Problemas resueltos de "Diseño del procesador"

Problema 6.1

CPU con ALU $\begin{cases} Z = X + Y \\ Z = \bar{X} \\ Z = \bar{Y} \end{cases}$

Expresar la secuencia de microoperaciones para restar:

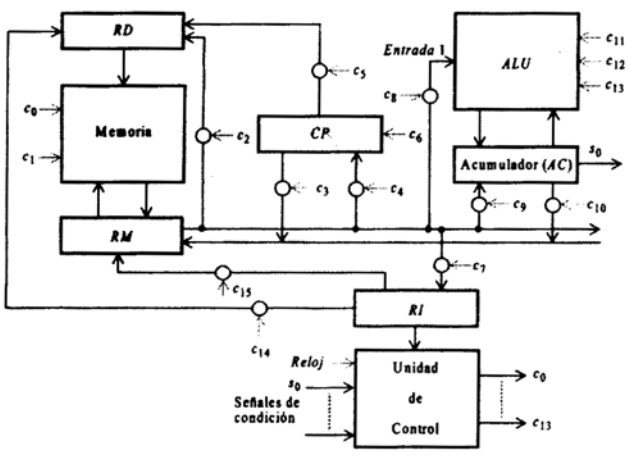
Resta $\Rightarrow$ $Z = X - Y = X + C2(Y) = X + (\bar{Y} + 1)$

Operaciones $\begin{cases} 1^{\circ}: & \bar{Y} \\ 2^{\circ}: & \bar{Y} + 1 \\ 3^{\circ}: & (\bar{Y} + 1) + X \end{cases}$

- Suponiendo que posee AC $\Rightarrow$ $\begin{cases} 1^{\circ}: & AC \leftarrow Y \\ 2^{\circ}: & AC \leftarrow \overline{AC} \\ 3^{\circ}: & AC \leftarrow AC + 1 \\ 4^{\circ}: & AC \leftarrow AC + X \end{cases}$

Problema 6.2

En la CPU de la figura obtener la secuencia para las instrucciones siguientes:



Señal de Control	Microoperación controlada
c_0	Leer de la memoria ($RM \leftarrow$ Memoria)
c_1	Escribir en la memoria (Memoria $\leftarrow (RM)$)
c_2	Transferir el contenido de *RM* a *RD* ($RD \leftarrow (RM)$)
c_3	Transferir el contenido de *CP* a *RM* ($RM \leftarrow (CP)$)
c_4	Transferir el contenido de *RM* a *CP* ($CP \leftarrow (RM)$)
c_5	Transferir el contenido de *CP* a *RD* ($RD \leftarrow (CP)$)
c_6	Incrementar en 1 el contenido de *CP* ($CP \leftarrow (CP) + 1$)
c_7	Transferir el contenido de *RM* a *RI* ($RI \leftarrow (RM)$)
c_8	Transferir el contenido de *RM* a la *Entrada* 1 de la ALU
c_9	Transferir el contenido de *RM* a *AC* ($AC \leftarrow (RM)$)
c_{10}	Transferir el contenido de *AC* a *RM* ($RM \leftarrow (AC)$)
c_{11}	$AC \leftarrow (AC)$ + *Entrada* 1
c_{12}	$AC \leftarrow (AC)$ AND *Entrada* 1
c_{13}	$AC \leftarrow (\overline{AC})$ (Complementar el contenido de *AC*)
c_{14}	Transferir el campo de dirección de *RI* a *RD* ($RD \leftarrow (RI$(dirección)))
c_{15}	Transferir el campo de datos de *RI* a *RM* ($RM \leftarrow (RI$(datos)))

Significado de las señales control de la CPU de la Figura 6.1

a) Cargar el AC con el contenido de una dir de memoria

AC ← (RM)

1°.- RD ← (RI (Operando)) ——→ C14

2°.- RM ← Mem ——→ C0

3°.- AC ← RM ——→ C9

b) AC a dir. de memoria

1°.- RD ← (RI (Operando)) ; RM ← AC ——→ C14 ; C10

2°.- Mem ← RM ——→ C1

c) AC ← AC + Mem.

