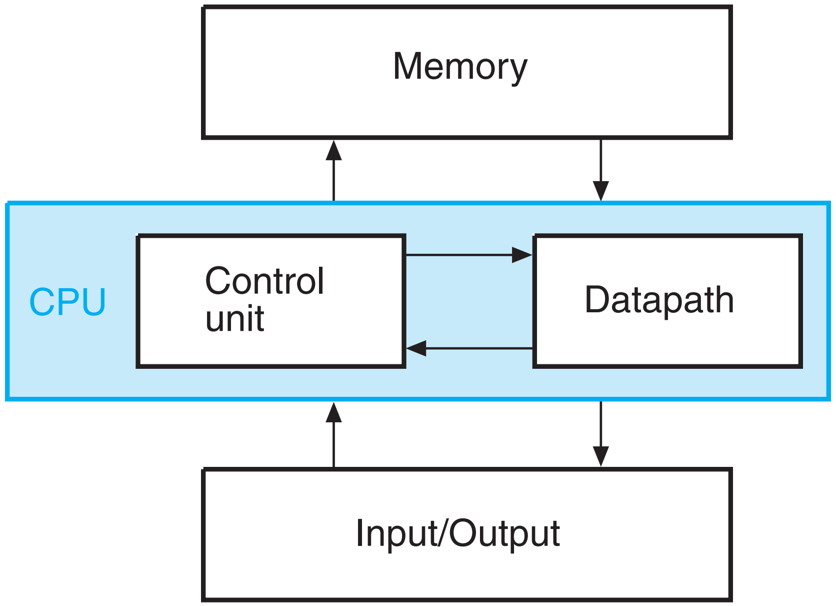


Fig. 2.1. Block Diagram of a Digital Computer

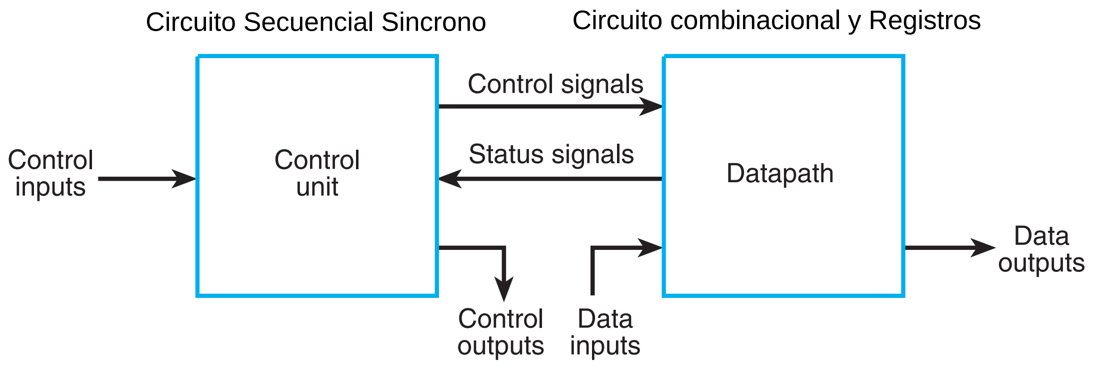


Fig. 2.2. Interaction between Datapath and Control Unit

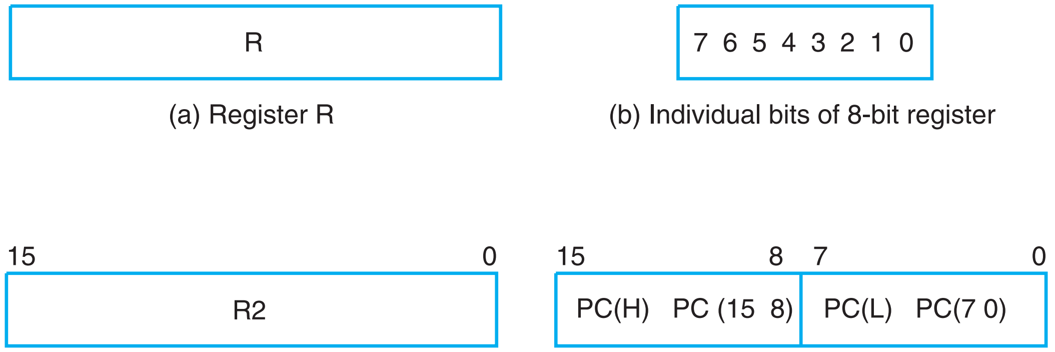

R
(a) Register R
7 6 5 4 3 2 1 0
(b) Individual bits of 8-bit register
15
0
R2
(c) Numbering of 16-bit register
15
8
7
0
PC(H) PC (15 8)
PC(L) PC(7 0)
(d) Two-part 16-bit register

Fig. 2.3. Block Diagrams of Registers

Operador Reemplazo <-

R2<-R1. Copia contenido R1 en R2 (R1 no se modifica)

Seæal de control K1

If (K1=1) then (R2<-R1)

