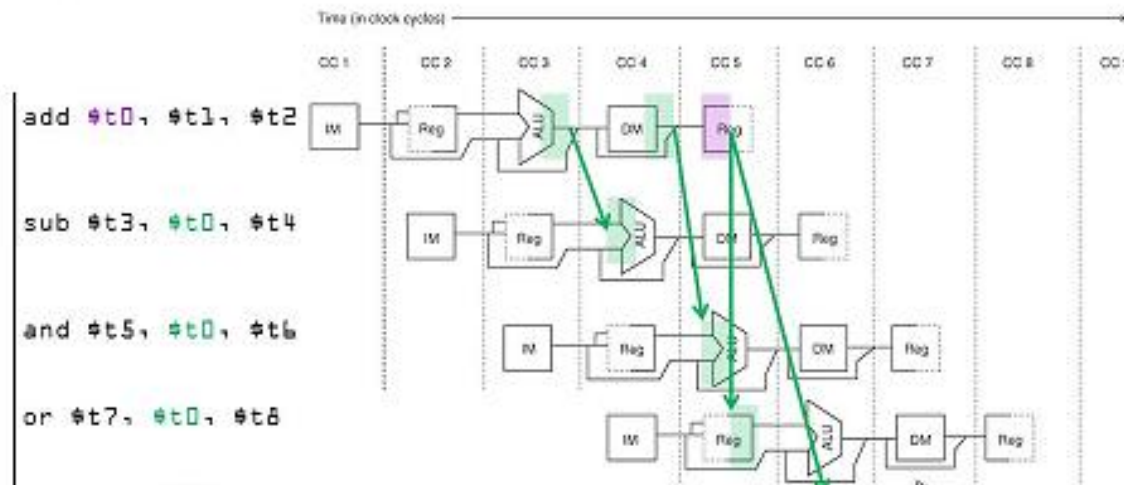
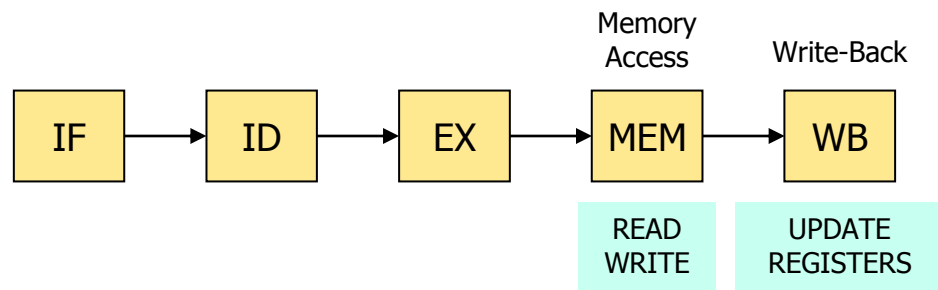


Problemas en los Pipelines: De Estructura

- Problemas de estructura: **Necesito más hardware**
 - Ocurren cuando dos instrucciones requieren una misma unidad estructural (por ejemplo la ALU)
 - i1.JMP A = 0 ← Requiere la ALU para comparar
 - i2.CMP A,B ← Requiere la ALU para comparar
 - En nuestro procesador sería:
 - ADD AX, BX
 - MUL CX

Solución al problema de Datos: **Forwarding**

- **Forwarding (bypassing o short-circuiting)**: Se envía el dato necesario de manera anticipada, sin esperar a que se actualice el registro final. Lo manda antes (desde la etapa MEM a donde se lo requiera)





Solución al problema de Control: 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



