

TEMA 4 “Unidad de Control”

0. Visión Global

La Unidad de Control es un bloque de la Arquitectura de Von Neumann.

Funciones de la Unidad de Control:

- Asegurar la ejecución de los programas (en M. principal).
- Generación de las secuencias de microórdenes para la ejecución de todas y cada una de las instrucciones de la computadora.
- Captar y decodificar cada instrucción.
- Ejecución de la siguiente Instrucción del programa.

Técnicas de diseño de la Unidad de Control:

- **Cableada:**
 - Mediante Registros de Desplazamiento.
 - Como Sistema Secuencial Síncrono.
 - Mediante Decodificadores de Tiempo e Instrucción.
- **Microprogramada:**
 - Mediante ROM de control:
 - Microprogramación horizontal.
 - Microprogramación vertical.

1. Diseño de la Unidad de Control Microprogramación

ROM de Control: Cada palabra está compuesta por todas las señales de microoperación del sistema.

- En el caso de la Computadora Mejorada, hay 18 microoperaciones => la anchura de la ROM de Control será de 18 bits.

Las instrucciones estarán compuestas por varios pasos (ciclos), tanto de búsqueda como de ejecución.

- Cada paso será una palabra de la Memoria de Control.

Tras la ejecución de un paso es necesario saber qué paso será el siguiente que se tendrá que ejecutar:

- Añadir una ROM con el comportamiento del flujo de microoperaciones: Control de la dirección de bifurcación.

1.1. Codificación de las microinstrucciones

Las **microinstrucciones** proporcionan las señales de control a los distintos elementos que existen dentro del sistema.

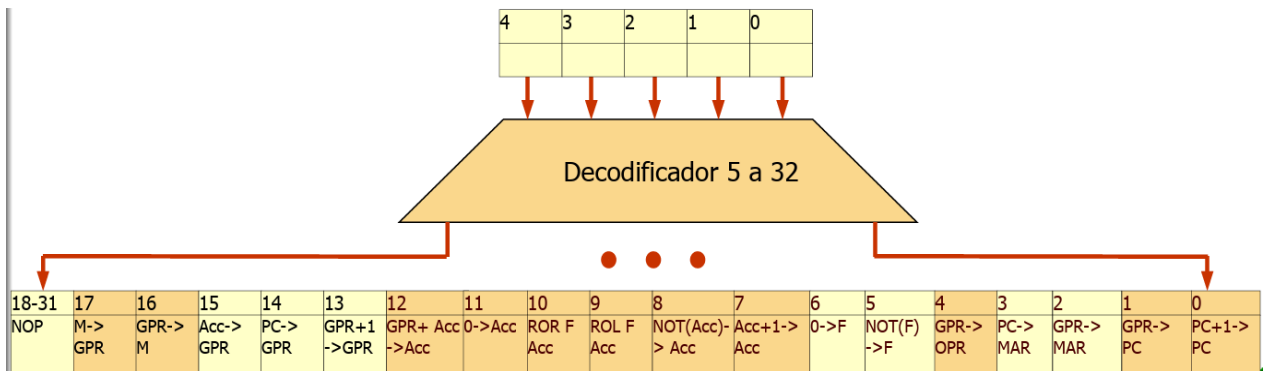
Las señales de control que gobiernan una misma unidad se suelen agrupar en campos dentro de la codificación de las microinstrucciones.

El formato de las microinstrucciones:

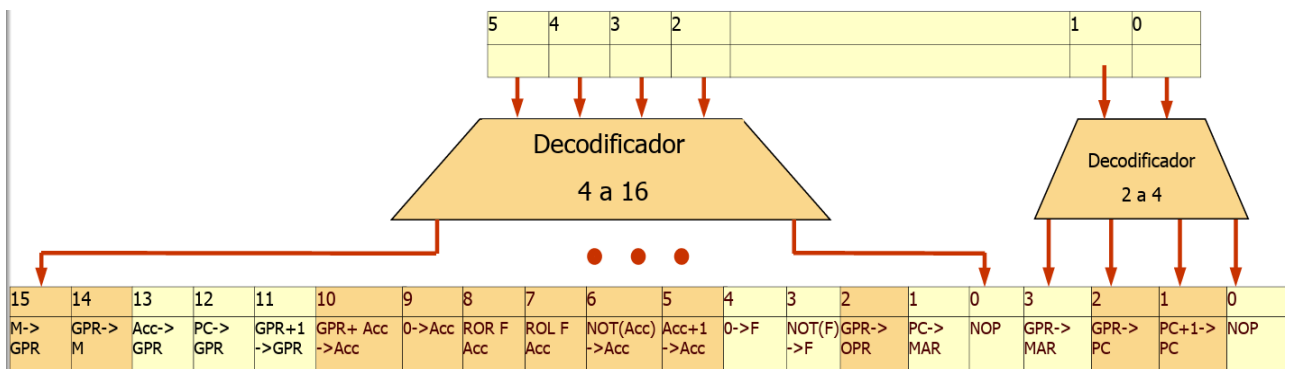
- **Formato no codificado:** Las micropalabras tienen un bit para cada señal de control.
 - Ej. La Computadora Mejorada tiene 18 microoperaciones, por lo tanto, las micropalabras de su Unidad de Control tendrían 18 bits.
 - Problema: Espacio (ROM con palabras muy grandes).

