CIRCUITOS SECUENCIALES SINCRONOS

Diagrama de Bloques de un autómata de Mealy

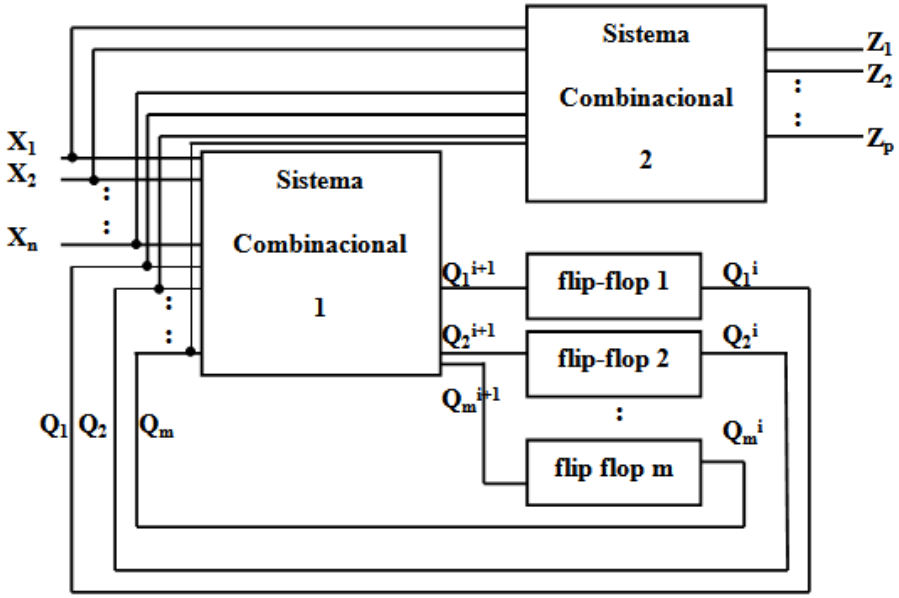
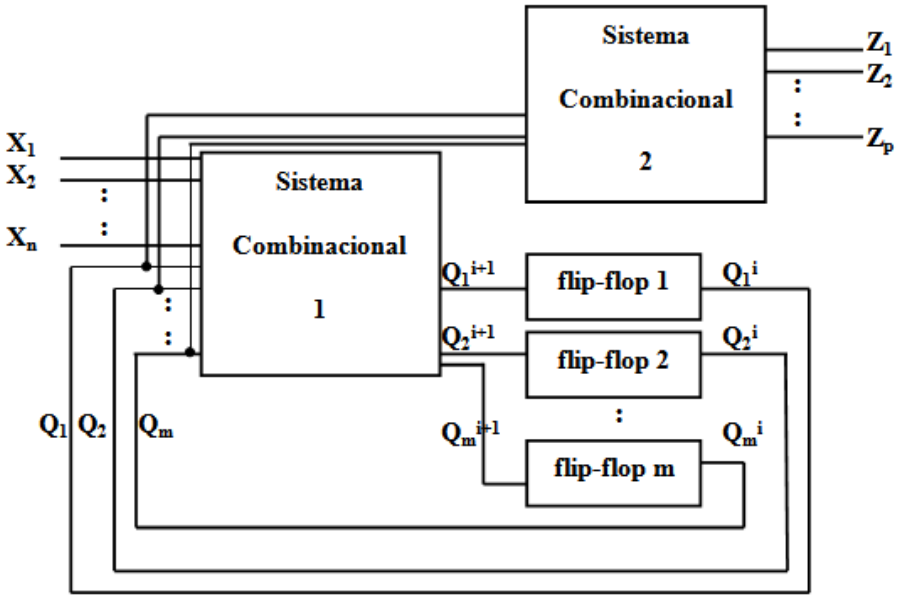
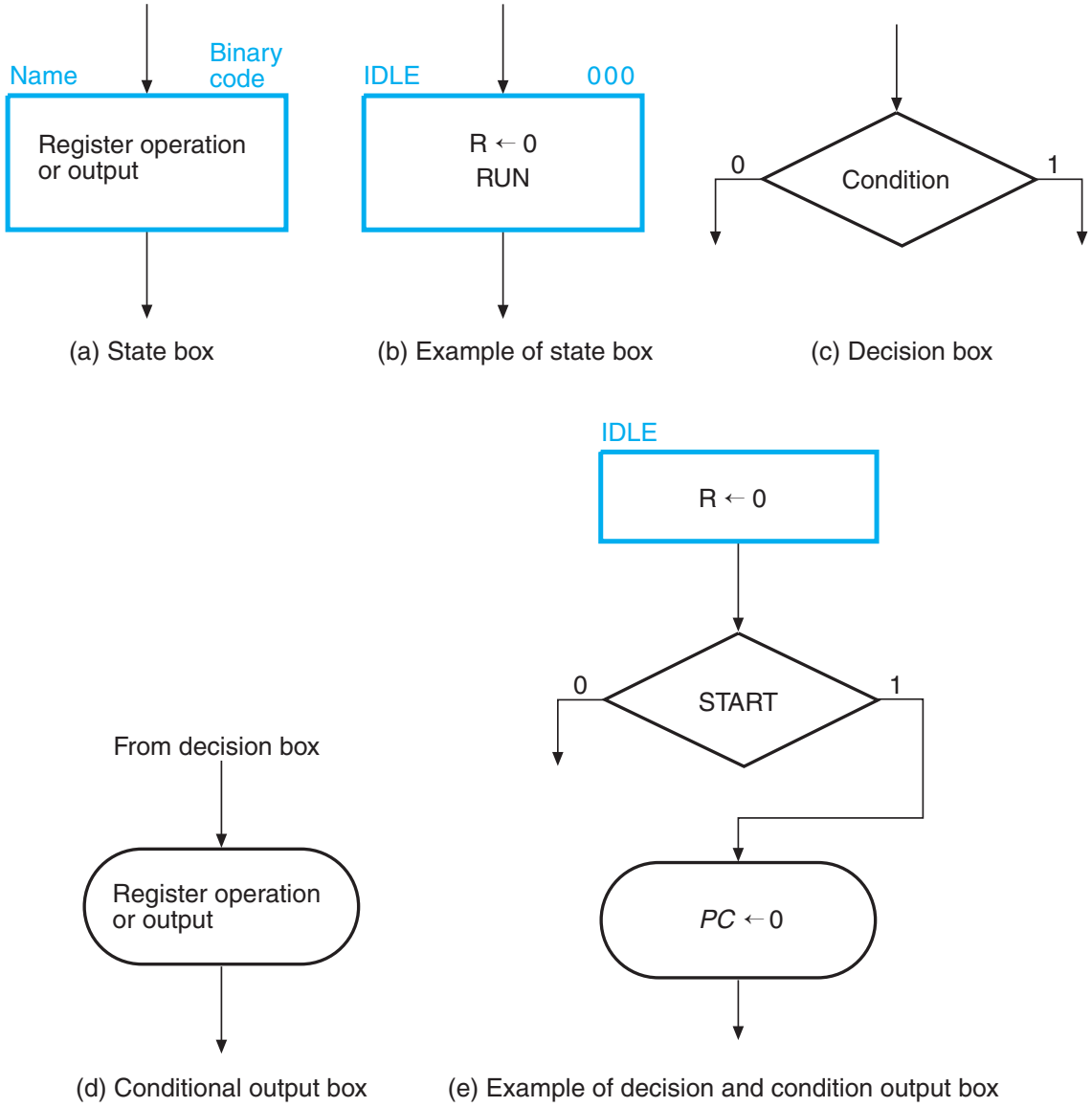


Diagrama de bloques de un autómata de Moore



Otras Maneras de diseñar un circuito secuencial síncrono

- Un flip-flop por estado (1 caliente, registro de desplazamiento)
- Sustituyendo las puertas logicas que forman parte de los sistemas combinacionales por cualquier bloque combinacional, en especial:
  - Multiplexores (inversores)
  - Decodificador + puertas lógicas
  - Memoria de acceso aleatorio
- Sustituyendo los flip-flops por un contador de módulo adecuado



(a) State box (b) Example of state box (c) Decision box

(d) Conditional output box (e) Example of decision and condition output box

Fig. 3-1 ASM Chart Elements

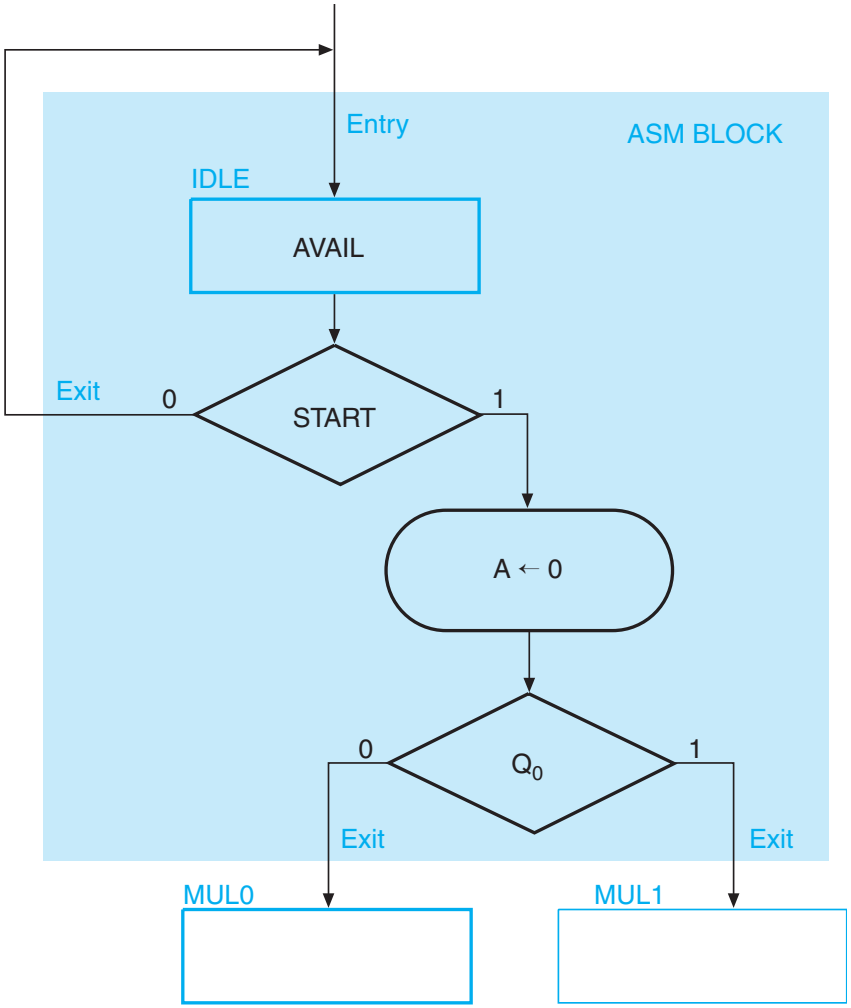


Fig. 3-2 ASM Block

SALIDAS:

Moore - AVAIL

Mealy - A (Clear Control of A register)

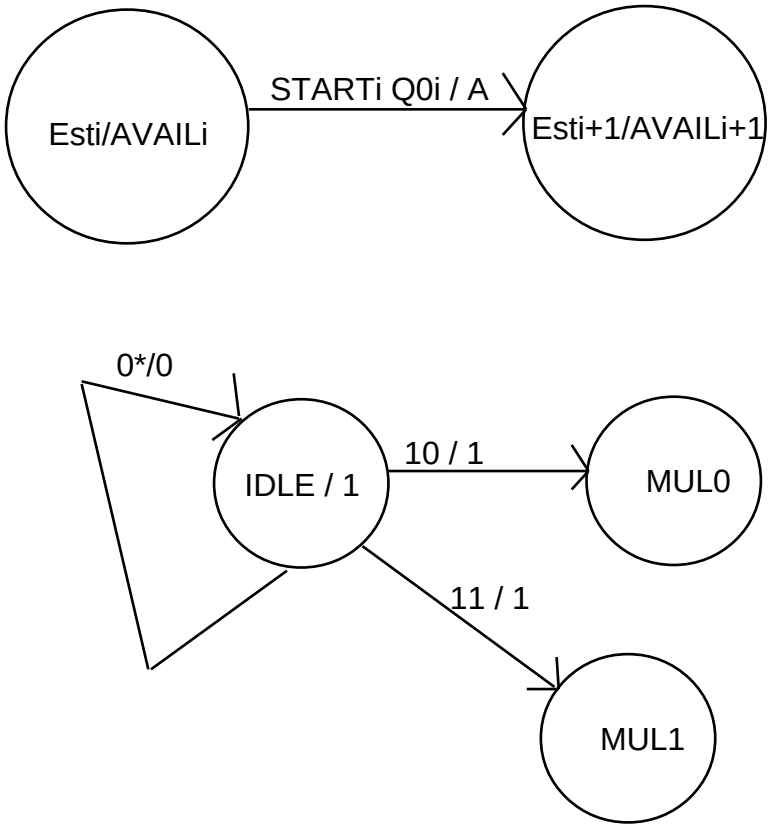


Diagrama de flujo equivalente

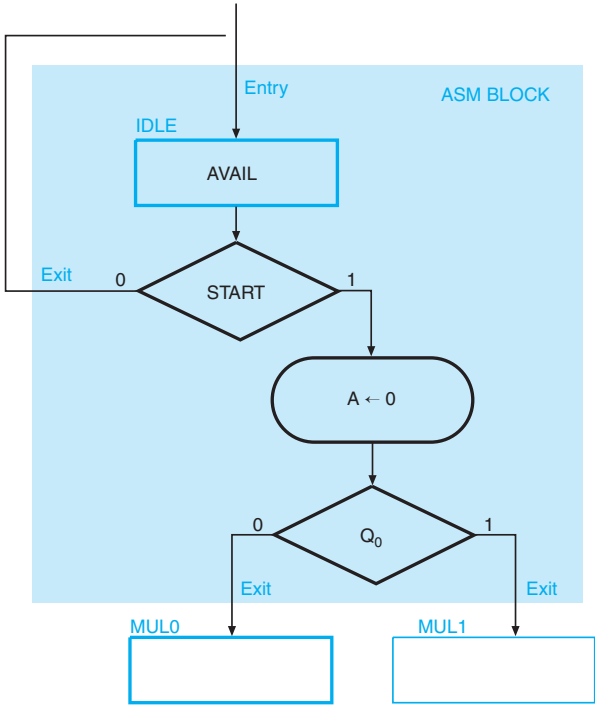


Fig. 3-2 ASM Block

Bucle MUL1 Transition

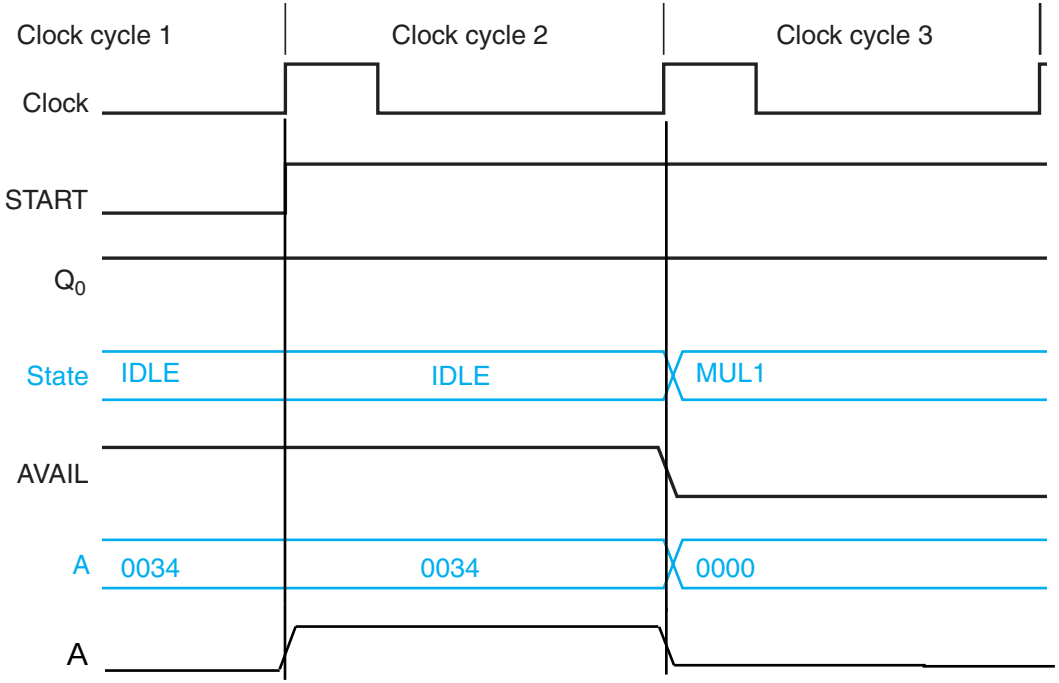


Fig. 3-3 ASM Timing Behavior

A clear control terminal in cycle 2,
but A register is cleared in cycle 3

23	10111	Multiplicand
19	10011	Multiplier
	10111	
	10111	
	00000	
	00000	
	10111	
437	110110101	Product

MODIFICACIONES:

1.- Productos Parciales

2.- Desplazamiento derecha suma parcial

3.- +0*multip - solo desplazamiento derecha

n bits x n bits = 2n bits

Fig. 3-4 Hand Multiplication Example

23	10111	Multiplicand		
19	10011	Multiplier		
	00000	Initial partial product	inicializacion a 0, cont n-1	
	10111	Add multiplicand, since multiplier bit is 1		
	10111	Partial product after add and before shift	1st add	
	010111	Partial product after shift	1st shift	
	10111	Add multiplicand, since multiplier bit is 1		
	1000101	Partial product after add and before shift[a]	acarreo	2nd add
	1000101	Partial product after shift		2nd shift
	01000101	Partial product after shift		3rd add, 3rd shift
	001000101	Partial product after shift		4th add, 4th shift
	10111	Add multiplicand, since multiplier bit is 1		5th add
	110110101	Partial product after add and before shift		5th shift
437	0110110101	Product after final shift		

a. Note that overflow temporarily occurred.