Solución al problema de Control: Predicción de saltos

■ Predictores de saltos: 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

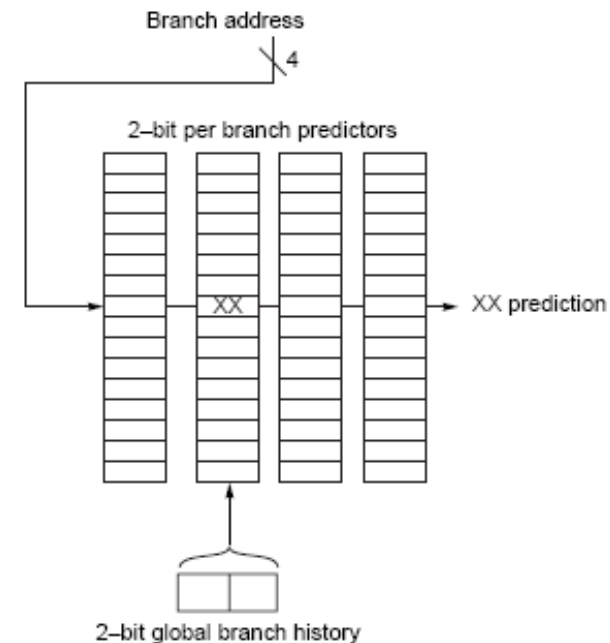
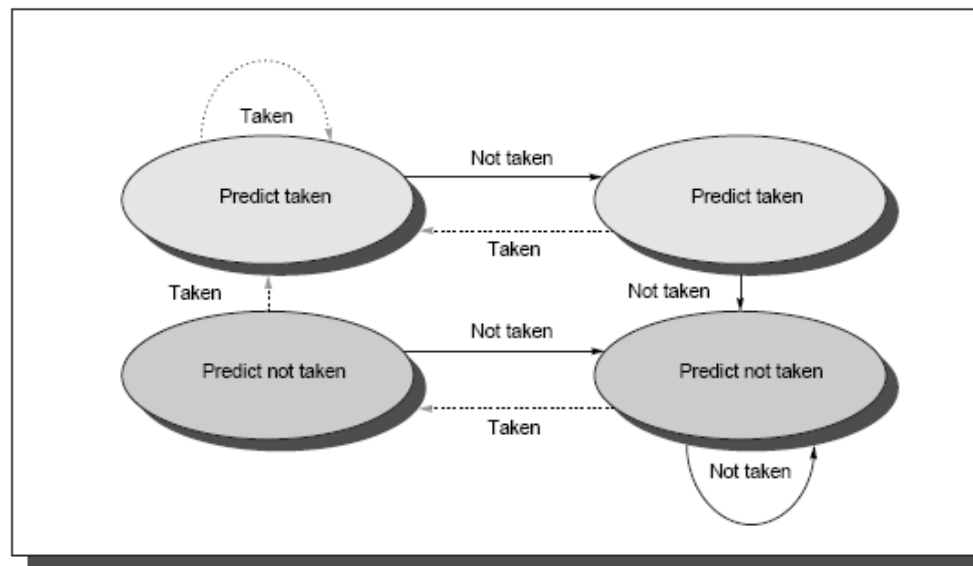
■ Predictores de saltos: Dinámicos

- 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

Solución al problema de Control: Predicción de saltos

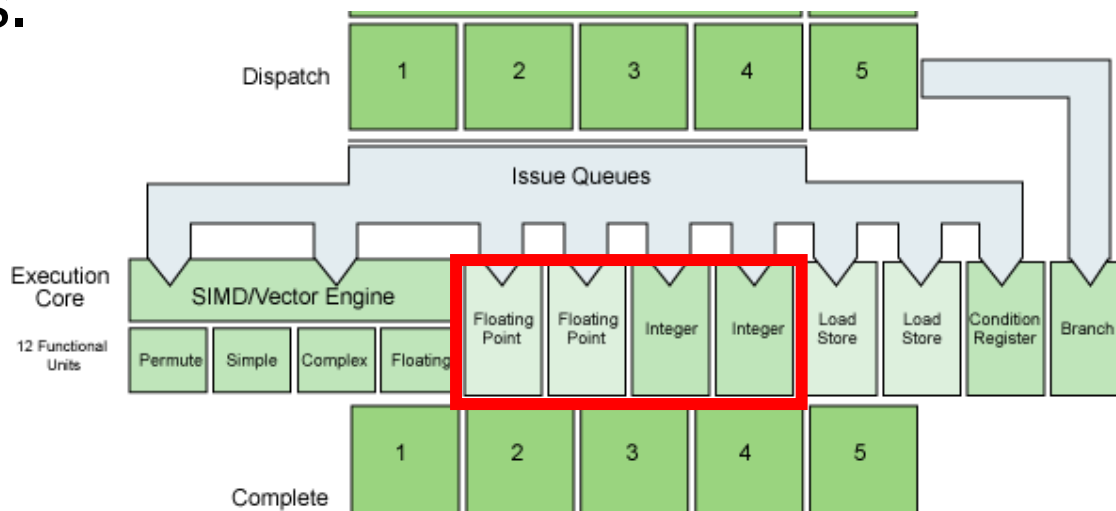
■ Predictores de saltos dinámicos

- 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



Solución al problema de Estructura: Más unidades

- Agregaremos más estructuras.
- Si por ejemplo, tengo que realizar dos instrucciones en donde debe intervenir la ALU, debiera esperar a que finalice la primera para poder ingresar la segunda.
- Por ello, se pueden integrar en lugar de una ALU, 2 ALUs.