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M->GPR	GPR->M	Acc->GPR	PC->GPR	GPR+1->GPR	GPR+Acc->Acc	0->Acc	ROR F Acc	ROL F Acc	NOT(Acc)->Acc	Acc+1->Acc	0->F	NOT(F)->F	GPR->OPR	PC->MAR	GPR->MAR	GPR->PC	PC+1->PC

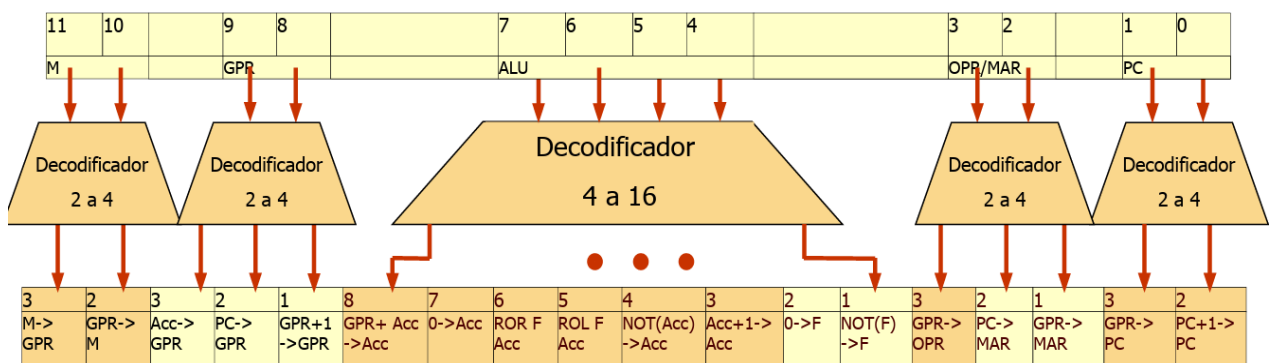
- **Formato completamente codificado:** Se codifica la activación de todos sus señales de control con menos bits (aunque en cada codificación solo se activa 1 única señal de control) .
 - Ej. La Computadora Mejorada tiene 18 microoperaciones, por lo tanto, con 5 bits nos bastaría.



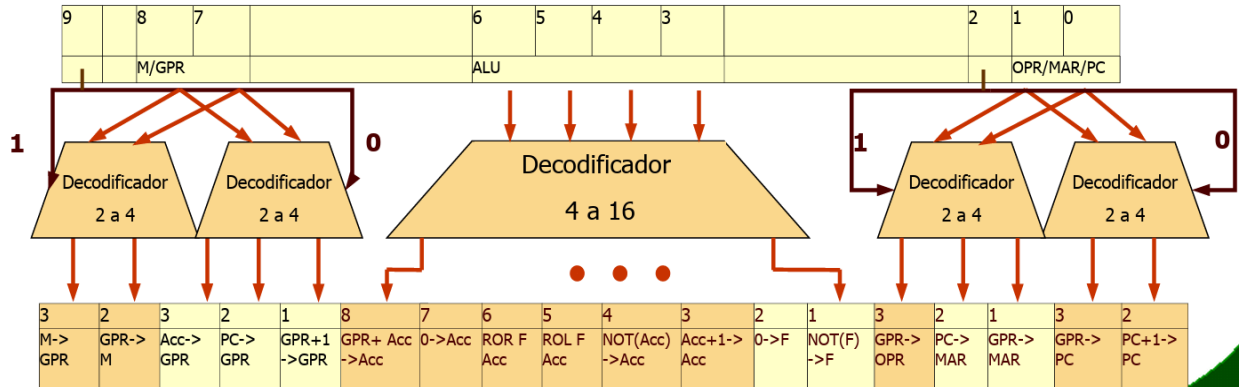
- **Formato codificado por trozos:** Se codifica la activación señales de control con menos bits agrupadas en campos (esto permite que varias señales se activen a la vez).
 - La microinstrucción es un poco más grande, pero los decodificadores son de menor tamaño.



Los campos deben ser consistentes para que la división sea útil. En todos los decodificadores hay que dar la oportunidad de “no hacer nada” (NOP).



- **Formato con solapamientos:** Si hay señales excluyentes entre sí, se pueden solapar campos, reduciendo el tamaño de las microinstrucciones.
 - Un bit indica si el resto de bits corresponden con el campo 1 o con el campo 2.



- **Nanoprogramación.**

2. Tipos de Microprogramación

Microprogramación horizontal:

- Formato no codificado.
- Microinstrucciones muy largas.
- Alto grado de paralelismo.
- Alto consumo de memoria de control.
- Más rápidas.

Microprogramación horizontal (formato no codificado):

Las micropalabras tienen un bit para cada señal de control.

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M->GPR	GPR->M	Acc->GPR	PC->GPR	GPR+1->GPR	GPR+Acc->Acc	0->Acc	ROR F Acc	ROL F Acc	NOT(Acc)->Acc	Acc+1->Acc	0->F	NOT(F)->F	GPR->OPR	PC->MAR	GPR->MAR	GPR->PC	PC+1->PC

- Ej. La activación de la microoperación “GPR+Acc->Acc” se codifica como 00 0001 0000 0000b = 01000h.