Fig. 3-5 Hardware Multiplication Example

23	10111	Multiplicand
	10011	Multiplier
	00000	Initial partial product
	10111	Add multiplicand
	10111	Partial product after add
	010111	Partial product after shift
	10111	Add multiplicand
	1000101	Partial product after add
	1000101	Partial product after shift
	01000101	Partial product after shift
	001000101	Partial product after shift
	10111	Add multiplicand
	110110101	Partial product after add
437	0110110101	Product after final shift

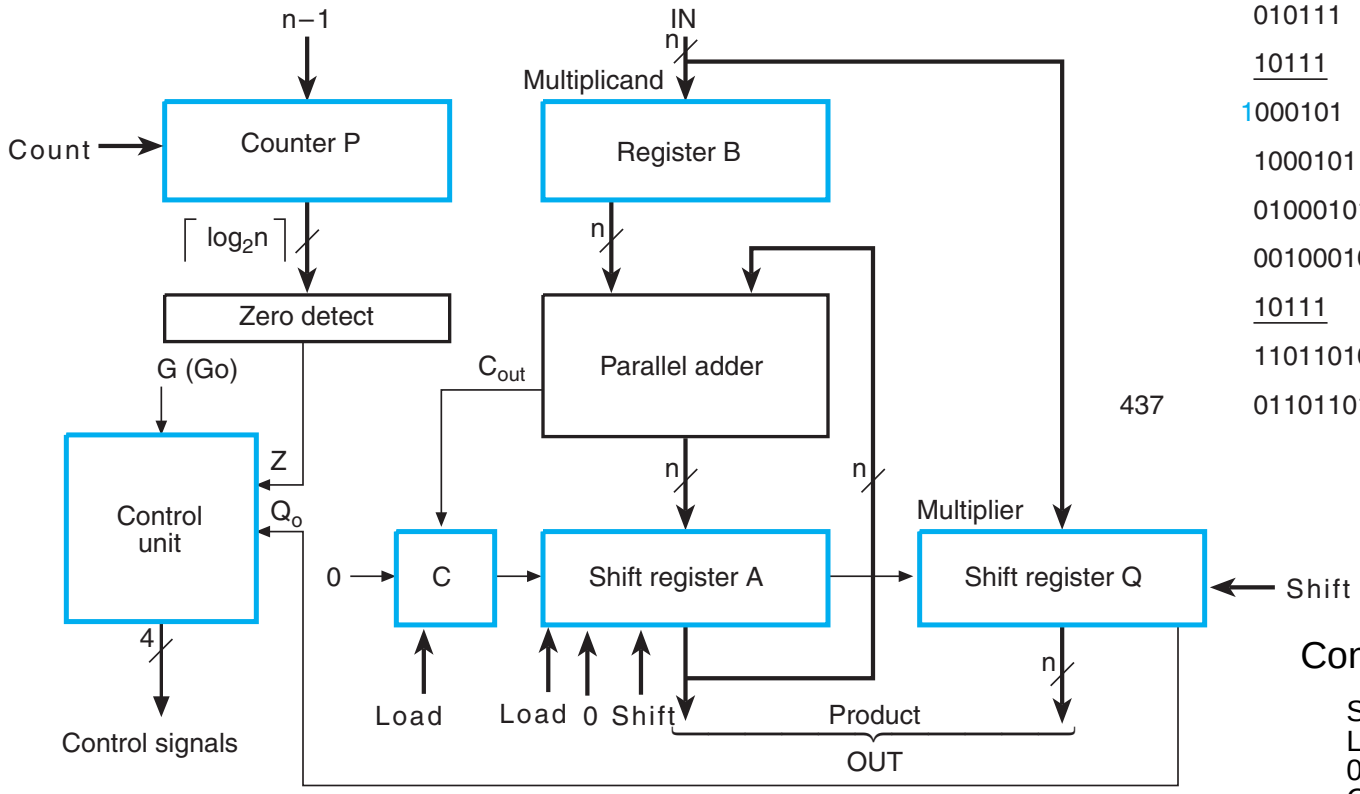


Fig. 3-6 Block Diagram for Binary Multiplier

Control Terminals (Output):

Shift
Load
0
Count

Control Input:

G
Q_0
Z

B REGISTER

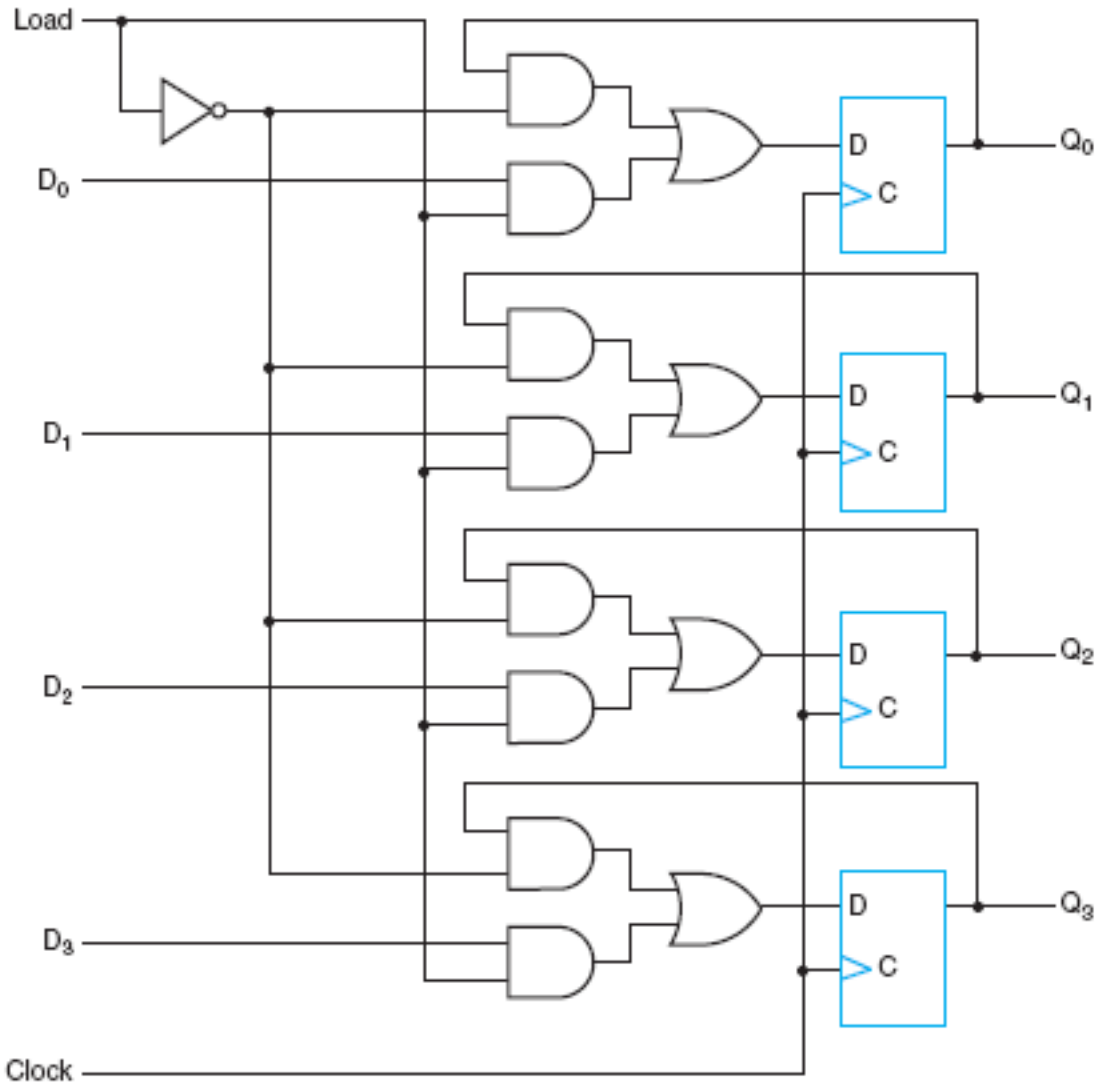


Fig 3.7. 4-Bit Register with Parallel Load

COUNTER P

DECREMENT INSTEAD OF

INCREMENT

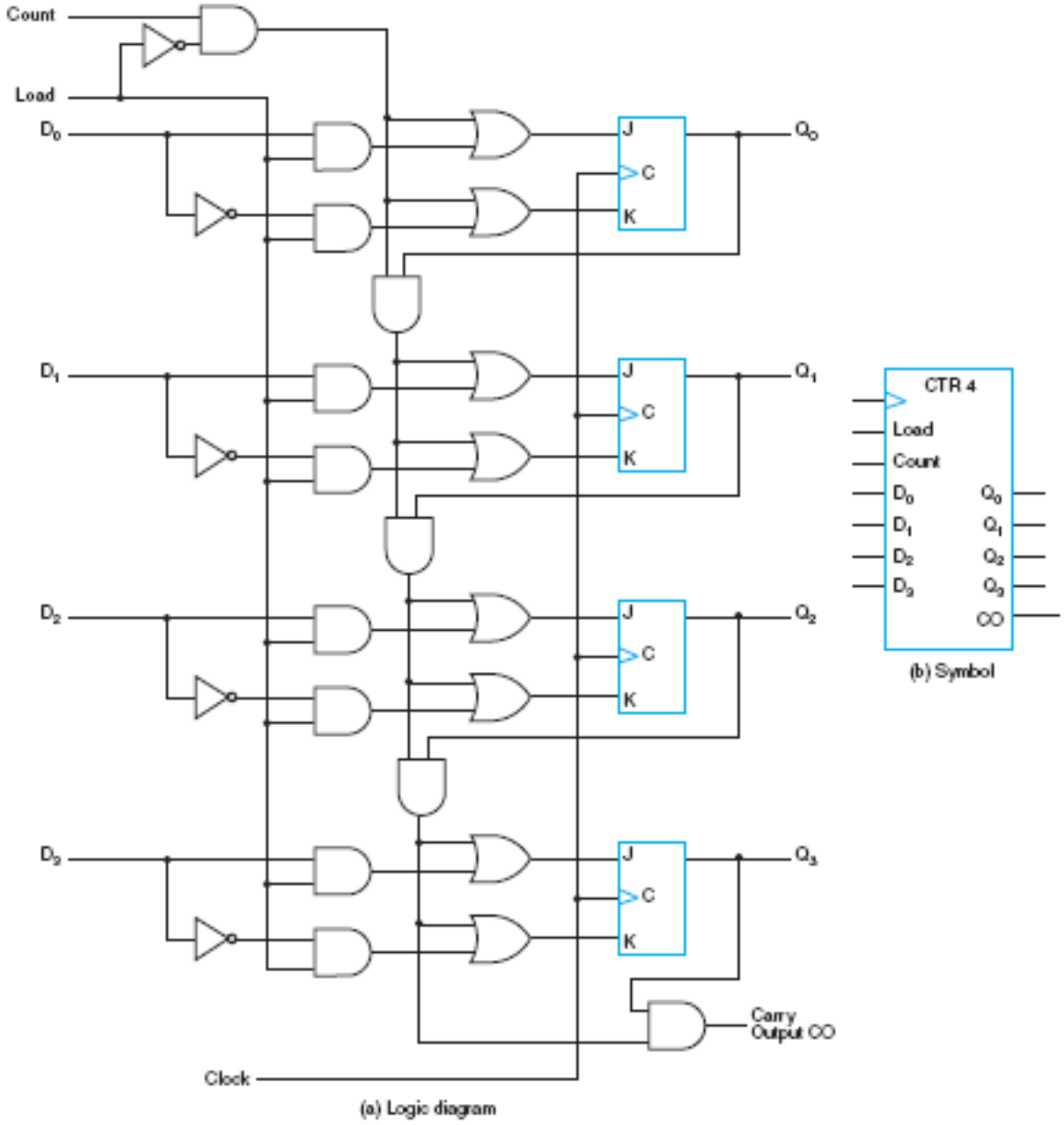


Fig 3.8. 4-Bit Binary Counter with Parallel Load

Q REGISTER

A REGISTER HAS ALSO CLEAR CONTROL

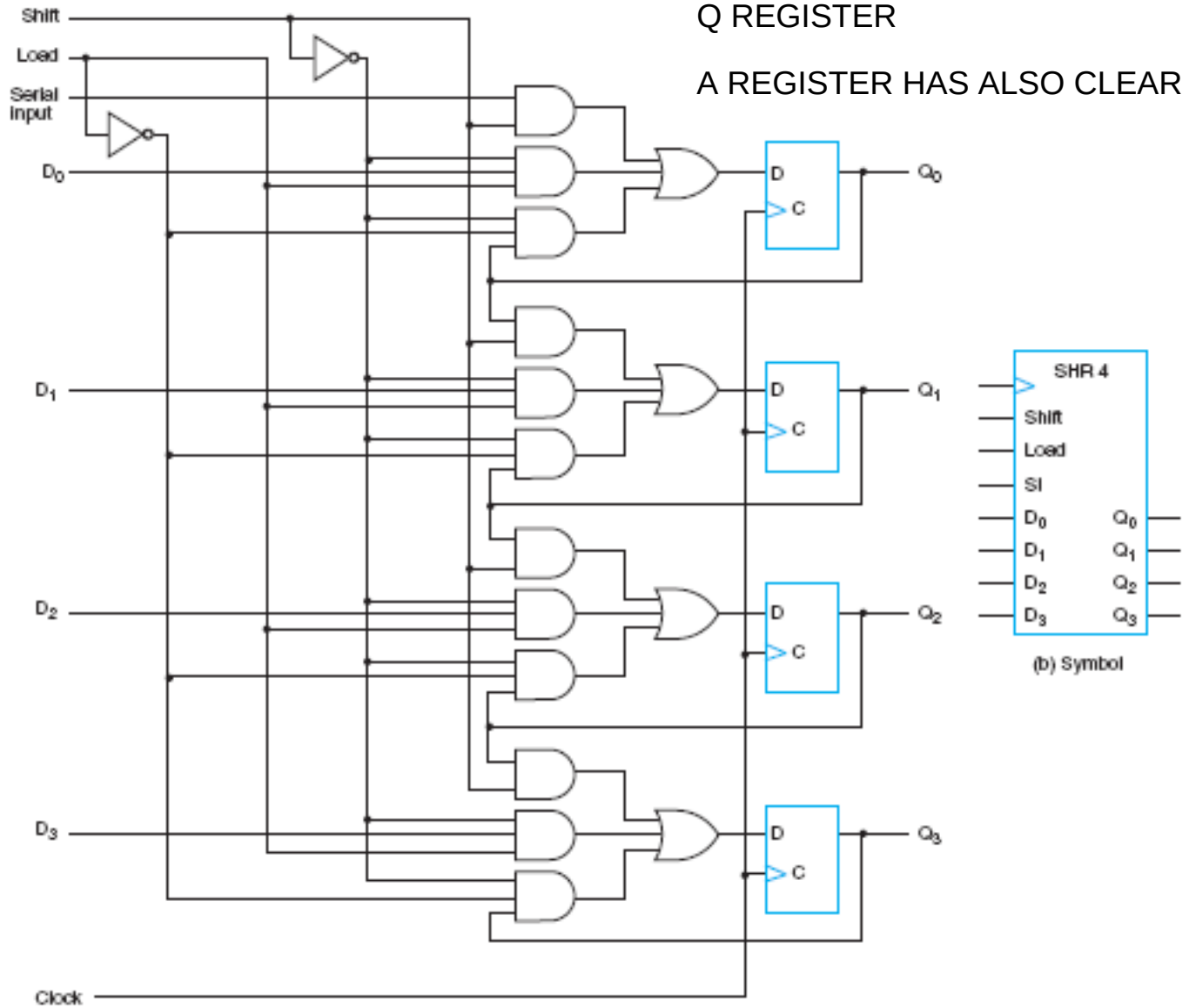


Fig 3.9. 4-Bit Shift Register with Parallel Load

Diseæo Controlador - Cableado

- Control microoperaciones - Salidas de control

- Secuenciacion de la unidad de control

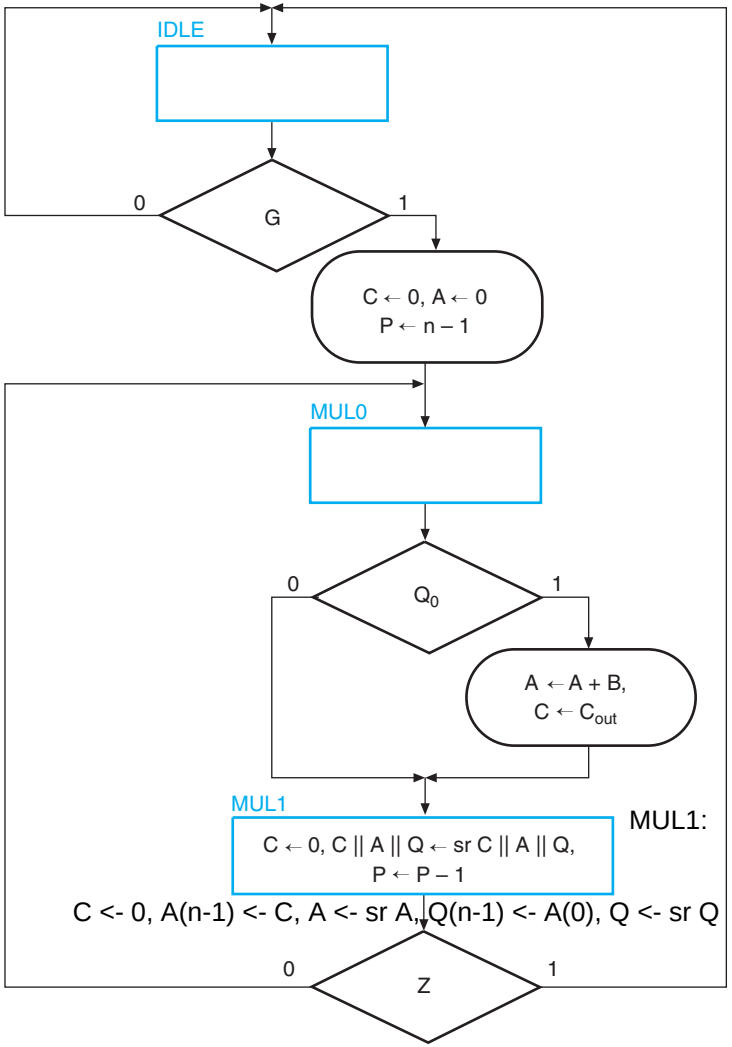
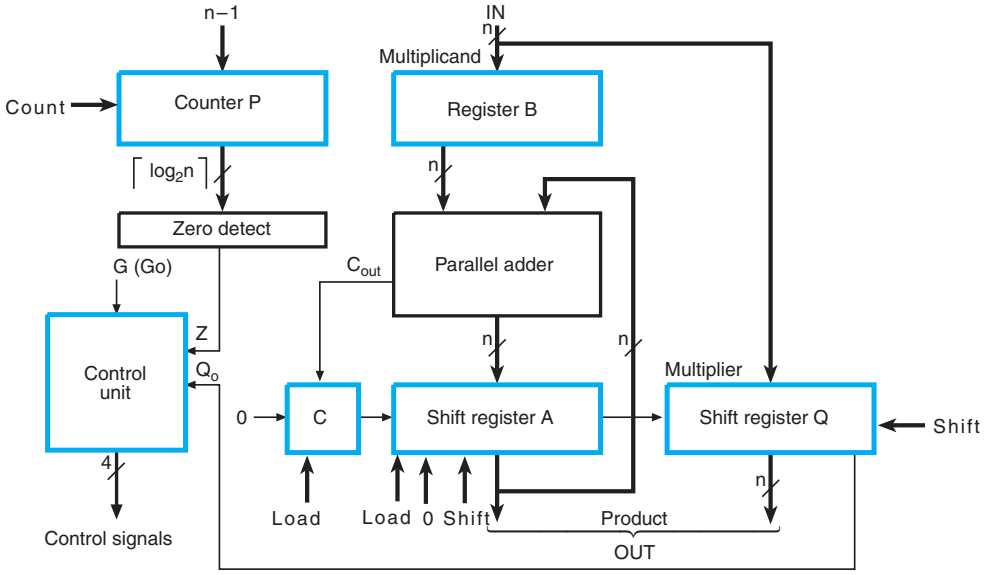
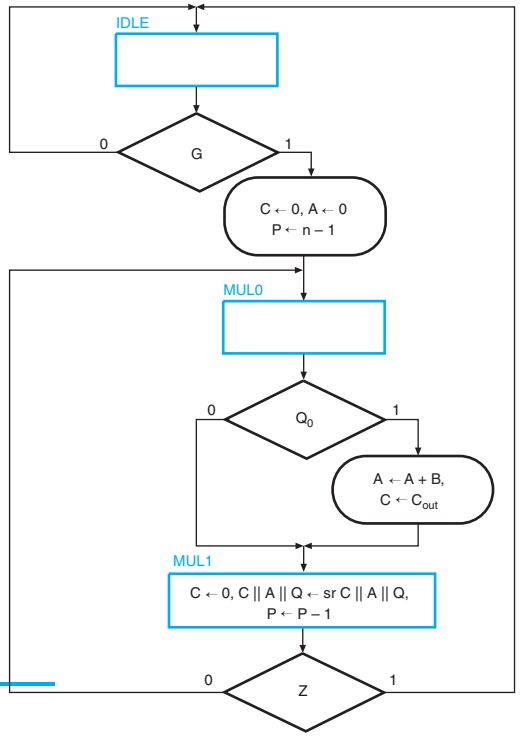


Fig. 3-10 ASM Chart for Binary Multiplier

23	10111	Multiplicand
	10011	Multiplier
	00000	Initial partial product
	10111	Add multiplicand
	10111	Partial product after add
	010111	Partial product after shift
	10111	Add multiplicand
	1000101	Partial product after add
	1000101	Partial product after shift
	01000101	Partial product after shift
	001000101	Partial product after shift
	10111	Add multiplicand
	110110101	Partial product after add
437	0110110101	Product after final shift





- Control microoperaciones - Salidas de control

(circuito combinacional 2)

TABLE 3-1
Control Signals for Binary Multiplier

	Block Diagram Module	Microoperation	Control Signal Name	Control Expression
	Register A:	$A \leftarrow 0$	Initialize	$IDLE \cdot G$
		$A \leftarrow A + B$	Load	$MUL0 \cdot Q_0$
		$C \| A \| Q \leftarrow \text{sr } C \| A \| Q$	Shift_dec	MUL1
No terminal control	Register B:	$B \leftarrow IN$	Load_B	LOADB
	Flip-Flop C:	$C \leftarrow 0$	Clear_C	$IDLE \cdot G + MUL1$
		$C \leftarrow C_{out}$	Load	—
No terminal control	Register Q:	$Q \leftarrow IN$	Load_Q	LOADQ
		$C \| A \| Q \leftarrow \text{sr } C \| A \| Q$	Shift_dec	—
	Counter P:	$P \leftarrow n - 1$	Initialize	—
		$P \leftarrow P - 1$	Shift_dec	—

4 SALIDAS:

Initialize
Load
Shift_dec
Clear_C

Table 3-1 Control Signals for Binary Multiplier

- Secuenciacion de la unidad de control

Estado ciclo siguiente (circuito combinacional 1)

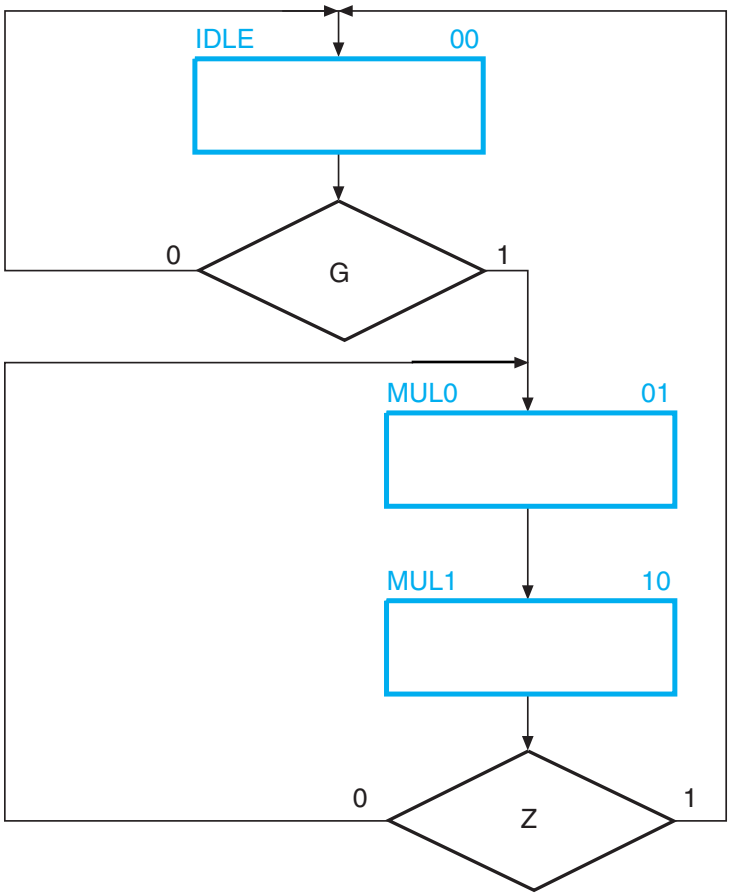


Fig. 3-11 Sequencing Part of ASM Chart for the Binary Multiplier

3 ESTADOS:

IDLE
MUL0
MUL1

2 ENTRADAS:

G
Z

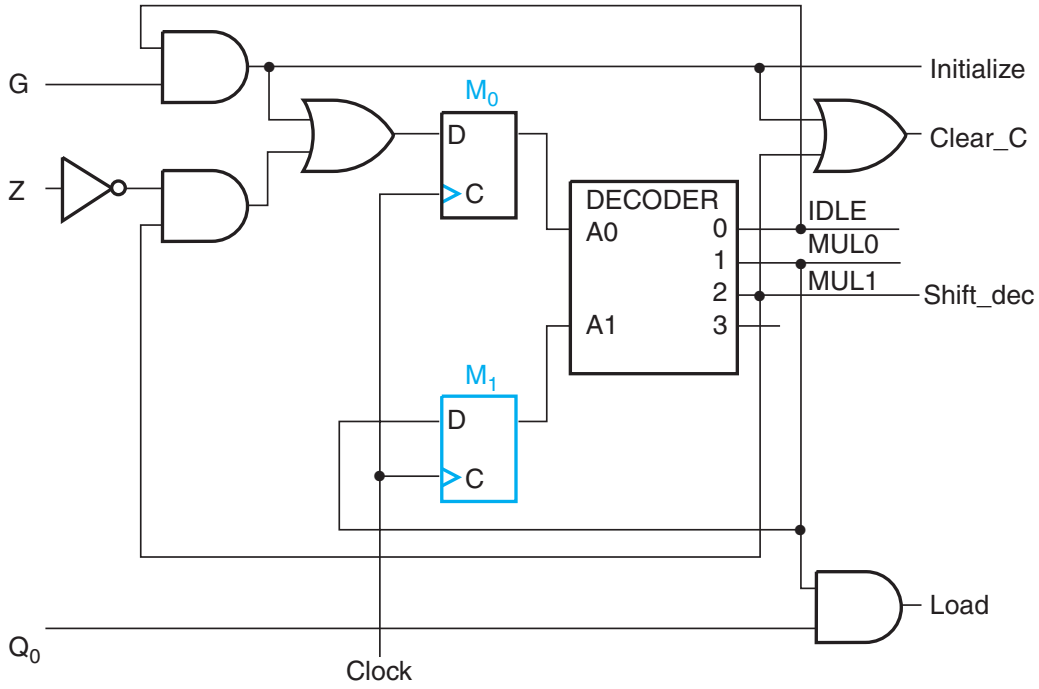
2 FLIP-FLOPS + 1 DECOD (2 IN - 4 OUT)

Present state			Inputs		Next state		Decoder Outputs		
Name	M_1	M_0	G	Z	M_1	M_0	IDLE	MUL0	MUL1
IDLE	0	0	0	×	0	0	1	0	0
	0	0	1	×	0	1	1	0	0
MUL0	0	1	×	×	1	0	0	1	0
MUL1	1	0	×	0	0	1	0	0	1
	1	0	×	1	0	0	0	0	1
—	1	1	×	×	×	×	×	×	×

Table 3-2 State Table for Sequence Register and Decoder Part of Multiplier Control Unit

$D_{M1} = MUL0$

$D_{M0} = IDLE*G + MUL1*\overline{Z}$



$D_{M1} = MUL0$

$D_{M0} = IDLE*G + MUL1*\overline{Z}$

Initialize	$IDLE \cdot G$
Load	$MUL0 \cdot Q_0$
Shift_dec	$MUL1$
Clear_C	$IDLE \cdot G + MUL1$

Fig. 3-12 Control Unit for Binary Multiplier Using a Sequence Register and a Decoder

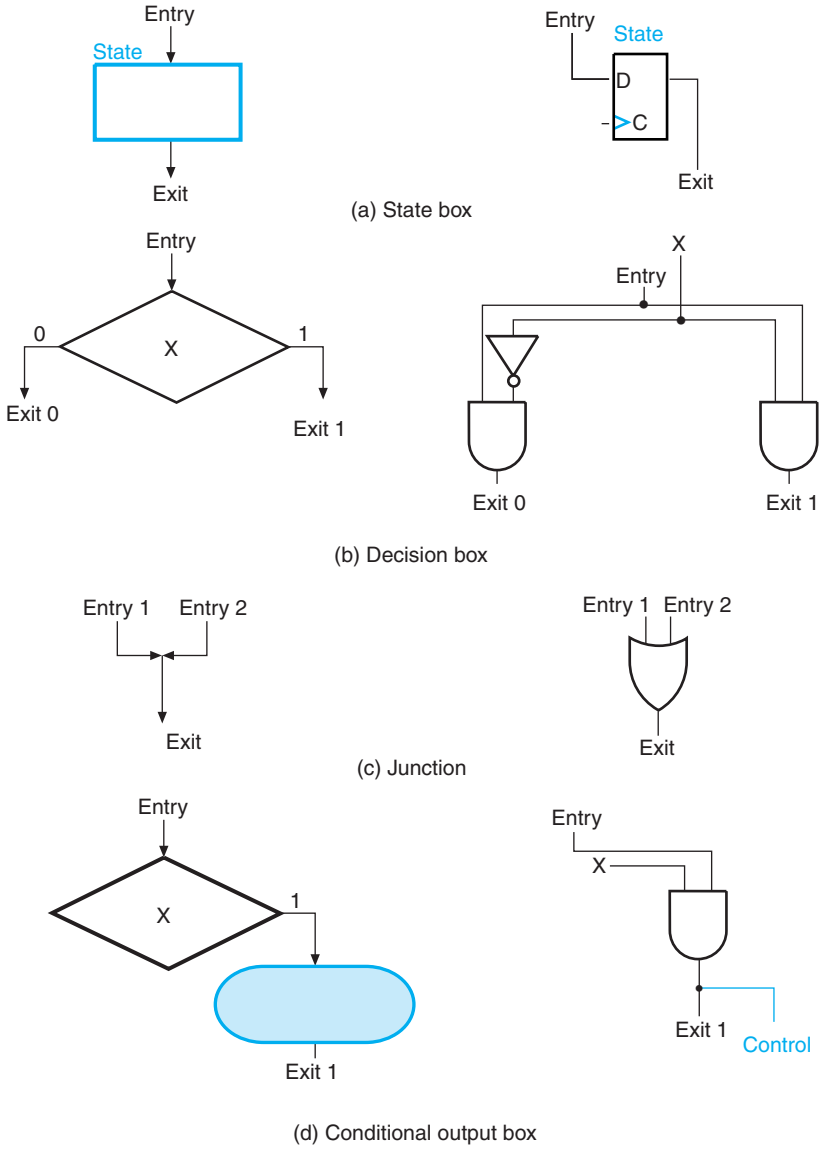
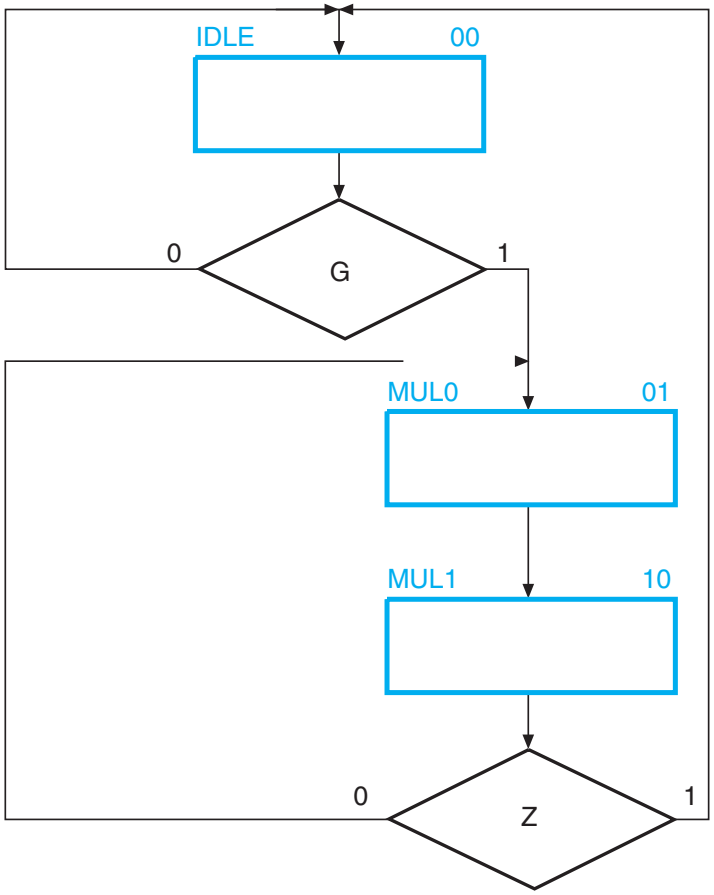


Fig. 3-13 Transformation Rules for Control Unit with One Flip-Flop per State



Control Signal Name	Control Expression
Initialize	$\text{IDLE} \cdot G$
Load	$\text{MUL0} \cdot Q$
Shift_dec	MUL1
Clear_C	$\text{IDLE} \cdot G + \text{MUL1}$