1°.- RD ← (RI (Operando)) ——→ C14

2°.- RM ← Mem ——→ C0

3°.- Entrada 1 ← RM , AC ← AC + Entrada 1 ——→ C8 , C11

d) AND de AC y un operando directo

1°.- RD ← (RI (Operando)) ——→ C14

2°.- RM ← Mem ——→ C0

3°.- Entr. 1 ← RM , AC ← AC "AND" Entr. 1 ——→ C8 , C12

e) Bifurcar en modo de dir. inmediato

1°.- RM ← (RI (Operando)) ——→ C15

2°.- CP ← RM ——→ C4

f) Bifurcar en modo de dir. directo

1°.- RD ← (RI (Operando)) ——→ C14

2°.- RM ← Mem ——→ C0

3°.- CP ← RM ——→ C4

g) Bifurcar si AC = 0 en modo dir. inmediato

1°.- RM ← (RI (Operando)) ——→ C15

2°.- Si SO = 1 CP ← RM ——→ C4

h) AC ← $\overline{AC}$

1°.- AC ← $\overline{AC}$ ——→ C13

Problema 6.3

Para la CPU indicada escribir las secuencias de operaciones para:

a) Operando inmediato

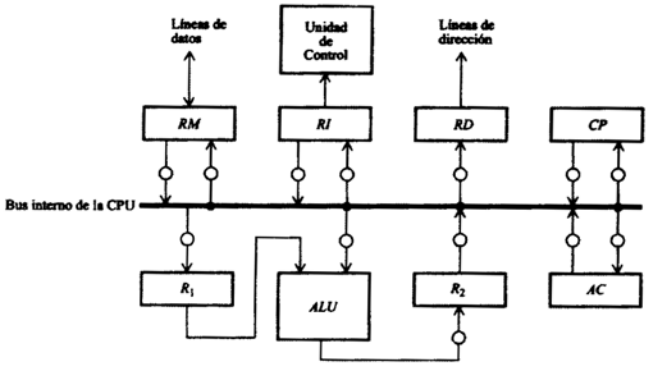
⇓

AC ← AC + Oper. inmediato

1º.- R1 ← (RI(operando))

2º.- R2 ← (AC) + (R1)

3º.- AC ← (R2)



Estructura de una CPU con un bus interno

R1 "ALU" AC ⟶ R2

b) Un operando con dir. directo ⇒ AC ← (AC) + (Mem)

1º.- RD ← (RI(Dirección))

2º.- RM ← Memoria

3º.- R1 ← (RM)

4º.- R2 ← (AC) + (R1)

5º.- AC ← (R2)

c) Un operando con dir. indirecto ⇒ AC ← (AC) + ((Mem))

1º.- RD ← (RI(Dirección))

2º.- RM ← Mem.

3º.- RD ← (RM)

4º.- RM ← Mem.

5º.- R1 ← (RM)

6º.- R2 ← (AC) + (R1)

7º.- AC ← (R2)

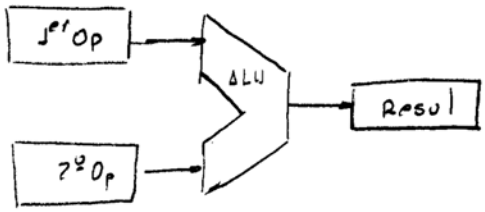
Problema 6.4

Escribir una secuencia de instrucciones para calcular

$$a := [(a*b+c) - d/f]/(a*d) + b \Rightarrow a := \frac{(a.b)+c-\frac{d}{f}}{(a.d)} + b$$

a) Instrucciones de 3 direcciones

(Accesos a memoria)