Dir. CROM	Etiqueta	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Codificación
FETCH	PC->MAR															1				00008
1	M->GPR	1																		20000
2	GPR->OPR PC+1->PC														1				1	00011
ADD	GPR->MAR																1			00004
4	M->GPR	1																		20000
5	GPR+Acc->Acc						1													01000
ADDI	GPR->MAR																1			00004
7	M->GPR	1																		20000
8	GPR->MAR																1			00004
9	M->GPR	1																		20000
10	GPR+Acc->Acc						1													01000
STA	GPR->MAR																1			00004
12	Acc->GPR			1																08000
13	GPR->M		1																	10000
JMP	GPR->PC																	1		00002
JMPI	GPR->MAR																1			00004
16	M->GPR	1																		20000
17	GPR->PC																	1		00002

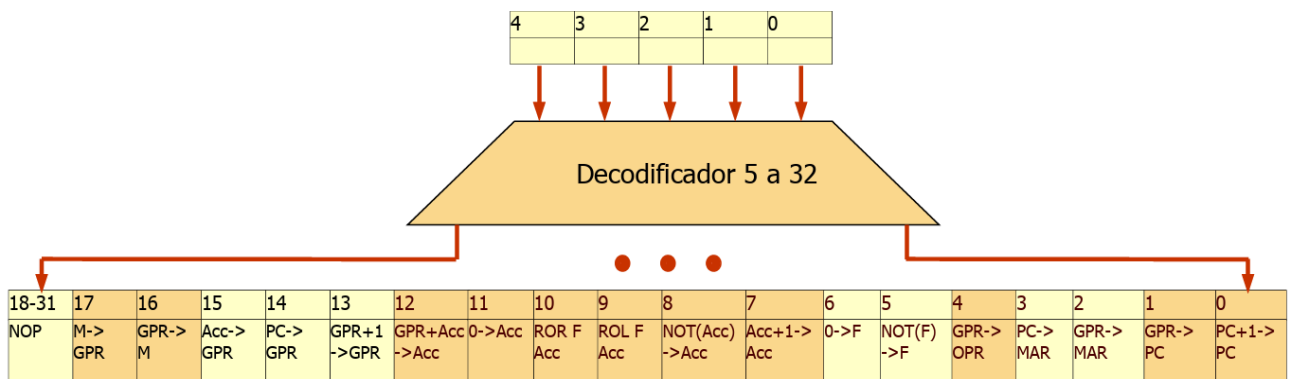
Continuaría con el resto de instrucciones

Microprogramación vertical:

- Formato codificado.
- Microinstrucciones cortas.
- Bajo paralelismo.
- Menor consumo de memoria de control.
- Menos rápidas.

Microprogramación vertical con formato completamente codificado:

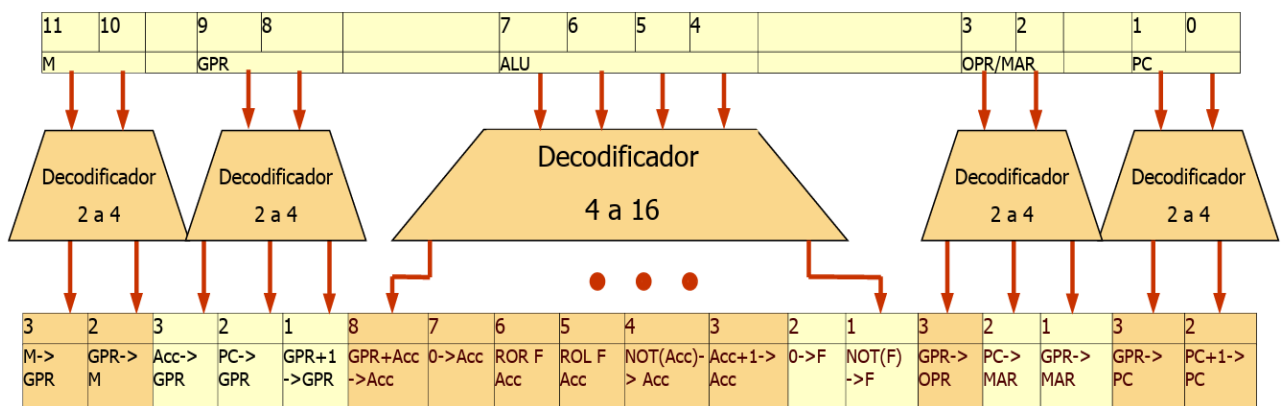
Hay 1 valor diferente para cada microinstrucción por cada microoperación que se active.



- Ej. La activación de la microoperación “GPR+Acc->Acc” se codifica como 12, ya que activa esa posición (01100b = 0Ch).
- Tamaño:
 - N° de palabras = n = 19 (Real: 32).
 - N° bits por palabra = W = 5.
 - Tamaño: 95 bits (Real: 160 bits).
- No se pueden ejecutar más de una microoperación en cada microinstrucción.(Ver FETCH).

Dir. CROM	Etiqueta	Código	Codificación	Hexad.
FETCH	PC->MAR	3	00011	03
1	M->GPR	17	10001	11
2	PC+1->PC	0	00000	00
3	GPR->OPR	4	00100	04
ADD	GPR->MAR	2	00010	02
5	M->GPR	17	10001	11
6	GPR+Acc->Acc	12	01100	0C
ADDI	GPR->MAR	2	00010	02
8	M->GPR	17	10001	11
9	GPR->MAR	2	00010	02
10	M->GPR	17	10001	11
11	GPR+Acc->Acc	12	01100	0C
STA	GPR->MAR	2	00010	02
13	Acc->GPR	15	01111	0F
14	GPR->M	16	10000	10
JMP	GPR->PC	1	00001	01
JMPI	GPR->MAR	2	00010	02
17	M->GPR	17	10001	11
18	GPR->PC	1	00001	01