K1: R2<-R1

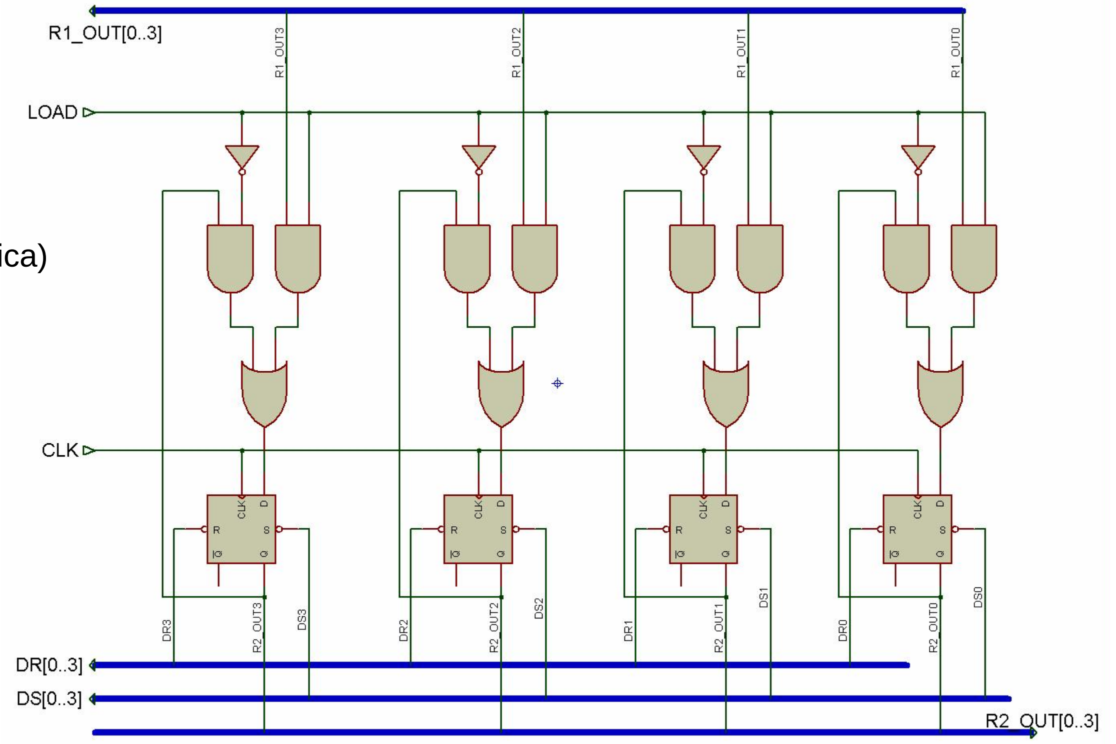


Fig. 2.4. - R2 register implemented with flip-flops D

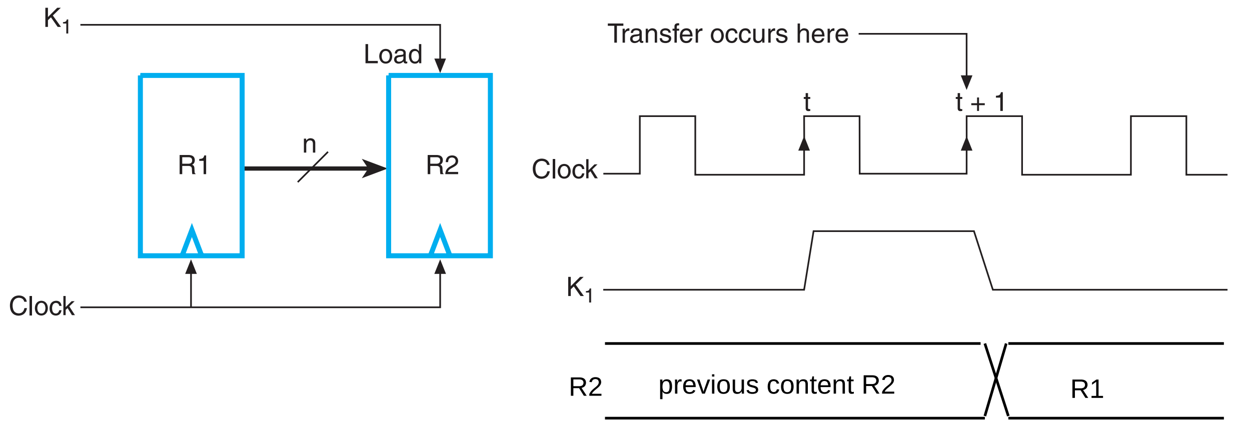


Fig. 2.5. Transfer from $R1$ to $R2$ when $K_1 = 1$

☐ TABLE 2-1
Basic Symbols for Register Transfers

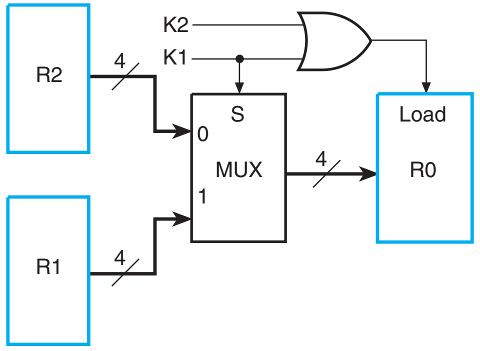
Symbol	Description	Examples
Letters (and numerals)	Denotes a register	AR, $R2$, DR, IR
Parentheses	Denotes a part of a register	$R2(1)$, $R2(7:0)$, $AR(L)$
Arrow	Denotes transfer of data	$R1 \leftarrow R2$
Comma	Separates simultaneous transfers	$R1 \leftarrow R2$, $R2 \leftarrow R1$
Square brackets	Specifies an address for memory	$DR \leftarrow M[AR]$

Table 2.1. Basic Symbols for Register Transfers

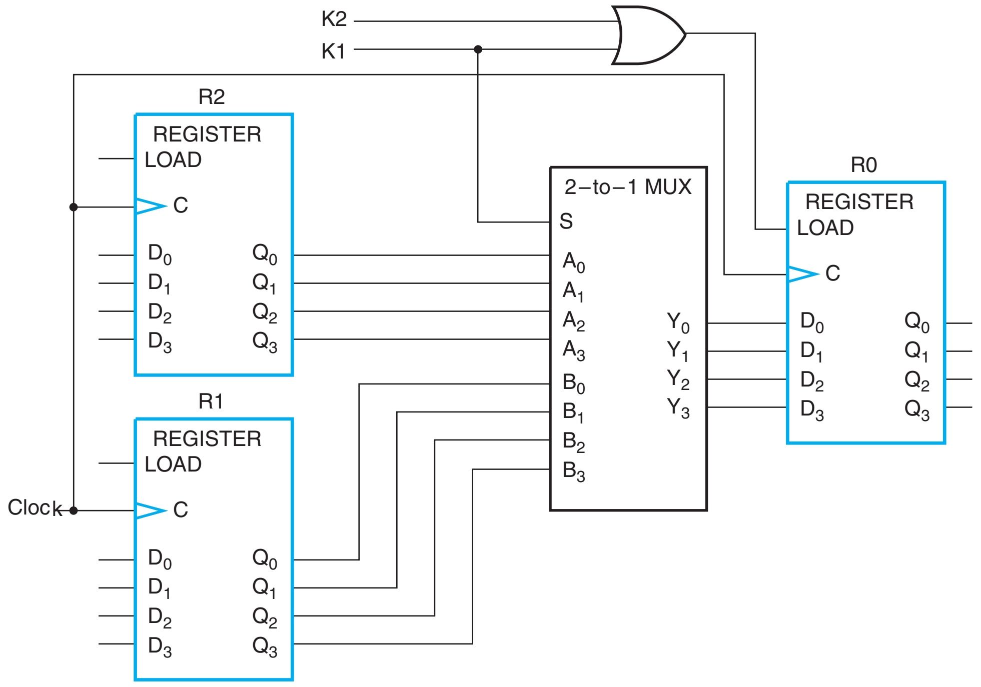
If (K1=1) then (R0 <- R1) else if (K2=1) then (R0 <- R2)