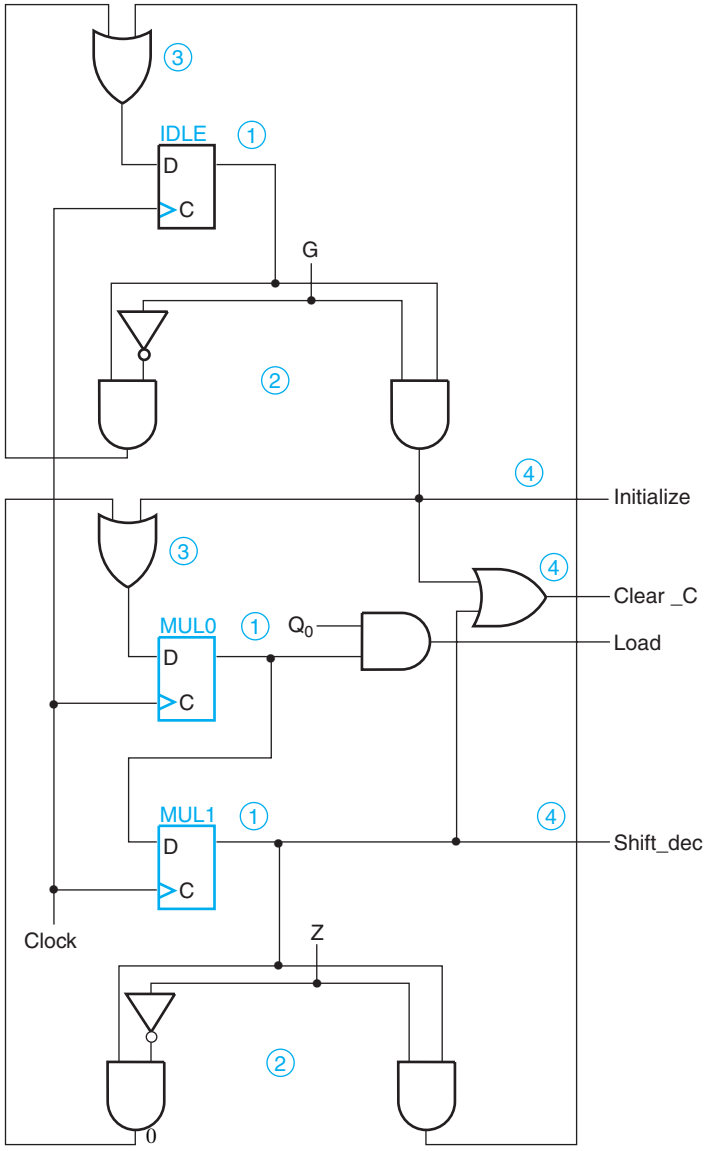
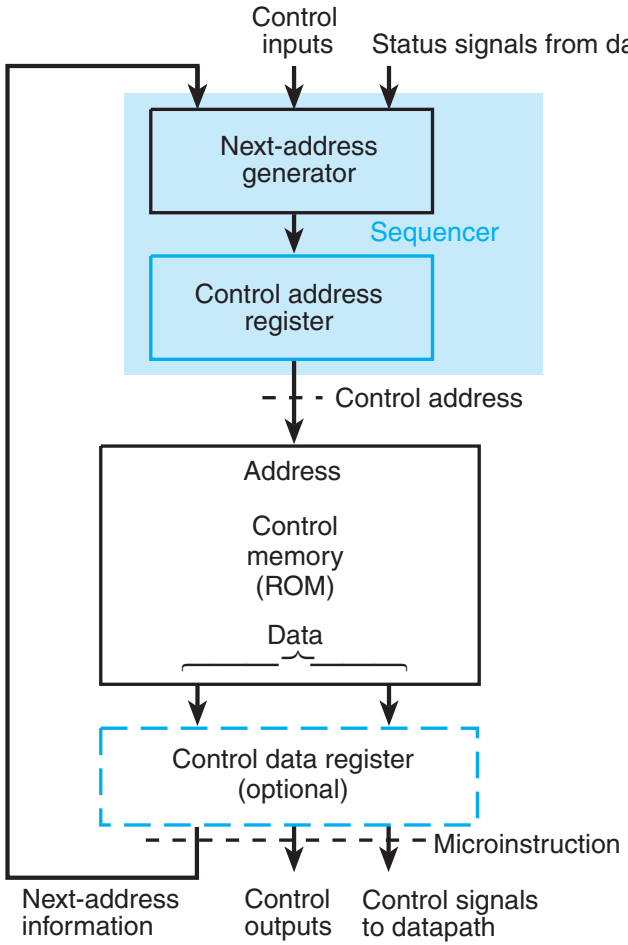


Fig. 3-14 Control Unit with One Flip-Flop per State for the Binary Multiplier

INICIALIZACION

UNIDAD DE CONTROL MICROPROGRAMADA

- aquella cuyos valores de control binario se almacenan como palabras en memoria
- cada palabra contiene una microinstruccion
- secuencia de microinstrucciones constituye un microprograma



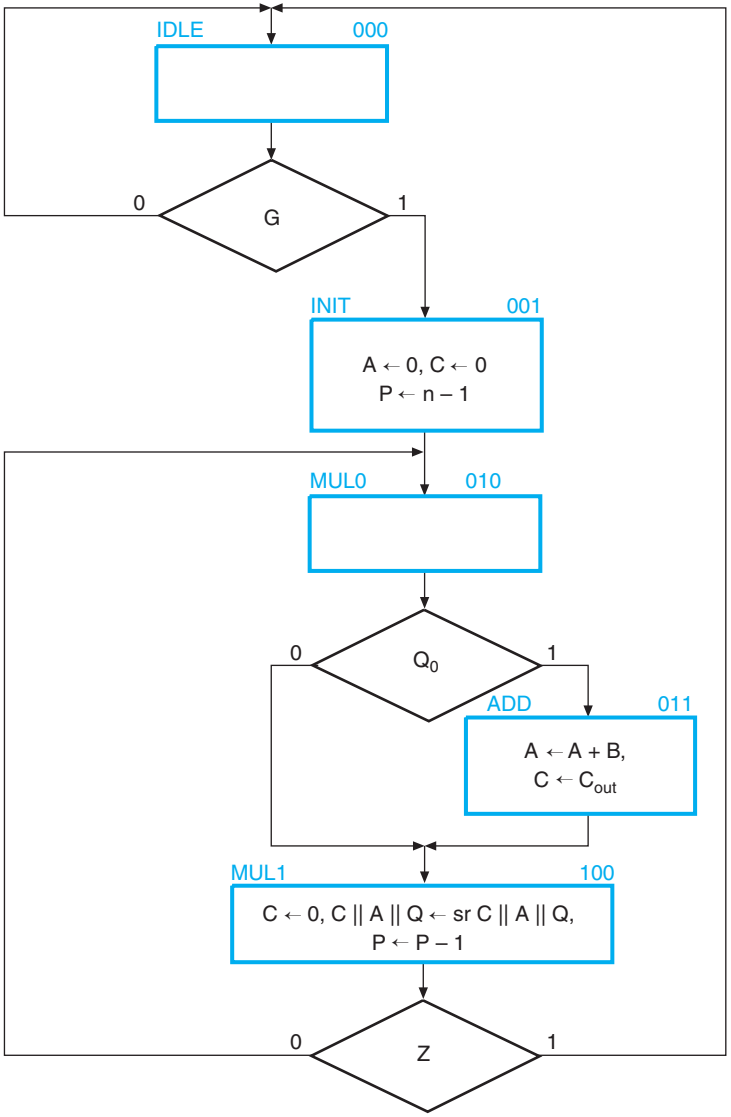
CAR - variables de estado

ROM (circ comb 2)

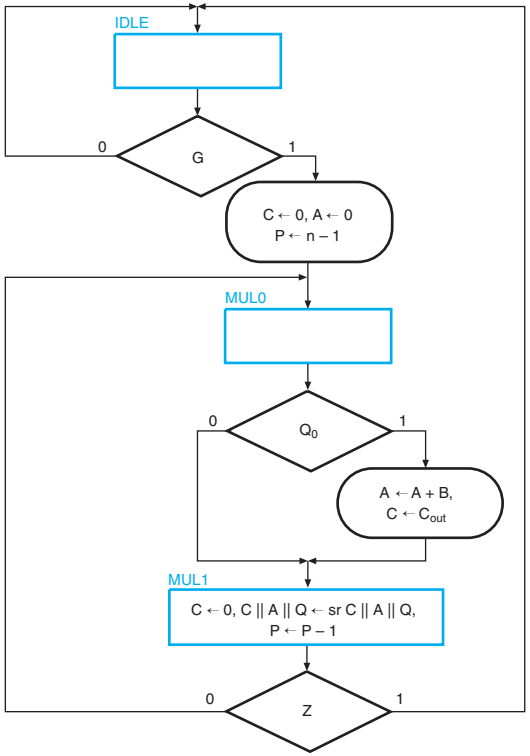
ROM + Next-address generator (circ comb 1)

Automata de Moore

Fig. 3-15 Microprogrammed Control Unit Organization

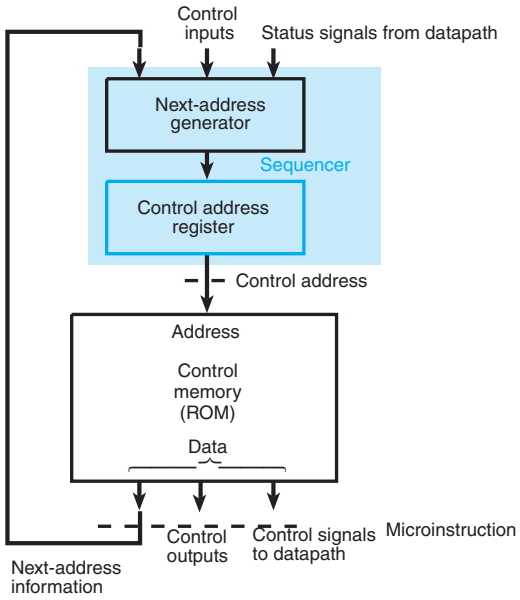


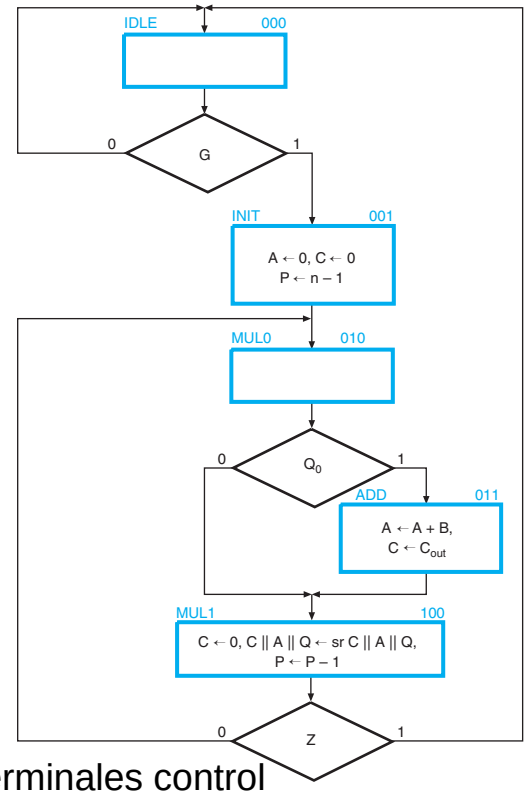
Automata de Moore



Automata de Mealy

Fig. 3-16 ASM Chart for Microprogrammed Binary Multiplier Control Unit





numero direcciones ROM = numero microinstrucciones = numero estados

numero bits / direccion = palabra control (microinstruccion) = secuenciacion + terminales control

TABLE 3-3
Control Signals for Microprogrammed Multiplier Control

Control Signal	Register Transfers	States in Which Signal is Active	Micro-instruction Bit Position	Symbolic Notation
Initialize	$A \leftarrow 0, P \leftarrow n-1$	INIT	0	IT
Load	$A \leftarrow A+B, C \leftarrow C_{\text{out}}$	ADD	1	LD
Clear_C	$C \leftarrow 0$	INIT, MUL1	2	CC
Shift_dec	$C \Vert A \Vert Q \leftarrow \text{sr } C \Vert A \Vert Q, P \leftarrow P-1$	MUL1	3	SD

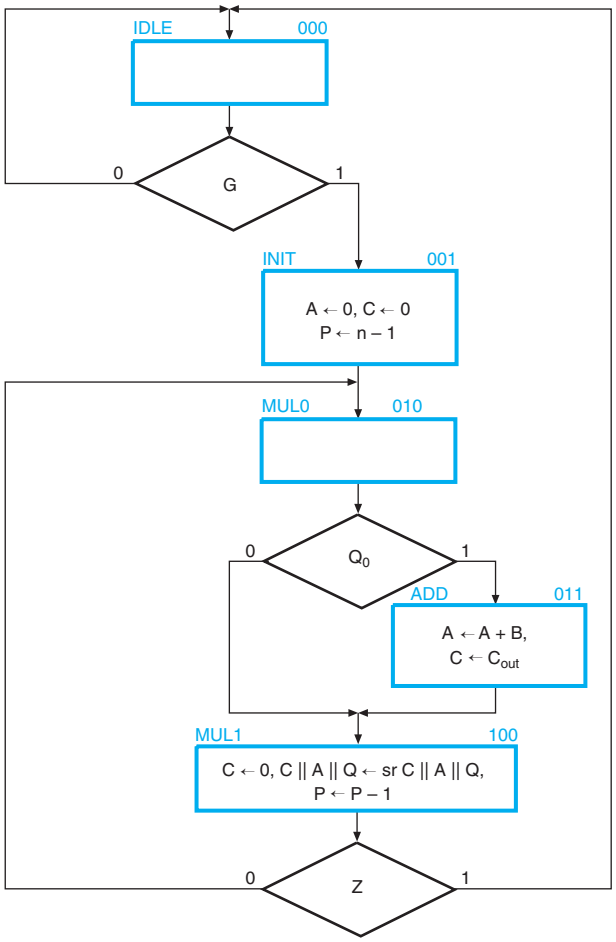
Table 3-3 Control Signals for Microprogrammed Multiplier Control

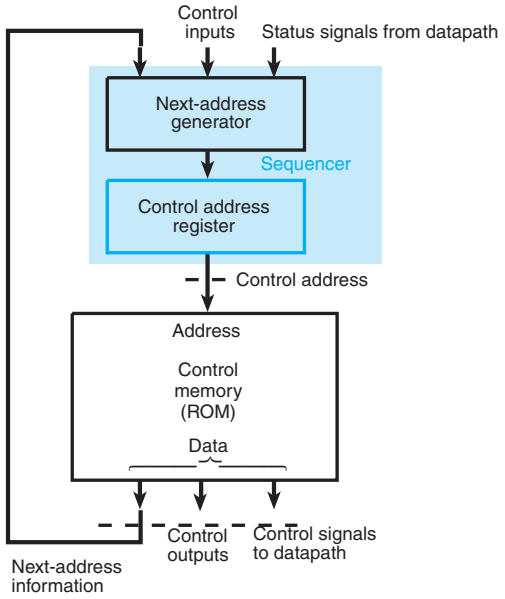
Fig. 4-17 Microinstruction Control Word Format

TABLE 3-4
SEL Field Definition for Binary Multiplier Control Sequencing

SEL		
Symbolic notation	**Binary Code**	**Sequencing Microoperations**
NXT	00	$CAR \leftarrow NXTADD0$
DG	01	$\overline{G}$: $CAR \leftarrow NXTADD0$ G: $CAR \leftarrow NXTADD1$
DQ	10	$\overline{Q_0}$: $CAR \leftarrow NXTADD0$ Q_0: $CAR \leftarrow NXTADD1$
DZ	11	$\overline{Z}$: $CAR \leftarrow NXTADD0$ Z: $CAR \leftarrow NXTADD1$