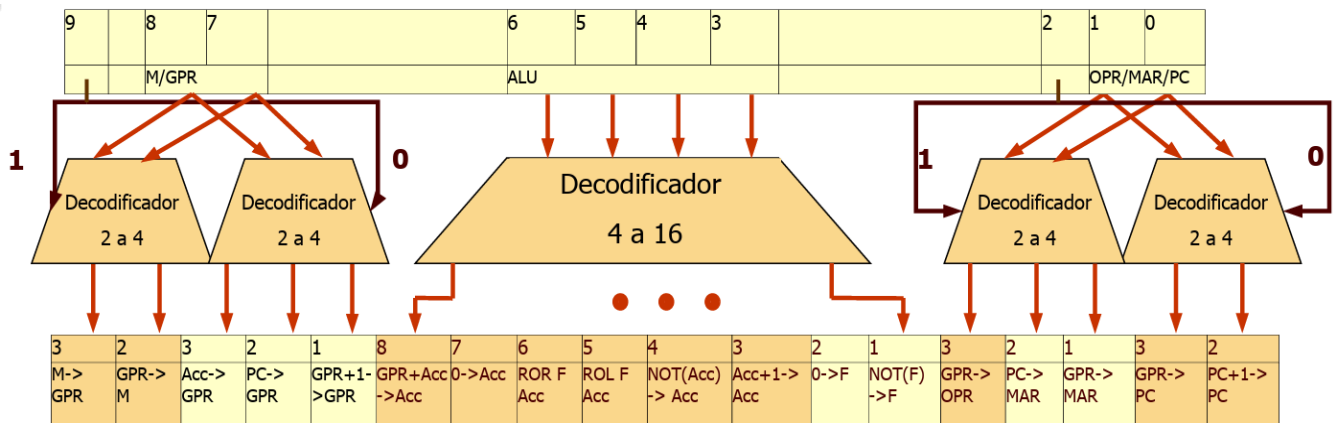
Microprogramación vertical con formato codificado por trozos:



- Ej. La activación de la microoperación “GPR+Acc->Acc” se codificaría como 00|00|1000|00|00 => 080h.
- Tamaño:
 - n = 18 (Real: 32).
 - W = 12.
 - Tamaño: 216 bits (Real: 384 bits).

Dir. CROM	Etiqueta	11	10	9	8	7	6	5	4	3	2	1	0	Codificación
FETCH	PC-> MAR									1	0			008
1	M->GPR	1	1											C00
2	GPR->OPR PC+1 -> PC									1	1	1	0	00E
ADD	GPR->MAR									0	1			004
4	M->GPR	1	1											C00
5	GPR+Acc->Acc					1	0	0	0					080
ADDI	GPR->MAR									0	1			004
7	M->GPR	1	1											C00
8	GPR->MAR									0	1			004
9	M->GPR	1	1											C00
10	GPR+Acc->Acc					1	0	0	0					080
STA	GPR->MAR									0	1			004
12	Acc->GPR			1	1									300
13	GPR->M	1	0											800
JMP	GPR->PC											1	1	003
JMPI	GPR->MAR									0	1			004
16	M->GPR	1	1											C00
17	GPR->PC											1	1	003

Microprogramación vertical con formato con solapamientos:

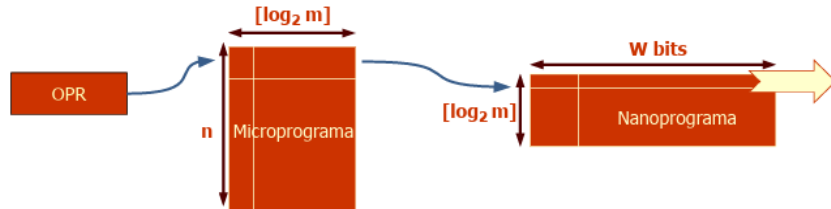


- Ej. La activación de la microoperación "GPR+Acc->Acc" se codificaría como 00|0 100|0 000 => 040h.
- Tamaño:
 - n = 18 (Real: 32).
 - W = 10.
 - Tamaño: 180 bits (Real: 320 bits).

Dir. CROM	Etiqueta	9	8	7	6	5	4	3	2	1	0	Codificación
FETCH	PC->MAR								1	1	0	006
1	M->GPR PC+1->PC	1	1	1					0	1	0	382
2	GPR->OPR								1	1	1	007
ADD	GPR->MAR								1	0	1	005
4	M->GPR	1	1	1								380
5	GPR+Acc->Acc				1	0	0	0				040
ADDI	GPR->MAR								1	0	1	005
7	M->GPR	1	1	1								380
8	GPR->MAR								1	0	1	005
9	M->GPR	1	1	1								380
10	GPR+Acc->Acc				1	0	0	0				040
STA	GPR->MAR								1	0	1	005
12	Acc->GPR	0	1	1								180
13	GPR->M	1	1	0								300
JMP	GPR->PC								0	1	1	003
JMPI	GPR->MAR								1	0	1	005
16	M->GPR	1	1	1								380
17	GPR->PC								0	1	1	003

Microprogramación vertical: Nanoprogramación.

- Objetivo: Reducir el tamaño de la memoria de control.
- Implica una memoria a 2 niveles.
- Se construye una memoria de m palabras de W bits, que contendrá las m microinstrucciones “únicas”.
- Se construye la memoria de control sustituyendo los W bits por la dirección única de la microinstrucción asociada en cada paso del microprograma.
- El tamaño total será: $n \cdot \lceil \log_2 m \rceil + \lceil \log_2 m \rceil \cdot W$ bits.