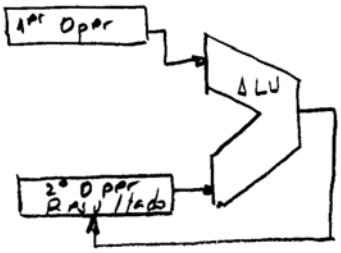


Un acceso para RI ← Mem

1º:	Mem ← a . b	
	M[X] ← M[A] * M[B]	④
2º:	M[X] ← M[C] + M[X]	④
3º:	M[Y] ← M[D] / M[F]	④
4º:	M[X] ← M[X] - M[Y]	④
5º:	M[Y] ← M[A] * M[D]	④
6º:	M[X] ← M[X] / M[Y]	④
7º:	M[X] ← M[X] + M[B]	④

→ 4 * 7 = 28 accesos

b) Instrucciones de 2 direcciones

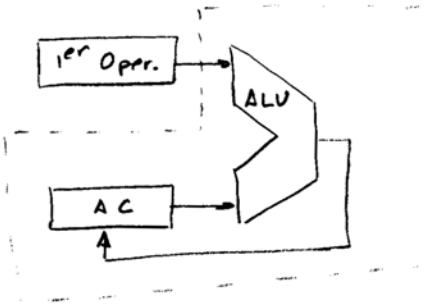


1º.-	M[X] ← M[A]	③
2º.-	M[X] ← M[X] * M[B]	④
3º.-	M[X] ← M[X] + M[C]	④
4º.-	M[Y] ← M[D]	③
5º.-	M[Y] ← M[Y] / M[F]	④
6º.-	M[X] ← M[X] - M[Y]	④
7º.-	M[A] ← M[A] * M[D]	④
8º.-	M[X] ← M[X] / M[A]	④
9º.-	M[X] ← M[X] + M[B]	④
10º.-	M[A] ← M[X]	③

Se almacena en A pq ya no hace falta

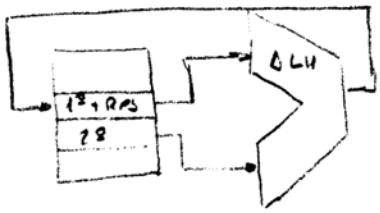
⇒ (7 * 4) + (3 * 3) = 37

c) Instrucciones de una dirección



1º: AC ← M[A] (2)
2º: AC ← AC * M[B] (2)
3º: AC ← AC + M[C] (2)
4º: M[X] ← AC (2)
5º: AC ← M[C] (2)
6º: AC ← AC / M[F] (2)
7º: M[Y] ← AC (2)
8º: AC ← M[X] (2)
9º: AC ← AC - M[Y] (2)
10º: M[Y] ← AC (2)
11º: AC ← M[A] (2)
12º: AC ← AC * M[D] (2)
13º: M[X] ← AC (2)
14º: AC ← M[Y] (2)
15º: AC ← AC - M[Y] (2)
16º: AC ← AC + M[B] (2)
17º: M[AC] ← AC (2) ⇒ 34 accesos

d) Instrucciones de cero direcciones (pila)



1º: M[A] (push M[A]) (2)
2º: M[D], M[B] (2)
3º: M[A] * M[B] (1)
4º: M[A] * M[B], M[C] (2)
5º: M[A] * M[B] + M[C] (1)
6º: M[A] * M[B] + M[C], M[D] (2)
7º: M[A] * M[B] * M[C], M[D], M[F] (2)
8º: M[A] * M[B] + M[C], M[D] / M[F] (1)
9º: (M[A] * M[B] + M[C]) - (M[D] / M[F]) (1)
10º: ←——————→, M[A] (2)
11º: ←——————→, M[A], M[D] (2)
12º: ←——————→, M[A] * M[D] (1)

13º	$((M[A] \cdot M[B] + M[C]) - (M[D] / M[F])) / (M[A] \cdot M[D])$	①
14º	⟵ ..... , M[B]	②
15º	⟵ ..... + M[B]	①
16	Pop M[A]	②