Table 3-4 SEL Field Definition for Binary Multiplier Control Sequencing





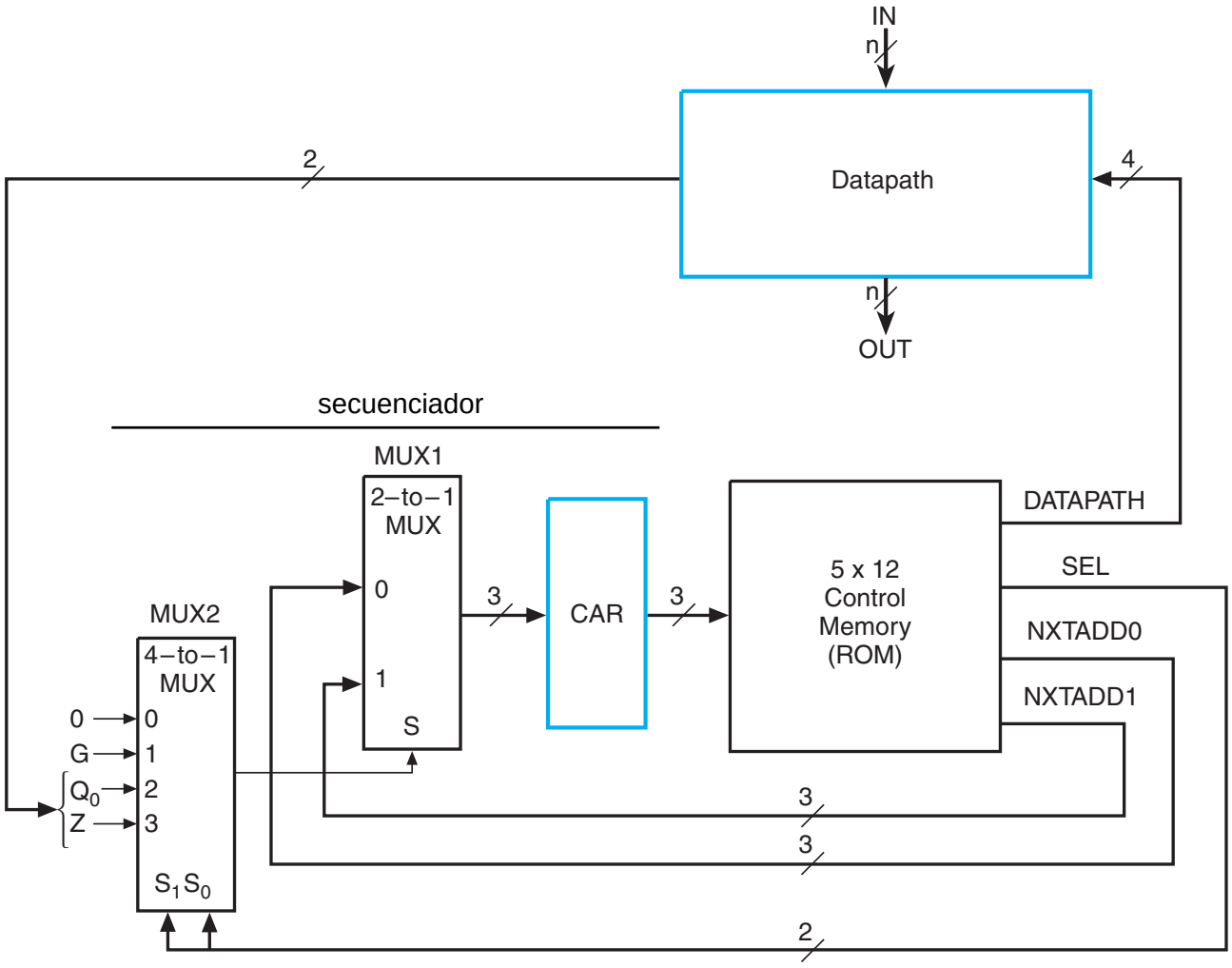


Fig. 3-18 Microprogrammed Control Unit for Multiplier

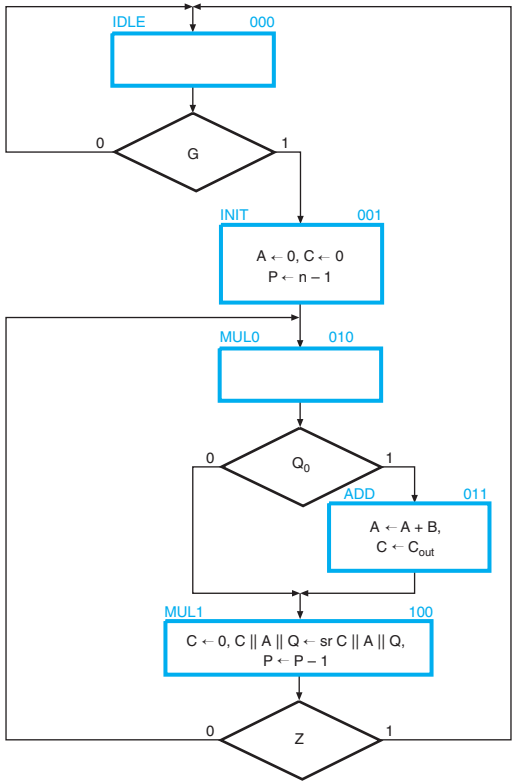


TABLE 3-5
Register Transfer Description of Binary Multiplier Microprogram

Address	Symbolic transfer statement	
	Control signals	Next State
IDLE		G: $CAR \leftarrow$ INIT, $\overline{G}$: $CAR \leftarrow$ IDLE
INIT	$C \leftarrow 0, A \leftarrow 0, P \leftarrow n-1$	$CAR \leftarrow$ MUL0
MUL0		Q_0: $CAR \leftarrow$ ADD, $\overline{Q}_0$: $CAR \leftarrow$ MUL1
ADD	$A \leftarrow A + B, C \leftarrow C_{out}$	$CAR \leftarrow$ MUL1
MUL1	$C \leftarrow 0, C \Vert A \Vert Q \leftarrow$ sr $C \Vert A \Vert Q$ $P \leftarrow P - 1$	Z: $CAR \leftarrow$ IDLE, $\overline{Z}$: $CAR \leftarrow$ MUL0,

Table 3-5 Register Transfer Description of Binary Multiplier Microprogram

TABLE3-6
Symbolic Microprogram and Binary Microprogram for Multiplier

Address	NXTADD1	NXTADD0	SEL	DATAPATH	Address	NXTADD1	NXTADD0	SEL	DATAPATH
IDLE	INIT	IDLE	DG	None	000	001	000	01	0000
INIT	—	MUL0	NXT	IT, CC	001	000	010	00	0101
MUL0	ADD	MUL1	DQ	None	010	011	100	10	0000
ADD	—	MUL1	NXT	LD	011	000	100	00	0010
MUL1	IDLE	MUL0	DZ	CC, SD	100	000	010	11	1100

Table 3-6 Symbolic Microprogram and Binary Microprogram for Multiplier

ARQUITECTURA DE UN ORDENADOR SENCILLO

OBJETIVO: Estudio de la arquitectura de un ordenador sencillo como base para ilustrar distintos enfoques del diseæo de control

DEFINICIONES:

INSTRUCCION u OPERACION: Conjunto de bits que ordena a la computadora ejecutar una operacion especifica

PROGRAMA: Lista de instrucciones que especifica las operaciones, operandos y secuencia del proceso

PROGRAMA + DATOS (operandos, intermedios y resultado) se guardan en una memoria RAM

UNIDAD DE CONTROL: Lee cada instruccion, la decodifica y ejecuta generando una secuencia de microoperaciones (palabras de control) sobre la ruta de datos

CAMPOS: La instruccion se divide en campos (subgrupos de bits) que especifican entre otras cosas:

CODIGO DE OPERACION (OPCODE) - Instruccion - n bits => 2^n operaciones

OPERANDOS - Se pueden definir:

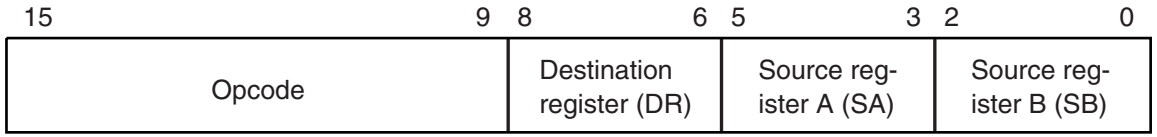
Explicitamente - Aparecen en los codigos de operandos de la instruccion. I.e. D <- A+B

Implicitamente - Se deducen del mismo codigo de operacion. I.e. A <- A+1

ARCHIVO DE REGISTROS - 8 REGISTROS => 3 bits campo operandos

FORMATO REGISTRO

DR <- SA +SB
M[SA] <- SB

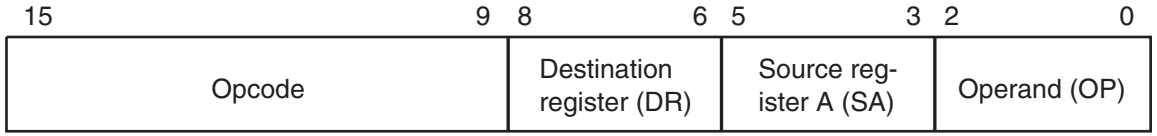


(a) Register

FORMATO INMEDIATO

Operando inmediato = cte

DR <- SA + OP

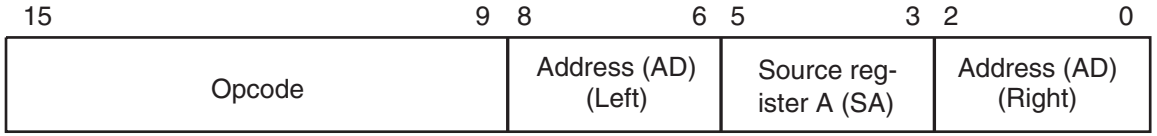


(b) Immediate

FORMATO BIFURCACION

AD - offset de direccion complemento a dos

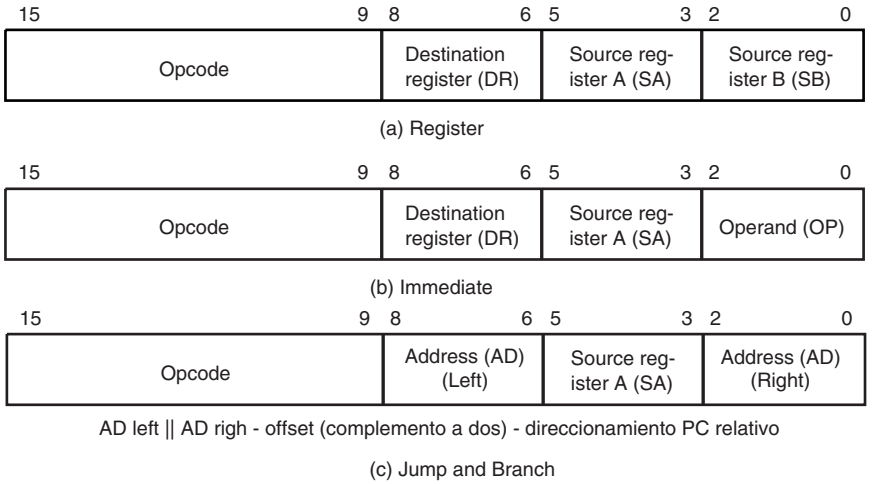
if SA=0 PC <- PC+AD



AD left || AD righ - offset (complemento a dos) - direccionamiento PC relativo

(c) Jump and Branch

Fig. 3-19 Three Instruction Formats



EJEMPLO

MEMORIA: 16 bits/palabra

INSTRUCCION: 16 bits (OPCODE - 7 bits) => 1 instruccion / posicion memoria

Normalmente 1 instruccion = 32 o 64 bits => 2 o 4 direcciones de memoria

Decimal address	Memory contents	Decimal opcode	Other specified fields	Operation
25	0000101 001 010 011	5 (Subtract)	DR:1, SA:2 SB:3	R1 ← R2 − R3
35	0100000 000 100 101	32 (Store)	SA:4 SB:5	M [R4] ← R5
45	1000010 010 111 011	66 (Add Immediate)	DR:2 SA:7 OP:3	R2 ← R7 + 3
55	1100011 101 110 100	96 (Branch on zero)	AD: 44 SA:6	If R6 = 0, PC ← PC − 20
70	0000000 011 000 000	Data = 192. After execution of instruction in 35, Data = 80.		

(R4 = 70, R5 = 80)

Fig. 3-20 Memory Representation of Instructions and Data

OPERACION DE ORDENADOR v.s. MICROOPERACION DE CONTROLADOR

OPERACIONES ORDENADOR

- Se almacenan en memoria principal (RAM)
- Contador de Programa (PC) accede a ellas y las transfiere al controlador
- Constan de 1 o varias microoperaciones

MICROOPERACIONES CONTROLADOR

- Se almacenan en la memoria de control (ROM) o se implementan con puertas logicas
- La unidad de control las ejecuta
- Tienen una duracion de 1 ciclo de reloj

EJEMPLOS:

1. RUTA DE DATOS + CONTROL CABLEADO (ciclo sencillo)
2. RUTA DE DATOS PIPELINE + CONTROL PIPELINE
3. RUTA DE DATOS + CONTROL MICROPROGRAMADO (ciclos multiples)

INSTRUCCIONES: 3 formatos diferentes mostrados anteriormente

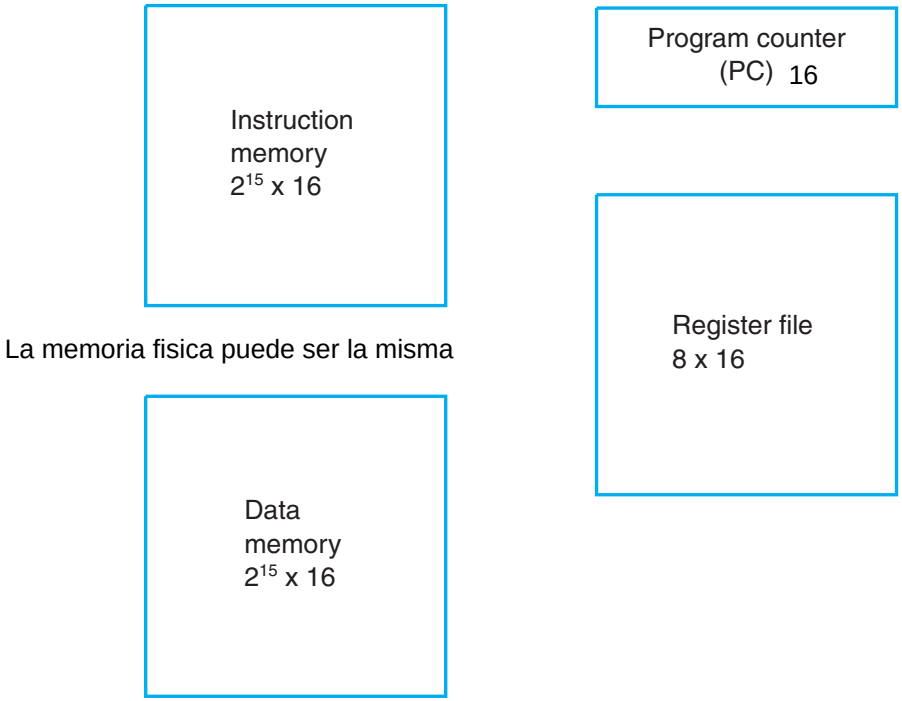
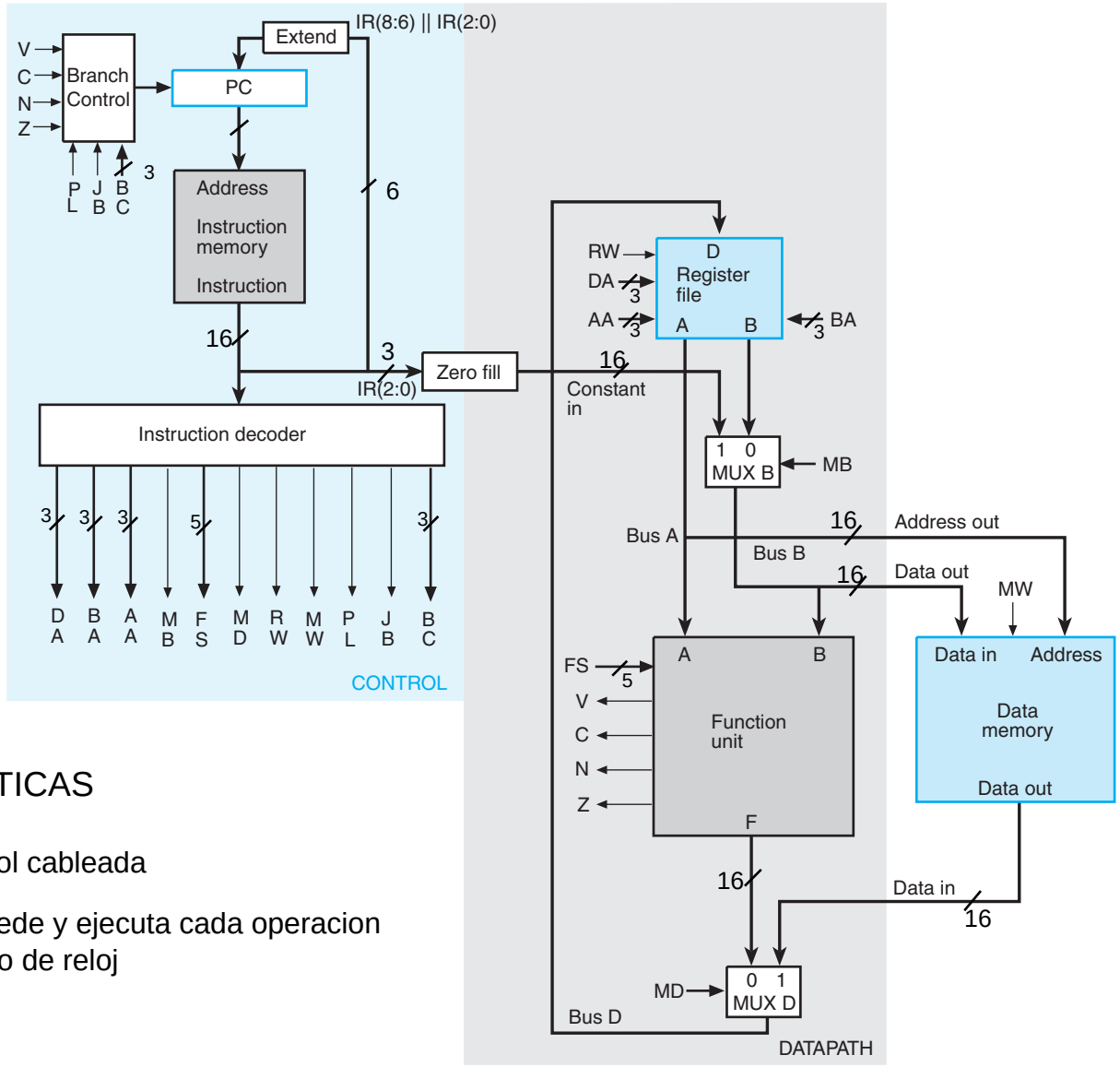


Fig. 3-21 Storage Resource Diagram for a Simple Computer

1 - CONTROL CABLEADO DE CICLO SENCILLO



CARACTERISTICAS

- Unidad de control cableada
- Controlador accede y ejecuta cada operacion en un unico ciclo de reloj

Fig. 3-22 Block Diagram for a Single-Cycle Computer

PL	JB	
0	*	-> no jump or branch instruction
1	1	-> unconditional jump (relative PC)
1	0	-> conditional jump (state bits)

TABLE 3-7
Truth Table for Instruction Decoder Logic

	Instruction Bits			Control Word Bits					
Instruction Function Type	**Bit 15**	**Bit 14**	**Bit 13**	**MB**	**MD**	**RW**	**MW**	**PL**	**JB**
ALU function using registers	0	0	0	0	0	1	0	0	X
Shifter function using registers	0	0	1	0	0	1	0	0	X
Memory write using register data	0	1	0	0	X	0	1	0	X
Memory read using register data	0	1	1	0	1	1	0	0	X
ALU operation using a constant	1	0	0	1	0	1	0	0	X
Shifter function using a constant	1	0	1	1	0	1	0	0	X
Conditional Branch	1	1	0	X	X	0	0	1	0
Unconditional Jump	1	1	1	X	X	0	0	1	1

FS = 00000
Para transferir la palabra que condiciona el salto

Table 3-7 Truth Table for Instruction Decoder Logic

Bit 11	Bit 10	Bit 9	
0	0	0	Z
0	0	1	N
0	1	0	C
0	1	1	V
1	0	0	$\overline{Z}$
1	0	1	$\overline{N}$
1	1	0	$\overline{C}$
1	1	1	$\overline{V}$