K1: R0 <- R1, $\overline{K1}$K2 : R0 <- R2

K1	K2	Operation
0	0	No transf
0	1	R0 <- R2
1	0	R0 <- R1
1	1	R0 <- R1

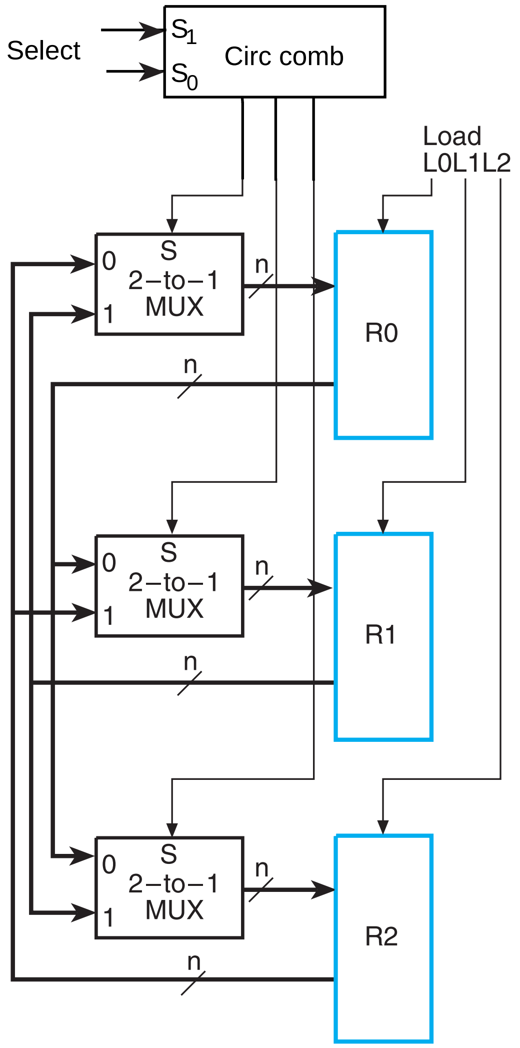


(a) Block diagram



(b) Detailed logic

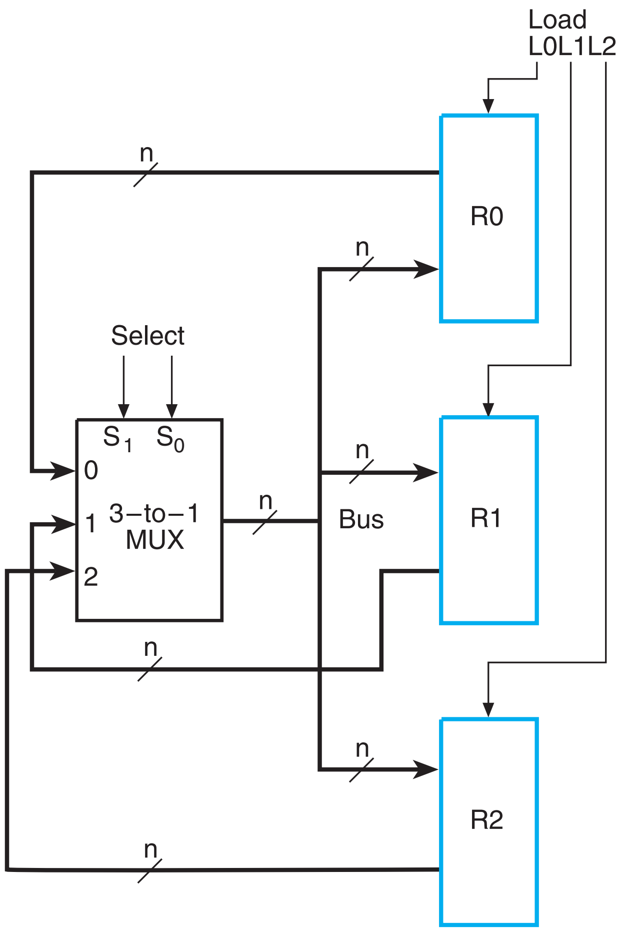
Fig 2.6. Use of Multiplexers to Select between Two Registers-Source



(a) Dedicated multiplexers

3 [2n (AND) + 1n (OR)] = 9n

Simultaneous Transfers



(b) Single Bus

3n (AND) + 1n (OR) = 4n

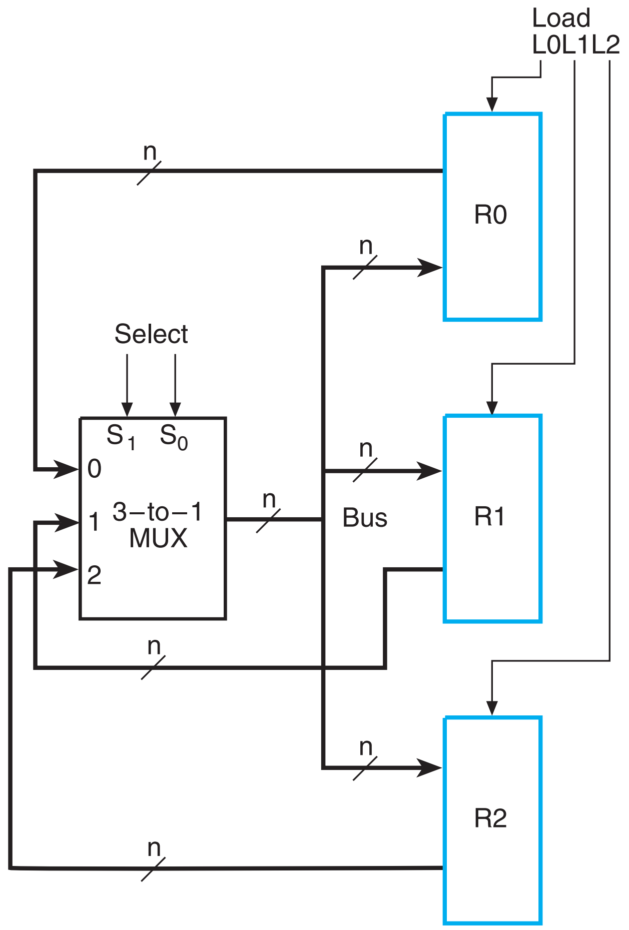
Simplicity

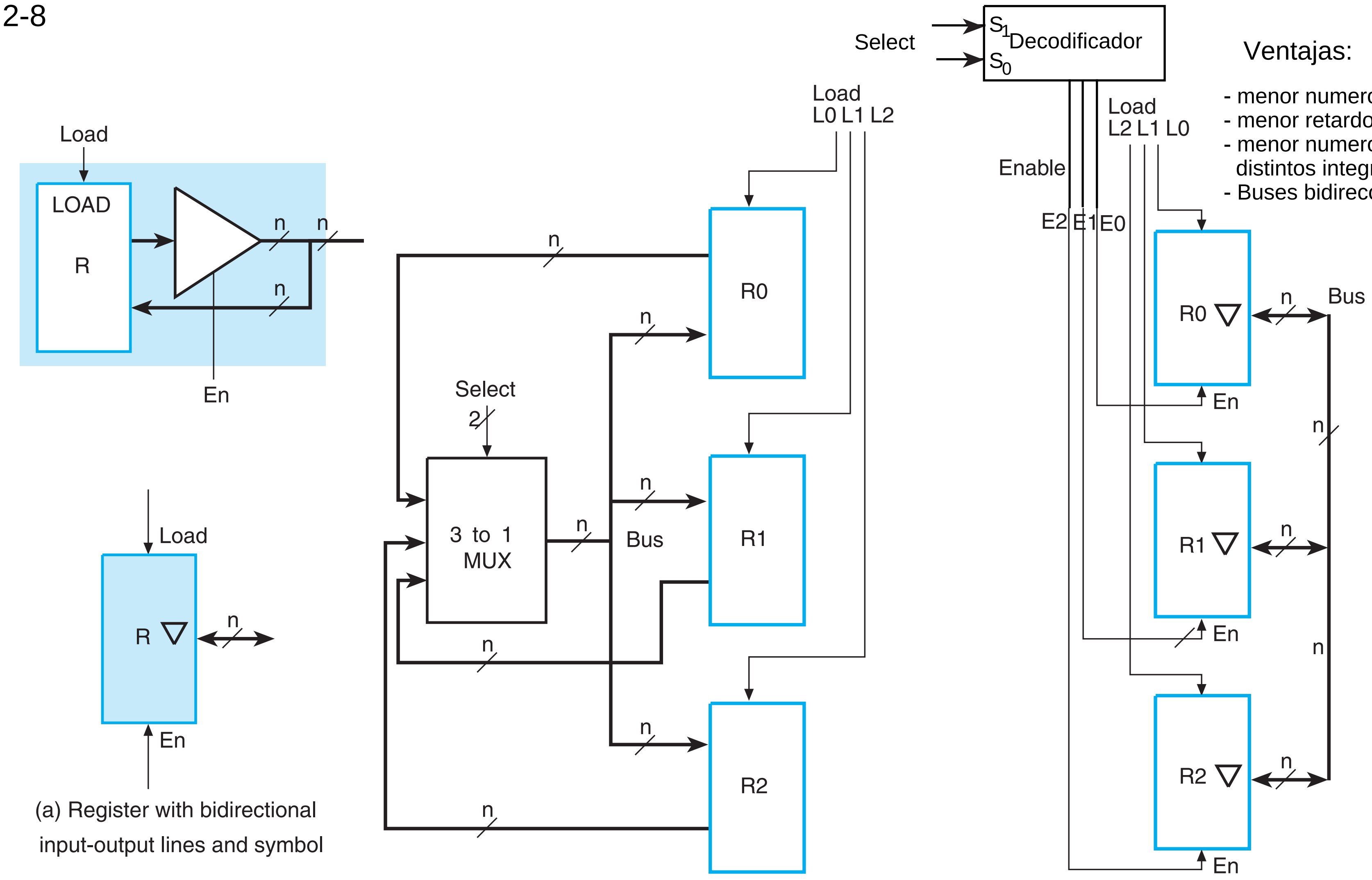
Fig 2.7. Single Bus versus Dedicated Multiplexers

