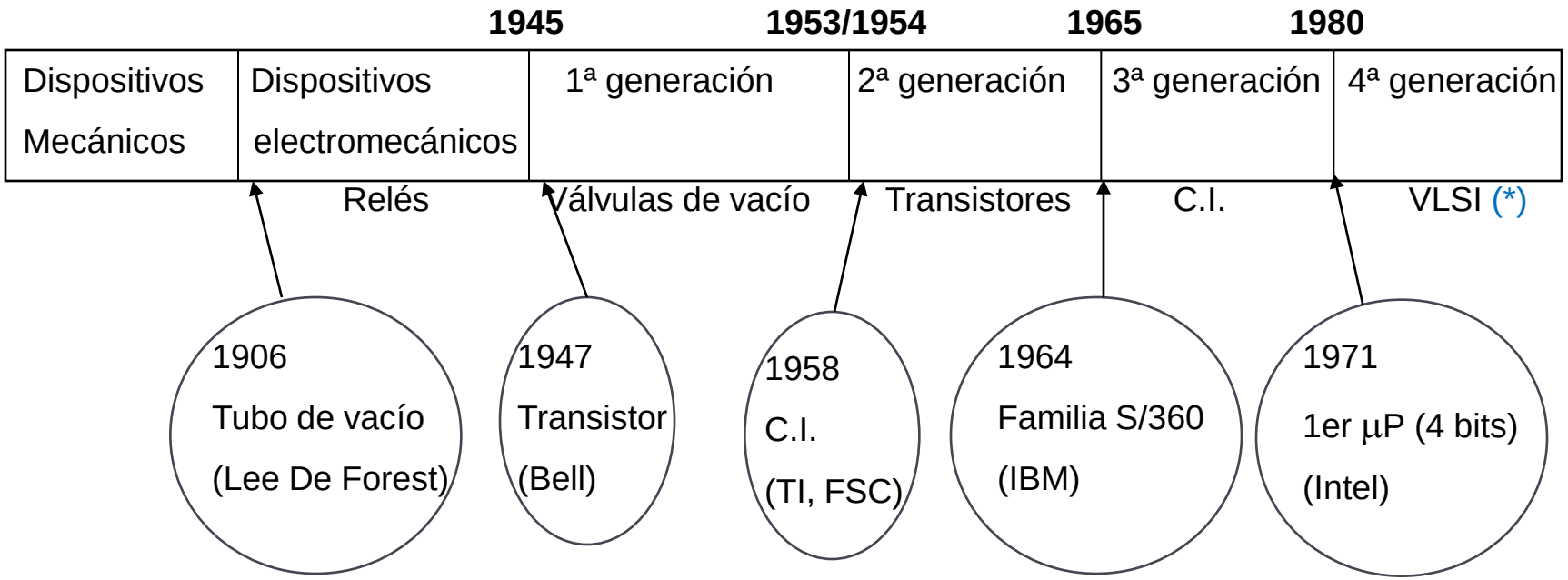
1-ARQUITECTURA DE VON NEUMANN

Tema 1

ÍNDICE

Introducción histórica



Introducción histórica

ESCALAS DE INTEGRACIÓN

- SSI: *Small Scale of Integration*, menos de 12 puertas lógicas
- MSI: *Medium S. I.*, entre 12 y 100 (mux, sumadores,…)
- LSI: *Large S. I.*, entre 100 y 1 000 (operaciones esenciales de una calculadora)
- (*) VLSI: *Very Large S.I.*, de 1 000 a 10 000 (se impone al uso de componentes discretos)

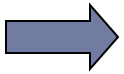
Introducción histórica

- Ley de Moore
  - *La densidad en los chips de silicio se duplica cada 18 meses*
- Ley de Rock
  - *El coste para fabricar semiconductores se duplica cada 4 años*

Von Neumann vs Harvard

- Arquitectura
  - Un ordenador debe interpretar las instrucciones que recibe, ejecutarlas y devolver un resultado.

¿Qué necesita?



Von Neumann vs Harvard

- Arquitectura

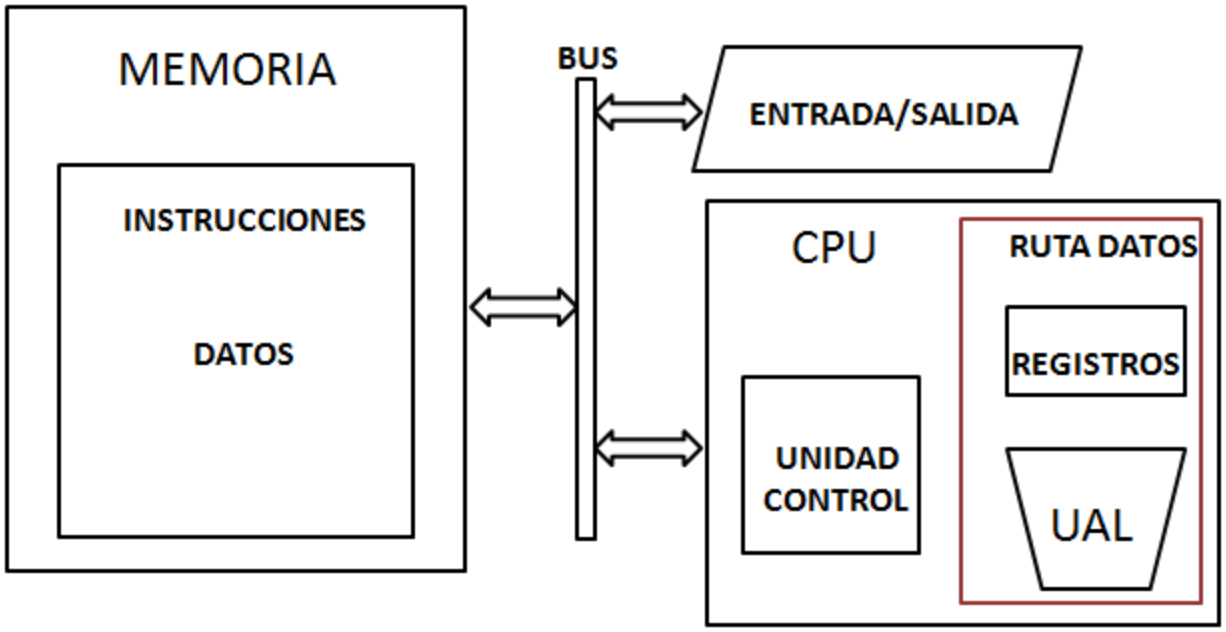
-Poder recoger y devolver datos (e instrucciones) → entrada/salida. Teclado, monitor, altavoz,…

-Almacenar datos e instrucciones → memoria

-Interpretar instrucciones y ejecutar operaciones → unidad de control y unidad aritmético-lógica

Von Neumann vs Harvard

- Von Neumann



Von Neumann vs Harvard

- Von Neumann
  - Memoria unificada para datos e instrucciones
  - Todas las instrucciones por la UAL

  → supone limitaciones
    - Arquitecturas no-von Neumann (paralelismo)
    - Programabilidad

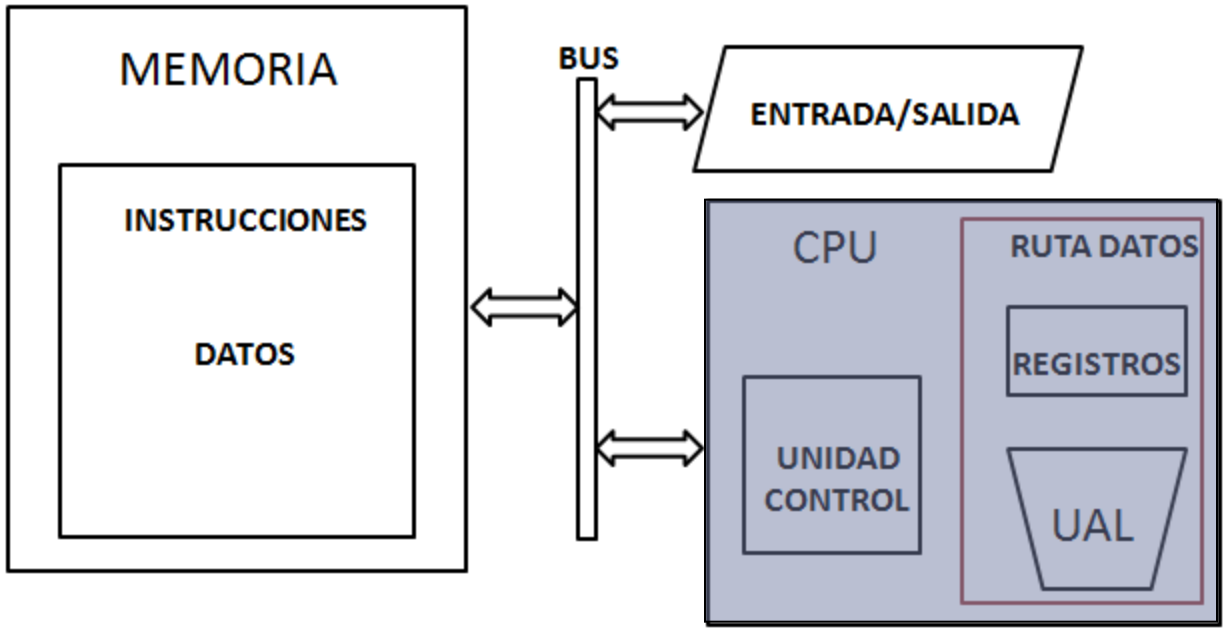
Von Neumann vs Harvard

- Harvard

  La arquitectura Harvard posee una memoria de programa y otra de datos, cada una de ellas con sus correspondientes buses de direcciones, datos y control. A la memoria de programa se le proporcionan las direcciones de las instrucciones y en el bus de datos devuelve las propias instrucciones. A la memoria de datos se le proporcionan las direcciones de los datos y en el bus de datos se transmiten los datos del programa.