2.1. Tipos de Microprogramación: Microprogramación Horizontal. Ejemplo.

Supongamos instrucciones codificadas en 17 palabras de Control.

- $W = 18$ bits.
- $n = 17$ palabras.
- Tamaño: 306 bits.
- Implementación real:
 - Dirección 5 bits \Rightarrow 32 palabras.
 - Tamaño: 576 bits. (240 bits “vacíos”).
- Microinstrucciones “únicas”: 8
 - 00008.
 - 20000.
 - 00011.
 - 00004.
 - 01000.
 - 08000.
 - 10000.
 - 00002.

Dir. CROM	Microinstrucción
00000	00008
00001	20000
00010	00011
00011	00004
00100	20000
00101	01000
00110	00004
00111	20000
01000	00004
01001	20000
01010	01000
01011	00004
01100	08000
01101	10000
01110	00002
01111	00004
10001	20000
...	

2.2. Tipos de Microprogramación: Microprogramación Vertical. Ejemplo.

Tamaño ROM de Control:

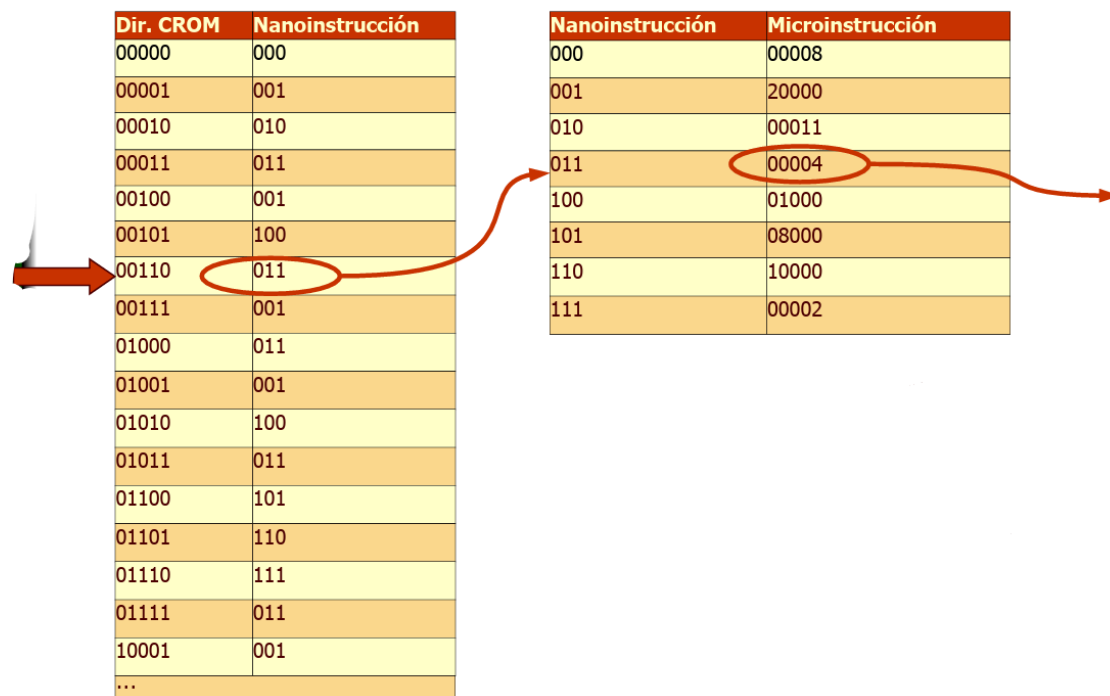
- $n = 17$ (Implementación real: 32).
- $W = 3$.
- Tamaño: 51 (Implementación real: 96).

Tamaño ROM de Nanoinstrucciones:

- $n = 8$.
- $W = 18$.
- Tamaño: 144 bits.

Tamaño Total: 195 bits (Real: 240 bits).

Reducción de Tamaño: 36.3% (Real: 58.3%).



2.3. Secuenciamiento en Microprogramación

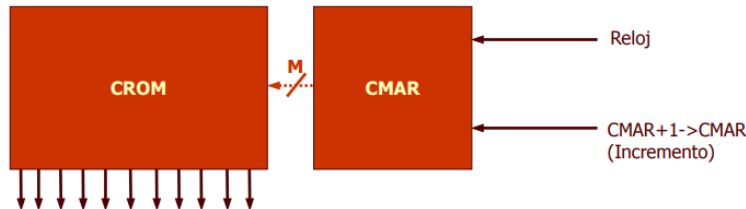
Secuencia dentro de un microprograma:

- Tras la ejecución de cada microinstrucción se tiene que determinar qué nueva microinstrucción se ha de ejecutar.
- Hay 3 posibilidades:
 - Pasar a ejecutar la microinstrucción que se encuentra en la dirección CROM consecutiva (Incremento).
 - Pasar a ejecutar la microinstrucción que se encuentre en una dirección CROM especificada explícitamente (Bifurcación).
 - Pasar a ejecutar la primera microinstrucción asociada al ciclo de ejecución de una instrucción determinada (Carga de Rutina).

- Además, la determinación de la siguiente microinstrucción puede ser:
 - Incondicional.
 - Condicional.

El secuenciamiento se consigue variando la dirección contenida en el registro de dirección de acceso a la CROM (CMAR: Control Memory Access Register).

Incremento: Pasar a la microinstrucción que se encuentra en la dirección CROM consecutiva => Incrementar el contenido de CMAR.