TABLE 2-2
Examples of Register Transfers Using the Single Bus in Figure 2-7(b)

	Select		Load		
Register Transfer	**S1**	**S0**	**L2**	**L1**	**L0**
$R0 \leftarrow R2$	1	0	0	0	1
$R0 \leftarrow R1,\ R2 \leftarrow R1$	0	1	1	0	1
$R0 \leftarrow R1,\ R1 \leftarrow R0$	Impossible				

Examples of Register Transfers Using the Single Bus in Figure 2-7(b)





(a) Register with bidirectional input-output lines and symbol

(b) Multiplexer bus

(c) Three-state bus using registers with bidirectional lines

Ventajas:

- menor numero de conexiones
- menor retardo de propagacion
- menor numero de pines si distintos integrados
- Buses bidireccionales

Fig 2.8. Three-State Bus versus Multiplexer Bus

Microoperacion: Operacion elemental ejecutada sobre datos almacenados en registros o en memoria

Tipos: Transferencia
- Aritmeticas
- Logicas
- Desplazamiento

☐ TABLE 2-3
Arithmetic Microoperations

Symbolic designation	Description
$R0 \leftarrow R1 + R2$	Contents of $R1$ plus $R2$ transferred to $R0$
$R2 \leftarrow \overline{R2}$	Complement of the contents of $R2$ (1's complement)
$R2 \leftarrow \overline{R2} + 1$	2's complement of the contents of $R2$
$R0 \leftarrow R1 + \overline{R2} + 1$	$R1$ plus 2's complement of $R2$ transferred to $R0$ (subtraction)
$R1 \leftarrow R1 + 1$	Increment the contents of $R1$ (count up)
$R1 \leftarrow R1 - 1$	Decrement the contents of $R1$ (count down)

Table 2.3. Arithmetic Microoperations

$\overline{X}K1 : R1 \leftarrow R1+R2$

$XK1 : R1 \leftarrow R1+\overline{R2}+1$

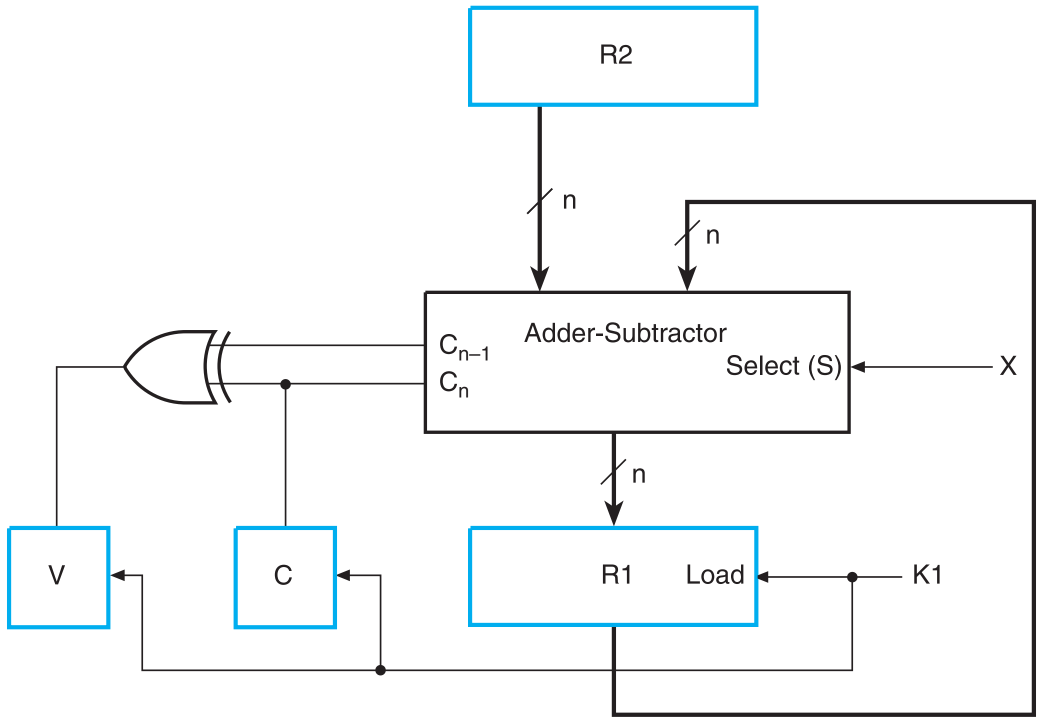


Fig 2.9. Implementation of Add and Subtract Microoperations

☐ TABLE 2-4
Logic Microoperations

Symbolic designation	Description
$R0 \leftarrow \overline{R1}$	Logical bitwise NOT (1's complement)
$R0 \leftarrow R1 \wedge R2$	Logical bitwise AND (clears bits)
$R0 \leftarrow R1 \vee R2$	Logical bitwise OR (sets bits)
$R0 \leftarrow R1 \oplus R2$	Logical bitwise XOR (complements bits)

Table 2.4. Logic Microoperations

```
AND - Borrado Selectivo   10101101     10101011   R1   Datos
                          00000000     11111111   R2   Mascara
                          --------     --------
                          00000000     10101011   R1*R2

OR -Puesta a Set Selectiva   10101101     10101011   R1  Datos
                             11111111     00000000   R2  Mascara
                             --------     --------
                             11111111     10101011   R1+R2

XOR - Complemento Selectivo   10101101     10101011   R1   Datos
                              00000000     11111111   R2   Mascara
                              --------     --------
                              10101101     01010100   R1⊕R2
```

☐ TABLE 2-5
Examples of Shifts

		Eight-bit examples	
Type	**Symbolic designation**	**Source *R*2**	**After shift: Destination *R*1**
shift left	$R1 \leftarrow \text{sl } R2$	10011110	00111100
shift right	$R1 \leftarrow \text{sr } R2$	11100101	01110010