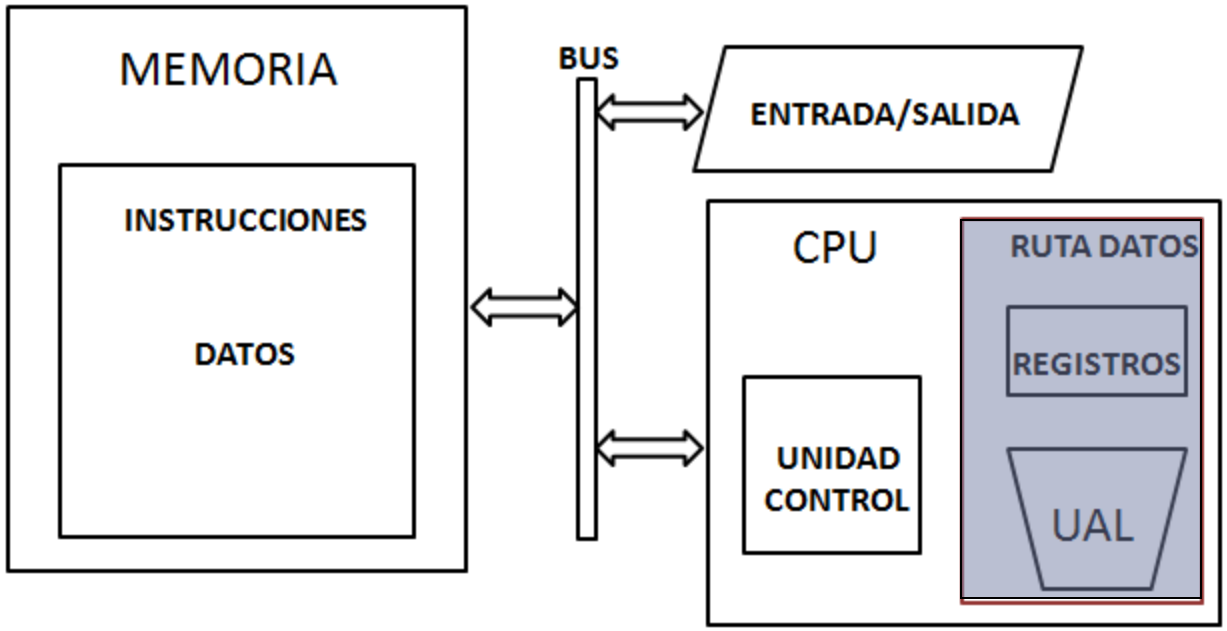
Arquitectura interna

- CPU

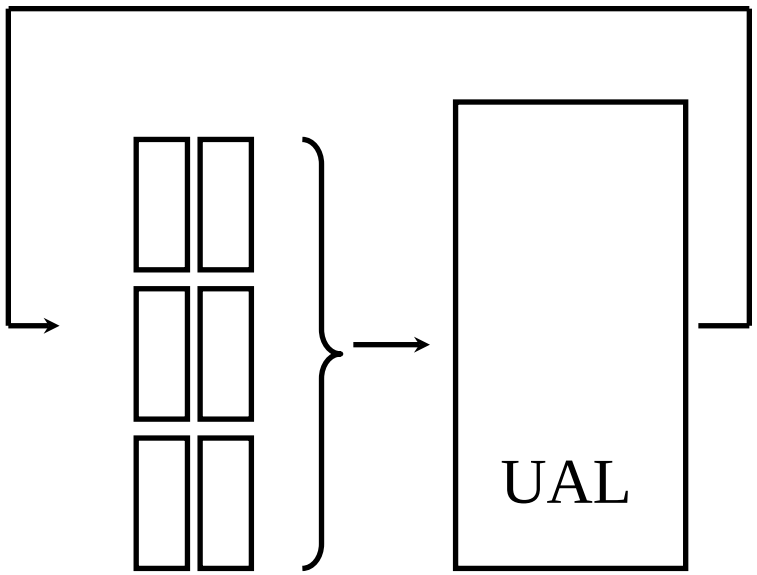


Arquitectura interna

- Ruta de datos: bloque de registros + unidad aritmético-lógica



Ruta de datos

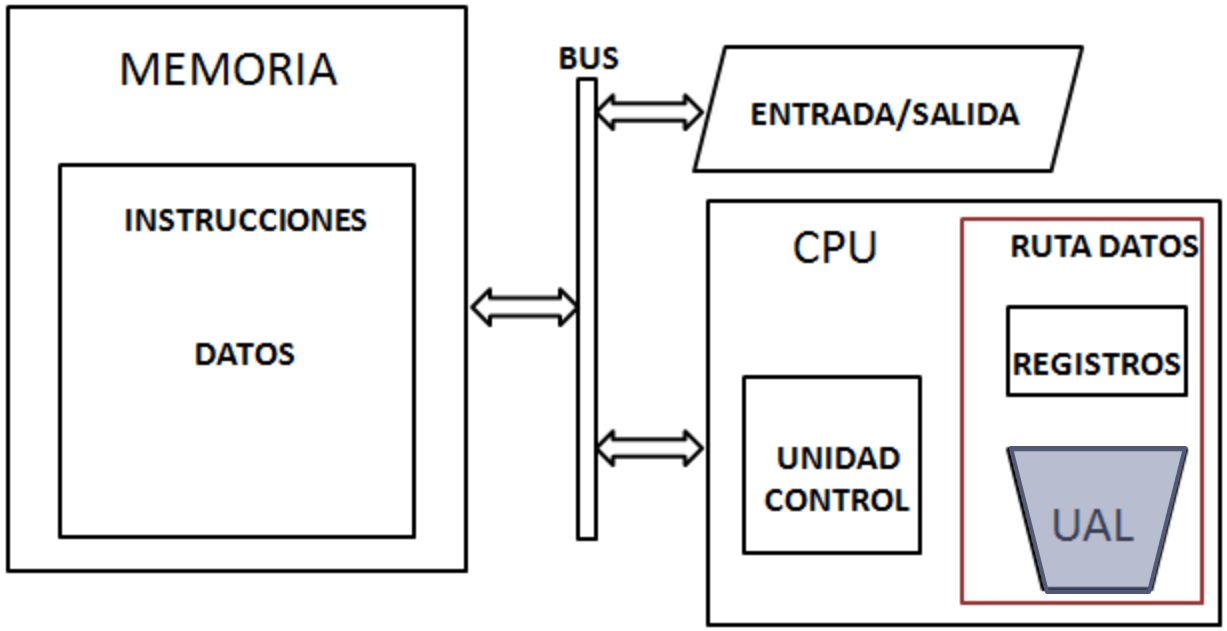


Bloque de registros

- La unidad aritmético-lógica tiene la lógica digital necesaria para ejecutar las instrucciones.
- Una única UAL para todos los registros, de algunos de ellos recoge los operandos y en alguno de ellos guarda el resultado.

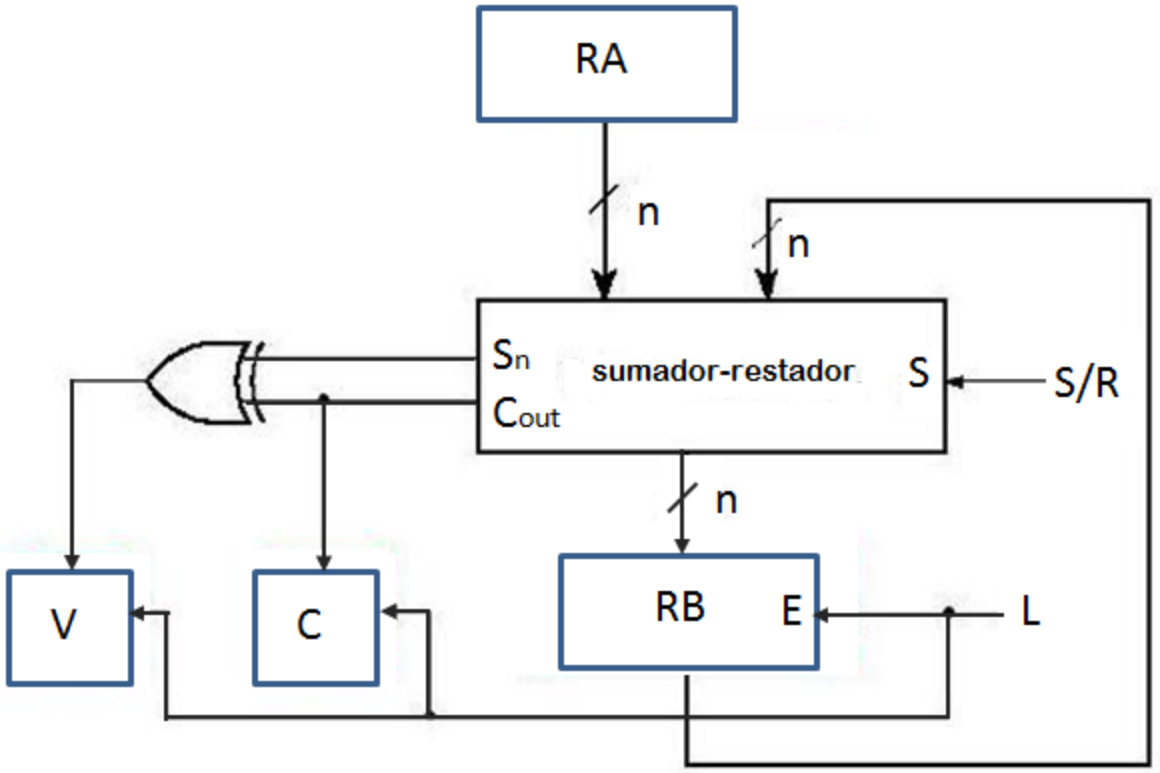
Arquitectura interna

- Unidad Aritmético-Lógica



Unidad aritmética

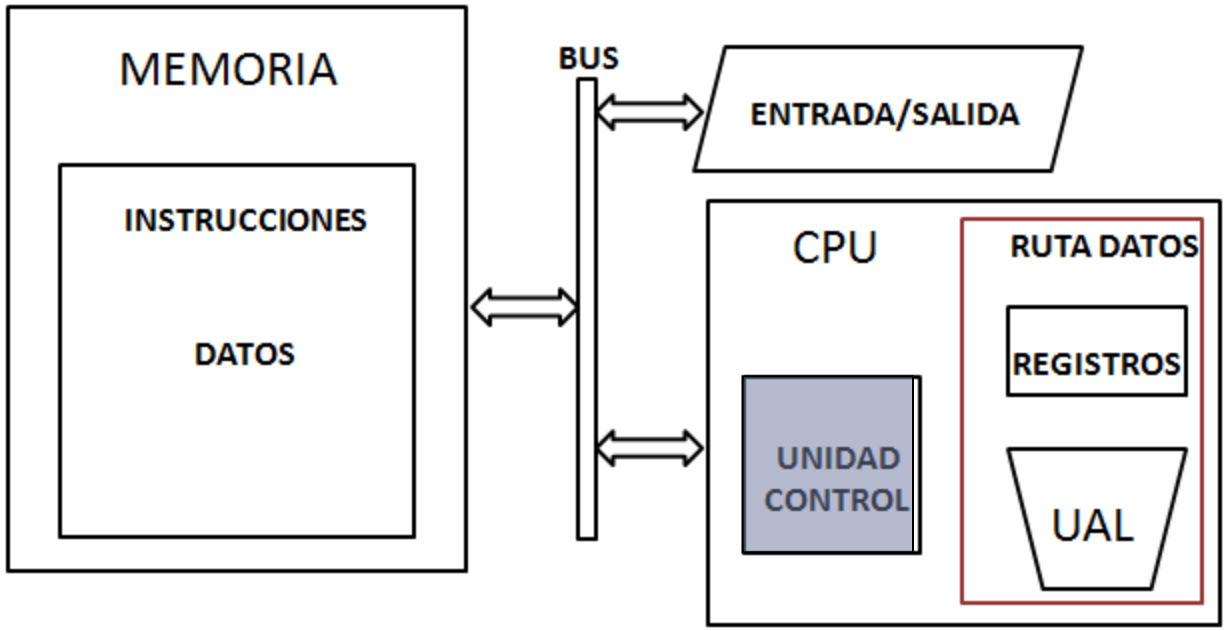
- Sumador/restador de n bits (la señal de control S/R elige la operación)
- Recoge los datos de RA y RB y carga el resultado en RB
- L habilita la carga en RB



- Los FFs V y C dan cuenta del desbordamiento y de la llevada respectivamente (*flags* de estado)

Arquitectura interna

- Unidad de Control



Tema 1

2-UNIDAD ARITMÉTICO LÓGICA

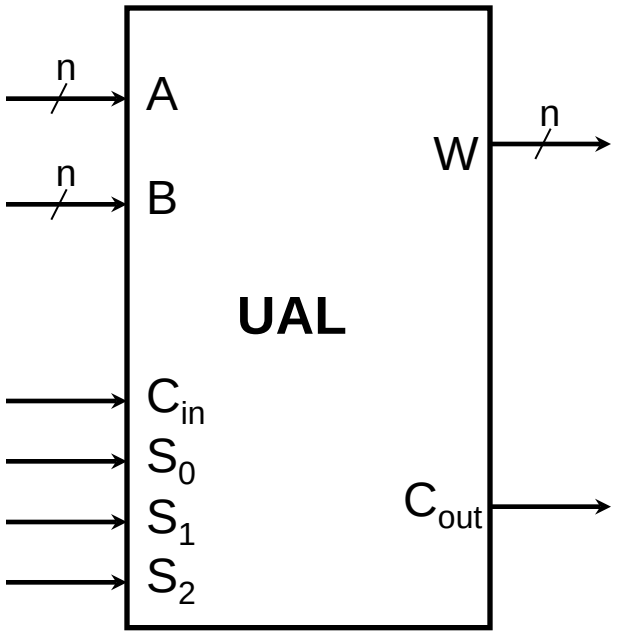
Estructura de Computadores OCW_2015
Nekane Azkona Estefanía

ÍNDICE

- Introducción histórica
- Von Neumann vs Harvard
- Ruta de datos
  - Unidad aritmética
  - Unidad lógica
- Unidad de control
  - *Formato de instrucción*
  - Control cableado
  - Control microprogramado
  - Computadora en canalización

Ruta de datos: Unidad Aritmético Lógica

- Circuito combinacional que realiza operaciones aritméticas y lógicas
- Las señales de control permiten la ejecución de la operación
- Circuito aritmético + circuito lógico=UAL



Símbolo de UAL de n bits

Ruta de datos: Unidad Aritmético Lógica

- Vamos a diseñar una unidad aritmético-lógica sencilla que realice las siguientes operaciones:

Operaciones	
Aritméticas	Decremento, Incremento Suma, Resta
Lógicas	NOT, AND, OR, XOR
Mover	RD←RB
Desplazamiento	Hacia la izquierda (1 bit) Hacia la derecha (1 bit)
Borrado	RD←0
De control de secuencia	Salto/Bifurcación incondicional Salto/Bifurcación condicionada a cero