25 accesos

Problema 6.5

Secuencia de instrucciones para $a = \dfrac{(a \cdot b) + c - \dfrac{d}{f}}{a \cdot d} + b$

supuesto que hay un banco de registros

a) Instrucciones de 3 direcciones

1º	R[1] ← M[A] · M[B]	③
2º	R[1] ← R[1] + M[C]	②
3º	R[2] ← M[D] / M[F]	③
4º	R[1] ← R[1] - R[2]	①
5º	R[2] ← M[A] · M[D]	③
6º	R[1] ← R[1] / R[2]	①
7º	M[A] ← R[1] + M[B]	③ → 16 accesos

b) Instrucciones de 2 direcciones

1º	R[1] ← M[A]	②
2º	R[1] ← R[1] · M[B]	②
3º	R[1] ← R[1] + M[C]	②
4º	R[2] ← M[D]	②
5º	R[2] ← R[2] / M[F]	②
6º	R[1] ← R[1] - R[2]	①
7º	R[2] ← M[A]	②
8º	R[2] ← R[2] · M[D]	②
9º	R[1] ← R[1] / R[2]	①
10º	R[1] ← R[1] + M[B]	②
11º	M[A] ← R[1]	①

20 accesos

c) 1 dirección con bancos de registros

1º	AC ← M[A]	(2)
2º	AC ← AC × M[B]	(2)
3º	AC ← AC + M[C]	(2)
4º	R[1] ← AC	(1)
5º	AC ← M[D]	(2)
6º	AC ← AC / M[F]	(2)
7º	R[2] ← AC	(1)
8º	AC ← R[1]	(1)
9º	AC ← AC - R[2]	(1)
10º	R[2] ← AC	(1)
11º	AC ← M[A]	(2)
12º	AC ← AC × M[D]	(2)
13º	R[1] ← AC	(1)
14º	AC ← R[2]	(1)
15º	AC ← AC / R[1]	(1)
16º	AC ← AC + M[B]	(2)
17º	M[A] ← AC	(2)
		26 accesos

Problema 6.6

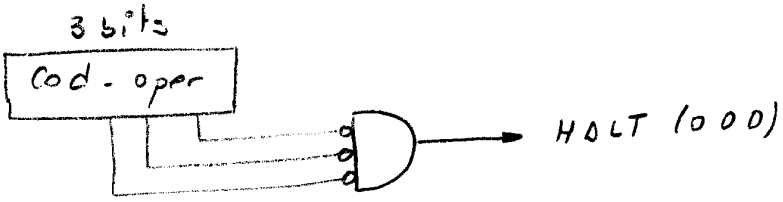
Ampliar SIMPLE1 con la instrucción HALT (SIMPLE1 = procesador del libro teoría)

- Lo que hay que hacer es que la fase de ejecución de HALT provoque que no salga de ella y además todas las señales de reloj sean 0.

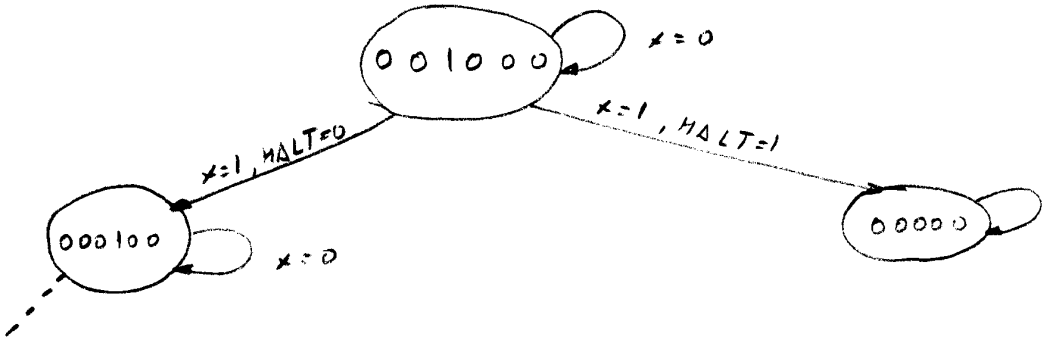
- Mirando en simple1 las instrucciones y el C.O. son

LDA (001), STA (010), ADD (011), SUB (100), MAB (101), BR (110), BRM (111) ⇒ Para HALT solo queda libre la combinación 000