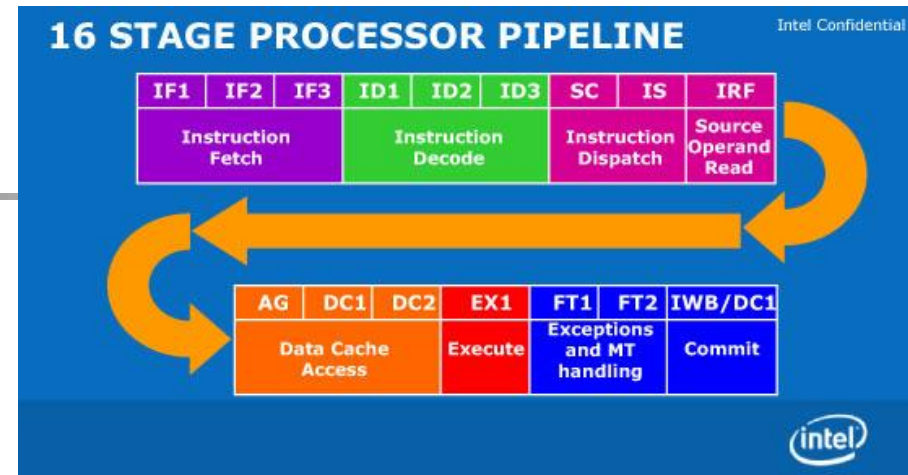


Superpipelines

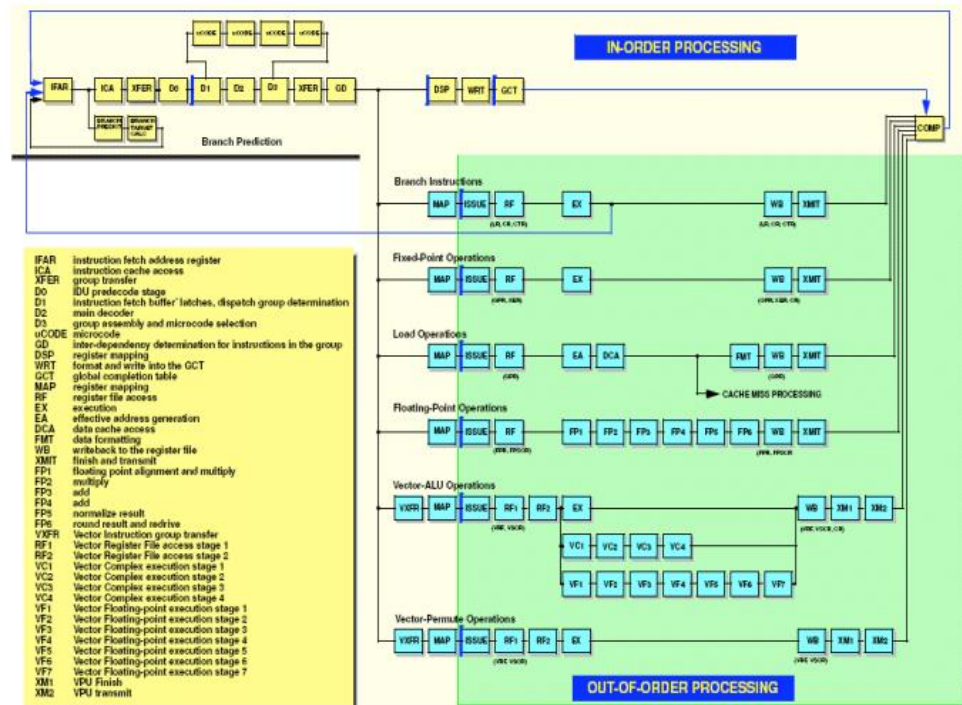
- Llamado “Pipeline Superescalar”. Presente en CPU modernos
- Pipelines con un gran número de pequeñas etapas para aumentar el paralelismo (profundidad del pipeline) y así mejorar la performance en la ejecución de instrucciones
- Características
 - Mayor cantidad de etapas (stages)
 - Múltiple despacho de instrucciones (más de una por ciclo de reloj)
 - Ejecución fuera de orden
 - Ejecución especulativa (sin saber si se ejecutarán finalmente)
 - Avanzados predictores de saltos