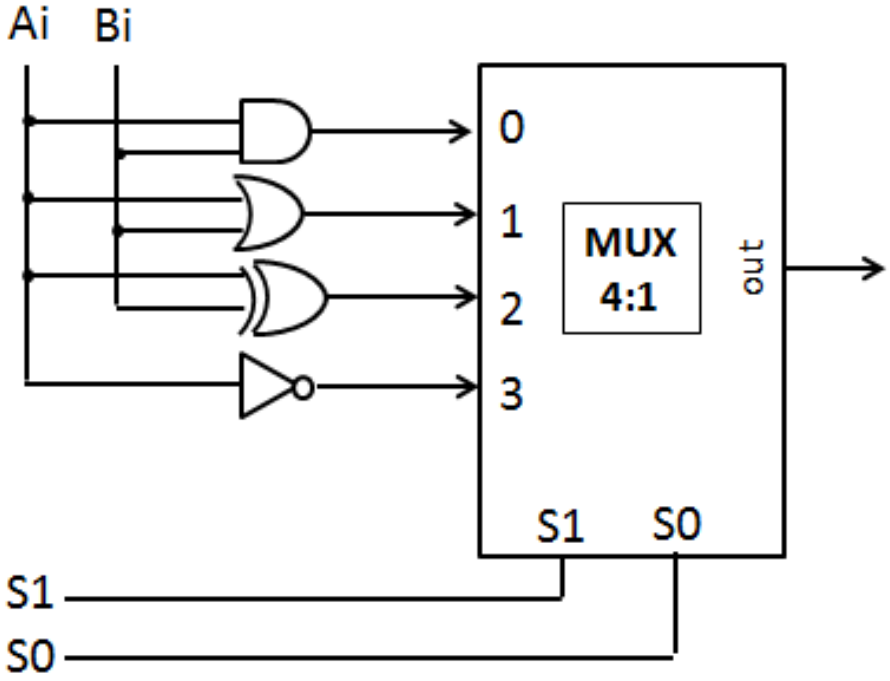
UAL: Circuito Lógico

- Las microoperaciones lógicas son a nivel de bit.
- La tabla muestra cuatro operaciones lógicas: AND, OR, XOR y NOT

S1	S0	Salida	Operación
0	0	A · B	AND
0	1	A + B	OR
1	0	A⊕ B	XOR
1	1	/A	NOT

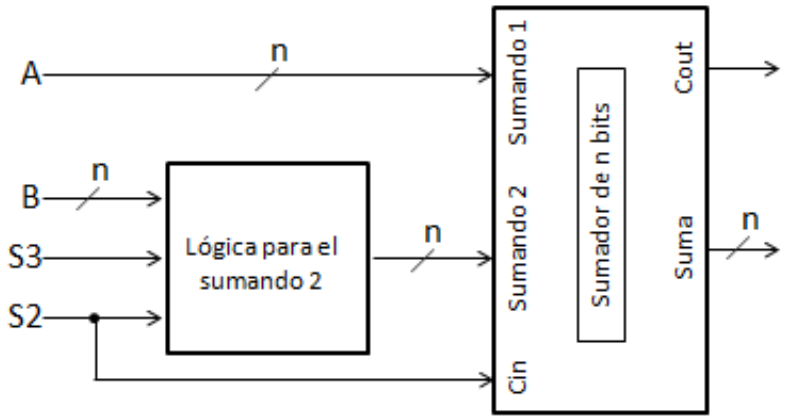
UAL: Circuito Lógico

- Cuatro puertas lógicas conectadas a la entrada de un multiplexor de 4 a uno formarían cada una de las etapas.



- *Si el dato tiene n bits son necesarias n etapas (todas conectadas con las mismas señales de control S1 y S0).*

UAL: Circuito Aritmético



S3	S2	A	B	Cin	Salida
0	0	A	Todo unos	0	A-1
0	1	A	Todo ceros	1	A+1
1	0	A	B	0	A+B
1	1	A	/B	1	A-B

El circuito aritmético está constituido por un sumador completo de n bits y la lógica necesaria para modificar el segundo sumando.

Siendo A (primer sumando) y B (segundo sumando) números binarios de n bits, el primer sumando siempre será A, el segundo debe variar para dar lugar a las distintas operaciones.

UAL: Circuito Aritmético

Una opción para la obtención del segundo sumando es utilizar un multiplexor de 4 a 1. Sus dos entradas de selección pondrán en la salida una de las entradas, y en cada una de las entradas conectaremos unos lógicos (Vcc), ceros lógicos (GND), B o B negada.

Como esta solución ya se ha utilizado para la etapa lógica, veamos otra alternativa…

UAL: Circuito Aritmético

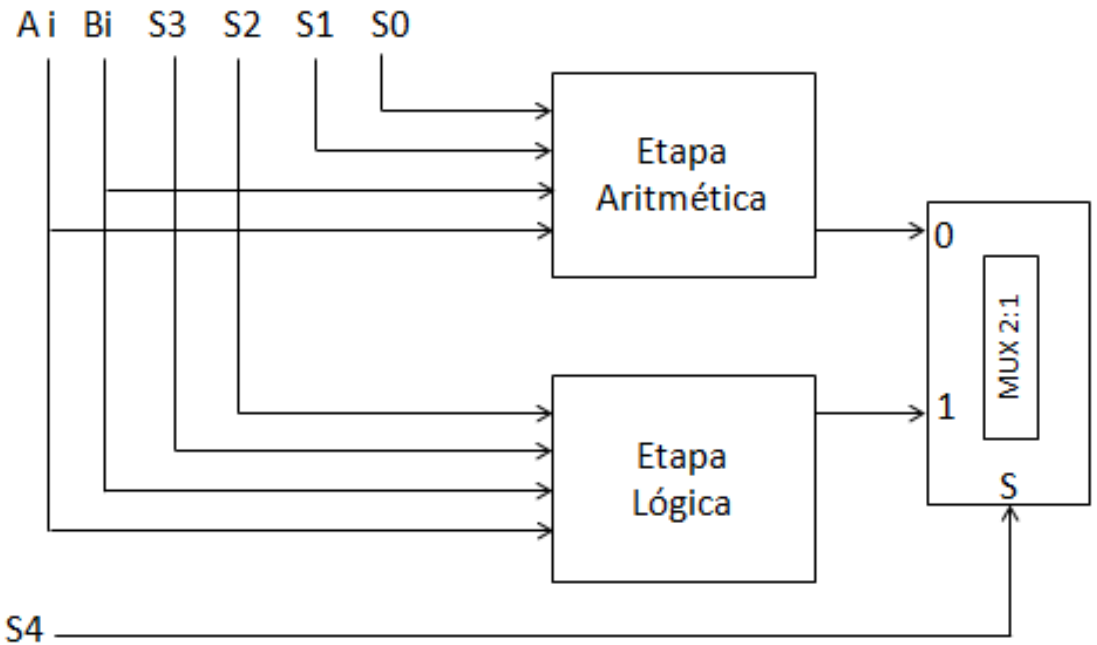
S3	S2	Bi	Segundo sumando
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Simplificando la función para el segundo sumando, nos queda que la lógica dentro de la caja que hemos denominado "Lógica para el segundo sumando" ha de ser:

$$\overline{S3} \cdot \overline{S2} + S3 \cdot (S2 \oplus Bi)$$

Bi representa el i-ésimo elemento del número B.

UAL: circuito completo (una etapa)



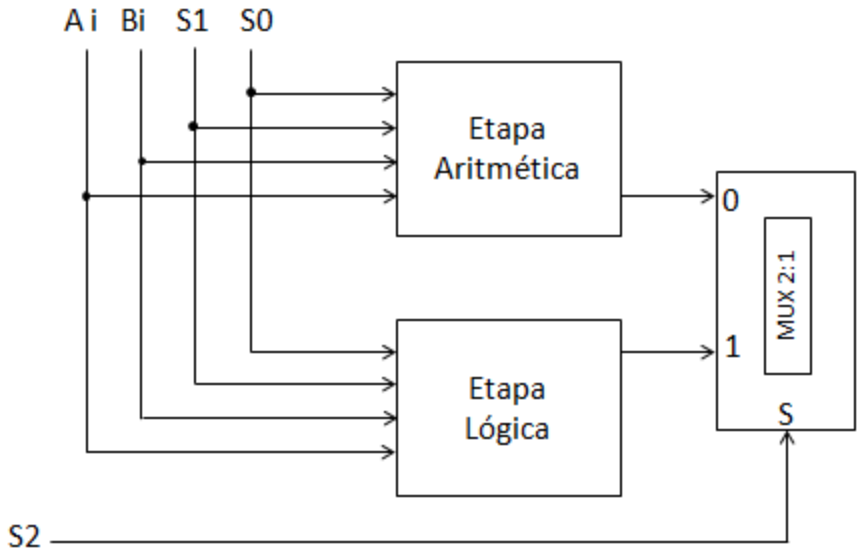
- Así combinando una etapa lógica y una aritmética tendrías una etapa de la UAL.

- Sin embargo vemos que necesitamos 5 variables (señales de selección S[4..0]) para 8 operaciones (4 aritmética y 4 lógicas)

Reordenamos para utilizar solo 3 variables

UAL: circuito completo (una etapa)

S2	S1	S0	Operación
0	0	0	A-1
0	0	1	A+1
0	1	0	A+B
0	1	1	A-B
1	0	0	NOT(A)
1	0	1	(A) AND (B)
1	1	0	(A) OR (B)
1	1	1	(A) XOR (B)