Table 2.5. Examples of Shifts

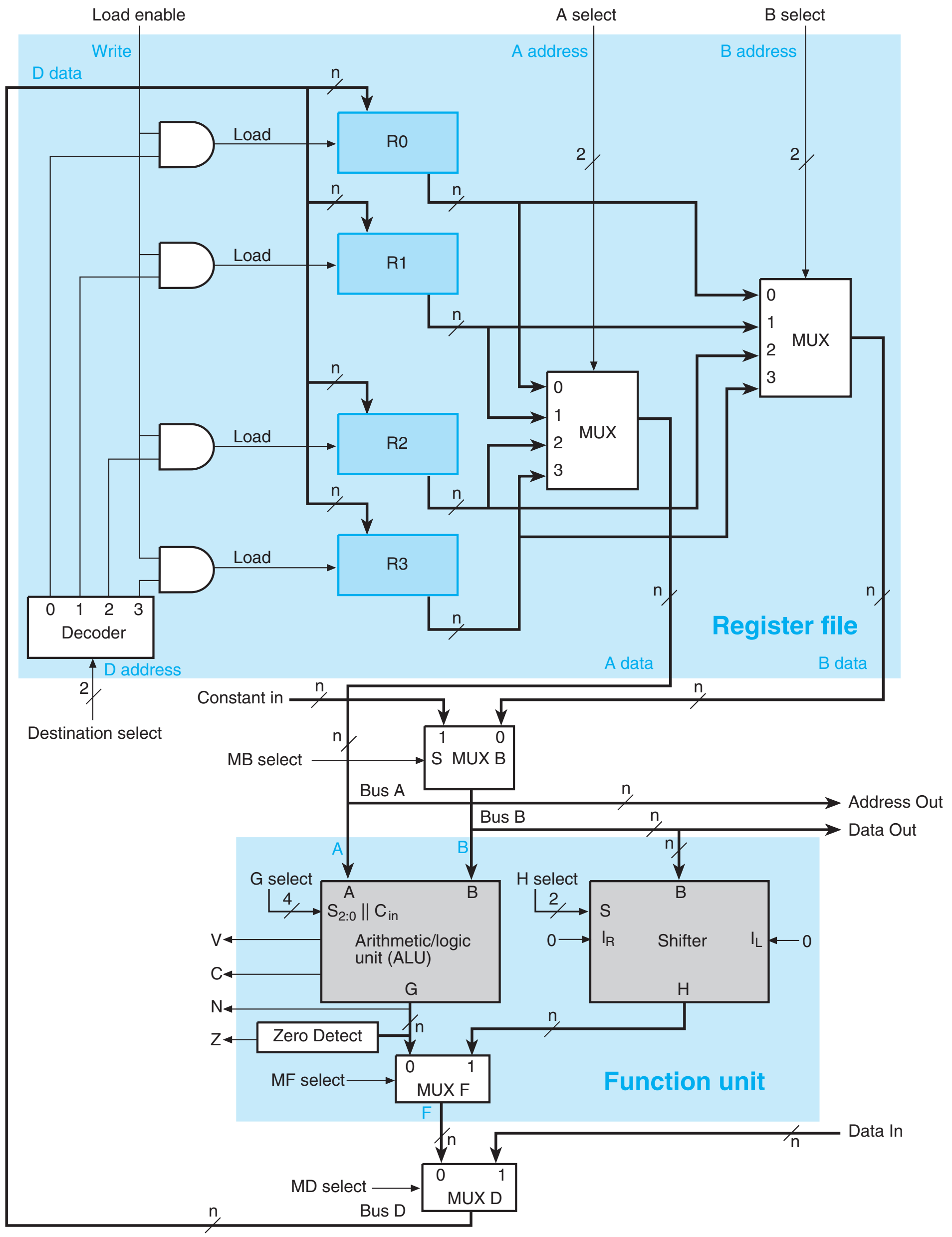
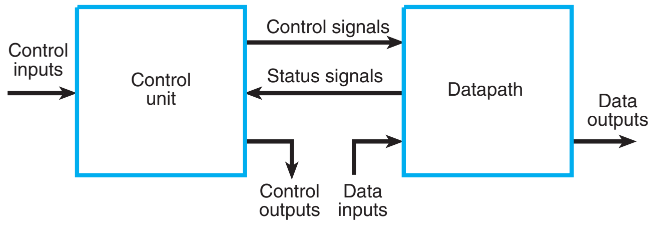


Fig 2.10. Block Diagram of a Datapath



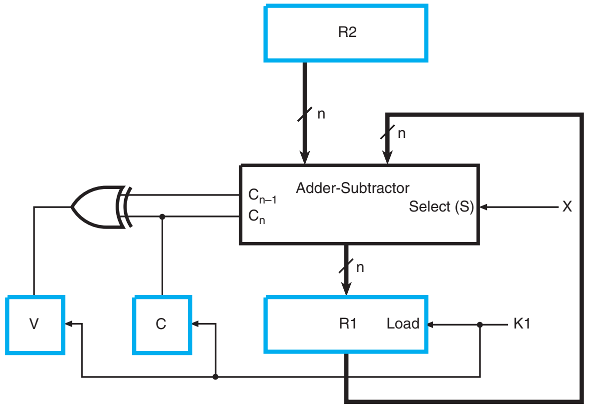
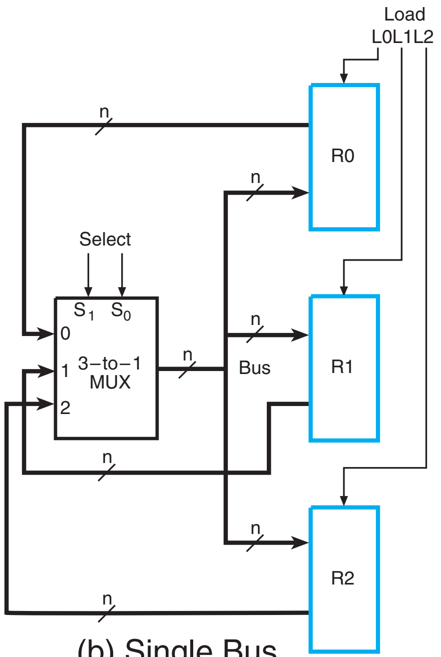
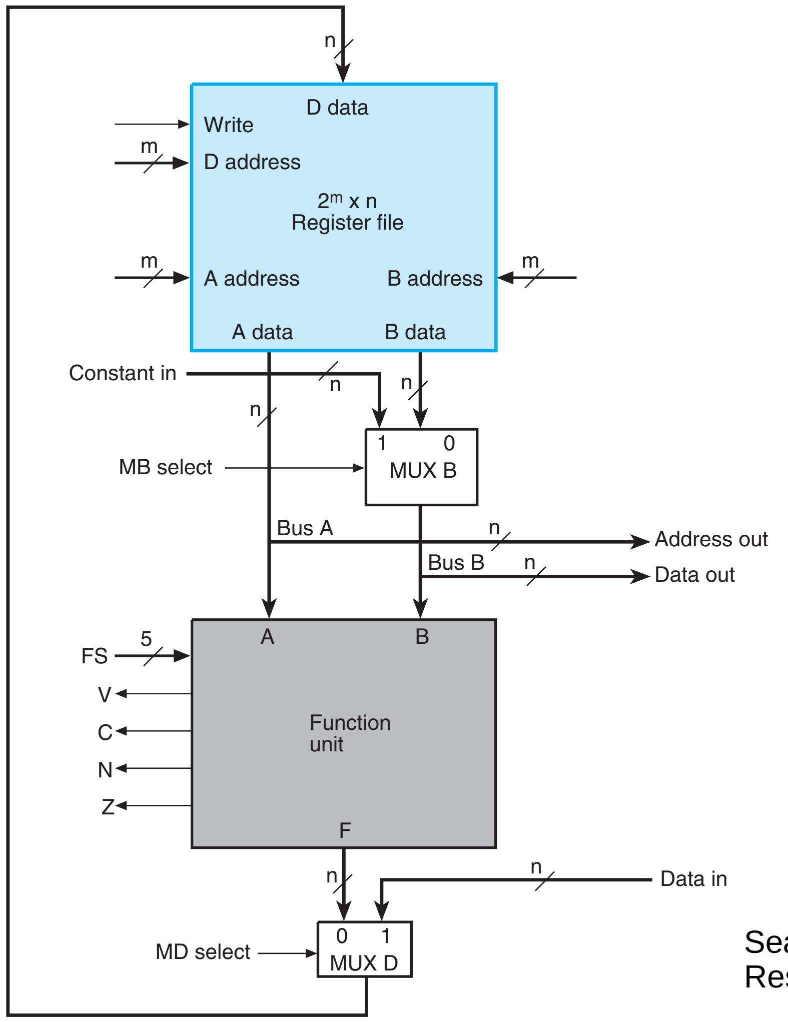