Bifurcación: Saltar a una microinstrucción que se encuentra en una dirección CROM en particular => Dicha dirección debe ser indicada (codificación de la dirección en la propia micropalabra) => Carga paralela de dicha dirección.

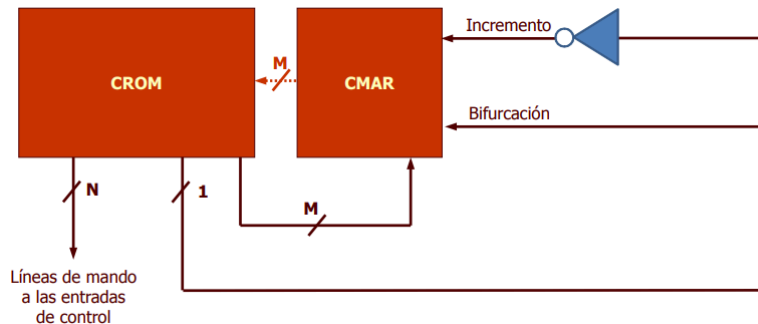


Tabla de Bifurcación del control actual:

S0	I	B	Comentario
0	1	0	Incremento
1	0	1	Bifurcación incondicional

Bifurcación condicional: Dependiendo de los bits de estado del sistema se querrá llevar un rumbo u otro en casos determinados.

S1	S0	Z	F	I	B	Comentario
0	0	X	X	1	0	Incremento
0	1	X	X	0	1	Bifurcación incondicional
1	0	0	X	0	1	Salta si Z=0
1	0	1	X	1	0	Incrementa si Z=1
1	1	X	0	1	0	Incrementa si F=0
1	1	X	1	0	1	Salta si F=1

Bloque actual de la LCB:



Carga de rutina: Saltar a la microinstrucción de una de las rutinas de las instrucciones => La dirección de inicio de cada instrucción debe conocerse => Carga paralela de dicha dirección.

Selección de tipo de secuenciamiento:

- Cada microinstrucción podrá seleccionar el tipo de secuenciación que deseen realizar.
- En función del conjunto de señales de secuenciación indicadas en la propia codificación de la micropalabra y de las señales de estado externas, se escogerá Incremento, Bifurcación o Carga de Rutina.
- Tras la última microinstrucción de cada instrucción se debe saltar al ciclo de búsqueda => Bifurcación incondicional a FETCH.

Tabla de Bifurcación final para el repertorio básico:

S2	S1	S0	Z	F	I	B	R	Comentario
0	0	0	X	X	1	0	0	Incremento
0	0	1	X	X	0	1	0	Bifurcación incondicional
0	1	0	0	X	0	1	0	Salta si Z=0
0	1	0	1	X	1	0	0	Incrementa si Z=1
0	1	1	X	0	1	0	0	Incrementa si F=0
0	1	1	X	1	0	1	0	Salta si F=1
1	X	X	X	X	0	0	1	Carga Rutina incondicional

2.4. Codificación de la Bifurcación

Cada microinstrucción debe incorporar las señales de secuenciación.

- En caso de bifurcación, también tendrá que incluir la dirección de salto.
- Esta incorporación se realiza concatenada a la codificación de las microinstrucciones.

Ejemplo: Supongamos 3 señales de control de bifurcación (S2, S1, S0) y una CROM de 256 palabras (8 bits de dirección).

10	9	8	7	6	5	4	3	2	1	0
S2	S1	S0	Dirección de Salto							

Así, con un formato con solapamientos, las micropalabras serían:

20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M/GPR			ALU				OPR/MAR/PC				S2	S1	S0	Dirección de Salto						

20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M/GPR			ALU				OPR/MAR/PC				S2	S1	S0	Dirección de Salto						

S2	S1	S0	Z	F	I	R	B
0	0	0	X	X	1	0	0
0	0	1	X	X	0	0	1
0	1	0	0	X	0	0	1
0	1	0	1	X	1	0	0
0	1	1	X	0	1	0	0
0	1	1	X	1	0	0	1
1	X	X	X	X	0	1	0

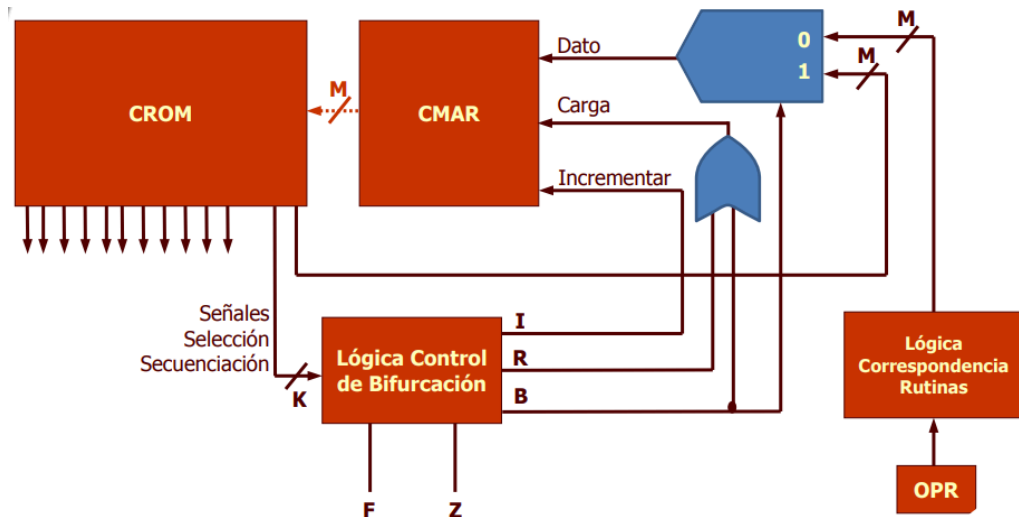
Ejemplo: Se desea codificar "GPR+Acc->Acc". Si Z=0, se bifurcará a la posición 24h; en caso contrario, pasaremos a la siguiente microinstrucción.