Decodificador ⇒ el de SIMPLE1 y añadir

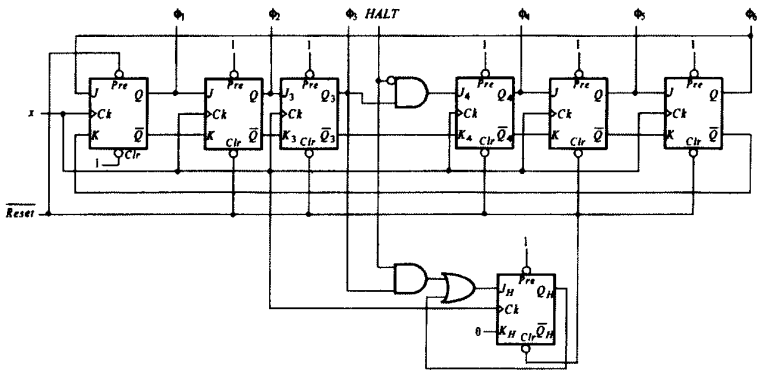


En el diagrama de estados desde el estado



Estado actual	x	HALT	Prox. estado
Φ_3 001000	0	-	Φ_3 (001000)
	1	0	Φ_4 (000100)
	1	1	HALT
HALT	-	1	HALT

Estado actual	HALT	Prox estado	J_4	K_4	J_H	K_H
Φ_3	1	Φ_H	0	-	1	-
	0	Φ_4	1	-	0	-
Φ_H	-	Φ_H	0	-	1	-



Circuito lógico del generador de subciclos modificado

PR. DP. 8

Problema 6.7

A partir de la arquitectura de SIMPLE1

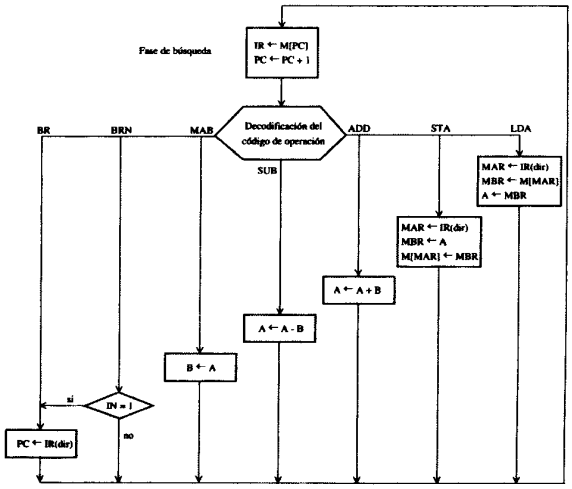
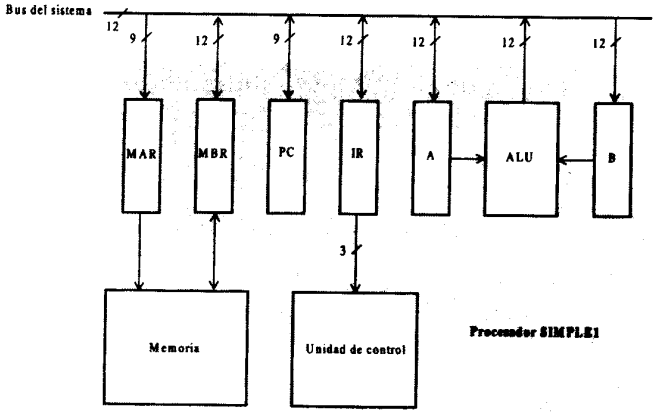


Diagrama de flujo del repertorio de instrucciones de SIMPLE1



a) Microoperaciones y señales de control para realizar B ← M[x]

La fase de búsqueda queda como en todas las instrucciones

ϕ_1, ϕ_2 y ϕ_3

		Señales de control
ϕ_4:	MAR ← IR(dir)	HIR, CMAR
ϕ_5:	MBR ← M[MAR]	R
ϕ_6:	B ← MBR	HMBR, CB