Fig 2.9. Implementation of Add and Subtract Microoperations



(b) Single Bus



Entradas/Salidas Control:

IN - Direccion A, B, D

MB, MD, FS

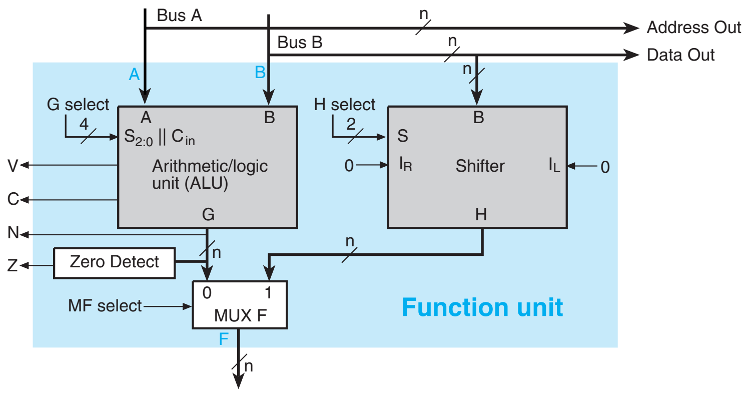
OUT - Bits Estado V, C, N, Z

Ejemplo: R1 <- R2 + R3

1 - Seleccion A R2 (direccion)
2 - Seleccion B R3 (direccion y MB)
3 - Seleccion FS (A+B)
4 - Seleccion MD (salida de unidad de funciones)
5 - Seleccion D R1 (direccion)
6 - Activacion Write

Seæales activas comienzo del ciclo de reloj.
Resultado aparece al comienzo del siguiente ciclo de reloj