- "GPR+Acc->Acc" se obtiene introduciendo 1000b en los bits 17-14 de la micropalabra de control.
- Para Bifurcar (B) si Z=0 o Incrementar (I) si Z=1; utilizamos 010 de las señales de control de bifurcación (bits 10-8 de la micropalabra).
- También habrá que codificar la posición de salto (24h), para que en el caso que se deba realizar la bifurcación se pueda realizar. La dirección se codifica en los bits 0-7 de la micropalabra de control.

20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0

La codificación sería 0010|0000|0010|0010|0100b = 20224h.

2.5. Control de Bifurcación



3. Habilitación de Microinstrucciones

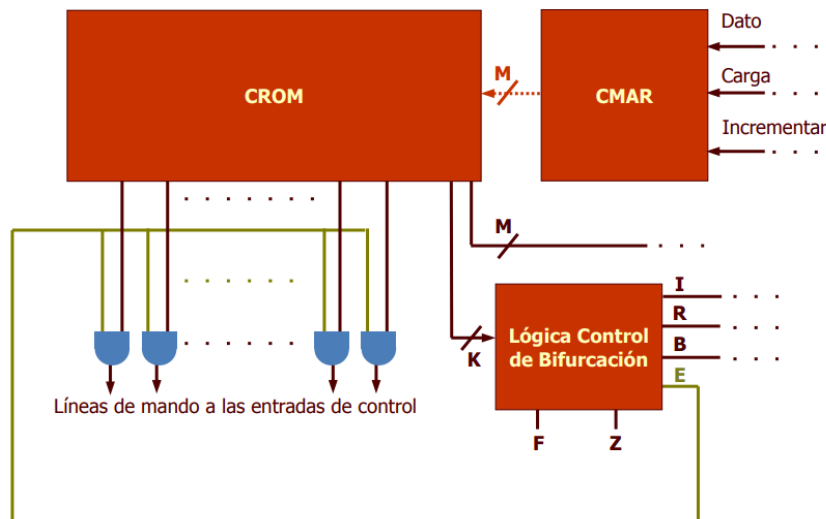
De manera adicional, se puede inhabilitar la ejecución de la microoperación seleccionada => Útil para ejecuciones condicionales.

Ejemplo:

- Si Z=0, ponemos a cero el acumulador y bifurcamos a una determinada posición.
- Si Z=1, no hacemos nada e incrementamos a la siguiente microinstrucción.

Para habilitar/inhabilitar la ejecución de microoperaciones seleccionadas, se introduce un bloque intermedio entre la CROM y las salidas, que permite el paso si la señal Enable está activa; en caso contrario, todas las salidas son 0.

S2	S1	S0	Z	F	I	B	R	E
0	0	0	X	X	1	0	0	1
0	0	1	X	X	0	1	0	1
0	1	0	0	X	0	1	0	1
0	1	0	1	X	1	0	0	1
0	1	1	X	0	1	0	0	1
0	1	1	X	1	0	1	0	1
1	0	0	X	X	0	0	1	1
1	0	1	0	X	0	1	0	1
1	0	1	1	X	1	0	0	0



3.1. Diseño de la Unidad de Control Cableado por Decodificadores Tiempo/Instrucción

El microprograma sigue existiendo => Codificación implícita mediante la existencia de cableado específico.

Para cada microoperación:

- Se activa exclusivamente en un determinado ciclo de tiempo de una instrucción.
- La activación puede ser incondicional o estar condicionada a un valor de estado.

Una microoperación se ejecuta: $M_n = I_i \cdot T_j \cdot C_k$

- **M_n** representa a una microoperación en particular de todas las que existen en el sistema.
- **I_i** representa la instrucción i -ésima del repertorio.
- **T_j** representa el ciclo j -ésimo de búsqueda o ejecución.
- **C_k** representa la condición k -ésima de estado.

“La microoperación M_n se ejecutará si la condición C_k es cierta durante el instante de ejecución T_j de la instrucción I_i del repertorio”.

La suma de todas las veces que se puede activar individualmente una microoperación, nos dará la lógica de activación completa.

Material necesario:

- Contador de tiempo.
 - Señales:
 - Incremento.
 - Carga paralelo.
- Decodificador de tiempo.
- Decodificador de instrucciones.
- Lógica combinacional (puertas AND, OR, NOT).

FETCH:

- PC->MAR.
- PC+1->PC.
- M->GPR.
- GPR->OPR.

ADD d: (Opcode 1)

- GPR->MAR.
- M->GPR.
- GPR+Acc->Acc.

ADDI d: (Opcode 2)

- GPR->MAR.
- M->GPR.
- GPR->MAR.
- M->GPR.
- GPR+Acc->Acc.

ISZ d: (Opcode 3)

- GPR->MAR.
- M->GPR.
- GPR+1->GPR.
- GPR->M.
- PC+1->PC (si Z = 1).