Desplazador

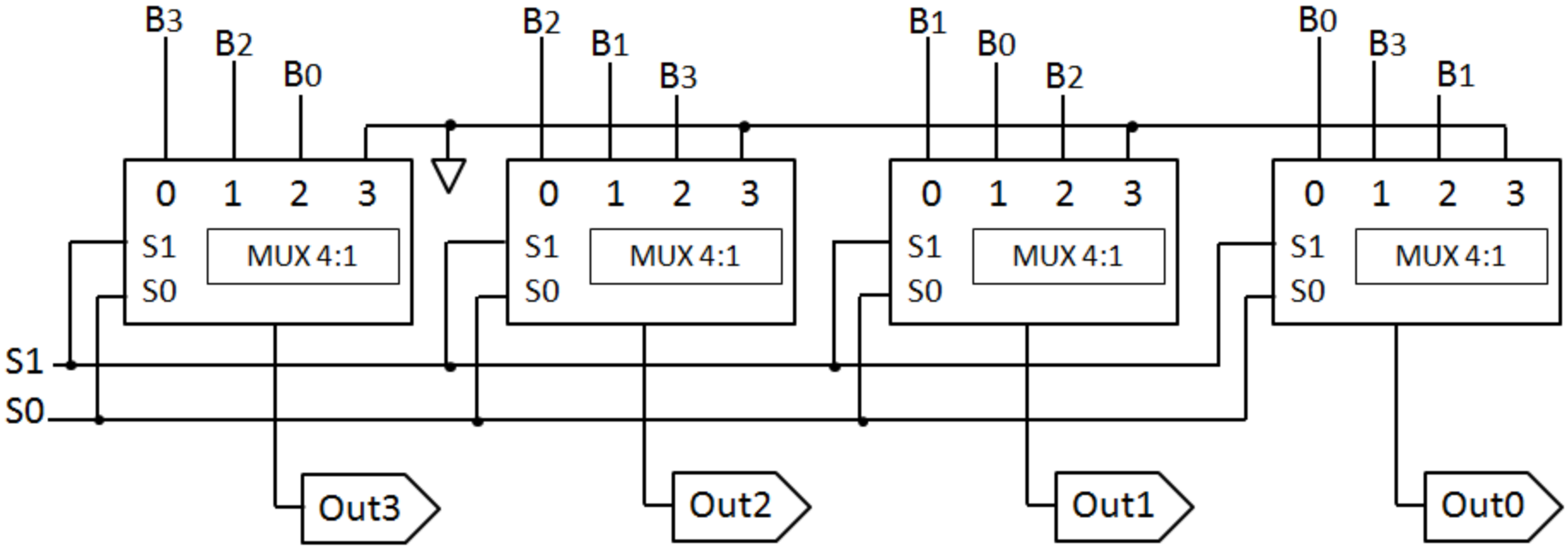


Tabla de las operaciones

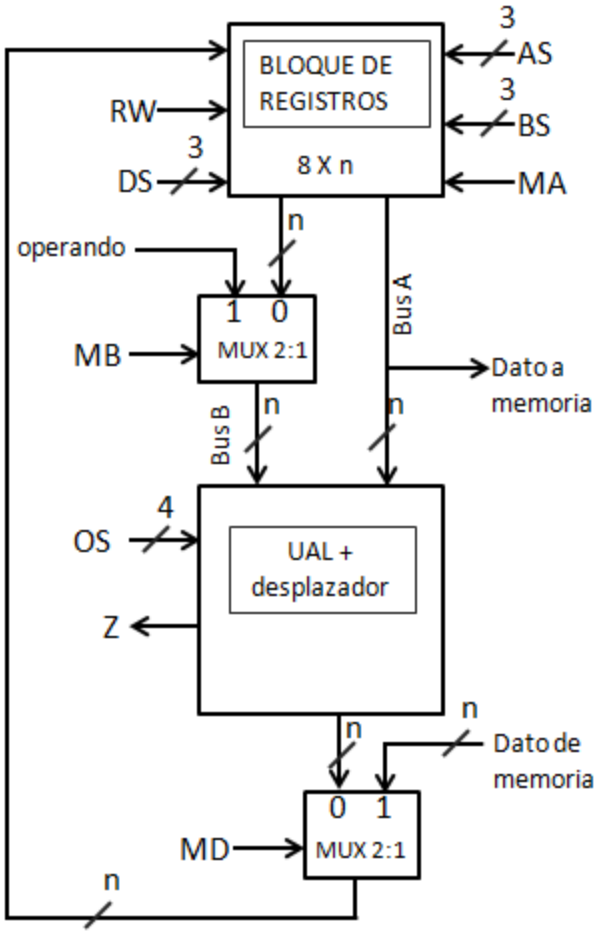
S3	S2	S1	S0	Operación	Descripción
0	0	0	0	DCR	A-1
0	0	0	1	INR	A+1
0	0	1	0	ADD	A+B
0	0	1	1	SUB	A-B
0	1	0	0	NOT	NOT (A)
0	1	0	1	AND	(A) AND (B)
0	1	1	0	OR	(A) OR (B)
0	1	1	1	XOR	(A) XOR (B)
1	x	0	0	MOV	Copiar un registro a otro
1	x	0	1	sl B	Desplazar una posición a la izqd.
1	x	1	0	sr B	Desplazar una posición a la dcha.
1	x	1	1	CLR	Borrar un registro

La tabla recoge las operaciones que tenemos por el momento, a falta de las de control de secuencia. Llamaremos OS[3..0] a las cuatro variables cuya combinación define una operación concreta.

Ruta de datos

•Se necesitan señales de control que seleccionen (direccionen) los registros que contienen los operandos que han de cargarse en los buses A y B de entrada a la UAL.
•En caso de operar con un dato inmediato (éste estará contenido en la instrucción), otra señal deberá elegir este valor (y no el contenido del registro) como entrada a la UAL.
•La UAL generará un resultado que pasará al bus D. En caso de que dicho resultado deba guardarse en un registro, serán necesarias tanto su dirección como un señal que habilite la escritura.
•La UAL activará en caso necesario el indicador, *flag,* de estado Z, que indica si el resultado de la operación ha sido igual a cero. Aunque en nuestro ejemplo no hay más que uno, por lo general existirán varios *flags* para indicar la llevada, el signo, la paridad, etc. Estos indicadores permiten realizar operaciones de control de secuencia condicionadas, necesarias para la implementación de bucles.

Ruta de datos



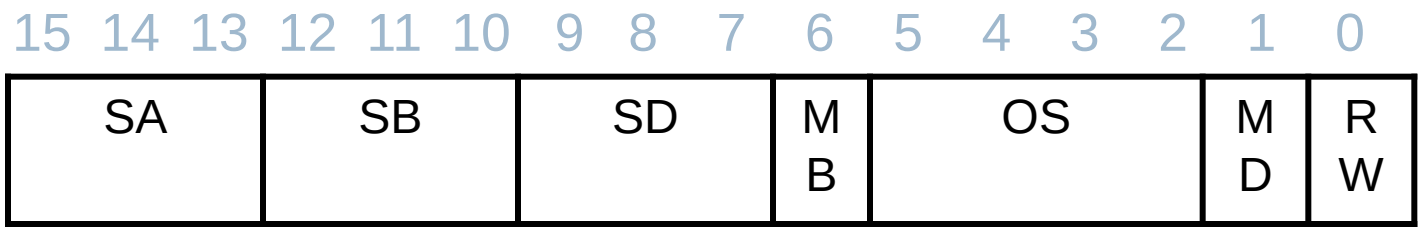
- En esta ruta de datos pueden realizarse 12 operaciones (4 lógicas, 4 aritméticas, una de movimiento entre registros, 2 de desplazamiento de un bit y una última de borrado de registro).

- Cada una de ellas y los operandos que intenvendrán estarán definidos por la combinación de las 17 señales de selección que rigen su funcionamiento.

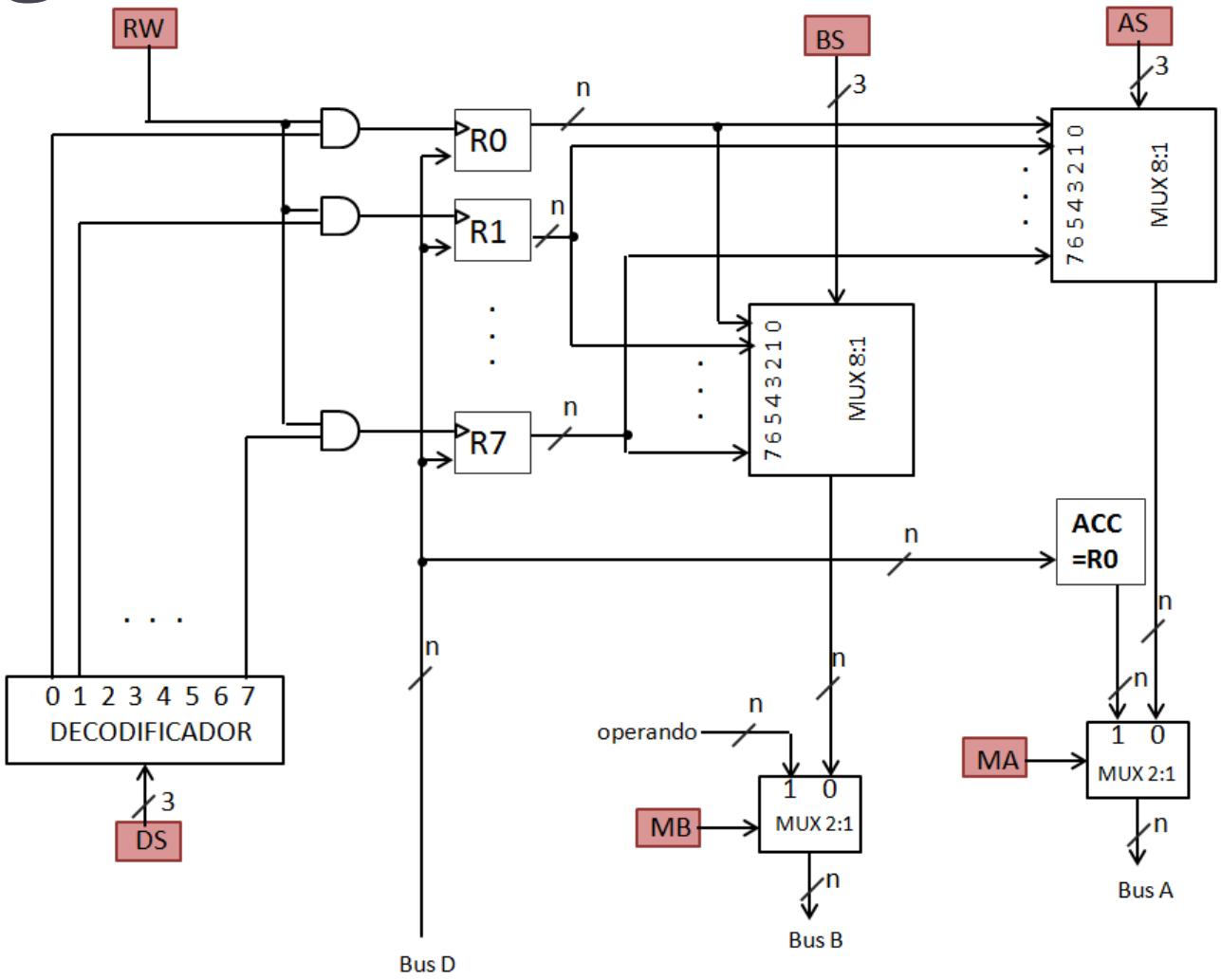
- La combinación de estas señales forman la **palabra de control** que se generará en la unidad de control a partir de la información contenida en la instrucción.

Ruta de datos: palabra de control

- Es la combinación de las señales de selección (veremos que podemos simplificar una , con lo que nos quedaremos con 16)
- Se divide en campos (7)
- Define y dirige las operaciones ejecutadas por la ruta de datos



Ruta de datos: bloque de registros



Instrucciones de control de secuencia

LPC	JB	CZ	Z	Carga en PC	
0	X	X	X	PC←PC+1	Ejecución secuencial (siguiente instrucción)
1	0	0	X	PC←nueva dir.	Salto sin condición (JMP)
1	0	1	0	PC←PC+1	Salto condicionado a Z (no cumple condición) (JZ, Z=0)
1	0	1	1	PC←nueva dir.	Salto condicionado a Z (cumple condición) (JZ, Z=1)
1	1	0	X	PC←PC+offset	Bifurcación incondicional (BR)
1	1	1	0	PC←PC+1	Bifurc. condicionada a Z (no cumple condición) (BZ, Z=0)
1	1	1	1	PC←PC+offset	Bifurc. condicionada a Z (cumple condición) (BZ, Z=1)

En nuestro diseño contemplamos la posibilidad de realizar tanto saltos como bifurcaciones (saltos a un entorno del PC); en ambos casos pueden ser incondicionales o depender de si el resultado de la operación fue o no igual a cero.

→ *Queda como ejercicio el diseño del circuito que cumpla lo descrito en la tabla.*

Ruta de datos: codificación de la palabra de control

AS,BS, DS		MB		OS					MD		RW	
R0	000	R	0	0	0	0	0	DCR	Result.	0	No escr.	0
R1	001	Cte	1	0	0	0	1	INR	Dato	1	Escribir	1
R2	010			0	0	1	0	ADD				
R3	011			0	0	1	1	SUB				
R4	100			0	1	0	0	NOT				
R5	101			0	1	0	1	AND				
R6	110			0	1	1	0	OR				
R7	111			0	1	1	1	XOR				
				1	x	0	0	MOV				
				1	x	0	1	sl B				
				1	x	1	0	sr B				
				1	x	1	1	CLR				

→ *Una tabla con todas las instrucciones al final del tema (glosario).*

R.D.: ejecución en canalización

La ejecución de una operación requiere el paso por cada uno de los elementos que componen la ruta de datos:

•Los operandos han de leerse del bloque de registros a los buses A y B.

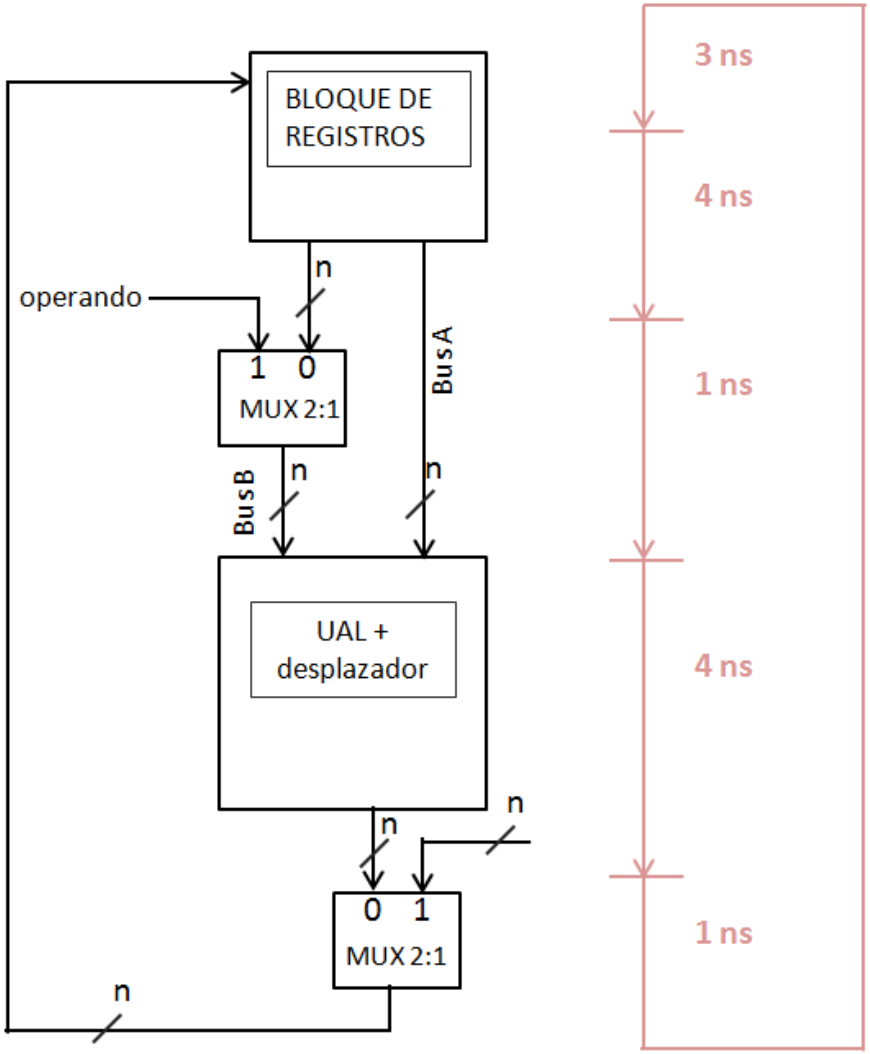
•Se debe seleccionar entre el contenido de un registro o un operando inmediato.