Fig 2.11. Block Diagram of Datapath Using the Register File and Function Unit



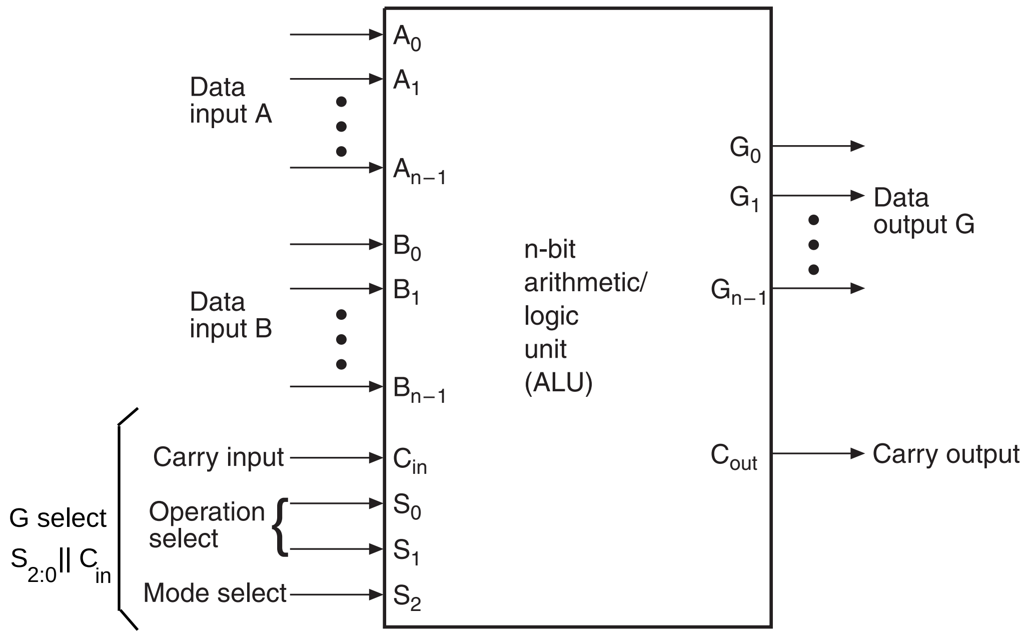


Fig 2.12. Symbol for an *n*-Bit ALU

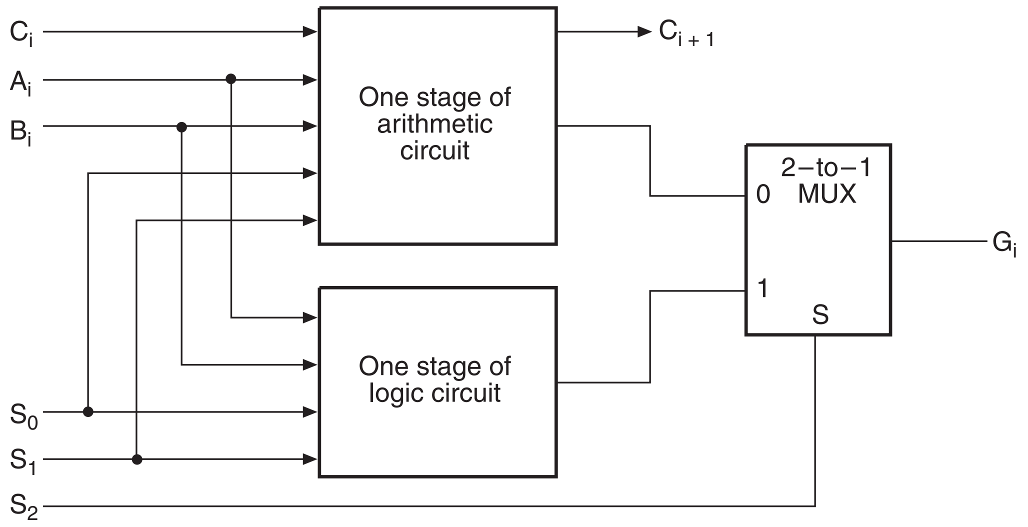


Fig 2.13. One Stage of ALU

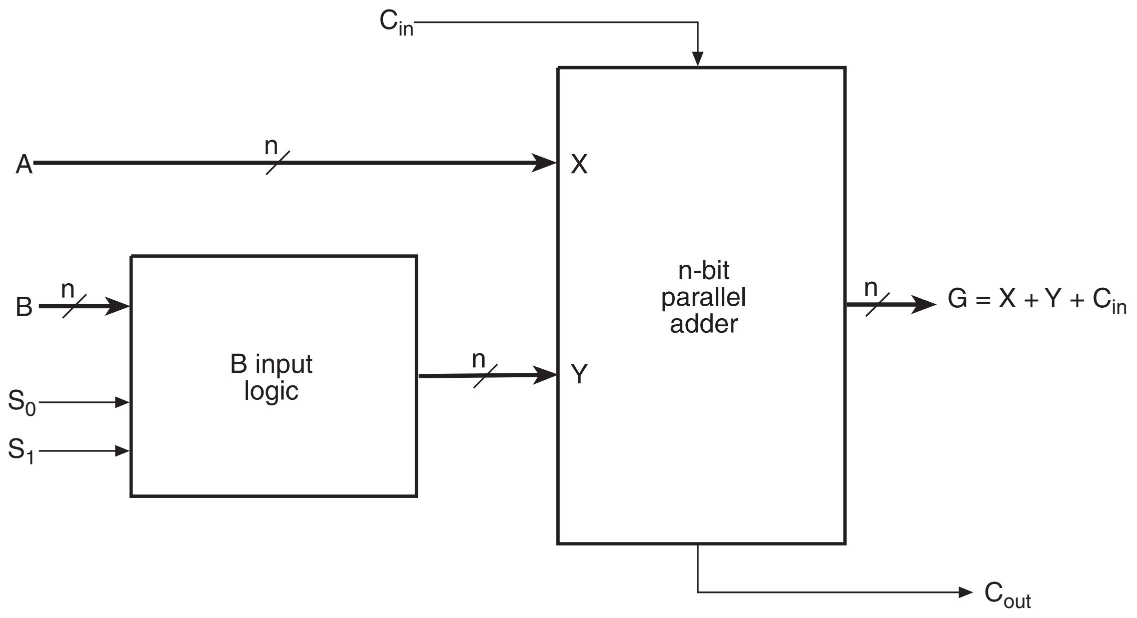


Fig. 2.14. Block Diagram of an Arithmetic Circuit

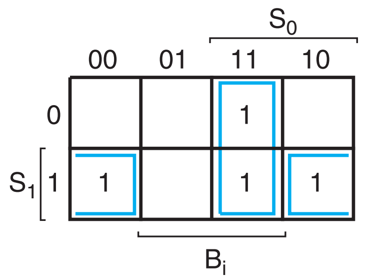
TABLE 2-6
Function Table for Arithmetic Circuit

Select		**Input**	**$G = A + Y + C_{in}$**	
S_1	**S_0**	**Y**	**$C_{in} = 0$**	**$C_{in} = 1$**
0	0	all 0's	$G = A$ (transfer)	$G = A + 1$ (increment)
0	1	B	$G = A + B$ (add)	$G = A + B + 1$
1	0	$\overline{B}$	$G = A + \overline{B}$	$G = A + \overline{B} + 1$ (subtract)
1	1	all 1's	$G = A - 1$ (decrement)	$G = A$ (transfer)

Table 2.6. Function Table for Arithmetic Circuit

Inputs			Output	
S_1	S_0	B_i	Y_i	
0	0	0	0	$Y_i = 0$
0	0	1	0	
0	1	0	0	$Y_i = B_i$
0	1	1	1	
1	0	0	1	$Y_i = \bar{B}_i$
1	0	1	0	
1	1	0	1	$Y_i = 1$
1	1	1	1	

(a) Truth table



(b) Map Simplification:
$Y_i = B_i S_0 + \bar{B}_i S_1$

Fig 2.15. B Input Logic for One Stage of Arithmetic Circuit

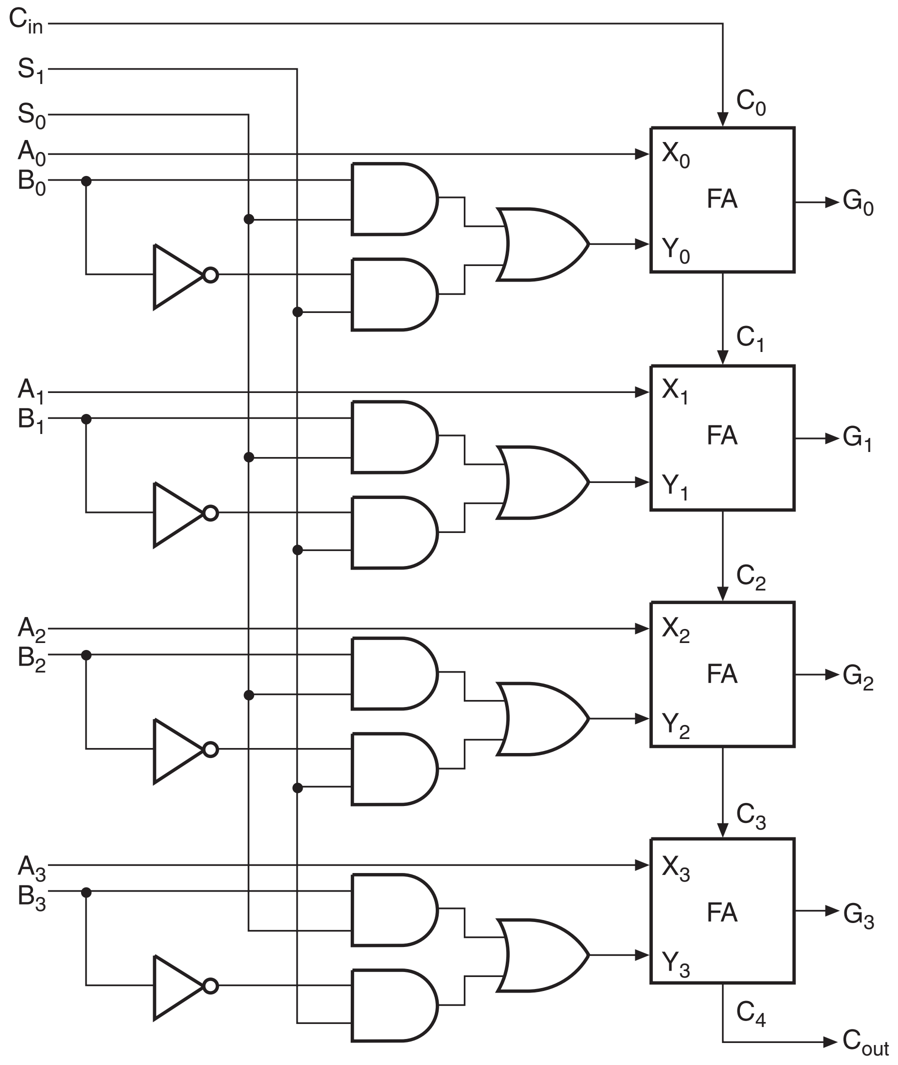
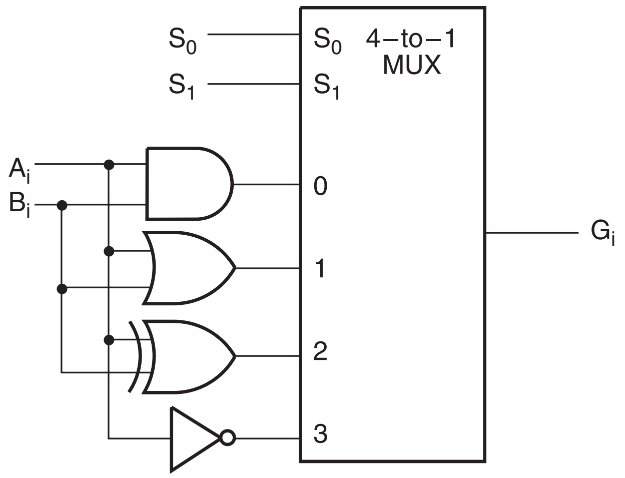