Superpipelines

- Core 2 Duo



- IBM PowerPC 970FX conocido como G5





Procesadores superescalares y el paralelismo espacial

- Procesadores con un único núcleo y múltiples Pipelines de instrucción. El núcleo contiene múltiples unidades de ejecución
- Se pueden ejecutar varias instrucciones simultáneamente en un mismo ciclo de reloj
- Cada Pipeline puede ser general o estar especializado en diferentes tipos de instrucciones
- Incorporan otras características:
 - Ejecución fuera de orden
 - Ejecución especulativa, etc.
- Múltiples pipelines en paralelo implica mantener un flujo de instrucciones continuo para alimentarlos

Procesadores superescalares y el paralelismo espacial

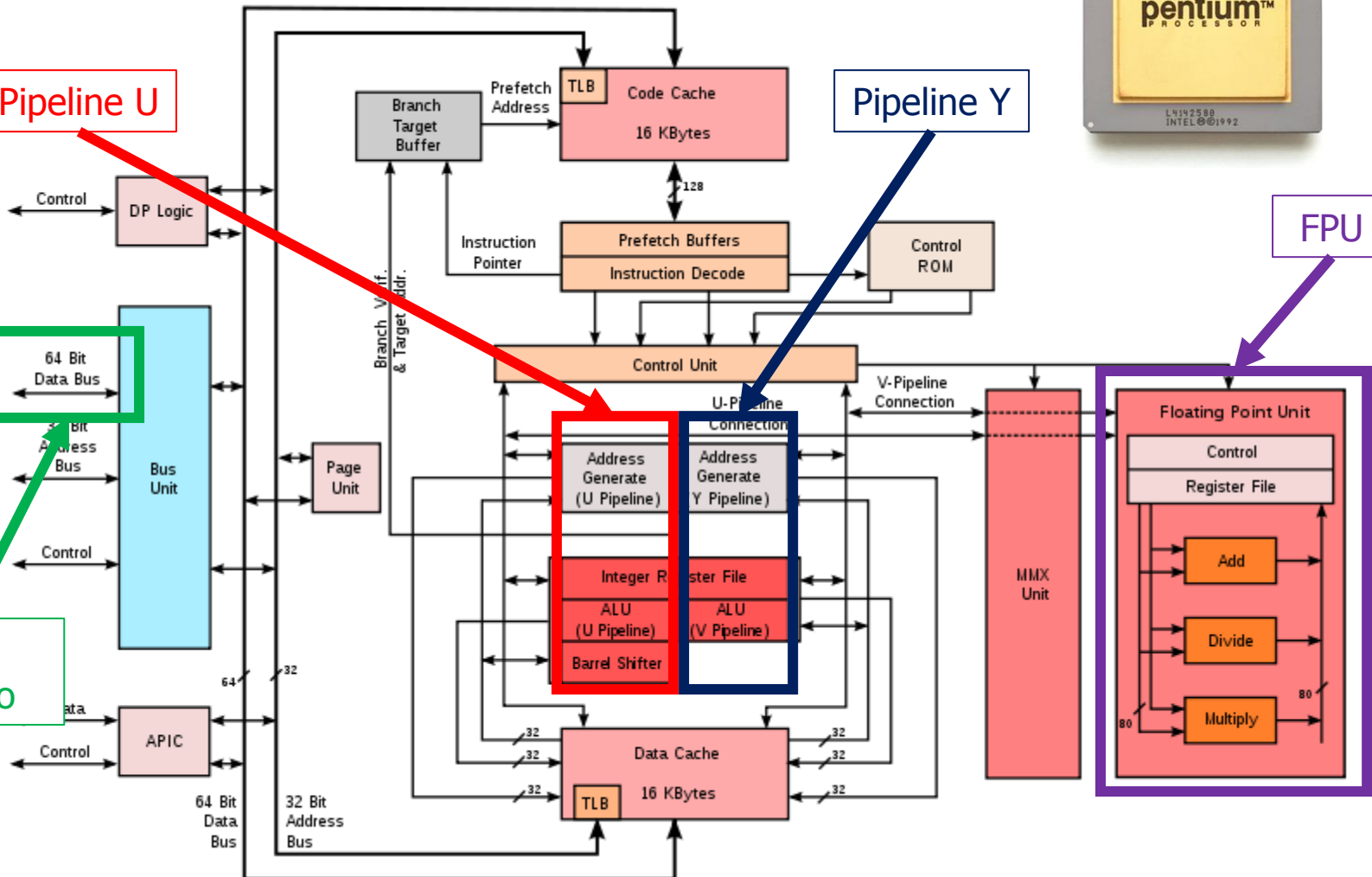


Pipeline U