$RW = \overline{Bit14} + \overline{Bit15} * Bit13$ $\quad$ $MB = Bit15$

$MW = \overline{Bit15} * Bit14 * \overline{Bit13}$ $\quad$ $MD = Bit14$

$PL = Bit15 * Bit14$

$JB = Bit13$

$BC2 = Bit11$ $\quad$ $BC1 = Bit10$ $\quad$ $BC0 = Bit9$

$FS4 = Bit13$ Bit13 coincide con el bit FS4 (seleccion del MUXF) de la Unidad de Funciones

$FS3 = Bit12$ $\quad$ $FS2 = Bit11$ $\quad$ $FS1 = Bit10$ $\quad$ $FS0 = Bit9$

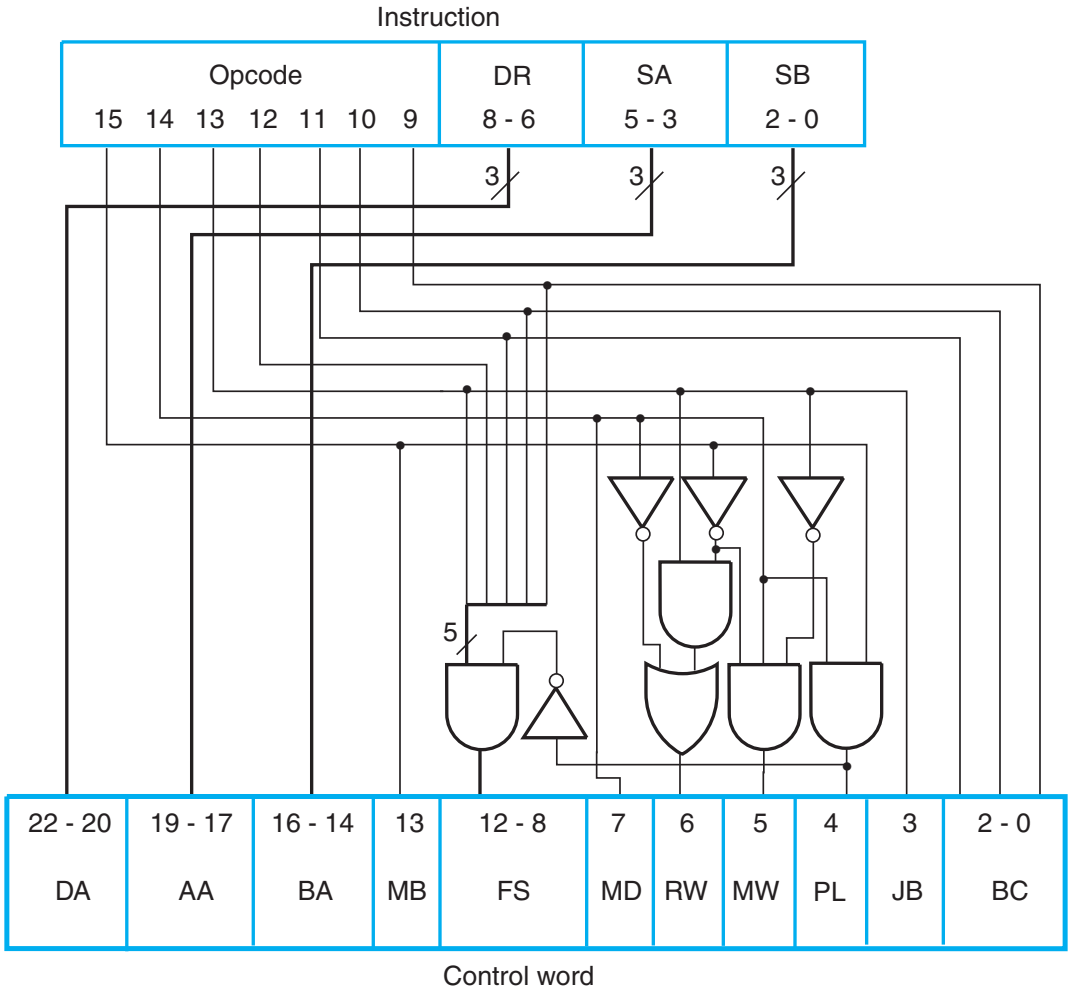


Fig. 3-23 Diagram of Instruction Decoder

Instruction Function Type	Instruction Bits			Control Word Bits					
	Bit 15	Bit 14	Bit 13	MB	MD	RW	MW	PL	JB
ALU function using registers	0	0	0	0	0	1	0	0	X
Shifter function using registers	0	0	1	0	0	1	0	0	X
Memory write using register data	0	1	0	0	X	0	1	0	X
Memory read using register data	0	1	1	0	1	1	0	0	X
ALU operation using a constant	1	0	0	1	0	1	0	0	X
Shifter function using a constant	1	0	1	1	0	1	0	0	X
Conditional Branch	1	1	0	X	X	0	0	1	0
Unconditional Jump	1	1	1	X	X	0	0	1	1

TABLE 3-8
Six Instructions for the Single-Cycle Computer

	Operation code	Symbolic name	Format	Description	Function	MB	MD	RW	MW	PL	JB
Fila 5	1000010	ADI	Immediate	Add immediate operand	$R[\text{DR}] \leftarrow R[\text{SA}] + \text{zf}\ I(2{:}0)$	1	0	1	0	0	0
Fila 4	0110000	LD	Register	Load memory content into register	$R[\text{DR}] \leftarrow M[R[\text{SA}]]$	0	1	1	0	0	1
Fila 3	0100000	ST	Register	Store register content in memory	$M[R[\text{SA}]] \leftarrow R[\text{SB}]$	0	1	0	1	0	0
Fila 2	0011000	SL	Register	Shift left	$R[\text{DR}] \leftarrow \text{sl}R[\text{SB}]$	0	0	1	0	0	1
Fila 1	0001110	NOT	Register	Complement register	$R[\text{DR}] \leftarrow \overline{R[\text{SA}]}$	0	0	1	0	0	0
Fila 7	1100000	BRZ	Jump/Branch	If R[SA] = 0, branch to PC + se AD	If R[SA] = $0, PC \leftarrow PC + \text{se}AD$, If R[SA] $\neq 0, PC \leftarrow PC + 1$	1	0	0	0	1	0

Table 3-8 Six Instructions for the Single-Cycle Computer

TABLE 3-9
Instruction Specifications for the Simple Computer

Instruction	Opcode	Mnemonic Format	Description
Move A	0000000	MOVEA RD, RA	R[DR] <- R[SA]
Increment	0000001	INC RD, RA	R[DR] <- R[SA] + 1
Add	0000010	ADD RD, RA, RB	R[DR] <- R[SA] + R[SB]
Substract	0000101	SUB RD, RA, RB	R[DR] <- R[SA] – R[SB]
Decrement	0000110	DEC RD, RA	R[DR] <- R[SA] - 1
Add and Increment	0000011	ADDINC RD, RA, RB	R[DR] <- R[SA] + R[SB] + 1
Add A+nB	0000100	ADDnB RD, RA, RB	R[DR] <- R[SA] + nR[SB]
AND	0001000	AND RD, RA, RB	R[DR] <- R[SA] ^ R[SB]
OR	0001010	OR RD, RA, RB	R[DR] <- R[SA] + R[SB]
Exclusive OR	0001100	XOR RD, RA, RB	R[DR] <- R[SA] ⊕ R[SB]
NOT	0001110	NOT RD, RA	R[DR] <- nR[SA]
Move B	0010000	MOVEB RD, RB	R[DR] <- R[SB]
Shift Right	0010100	SHR RD, RB	R[DR] <- SHR R[SB]
Shift Left	0011000	SHL RD, RB	R[DR] <- SHL R[SB]
Load	0110000	LD RD, RA	R[DR] <- M[R[SA]]
Store	0100000	ST RA, RB	M[R[SA]] <- R[SB]
Load Inmediate	1010000	LDI RD, OP	R[DR] <- OP
Add Inmediate	1000010	ADDI RD, RA, OP	R[DR] <- R[SA] + OP
Branch on Zero	1100000	BRZ RA, AD	If (R[SA] = 0) PC <- PC + se AD
Branch on Negative	1100001	BRN RA, AD	If (R[SA] < 0) PC <- PC + se AD
Branch on Carry *	1100010	BRC RA, AD	If (C = 1) PC <- PC + se AD
Branch on Overflow *	1100011	BRV RA, AD	If (V = 1) PC <- PC + se AD
Branch on non Zero	1100100	BRNZ RA, AD	If (R[SA] ≠ 0) PC <- PC + se AD
Branch on Positive	1100101	BRNN RA, AD	If (R[SA] ≥ 0) PC <- PC + se AD
Branch on non Carry *	1100110	BRNC RA, AD	If (C = 1) PC <- PC + se AD
Branch on non Overflow *	1100111	BRNV RA, AD	If (V = 1) PC <- PC + se AD
Jump	1110000	JMP OP	PC <- PC + se AD

PROGRAMA EJEMPLO

83 - (2 + 3)

Al principio de la ejecucion:

R3 = 248, M[248] = 2, M[249] = 83, M[250] <- resultado

OPCODE	DR	SA	SB	Registro que cambia
LD	R1	R3	-	R1 = 2
ADI	R1	R1	3	R1 = 5
NOT	R1	R1	-	
INC	R1	R1	-	R1 = -5
INC	R3	R3	-	R3 = 249
LD	R2	R3	-	R2 = 83
ADD	R2	R2	R1	R2 = 78
INC	R3	R3	-	R3 = 250
ST	-	R3	R2	M[250] = 78

Fig. 3-24 Example Program

Al final de la ejecucion:

R1 = -5

R2 = 78

R3 = 250

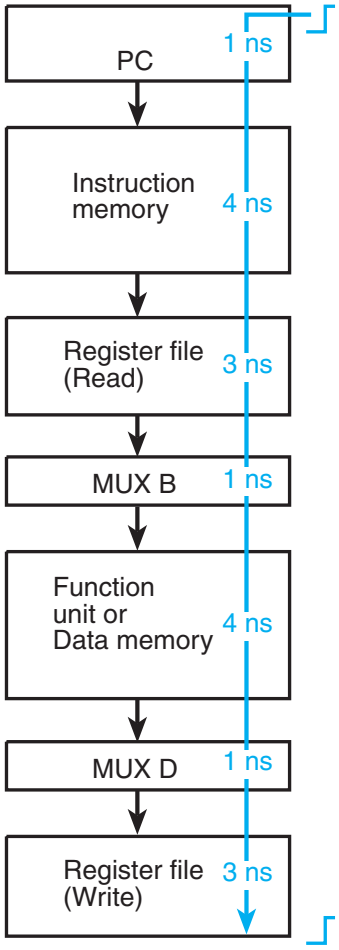
M[250] = 78

LIMITACIONES CONTROL CICLO SENCILLO

- ejecucion de operaciones complejas. I.e. Multiplicacion

- Diseæo habitual: Unica memoria que almacene instrucciones y datos
  => aumenta el numero de ciclos por instruccion

- duracion del periodo de reloj larga.



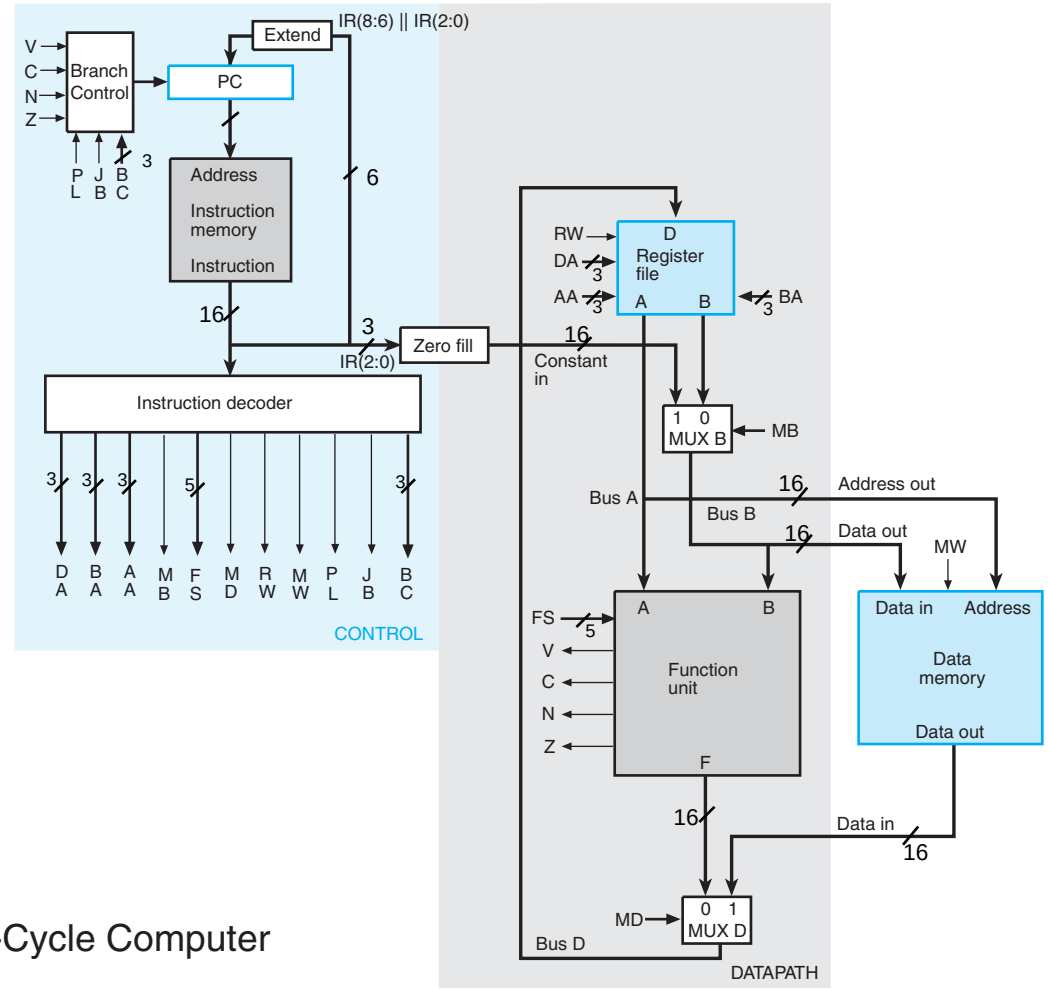
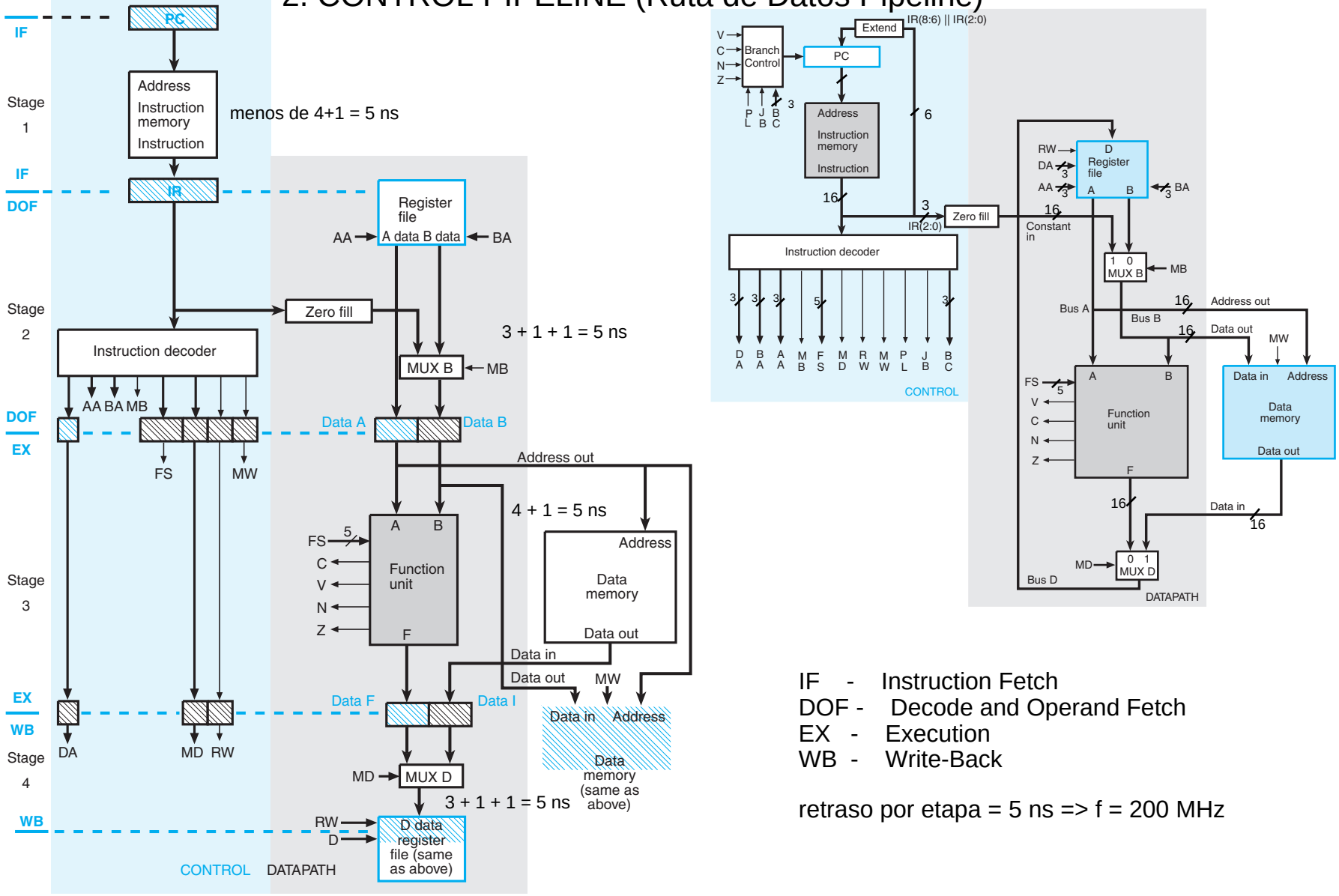