Señal de control	Microorden controlada
R	Leer de la memoria (MBR ← M[MAR])
W	Escribir en la memoria (M[MAR] ← MBR)
CMAR	Cargar el contenido del bus en MAR (MAR ← Bus)
HMBR	Habilitar el registro MBR (Bus ← MBR)
CMBR	Cargar el contenido del bus en MBR (MBR ← Bus)
HPC	Habilitar el registro PC (Bus ← PC)
CPC	Cargar el contenido del bus en PC (PC ← Bus)
IPC	Incrementar el contenido de PC (PC ← PC + 1)
HIR	Habilitar el registro IR (Bus ← IR)
CIR	Cargar el contenido del bus en IR (IR ← Bus)
HA	Habilitar el registro A (Bus ← A)
CA	Cargar el contenido del bus en A (A ← Bus)
CB	Cargar el contenido del bus en B (B ← Bus)
HALU	Habilitar la unidad aritmético-lógica
SUMA	Seleccionar la función de suma en la unidad aritmético-lógica
RESTA	Seleccionar la función de resta en la unidad aritmético-lógica

PR.DP.9

b) ¿Cómo se puede efectuar $B \leftarrow M[x]$ haciendo uso del repertorio de instrucciones siguiente?

Nemotécnico	Código binario	Instrucción	Acción
LDA x	LDA = 001	Carga directa	A ← M[x]
STA x	STA = 010	Almacenamiento directo	M[x] ← A
ADD	ADD = 011	Suma B a A	A ← A + B
SUB	SUB = 100	Resta B de A	A ← A - B
MAB	MAB = 101	Mueve A a B	B ← A
BR x	BR = 110	Salto incondicional a x	PC ← x
BRN x	BRN = 111	Salto a x si indicador negativo a 1	PC ← x si IN = 1

Repertorio de instrucciones de SIMPLE1

LDA x ⇒ A ← M[x]

MAB ⇒ B ← A

c) Modificaciones en la unidad de control para ampliar el repertorio a LDB x (B ← M[x])

Solo queda libre el C.O. 000 ⇒ LDB x (000)

Los estados ϕ_1 a ϕ_3 no cambian

ϕ_4 ⇒ MAR ← IR(dir)

ϕ_5 ⇒ MBR ← M[MAR]

ϕ_6 ⇒ B ← MBR

	IPC	CPC	HPC	CMAR	R	W	CMBR	HMBR	CIR	HIR	CA	HA	SUMA	RESTA	HALU	CB
Búsqueda	ϕ_3		ϕ_1	ϕ_1	ϕ_2			ϕ_3	ϕ_3							
LDB				ϕ_4	ϕ_5			ϕ_6		ϕ_4						ϕ_6
LDA				ϕ_4	ϕ_5			ϕ_6		ϕ_4	ϕ_6					
STA				ϕ_4		ϕ_6	ϕ_5			ϕ_4		ϕ_5				
ADD											ϕ_4		ϕ_4		ϕ_4	
SUB											ϕ_4			ϕ_4	ϕ_4	
MAB												ϕ_4				ϕ_4
BR		ϕ_4								ϕ_4						
BRN		ϕ_4 IN								ϕ_4 IN						

Matriz de instantes de activación de las señales de control, ampliada para incluir LDB

$CMAR = \phi_1 + \phi_4 LDB + \phi_4 LDA + \phi_4 STA$

$R = \phi_2 + \phi_5 LDB + \phi_5 LDA$

$HMBR = \phi_3 + \phi_6 LDB + \phi_6 LDA$

$HIR = \phi_4 LDB + \phi_4 LDA + \phi_4 STA + \phi_4 BR + \phi_4 \cdot IN \cdot BRN$

$CB = \phi_4 MAB + \phi_6 LDB$

d) Código para B ← M[125] $\quad 125_D = 1111101$

C.O.

000	001111101

Problema 6.8

En SIMPLE1 se quiere prescindir de MBR conectando directamente el bus de datos de la mem. al bus del sistema.

a) ¿Qué otro elemento se vería afectado?

La unidad de control ya que no deberá generar { HMBR, CMBR

b) ¿Cambios en el repertorio de instrucciones?