Pipeline Y

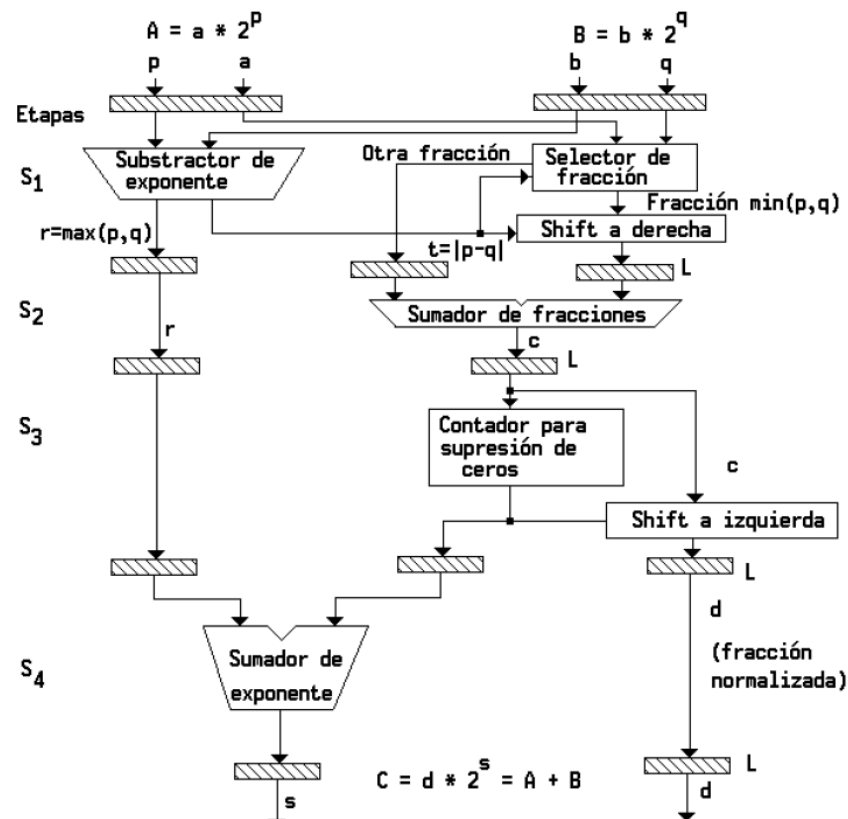
FPU

Bus ancho



Pipelines de cálculo: ALU punto flotante

- Pipeline Sumador de Punto Flotante en 4 etapas



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

Arquitecturas CISC y RISC



- CISC = Complex Instruction Set Computer
- Set de instrucciones complejo donde cada una es un conjunto de más de una micro-instrucción
- Todas las micro-instrucciones están en la Memoria de Control en la UDC
- Poseen múltiples modos de direccionamiento
- Las instrucciones tienen distintos tamaños, tardan distinto
- Los registros son de propósito específico
- IBM System/360, PDP-11, VAX, Motorola 68000, e Intel familia x86

Arquitecturas CISC y RISC

Complete 8086 instruction set

Quick reference:

	CMPSB				MOV		
AAA	CMPSW	JAE	JNBE	JPO	MOVS	RCR	SCASB
AAD	CWD	JB	JNC	JS	MOVSW	REP	SCASW
AAM	DAA	JBE	JNE	JZ	MUL	REPE	SHL
AAS	DAS	JC	JNG	LAHF	NEG	REPNE	SHR
ADC	DEC	JCXZ	JNGE	LDS	NOP	REPZ	STC
ADD	DIV	JE	JNL	LEA	NOT	REPZ	STD
AND	HLT	JG	JNLE	LES	OR	RET	STI
CALL	IDIV	JGE	JNO	LODSB	OUT	RETF	STOSB
CBW	IMUL	JL	JNP	LODSW	POP	ROL	STOSW
CLC	IN	JLE	JNS	LOOP	POPA	ROR	SUB
CLD	INC	JMP	JNZ	LOOPE	POPF	SAHF	TEST
CLI	INT	JNA	JO	LOOPNE	PUSH	SAL	XCHG
CMC	INTO	JNAE	JP	LOOPNZ	PUSHA	SAR	XLATB
CMP	IRET	JNB	JPE	LOOPZ	PUSHF	SBB	XOR
	JA				RCL		

Arquitecturas CISC y RISC



- RISC = Reduced Instruction Set Computer
- Basados en el modelo de Harvard
- Set de instrucciones sencillo y con pocas instrucciones, por ejemplo:
 - LOAD / STORE memoria y registros
 - LD, ST, LDI, STI
 - MOVIMIENTO DE DATOS
 - PUSH, POP, SWAP
 - CALCULATE para ejecutar operaciones
 - ADD, SUB, MUL, DIV, MOD (módulo), CMP
 - MANIPULACION DE BITS
 - , AND, OR, XOR, SHL, SHR
 - BRANCH para control de ejecución
 - BEQ (saltar si es igual), BNE (distinto), BGT (mayor que), JMP, NOP
 - ESPECIALES
 - Vectores, criptografía,

Arquitecturas CISC y RISC



- Cada instrucción realiza una micro-tarea (como cada micro-instrucción de los CISC)
- No hay memoria de control → no hay micro-programas
- En general, los registros son de propósito general. A veces hay:
 - PC, Instruction Reg., Stack Pointer, Status Reg., Link Reg., Exception Reg.
- En general, las instrucciones demandan un ciclo de reloj excepto:
 - Acceso a la memoria principal
 - Resolución de dependencias entre instrucciones
- Ciclo de instrucción
 - Inst. Fetch → Inst. Decode → Execution → Mem, Access → Write Back
- Mayor velocidad de ejecución y performance
- Hoy en día, las tecnologías CISC y RISC han convergido

Arquitecturas CISC y RISC



■ Ejemplo de un programa RISC escrito en assembler:

LOAD R1, [num1_addr] → Carga desde la memoria el dato num1 en R1

LOAD R2, [num2_addr] → Carga desde la memoria el dato num2 en R2

ADD R3, R1, R2 → Suma el contenido de R1 y R2 y el resultado lo guarda en R3

STORE R3, [result_addr] → Guarde el resultado R3 en la memoria

■ Un programa más complejo: Factorial de un número

LOAD R1, 5 → Va a calcular el factorial de 5

LOAD R2, 1 → Inicializa el factorial en 1

Bucle:

MUL R2, R2, R1 → Multiplica el resultado por el número

SUB R1, R1, 1 → Decrementa el número

CMP R1, 0 → ¿El número es cero?

BEQ bucle → Si es cero, salta a la etiqueta "done:"

BR termino → Si no es cero, salta a la etiqueta "termino:"

Termino:

STORE R2, [result] → Guarde el resultado R2 en la memoria

HALT → Fin

Arquitecturas CISC y RISC



■ Compiladores

- Cada procesador tiene su set de instrucciones
- Por lo general, no hay compatibilidad binaria de procesadores nuevos con sus predecesores dentro de una misma familia
- Por ejemplo, para procesadores ARM se utiliza comúnmente el GCC (GNU Compiler Collection). Genera código para otras arquitecturas y tiene optimizaciones para distintas versiones de procesadores ARM
- No es necesario tener una versión diferente de compilador para cada chip RISC pero, se debe informar al compilador en que procesador se ejecutará el binario para que aplique las optimizaciones pertinentes

`-march=armv7-a` → Versión de la arquitectura ARM (ARMv7-A)

`-mtune=cortex-a9` → Variante de procesador ARM (Cortex-A9)