Fig. 3-25 Worst Case Delay Path in Single-Cycle Computer

Instruccion con lectura de operandos desde registro
ejecucion de instruccion en unidad de funciones y
finalmente escritura en registro

2. CONTROL PIPELINE (Ruta de Datos Pipeline)



IF - Instruction Fetch
DOF - Decode and Operand Fetch
EX - Execution
WB - Write-Back

retraso por etapa = 5 ns => f = 200 MHz

Fig. 3-27 Block Diagram of Pipelined Computer

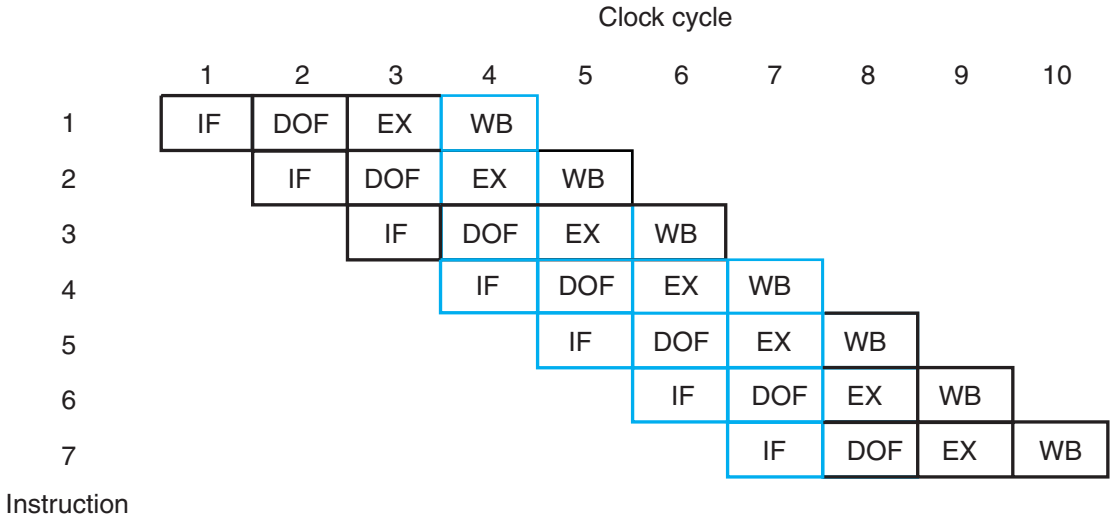


Fig. 4-27 Pipeline Execution Pattern of Register Number Program

pipeline: 7 instrucciones, 10 ciclos reloj (5 ns) = 50 ns
no pipeline: 7 instrucciones, 7 ciclos de reloj (17 ns) = 119 ns

pipeline 2.4 veces mas rapida

pipeline: 4 instrucciones, 4 ciclos (5 ns) = 20 ns
no pipeline: 4 instrucciones, 4 ciclos (17 ns) = 68 ns

pipeline 3.4 veces mas rapida

no llega a 4 por los retrasos introducidos por las plataformas

Diseæo complicado:

- Instrucciones con diferentes duraciones
- Instrucciones de salto
- Operandos que todavia no han sido calculados

3 - CONTROL MICROPROGRAMADO DE CICLOS MULTIPLES

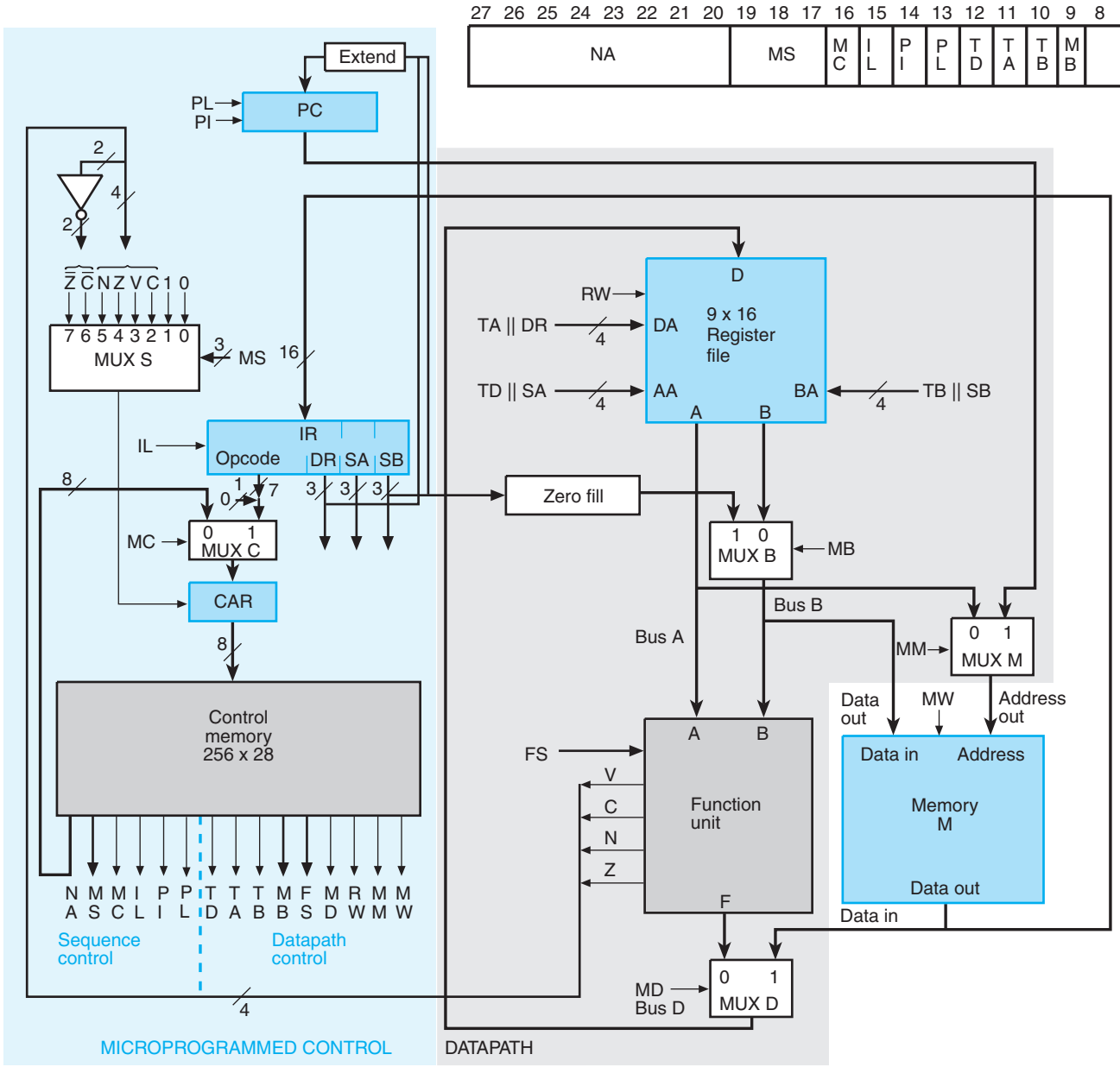


Fig. 3-28 Multiple-Cycle Microprogrammed Computer

Modificaciones sobre la de ciclo sencillo:

Memoria de instrucciones y datos:

- acceso desde memoria M a registro IR (instruction register) Terminal IL

- Mux M - accede al bus de direcciones de memoria M desde PC (Program counter) o desde archivo registros terminal MM

Archivo de registros:

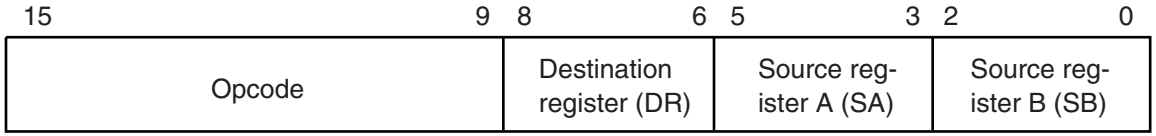
- registro oculto R8 (para instrucciones de ciclo multiple) Terminales TA, TB y TD

Unidad de Control:

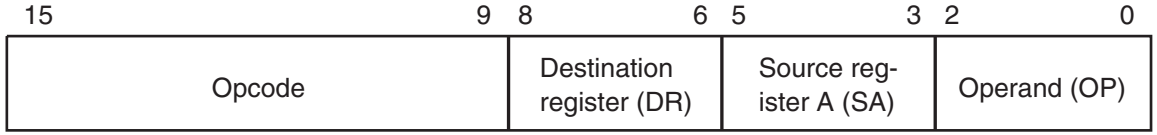
- contador del programa PC(direccion siguiente instruccion) Terminales PL (load), PI (increment)

- registro de instruccion IR (instruccion que se ejecuta) Terminal IL (Load) (ya indicado)

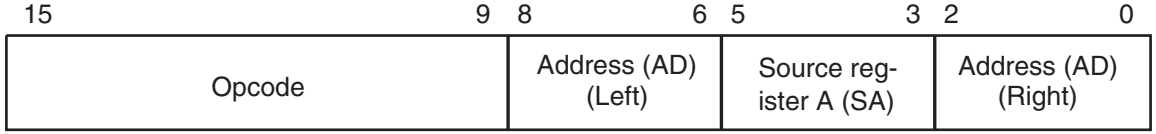
- CAR - es un contador que se incrementa (bit mas signif) o se carga con la salida de MUX C Terminales - MC, MS (3 bits)



(a) Register



(b) Immediate



AD left || AD righ - offset (complemento a dos) - direccionamiento PC relativo

(c) Jump and Branch

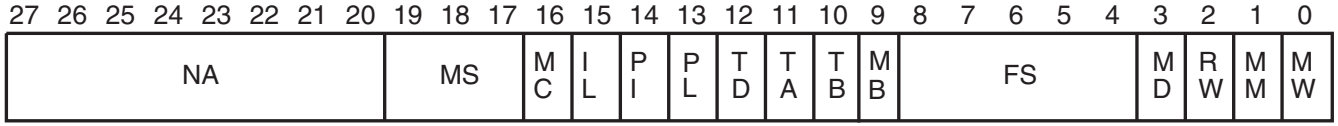


Fig. 3-29 Format for Microinstruction

TABLE 4-10
Control Word Information for Datapath

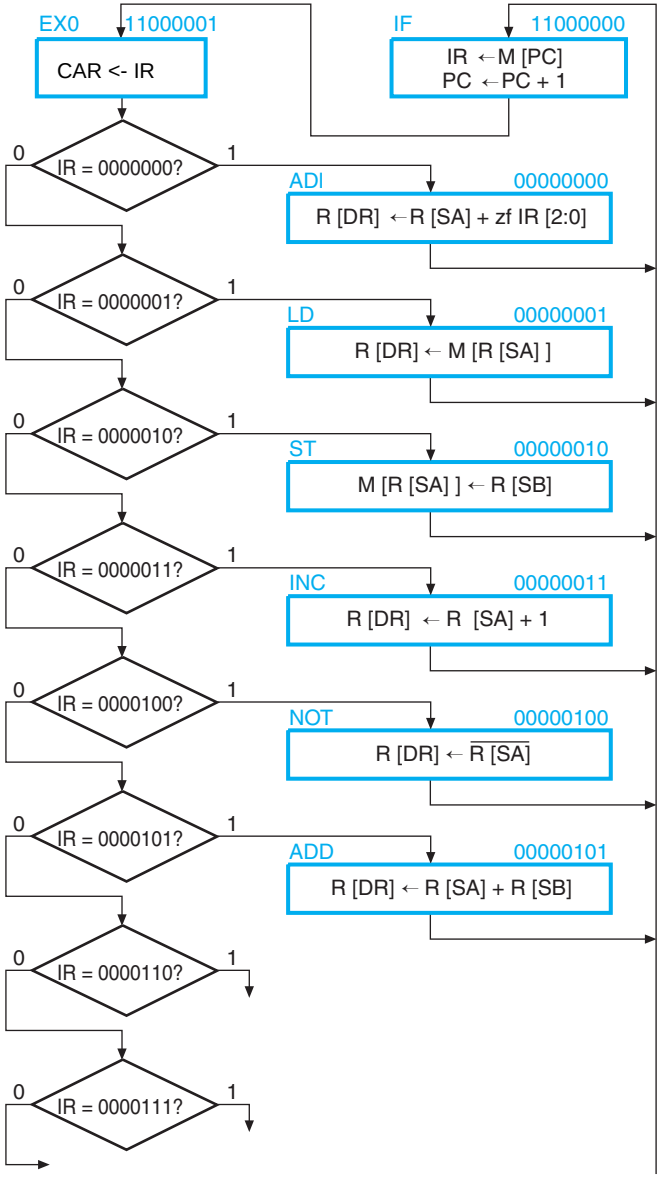
TD	TA	TB	MB		FS		MD	RW	MM	MW	
Select	Select	Select	Select	Code	Function	Code	Select	Function	Select	Function	Code
R[DR]	R[SA]	R[SB]	Register	0	$F = A$	00000	FnUt	No write (NW)	Address	No write (NW)	0
R8	R8	R8	Constant	1	$F = A + 1$	00001	Data In	Write (WR)	PC	Write (WR)	1
					$F = A + B$	00010					
					$F = A + B + 1$	00011					
					$F = A + \overline{B}$	00100					
					$F = A + \overline{B} + 1$	00101					
					$F = A - 1$	00110					
					$F = A$	00111					
					$F = A \wedge B$	01000					
					$F = A \vee B$	01010					
					$F = A \oplus B$	01100					
					$F = \overline{A}$	01110					
					$F = B$	10000					
					$F = \text{sr } B$	10100					
					$F = \text{sl } B$	11000					

Table 3-9 Control Word Information for Datapath

TABLE 3-11
Control Information for Sequence Control Fields

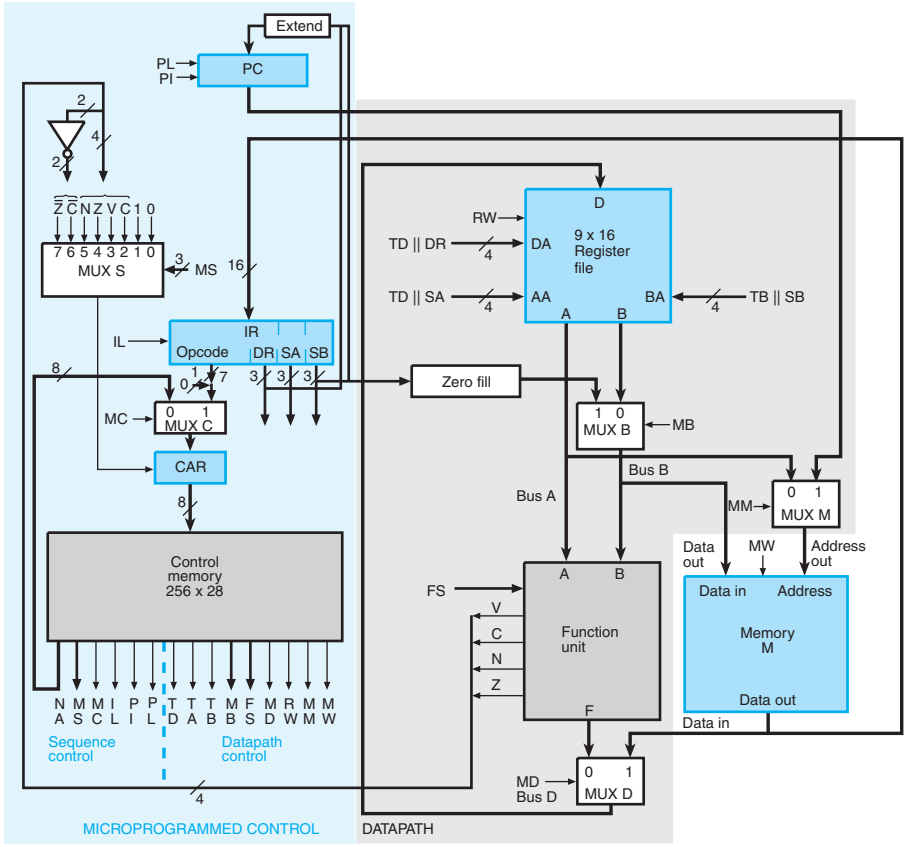
MS			MC		IL		PI		PL		
Action	Symbolic Notation	Code	Select	Symbolic Notation	Action	Symbolic Notation	Action	Symbolic Notation	Action	Symbolic Notation	Code
Increment *CAR*	CNT	000	NA	NXA	No load	NLI	No load	NLP	No load	NLP	0
Load *CAR*	NXT	001	Opcode	OPC	Load instr.	LDI	Increment PC	INP	Load PC	LDP	1
If $C = 1$, load *CAR*; else increment *CAR*	BC	010									
If $V = 1$, load *CAR*; else increment CAR	BV	011									
If $Z = 1$, load *CAR*; else increment *CAR*	BZ	100									
If $N = 1$, load *CAR*; else increment *CAR*	BN	101									
If $C = 0$, load *CAR*; else increment *CAR*	BNC	110									
If $Z = 0$, load *CAR*, else increment *CAR*	BNZ	111									