•Ha de realizarse la operación (aritmética, lógica o desplazamiento).

•Se debe seleccionar entre el resultado de la operación o un dato externo.

•El valor en el bus D debe reescribirse en un registro del bloque de registros.

R.D.: ejecución en canalización



La ejecución de una operación requiere 13 ns:

→t_{CLK}=13ns

→f_{CLK}=1/13ns=76.9MHz

R.D.: ejecución en canalización

La ejecución en canalización (pipeline) de una operación consiste en la división en etapas que podrán ejecutarse simultáneamente.

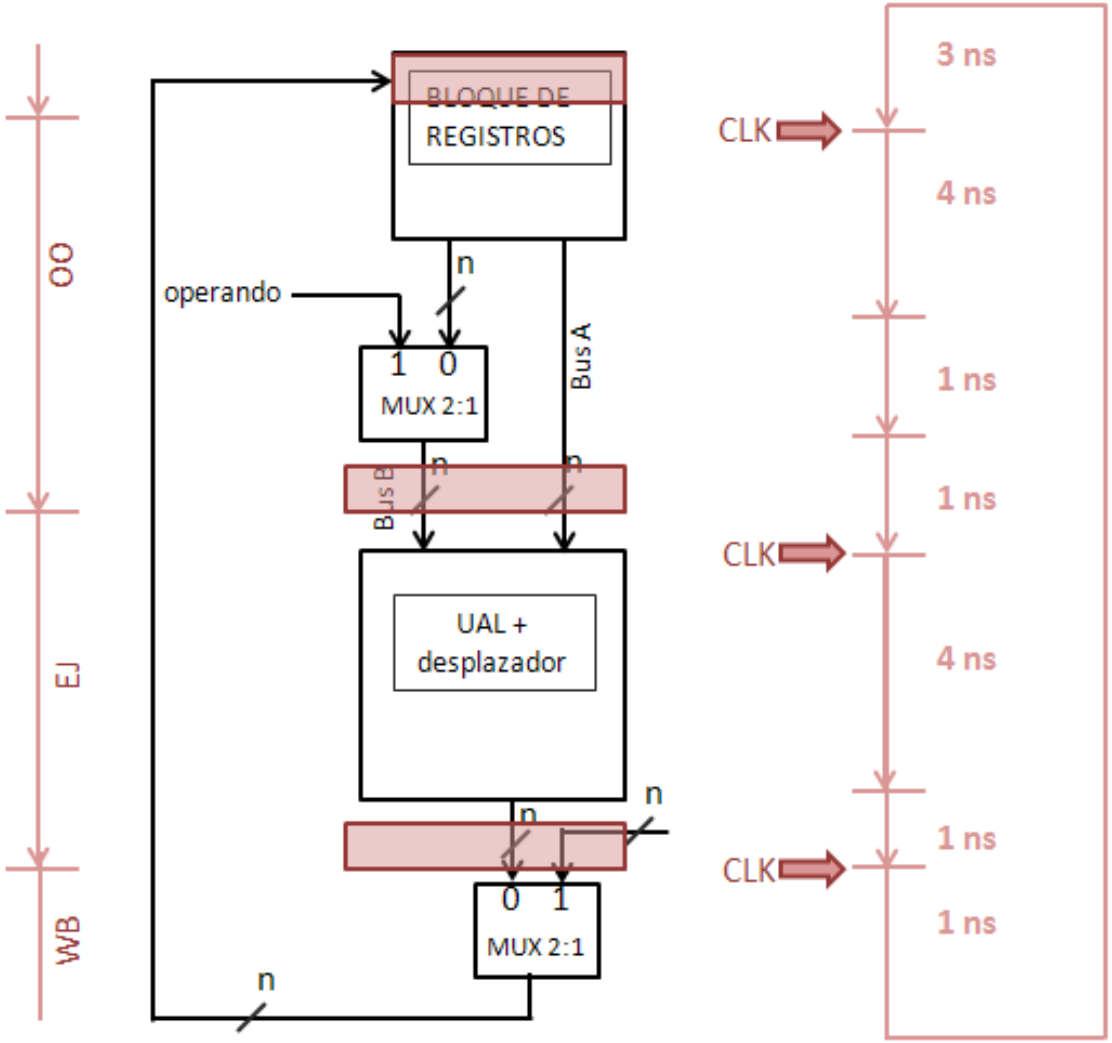
• Requiere añadir registros donde almacenar los valores hasta que la siguiente etapa los utilice.

• El nuevo reloj ha de tener un periodo tal que la etapa más lenta tenga tiempo de realizar su tarea → buscar etapas con tiempos comparables.

• Estos registros añaden un retardo, por lo que la ejecución de una sola instrucción tardará más. La ventaja se ve al ejecutar varias instrucciones.

• Nunca se obtendrá la máxima ventaja debido a:
    • el retardo añadido por los registros
    • el llenado y vaciado de la canalización

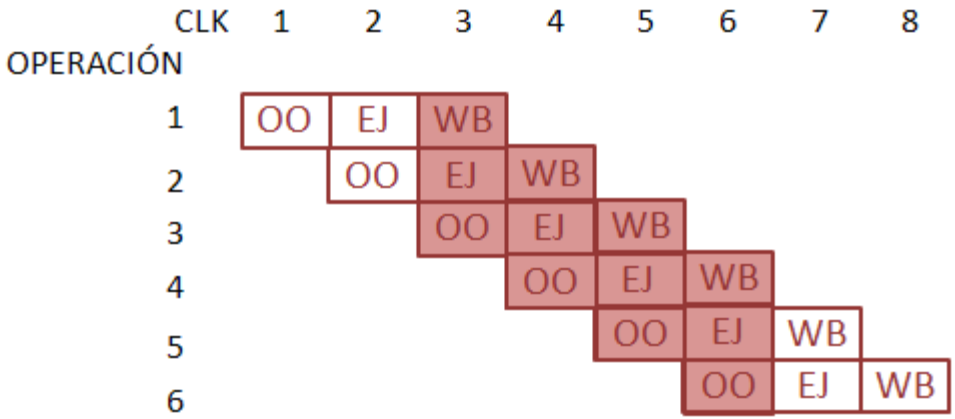
R.D.: ejecución en canalización



- El nuevo reloj tiene un periodo
  → Tmín=máx{4+1+1, 4+1, 1+3}=6ns.

- Ahora la ejecución de una operación requiere tres etapas: obtención de los operandos (OO), ejecución (EJ) y escritura del resultado (WB)
  → 3x6ns=18ns

R.D.: ejecución en canalización



• La ventaja de la ejecución en canalización se aprecia con la ejecución de una secuencia de operaciones. En el caso de 6 operaciones, requiere 8 ciclos de reloj x 6 ns = 48 ns → a 8 ns de media la instrucción, bastante más rápido que los 13 ns iniciales.
• Cuando la canalización está llena (coloreado) se ejecuta una instrucción cada 6 ns.
• El tiempo de llenado y vaciado de la canalización hace que la mejora sea menor a la óptima (en el ejemplo 8 ns en lugar de 6 ns). Cuanto mayor sea la secuencia de instrucciones, menos se notará este efecto.

Tema 1

3-UNIDAD DE CONTROL

Estructura de Computadores OCW_2015
Nekane Azkona Estefanía

ÍNDICE

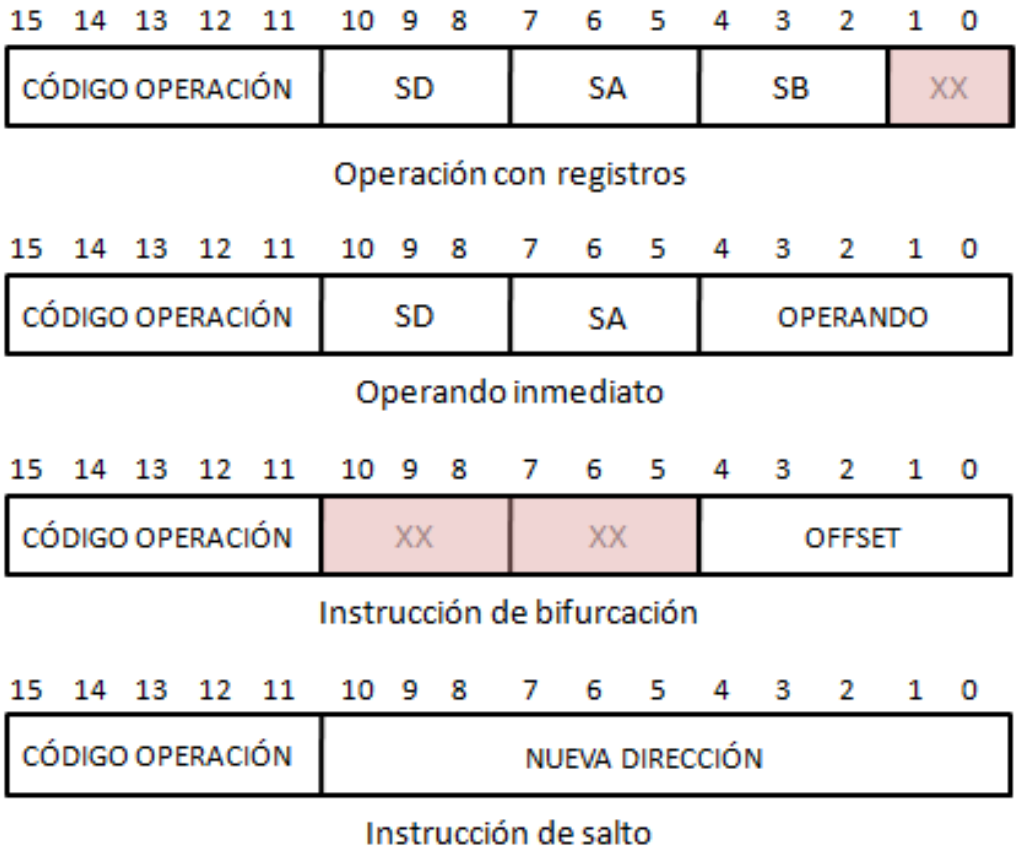
- Introducción histórica
- Von Neumann vs Harvard
- Ruta de datos
  - Unidad aritmética
  - Unidad lógica
- Unidad de control
  - *Formato de instrucción*
  - Control cableado
  - Control microprogramado
  - Computadora en canalización

Formato de instrucciones

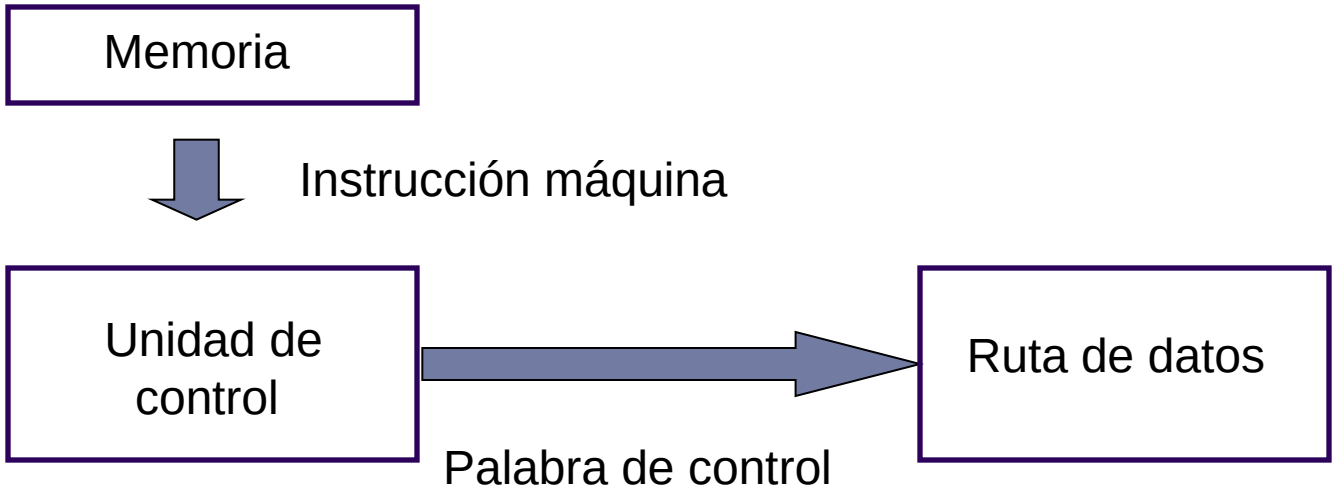
- Instrucción → conjunto de bits dividido en campos.
- *Campos* → cada uno se asocia a un elemento y define unas funciones.
- *Código de Operación (C.Op.)* → conjunto de m bits, puede representar hasta 2^m operaciones diferentes.
- Campos de *dirección* → dirección de los operandos
- La U.C. deberá generar a partir del conjunto de bits de una instrucción la secuencia de palabras de control necesarias para su ejecución.