En las instrucciones LDA x y STA x

c) Rediseñar aquellos elementos que se vean afectados

Búsqueda

Antes:
- $\Phi 1$: MAR ← PC
- $\Phi 2$: MBR ← M[MAR]
- $\Phi 3$: IR ← MBR, PC ← PC+1

Ahora:
- $\Phi 1$: MAR ← PC ⇒ HPC, CMAR
- $\Phi 2$: IR ← M[MAR], PC ← PC+1 ⇒ R, CIR, IPC

LDA x

Antes:
- $\Phi 4$: MAR ← IR(dir)
- $\Phi 5$: MBR ← M[MAR]
- $\Phi 6$: A ← MBR

Ahora:
- $\Phi 4$: MAR ← IR(dir) ⇒ CMAR, HIR
- $\Phi 5$: A ← M[MAR] ⇒ CA, R

STA x

Antes:
- $\Phi 4$: MAR ← IR(dir)
- $\Phi 5$: MBR ← A
- $\Phi 6$: M[MAR] ← MBR

Ahora:
- $\Phi 4$: MAR ← IR(dir) ⇒ HIR, CMAR
- $\Phi 5$: M[MAR] ← A ⇒ HA, W

Ahora las instrucciones se ejecutan como máximo en 4 ciclos ⇒

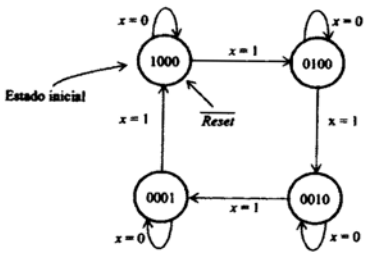
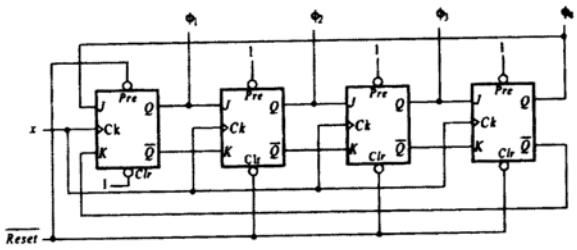


Diagrama de estado del contador en anillo módulo-4



La matriz de instantes de activación y de las señales de control será:

	IPC	CPC	HPC	CMAR	R	W	CIR	HIR	CA	HA	SUMA	RESTA	HALU	CB
Búsqueda	ϕ_2		ϕ_1	ϕ_1	ϕ_2		ϕ_2							
LDA				ϕ_3	ϕ_4			ϕ_3	ϕ_4					
STA				ϕ_3		ϕ_4		ϕ_3		ϕ_4				
ADD									ϕ_3		ϕ_3		ϕ_3	
SUB									ϕ_3			ϕ_3	ϕ_3	
MAB										ϕ_3				ϕ_3
BR		ϕ_3						ϕ_3						
BRN		ϕ_3 *IN*						ϕ_3 *IN*						

Tabla 6.8: Matriz de instantes de activación de las señales de control para cada instrucción de SIMPLE1

$IPC = \phi_2$
$CPC = \phi_3 \text{BR} + \phi_3 IN \text{BRN}$
$HPC = \phi_1$
$CMAR = \phi_1 + \phi_3 \text{LDA} + \phi_3 \text{STA}$
$R = \phi_2 + \phi_4 \text{LDA}$
$W = \phi_4 \text{STA}$
$CIR = \phi_2$
$HIR = \phi_3 \text{LDA} + \phi_3 \text{STA} + \phi_3 \text{BR} + \phi_3 IN \text{BRN}$
$CA = \phi_4 \text{LDA} + \phi_3 \text{ADD} + \phi_3 \text{SUB}$
$HA = \phi_4 \text{STA} + \phi_3 \text{MAB}$
$SUMA = \phi_3 \text{ADD}$
$RESTA = \phi_3 \text{SUB}$
$HALU = \phi_3 \text{ADD} + \phi_3 \text{SUB}$
$CB = \phi_3 \text{MAB}$

Problema 6.9

Cambios en SIMPLE1 para sustituir BRN por BRNN (bifurcar si no negativo)

Colocar una inversión en la entrada IN

Problema 6.10

Modificar SIMPLE1 para que bifurque al estado inicial desde cualquier estado dependiendo de la instrucción que esté ejecutando.