Table 3-10 Control Information for Sequence Control Fields



1 - Busqueda Instruccion - estados IF, EX0

2 - Ejecucion Instruccion (microprograma)



el primer estado de la microinstruccion (CAR) coincide con 0 + OPCODE

Fig. 3-30 ASM Diagram of the Multiple-Cycle Microprogrammed Computer

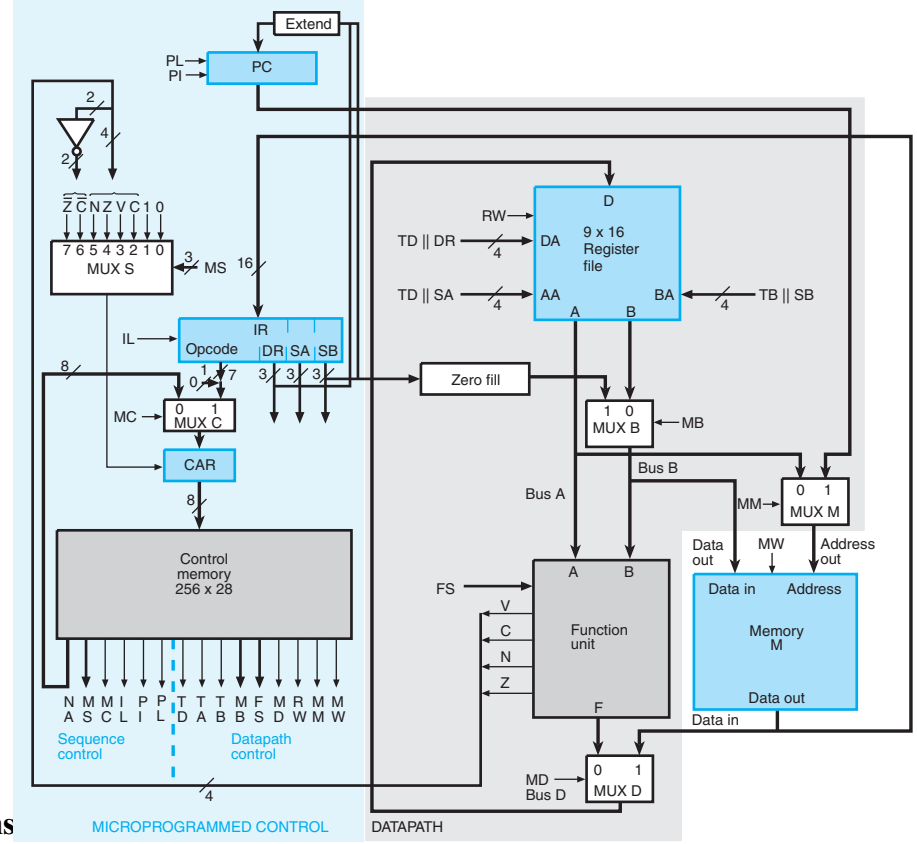


TABLE 3-12
Symbolic Microprogram for Fetch and Execution of Six Instructions

Address	NXT ADD	MS	MC	IL	PI	PL	TD	TA	TB	MB	FS	MD	RW	MM	MW
IF	EX0	CNT	—	LDI	INP	NLP	—	—	—	—	—	—	NW	PC	NW
EXO	—	NXT	OPC	NLI	NLP	NLP	—	—	—	—	—	—	NW	—	NW
ADI	IF	NXT	NXA	NLI	NLP	NLP	DR	SA	—	Constant	$F = A + B$	FnUt	WR	—	NW
LD	IF	NXT	NXA	NLI	NLP	NLP	DR	SA	—	—	—	Data	WR	MA	NW
ST	IF	NXT	NXA	NLI	NLP	NLP	—	SA	SB	Register	—	—	NW	MA	WR
INC	IF	NXT	NXA	NLI	NLP	NLP	DR	SA	—	—	$F = A + 1$	FnUt	WR	—	NW
NOT	IF	NXT	NXA	NLI	NLP	NLP	DR	SA	—	—	$F = \overline{A}$	FnUt	WR	—	NW
ADD	IF	NXT	NXA	NLI	NLP	NLP	DR	SA	SB	Register	$F = A + B$	FnUt	WR	—	NW

Table 3-11 Symbolic Microprogram for Fetch and Execution of Six Instructions

Address	NXT ADD	MS	MC	IL	PI	PL	TD	TA	TB	MB	FS	MD	RW	MM	MW
IF	EX0	CNT	—	LDI	INP	NLP	—	—	—	—	—	—	NW	PC	NW
EXO	—	NXT	OPC	NLI	NLP	NLP	—	—	—	—	—	—	NW	—	NW
ADI	IF	NXT	NXA	NLI	NLP	NLP	DR	SA	—	Constant	$F = A + B$	FnUt	WR	—	NW

TABLE 3-13
Binary Microprogram for Fetch and Execution of Six Instructions

Address	NXT ADD	MS	MC	IL	PI	PL	TD	TA	TB	MB	FS	MD	RW	MM	MW
192	193	000	0	1	1	0	0	0	0	0	00000	0	0	1	0
193	000	001	1	0	0	0	0	0	0	0	00000	0	0	0	0
000	192	001	0	0	0	0	0	0	0	1	00010	0	1	0	0
001	192	001	0	0	0	0	0	0	0	0	00000	1	1	0	0
002	192	001	0	0	0	0	0	0	0	0	00000	0	0	0	1
003	192	001	0	0	0	0	0	0	0	0	00001	0	1	0	0
004	192	001	0	0	0	0	0	0	0	0	01110	0	1	0	0
005	192	001	0	0	0	0	0	0	0	0	00010	0	1	0	0

Table 3-12 Binary Microprogram for Fetch and Execution of Six Instructions

Cada instrucción dura 3 ciclos de reloj:

1.- Carga en IR (IF)
2.- Decodificacion Instruccion (EX0)
3.- Ejecucion de la operacion

EJEMPLO: Instruccion con ejecucion en 2 ciclos de reloj

R[DR] <- M [M [R[SA]]] direccionamiento indirecto

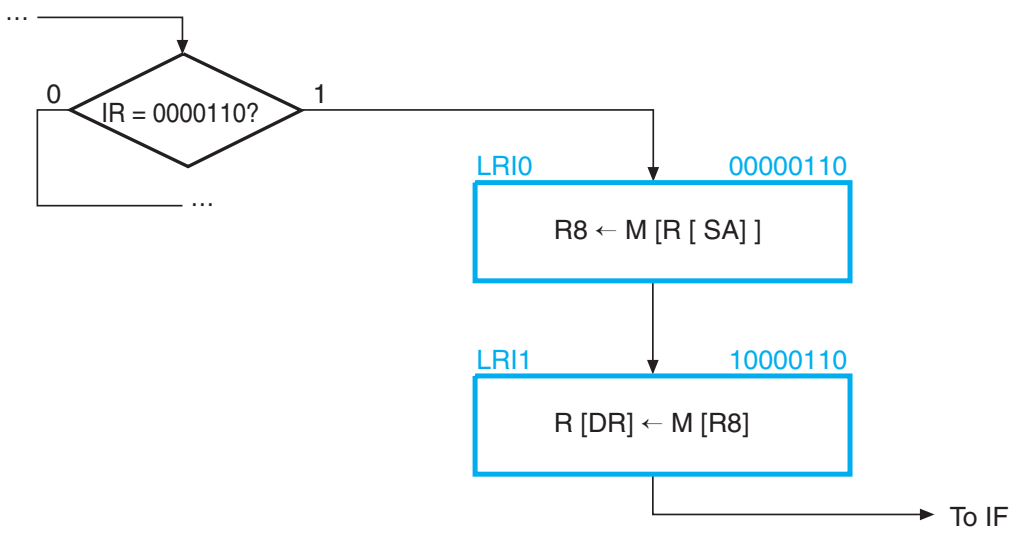
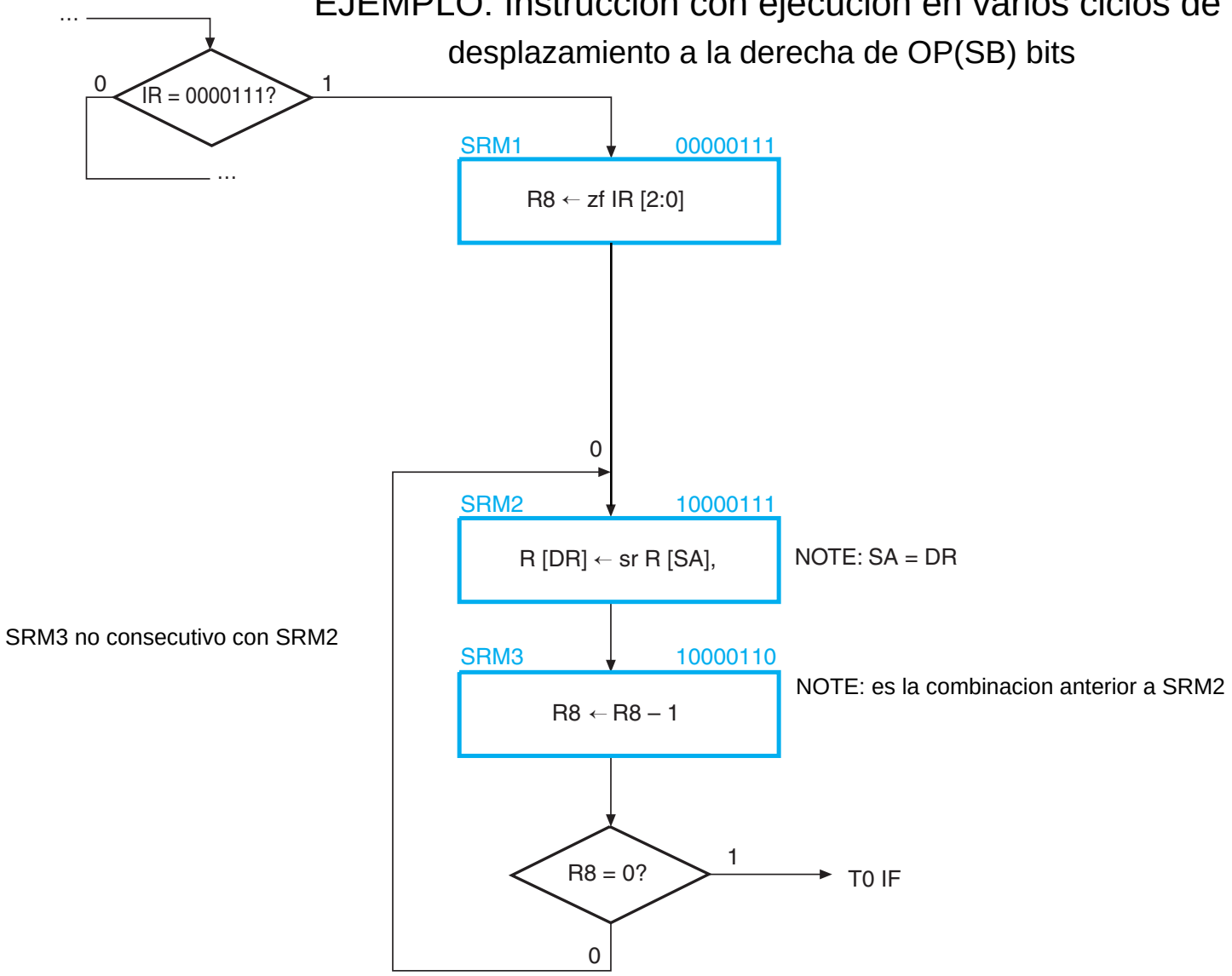


Fig. 3-31 ASM Chart for Register Indirect Instruction

EJEMPLO: Instruccion con ejecucion en varios ciclos de reloj

desplazamiento a la derecha de OP(SB) bits



La instrucción no contempla desplazamiento de 0 bits a la derecha. OP debe ser distinto de 0

Fig. 3-32 ASM Chart for Right-Shift Multiple Instruction

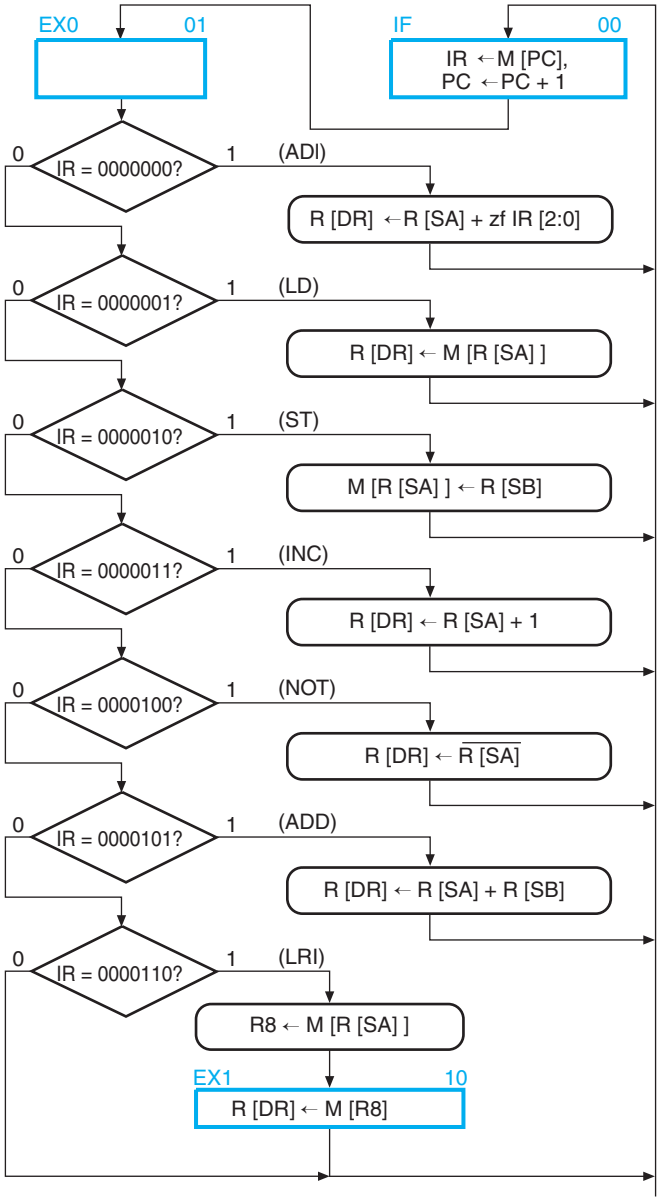


Fig. 3-33 ASM Chart for Multiple-Cycle, Decoder-Based Computer

Alternativa Cableada

solo implementa las 6 instrucciones de ciclo sencillo del ejemplo

y la de ciclo multiple de direccionamiento indirecto

Modificaciones:

- sustituyo CAR por un contador de 4 estados
  sustituyo MS y MC por el terminal CR (puesta a 0)
  codifico estado IF obligatoriamente como 00

- como no hay instrucciones de salto de programa
  elimino terminal PL (carga) del contador

- No hay transferencia de IR a CAR -
  Cada instruccion dura un ciclo de reloj menos ya que en el estado EX0 se decodifica y ejecuta la instruccion de ciclo sencillo o el primer ciclo de la de ciclo multiple

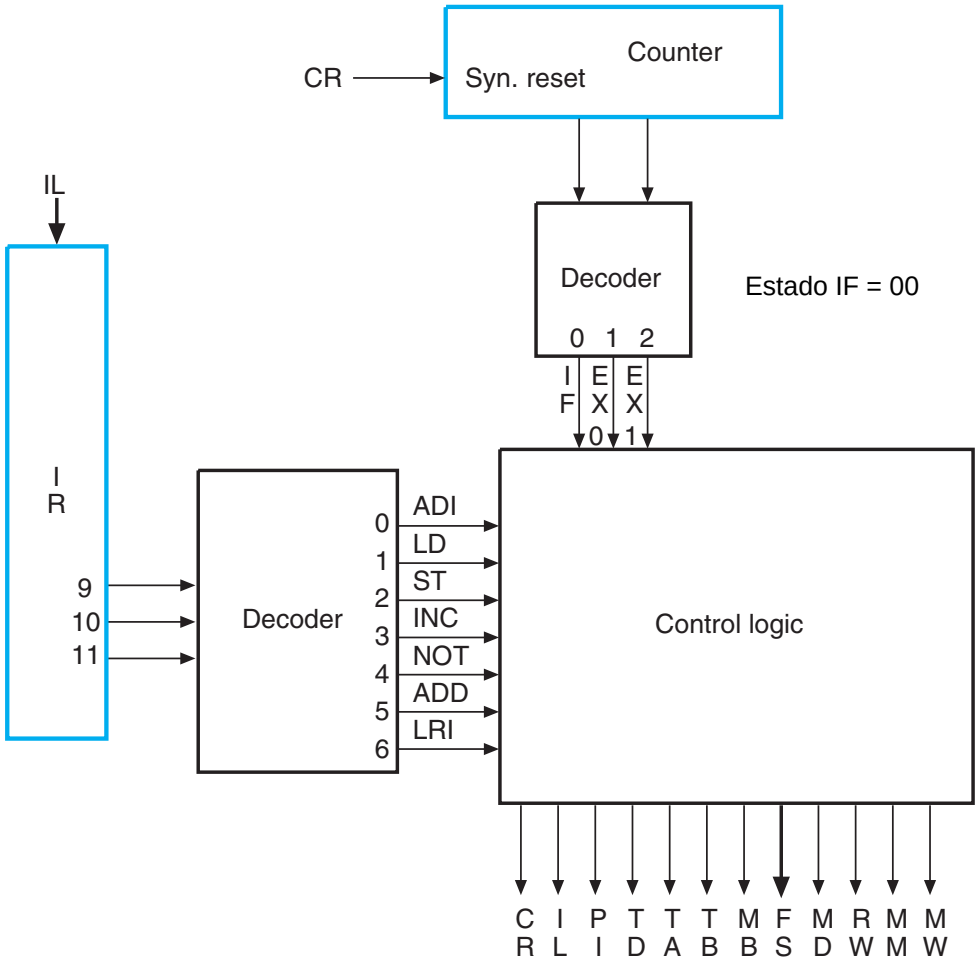


Fig. 3-34 Block Diagram of Hardwired Counter and Decoder-Based, Multiple-Cycle Control Unit