FETCH se realiza para **TODAS** las instrucciones y son siempre los 3 primeros ciclos:

- PC->MAR: t_0 .
- PC+1->PC: t_1 .
- M->GPR: t_1 .
- GPR->OPR: t_2 .

ADD:

- GPR->MAR: $l_1 \cdot t_3$.
- M->GPR: $l_1 \cdot t_4$.
- GPR+Acc->Acc: $l_1 \cdot t_5$.

ADDI:

- GPR->MAR: $l_2 \cdot t_3$.
- M->GPR: $l_2 \cdot t_4$.
- GPR->MAR: $l_2 \cdot t_5$.
- M->GPR: $l_2 \cdot t_6$.
- GPR+Acc->Acc: $l_2 \cdot t_7$.

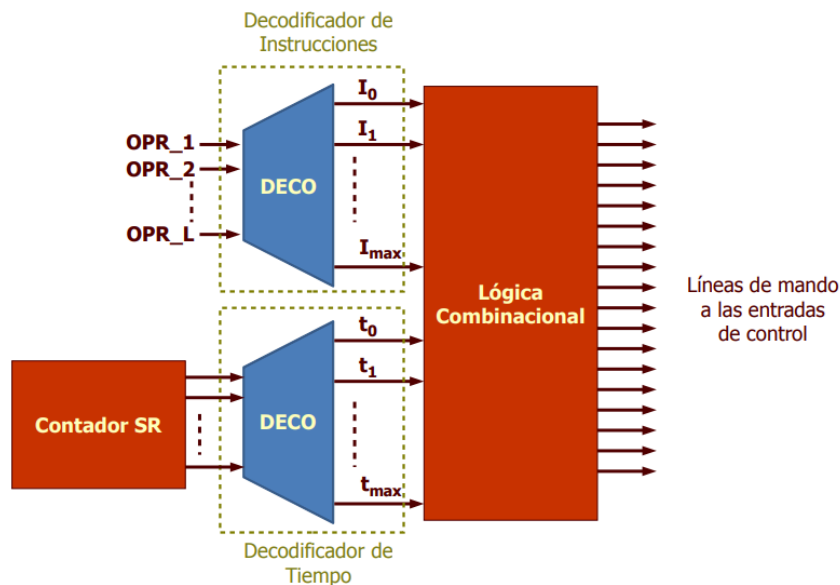
ISZ:

- GPR->MAR: $l_3 \cdot t_3$.
- M->GPR: $l_3 \cdot t_4$.
- GPR+1->GPR: $l_3 \cdot t_5$.
- GPR->M: $l_3 \cdot t_6$.
- PC+1->PC: $l_3 \cdot t_6 \cdot Z$.

Agrupación de microoperaciones:

- PC->MAR: t_0 .
- PC+1->PC: $t_1 + l_3 \cdot t_6 \cdot Z$.
- M->GPR: $t_1 + l_1 \cdot t_4 + l_2 \cdot t_4 + l_3 \cdot t_4$.
- GPR->OPR: t_2 .

- $GPR \rightarrow MAR: I_1 \cdot t_3 + I_2 \cdot t_3 + I_2 \cdot t_5 + I_3 \cdot t_3.$
- $GPR + Acc \rightarrow Acc: I_1 \cdot t_5 + I_2 \cdot t_7.$
- $GPR + 1 \rightarrow GPR: I_3 \cdot t_5.$
- $GPR \rightarrow M: I_3 \cdot t_6.$



4. Secuencia Cableada

En el diseño cableado mediante decodificadores de tiempo e instrucciones, la secuenciación se consigue modificando el contador de tiempo (denominado SR, Sequence Register).

El contador de tiempo tendrá 3 microoperaciones:

- Cuenta ascendente.
- Carga paralela.
- Puesta a 0.

Para cada ciclo de cada instrucción se ha de indicar tanto las microoperaciones que se activan como la secuenciación.

Ejemplo: Ciclo 4 (t_4), Instrucción ADDI (I_2)

- $I_2 \cdot t_4$ $GPR \rightarrow MAR$ $SR \rightarrow SR + 1.$
- De manera análoga a la microoperación “ $GPR \rightarrow MAR$ ”, se determinará de manera global la activación de “ $SR \rightarrow SR + 1$ ”.
- La carga paralela implicará la selección del valor que se ha de cargar: Selección codificada mediante el cableado de cada dígito binario.
- La vuelta al ciclo de búsqueda se realiza mediante la puesta a 0 síncrona.

4.1. Comparación de las técnicas de diseño

Cableado mediante registros de Desplazamiento:

- También llamada mediante “trenes de biestables”.
- Implementación sencilla.
- Alta flexibilidad.
- Facilidad de análisis.
- Bastante rápida.
- Minimiza la necesidad de lógica adicional.
- Alto uso de biestables (muy por encima del óptimo).

Cableado mediante Circuito Secuencial Síncrono:

- Tedioso y rígido.
- Difícil de analizar.
- Difícil de detectar errores.
- Garantiza la ejecución más rápida de todos los métodos.
- Minimiza el número de biestables.

Microprogramado mediante ROM de Control:

- Muy fácil de diseñar.
- Extremadamente flexible.
- Fácil de analizar.
- Es el mecanismo más lento.
- Costosa en circuitería (ROM, Registros, Lógica combinatorial, etc).

Cableado mediante Decodificadores de Tiempo e Instrucción:

- Sencillo de diseñar.
- Bastante flexible.
- Fácil de analizar.
- Tiene una ejecución bastante rápida.
- Bastante lógica combinatorial.