Formatos de instrucciones

Formatos de instrucción de la computadora de nuestro ejemplo:



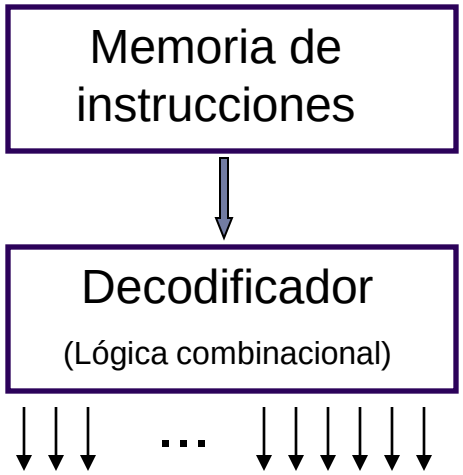
Unidad de control



La unidad de control es un “traductor”. Dos métodos:

1. Control cableado
2. Control microprogramado (control almacenado)

Unidad de control: control cableado



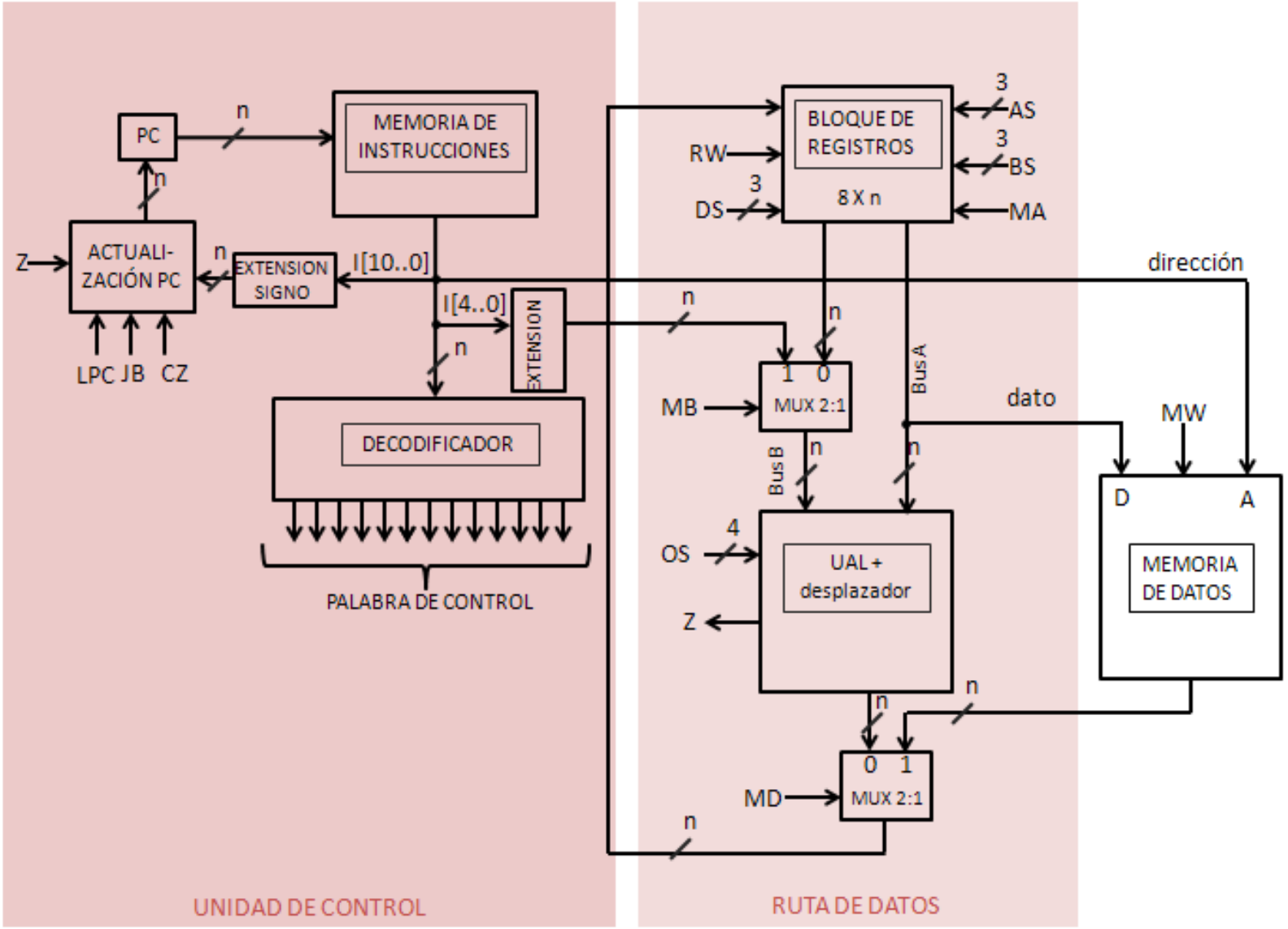
Características:

- ✘ Se diseña siguiendo los métodos clásicos de diseño lógico → laborioso y costoso (*)
- ✘ Difícil de modificar (¿rediseño?)

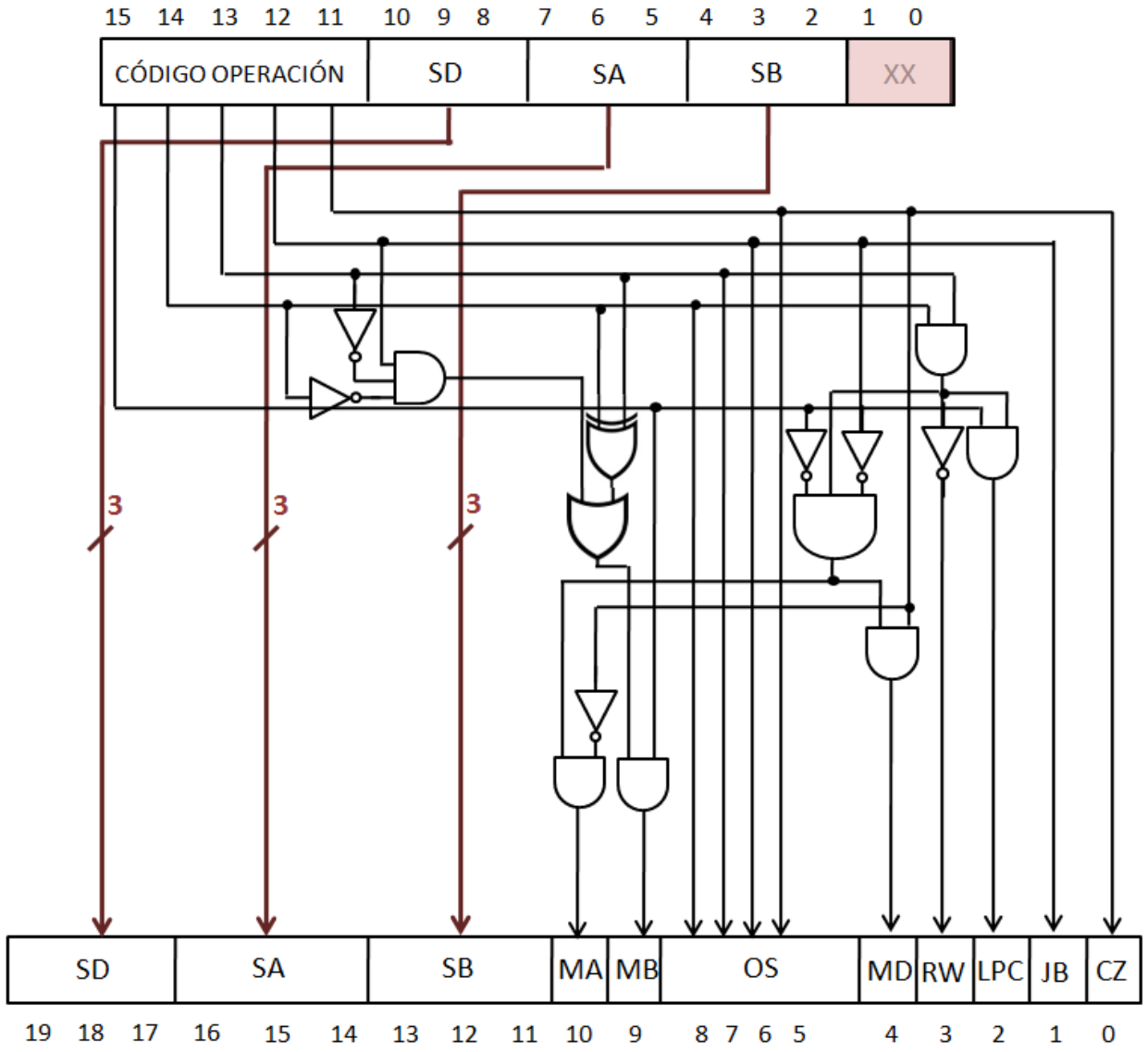
- ☑ En igualdad de condiciones es más rápida que la lógica almacenada

(*) Hoy en día los programas de diseño asistido por ordenador (CAD) para VLSI resuelven de manera automática la mayoría de las dificultades de diseño

Computador con control cableado y ciclo sencillo



Unidad de control: control cableado



Decodificador cableado

Unidad de control: control microprogramado

En el caso del control microprogramado, la palabra de control correspondiente a una instrucción está almacenada en una memoria de control. El código de operación se utiliza para direccionar la memoria que lo contiene.

En caso de implementar una unidad de control de ciclo múltiple, la instrucción puede requerir más de un ciclo de reloj para ejecutarse. Requiere registros adicionales no accesibles al usuario (registros del sistema), un registro para la instrucción; además permite unificar las memorias de datos e instrucciones → estos cambios hacen necesarias nuevas señales de control, dando lugar a una palabra de control mayor.

Unidad de control: control microprogramado de ciclo múltiple

- Una instrucción → un microprograma = conjunto de microinstrucciones ordenado
- Microcódigo en memoria (*firmware*)

<u>Características</u>:

1. Simplicidad conceptual (*)
2. Fácil de modificar (*)
3. Puede incluir instrucciones complejas
4. Pueden desarrollarse computadores que ejecuten diferentes juegos de instrucciones (también emuladores)

(*) El desarrollo de microcódigo es trabajoso, se puede decir que el ahorro en diseño hardware se paga por lado del firmware.