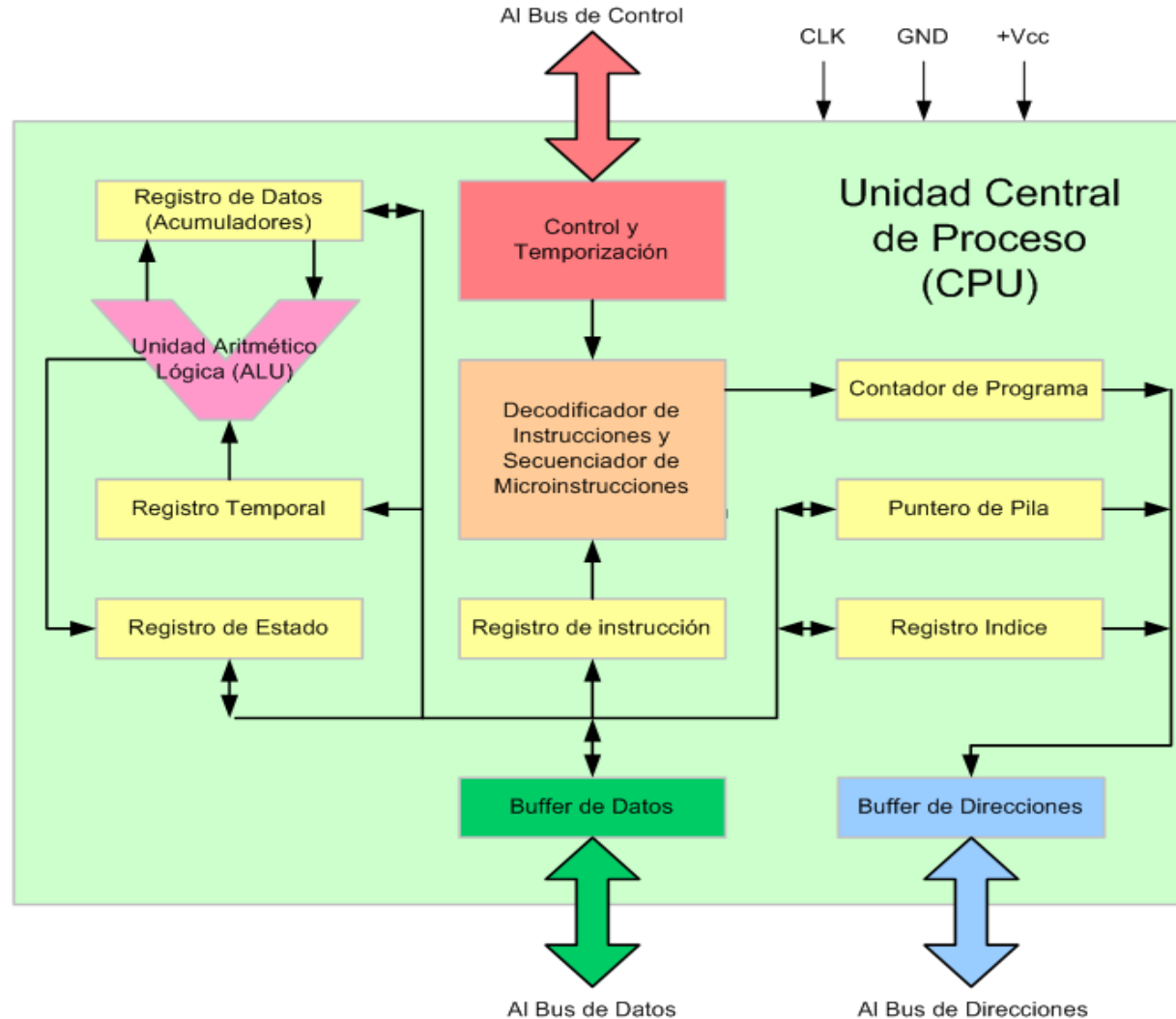
- Arquitecturas RISC avanzadas implementan Pipelines, ejecución en desorden, ejecución especulativa, etc.



Arquitecturas RISC y CISC

Característica	CISC	RISC
Cantidad de instrucciones en lenguaje máquina	Muchas	Pocas
Cantidad de modos de direccionamiento	Muchos	Pocos
Longitud de instrucciones	Varias	Mayormente fija
Cantidad de ciclos de reloj necesarios para ejecutar cada instrucción	Muchas, más de uno	Uno en casi todas
Instrucciones para acceder a la memoria y registros	Muchas	2, Load y Store
Registros de propósito específico	Si	En general no
Control microprogramado	Si	No

Diagrama en Bloques de una CPU



CPU

■ Diagrama PowerPC G5