Por ejemplo salvo LDA x y STA x todas las instrucciones tienen hasta $\Phi 4$, en LDA y STA llegan a $\Phi 6$. Esto supone que todas las instrucciones tienen 6 Φ estando en algunas $\Phi 5$ y $\Phi 6$ sin hacer nada.

Rediseñar el sistema para que en LDA y STA llegue a $\Phi 6$ y en el resto de $\Phi 4$ pase a $\Phi 1$

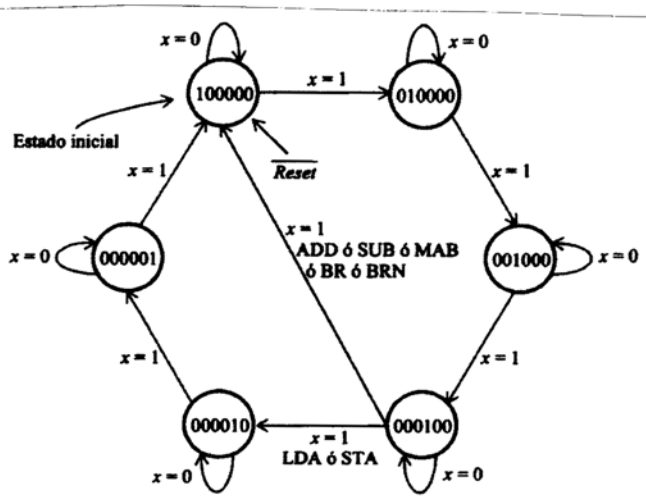


Diagrama de estados del generador de subciclos modificado

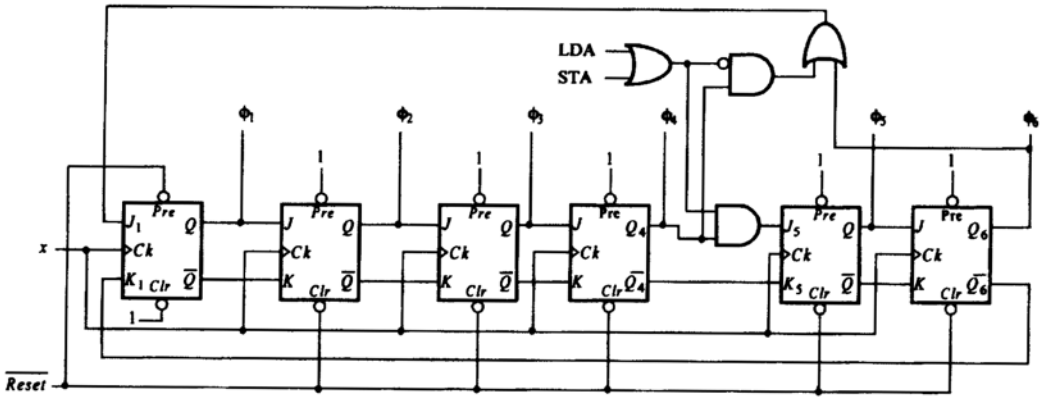
- Ahora la entrada al estado 6 (000010) debe de ser cuando estamos en el 4 (000100) y estén activadas LDA o STA (Decodificador del Cod. Operación)

- La entrada al estado 1 (100000) es desde el 6 (000001) o desde el 4 (000100) si no están activadas ni LDA ni STA

Est. actual	LDA "OR" STA	Prox esta	J_5	K_5	J_1	K_1
Φ_4	1	Φ_5	1	-	0	-
Φ_4	Φ	Φ_1	0	-	1	-
Φ_6	-	Φ_1	0	-	1	-

$J_1 = \Phi_4 \cdot \overline{LDA + STA} + \Phi_6 \qquad K_1 = \overline{Q}_6$

$J_5 = \Phi_4 \cdot (LDA + STA) \qquad K_5 = \overline{Q}_4$



Circuito lógico modificado del generador de subciclos