Unidad de control: control microprogramado

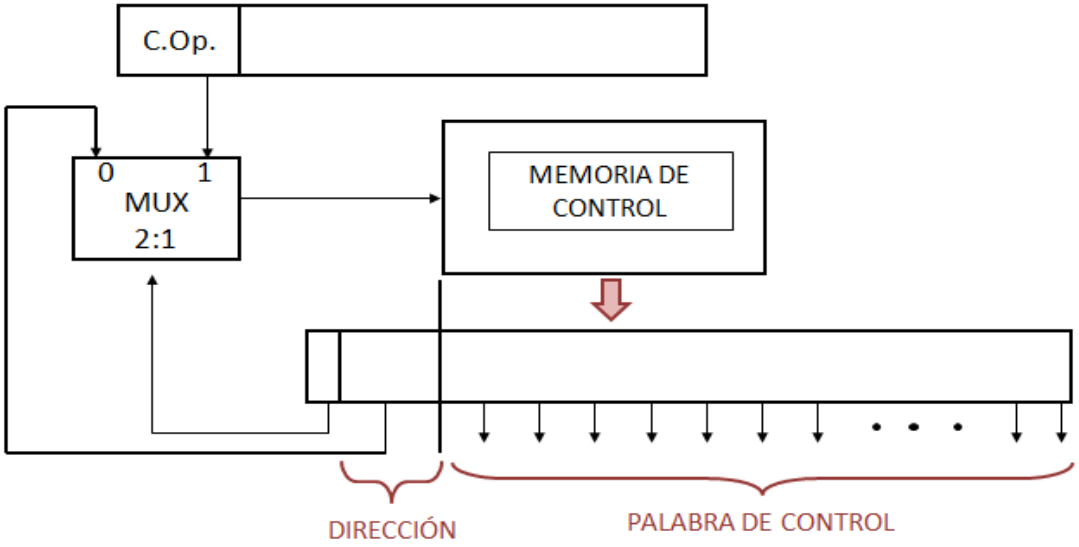
Condiciones básicas:

1. Memoria de control con capacidad para almacenar todos los microprogramas correspondientes a todas las instrucciones
2. Procedimiento para asociar a cada instrucción máquina el microprograma que le corresponde (Código de Operación→ dirección de la memoria de control)
3. Mecanismo para leer las instrucciones consecutivas de un microprograma (y salto a un nuevo microprograma al terminar)
4. Mecanismo de microbifurcación condicionada que permita ejecutar instrucciones máquina de bifurcación condicionada

Pueden resolverse mediante secuenciamiento implícito o explícito

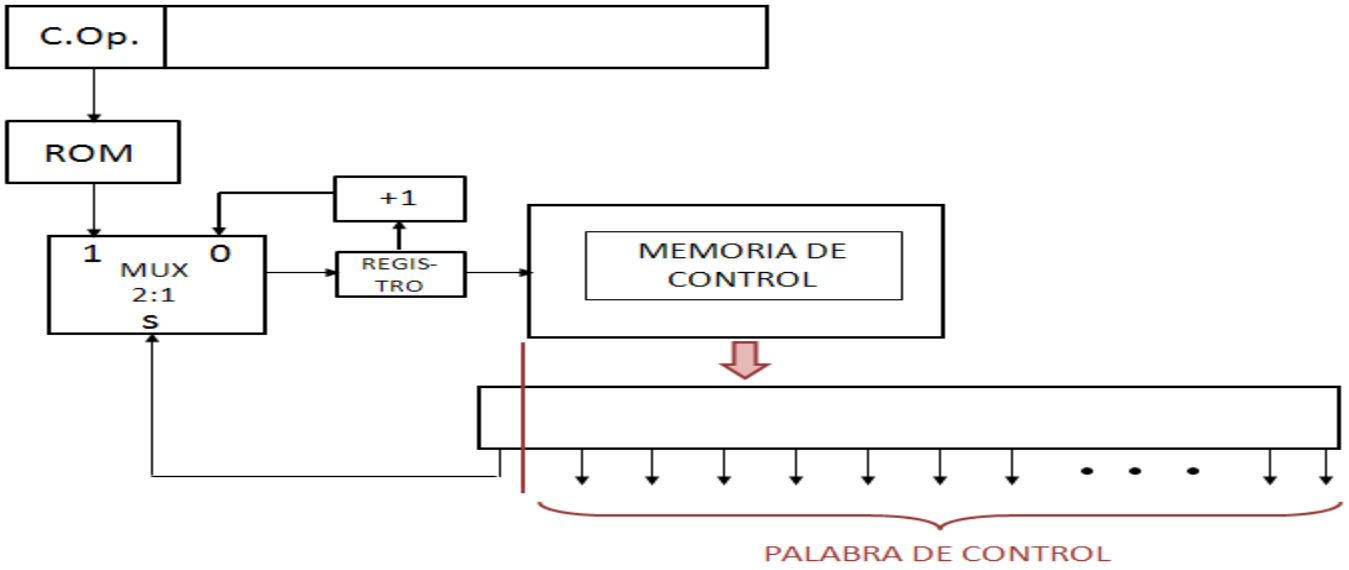
Unidad de control: control microprogramado

- Secuenciamiento explícito:
  - Cada microinstrucción tiene la dirección de la siguiente (no requiere mecanismo de secuenciación)
  - Las microinstrucciones pueden estar desordenadas
  - Requiere gran parte de la memoria de control

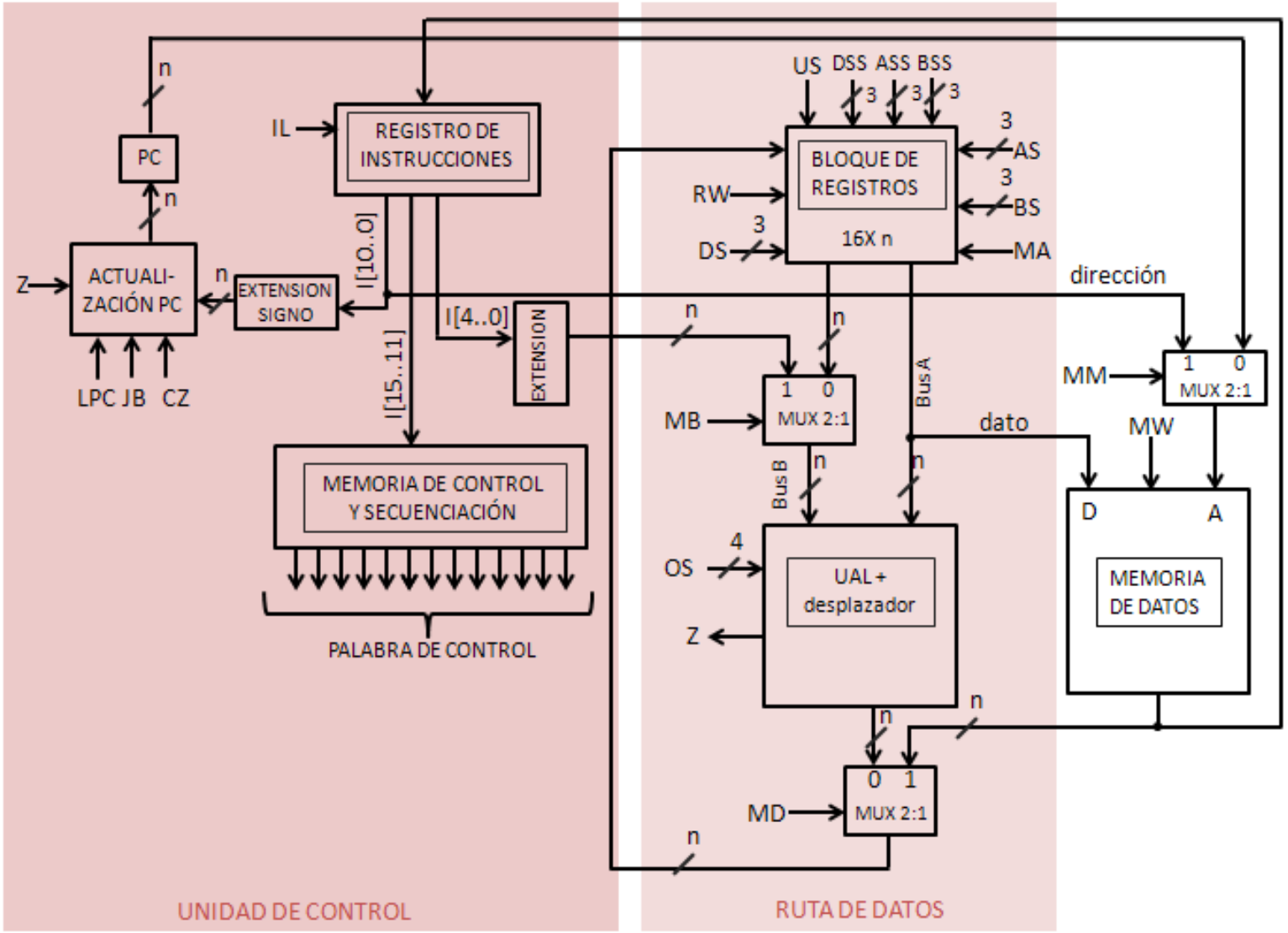


Unidad de control: control microprogramado

- Secuenciamiento implícito :
  - Las microinstrucciones de un microprograma han de estar almacenadas en orden en memoria
  - (2) añadir un pequeño decodificador entre el código de operación y el multiplexor de las direcciones
  - (3) un registro de microdirecciones y un incrementador



Computador de ciclo múltiple y control microprogramado



Unidad de control: control microprogramado

La nueva palabra de control ha de añadir las señales IL (para cargar el registro de instrucción con una nueva instrucción), US (para indicar si los siguientes tres bits direccionan un registro de usuario o de sistema), DSS, ASS, BSS (señales de 3 bits que direccionan los 8 nuevos registros no accesibles al usuario), MM (para seleccionar entre una dirección de un dato o el contenido del contador de programa, es decir una dirección de una instrucción).

Unidad de control: control microprogramado

Por último no pueden faltar las señales necesarias para la secuenciación, que serán diferentes según el secuenciamiento sea implícito o explícito. En relación con esto hay que tener en cuenta que el tamaño de la memoria de control será mayor que el caso de ciclo múltiple:

→en el número de posiciones, debido a que cada instrucción requerirá una secuencia de palabras de control (antes el tamaño era 2^5, siendo 5 el número de bits del código de operación; ahora ha de ser mayor).

→ en la anchura de palabra, ya que el hecho de ser de ciclo múltiple requiere unas modificaciones que hacen necesarias más señales de control, dando lugar a una palabra de control de más bits.

Computadora en canalización

Ciclo de ejecución básico de un computador:

1. Leer la instrucciones de memoria señalada por el PC → a un registro de la U.C.
2. Decodificar instrucción
3. Buscar los operandos (obtener la dirección efectiva)
4. Recuperar los operandos
5. Ejecutar la operación
6. Almacenar resultado
7. Volver al paso 1 a por la siguiente instrucción

Computadora en canalización

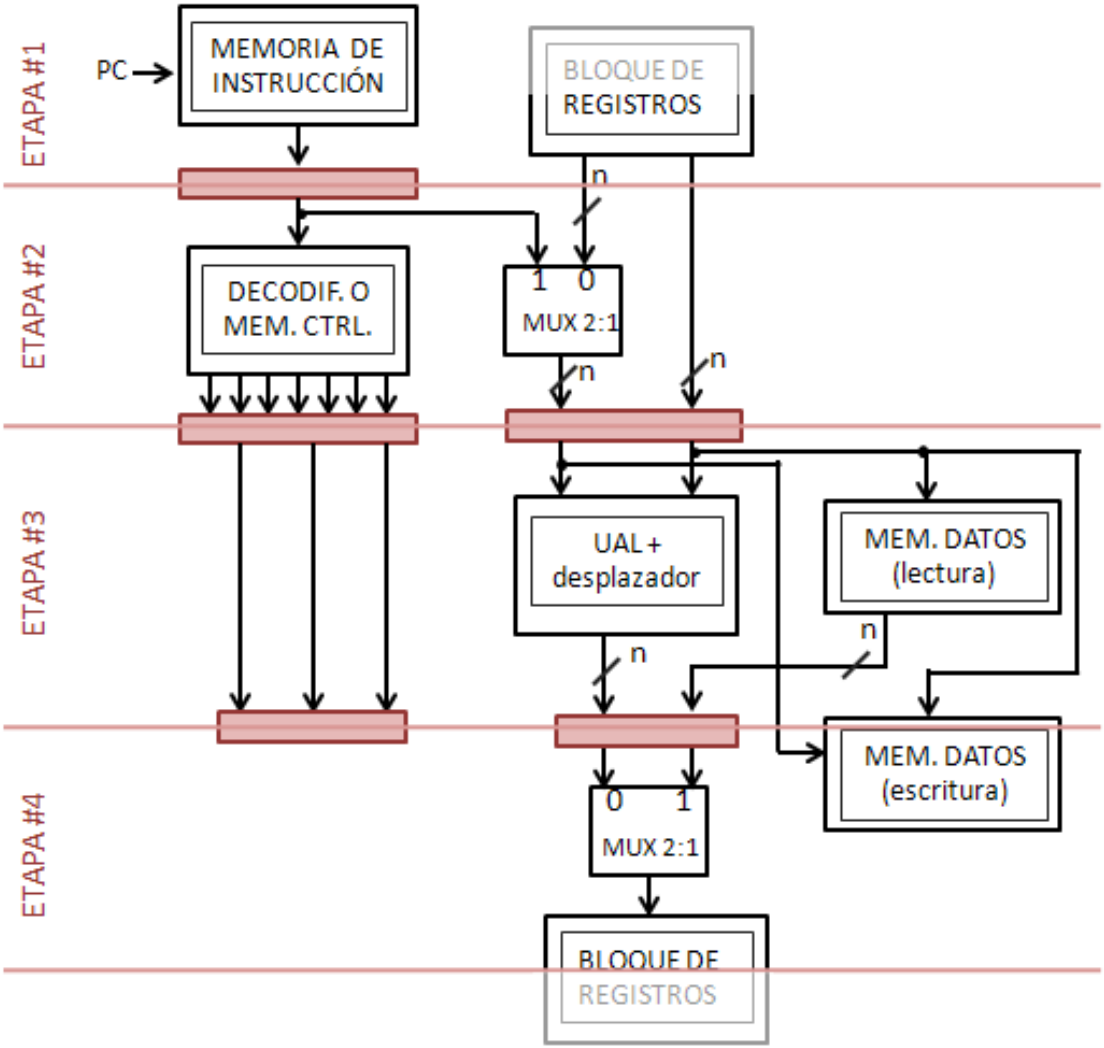
Ejecución SIN canalización:

- La ejecución de una instrucción requiere un periodo de reloj mayor o igual al tiempo de retardo de todos los elementos que intervienen en la ejecución de la misma
- Solo una de las partes funciona en cada momento

Ejecución en canalización:

- La ejecución de una instrucción se divide en etapas (han de añadirse registros para mantener los valores entre etapas)
- Se sustituye el reloj por uno cuyo periodo sea mayor o igual al tiempo de retardo de la etapa de mayor duración)
- Una vez llena la canalización, todas las etapas funcionan simultáneamente

Computadora en canalización



Computadora en canalización

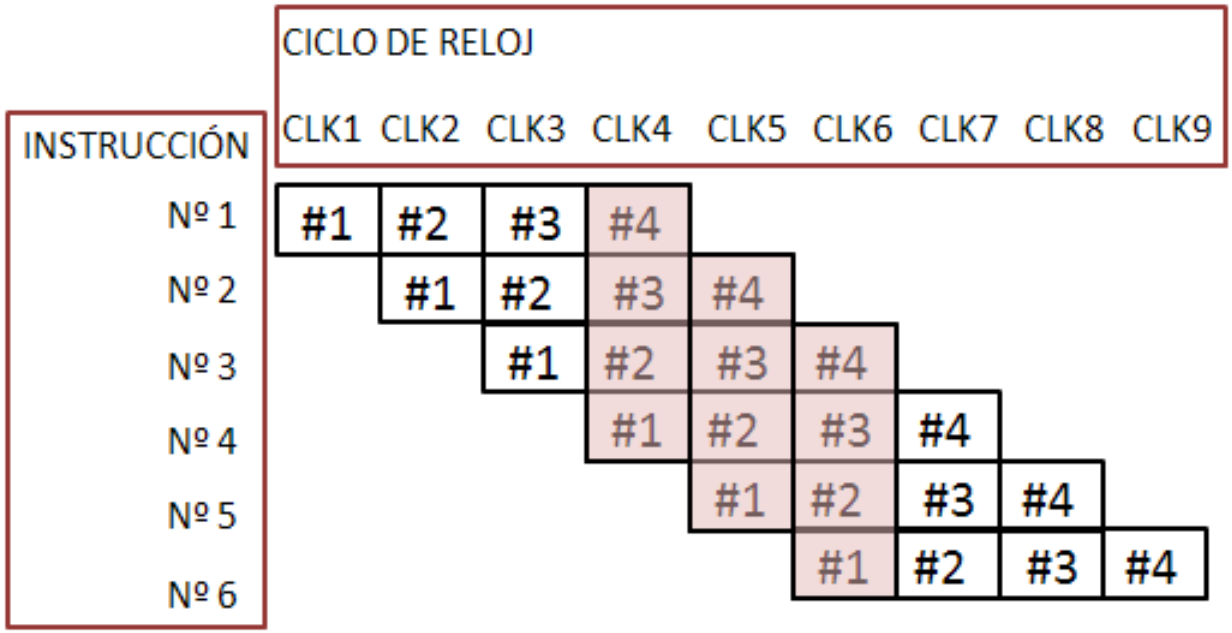
Etapa	Intervienen	Función
1	Contador de programa (PC) Memoria de instrucción	Obtener la instrucción de la memoria de instrucción y actualizar el PC
		Plataforma de canalización que juega el papel del registro de instrucción
2	Decodificador Bloque de registros	Decodificar IR→ señales de control: **SA, SB, MA** y **MB** para obtención de datos El resto:
		En este registro se guardan señales de control que usaremos más tarde
3	Unidad de ejecución Lectura/Escritura en memoria de datos	Operación de UAL, desplazamiento o de memoria → **OS** y **MW** se usan aquí La lectura de memoria de datos se realiza en esta etapa → dato direccionado→ Salida de datos
		Resultados de la etapa 3, señales de control de etapa 4 Escritura en memoria de datos es parte de esta plataforma
4	Escritura en bloque de registros	**SD**, **MD** y **RW** se usan en la etapa de escritura

Computadora en canalización: ejecución en canalización

- Las plataformas de canalización equilibran los retardos:
  - Si cada etapa ≤ 5ns ➔ Reloj: f_{max}= 20Mhz
  - Antes una instrucción = 17ns
  - Ahora una instrucción 4x5 = 20ns

Un única instrucción tarda más, pero la ventaja está en que pueden ejecutarse varias en paralelo.

Computadora en canalización: ejecución en canalización

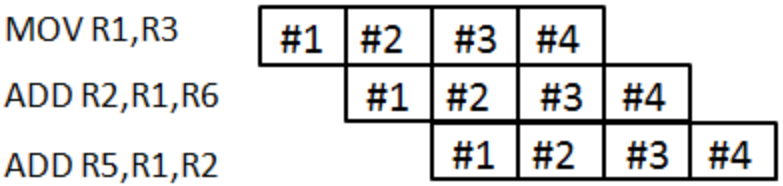


9 ciclos x 5 ns=45 ns → a una media de 7,5 ns la instrucción.Mejor que los 17ns iniciales

- La ejecución de una sola instrucción tarda más
  - Cuando la canalización está llena 'se ejecuta 1 instrucción' cada ciclo de reloj
  - La mejora no es óptima debido al tiempo de llenado y vaciado

Computadora en canalización: ejecución en canalización

Si una instrucción requiere un operando resultado de una instrucción anterior, puede suceder que debido a la ejecución en canalización que este dato aún no se haya reescrito → riesgo de datos

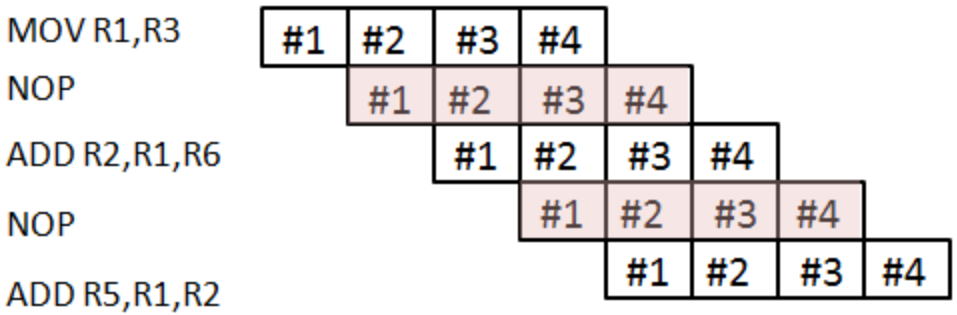


	R1	R2	R3	R4	R5	R6
Inicialmente	1	2	3	4	5	6
Resultado esperado	3	9	3	4	12	6
Resultado obtenido	3	7	3	4	5	3

Supongamos que inicialmente Ri=i, tras la primera instrucción R1 debería ser 3; tras la segunda R2=9 y finalmente $R5=12_{10}$

Sin embargo, cuando la segunda instrucción recupera sus operando (etapa 2) la primera aún no los ha reescrito (etapa 4) por lo que R1 vale 1; cuando la tercera instrucción recupera sus operandos R1 ya vale 3, pero R2 sigue valiendo 2 → tras la ejecución del programa R5=5 → el resultado es imprevisible.

Computadora en canalización: solución sw al riesgo de datos



- Retrasar las instrucciones para dar tiempo a que se realice la reescritura antes que la siguiente lectura. En el ejemplo con instrucciones NOP (no operación).
- Alarga el programa lo que supone un retardo (evitable, aunque complicado, si en lugar de instrucciones NOP se reordena el código para aprovechar y ejecutar otra instrucción).
- Implica que el programador conozca los detalles de la ejecución en la canalización.

Computadora en canalización: solución hw 1 al riesgo de datos

- Se añade la lógica necesaria para detectar instrucciones que requieren un operando que está en un registro que hay que escribir en el siguiente período (instrucción anterior).
- Si es necesario se detiene durante un periodo de reloj la canalización evitando el riesgo de datos. En las siguientes etapas se dice que se ha introducido una burbuja en la canalización.
- Libera al programador (no es necesario añadir instrucciones NOP al programa)…
- …pero en cuanto al retardo, la penalización es la misma.

Computadora en canalización: solución hw 2 al riesgo de datos

- Cuando se da un riesgo de datos, el dato que buscamos no está todavía en el registro, pero ya se ha calculado (está en la entrada del multiplexor D).
- Añadimos un multiplexor extra que haciendo uso de la lógica implementada en la solución hardware 1 lleve el dato directamente al bus A o al B.
- Libera al programador .
- No necesitamos más períodos de reloj.

Glosario

Instrucciones contempladas por la computadora diseñada.

	DESCRIPCIÓN	CÓDIGO DE OPERACIÓN	ENSAMBLADOR
1	(RD)←(RA)-1	00000	DCR RD, RA
2	(RD)←(RA)+1	00001	INR RD, RA
3	(RD)←(RA)+(RB)	00010	ADD RD, RA, RB
4	(RD)←(RA)-(RB)	00011	SUB RD, RA, RB
5	(RD)←/(RA)	00100	MVN RD, RA
6	(RD)←(RA) AND (RB)	00101	AND RD, RA, RB
7	(RD)←(RA) OR (RB)	00110	OR RD, RA, RB
8	(RD)←(RA) XOR (RB)	00111	XOR RD, RA, RB
9	(RD)←(RB)	01000	MOV RD, RB
10	(RD)←shift left (RB)	01001	SL RD, RB
11	(RD)←shift right (RB)	01010	SR RD, RB
12	(RD)← CERO	01011	CLR RD
13	[M]←(AC)	01100	STA
14	(AC)←[M]	01101	LDA
- - -		01110	- - -
- - -		01111	- - -
- - -		10000	- - -
- - -		10001	- - -
15	(RD)←(RA)+OP	10010	ADI RD, RA, OP
16	(RD)←(RA)-OP	10011	SUI RD, RA, OP
- - -		10100	- - -
17	(RD)←(RA) AND OP	10101	ANI RD, RA, OP
18	(RD)←(RA) OR OP	10110	ORI RD, RA, OP
19	(RD)←(RA) XOR OP	10111	XRI RD, RA, OP
20	(RD)←OP	11000	MVI RD, OP
21	(RD)←shift left OP	11001	SLI RD, OP
22	(RD)←shift right OP	11010	SRI RD, OP
- - -		11011	- - -
23	PC← ADDRESS	11100	JMP
24	PC←ADDRESS (Z=1) PC←PC+1 (Z=0)	11101	JZ
25	PC←PC+offset	11110	BR
26	PC←PC+offset (Z=1) PC←PC+1 (Z=0)	11111	BRZ

Glosario

RD: Ruta de Datos
UC: Unidad de Control

SD: Selección D
SA: Selección A
SB: Selección B
MA: Multiplexor A
MB: Multiplexor B
OS: Operation Selection (*Selección de la Operación*)
MD: Multiplexor D
RW: Register Write (*Escritura en Registro*)
LPC: Load Program Counter (*Carga Contador de Programa*)
JB: Jump/Branch (*Salto/Bifurcación*)
CZ: Conditioned to Z (*Condicionado a Z*)