Fig 2.16. Logic Diagram of a 4-Bit Arithmetic Circuit



(a) Logic Diagram

S_1	S_0	Output	Operation
0	0	$G = A \wedge B$	AND
0	1	$G = A \vee B$	OR
1	0	$G = A \oplus B$	XOR
1	1	$G = \overline{A}$	NOT

(b) Function Table

Fig 2.17. One Stage of Logic Circuit

TABLE 2-7
Function Table for ALU

Operation Select					
S_2	S_1	S_0	C_{in}	Operation	Function
0	0	0	0	$G = A$	Transfer *A*
0	0	0	1	$G = A + 1$	Increment *A*
0	0	1	0	$G = A + B$	Addition
0	0	1	1	$G = A + B + 1$	Add with carry input of 1
0	1	0	0	$G = A + \overline{B}$	*A* plus 1's complement of *B*
0	1	0	1	$G = A + \overline{B} + 1$	Subtraction
0	1	1	0	$G = A - 1$	Decrement *A*
0	1	1	1	$G = A$	Transfer *A*
1	0	0	X	$G = A \wedge B$	AND
1	0	1	X	$G = A \vee B$	OR
1	1	0	X	$G = A \oplus B$	XOR
1	1	1	X	$G = \overline{A}$	NOT (1's complement)

Table 2.7. Function Table for ALU

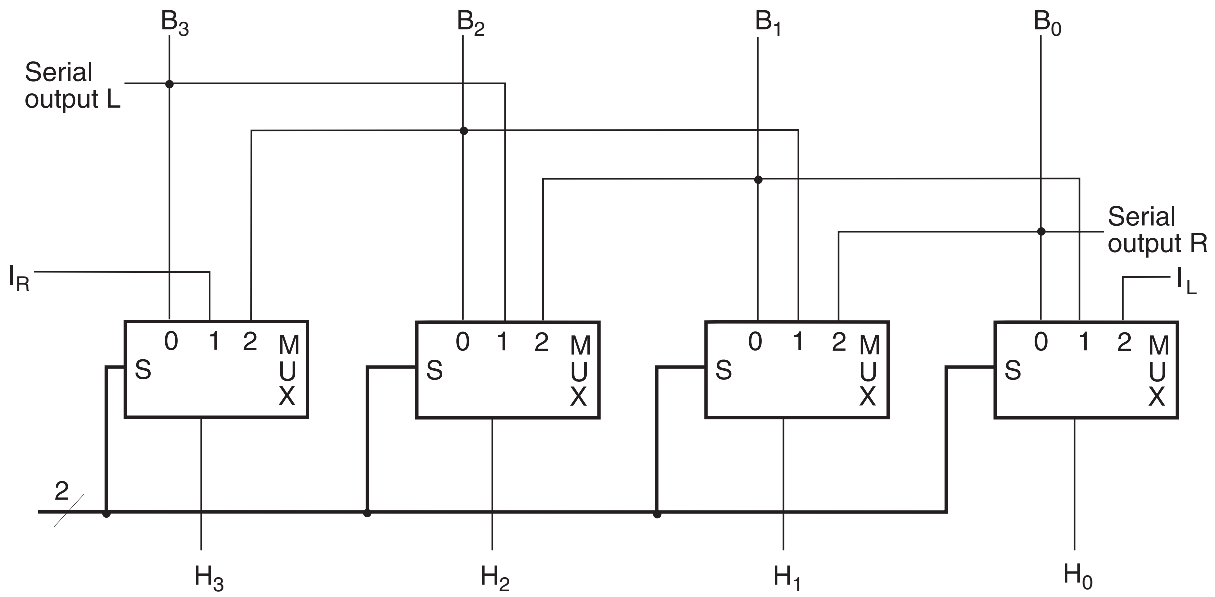


Fig 2.18. 4-Bit Basic Shifter

TABLE 2-8

Select				Eight-bit examples	
S_1	S_0	Type		Source	After shift: Destination
0	0	no shift		10011110	10011110
0	1	shift right	sr	11100101	01110010
1	0	shift left	sl	10011110	00111100

Table 2.8. Examples of Shifts

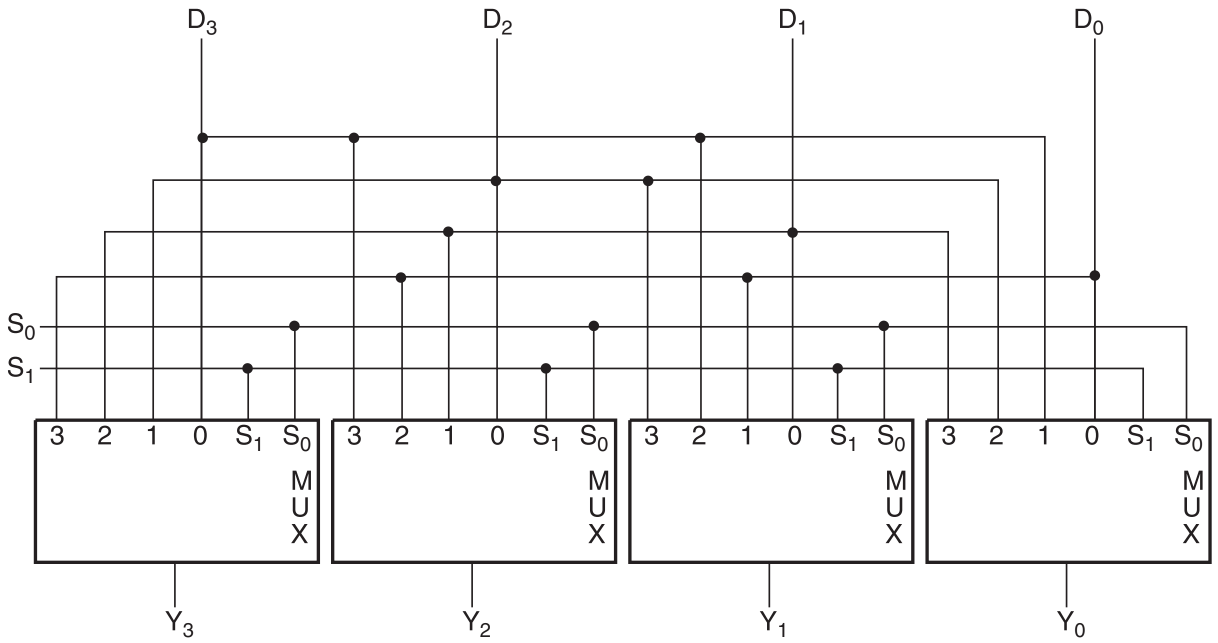


Fig 2.19. 4-Bit Barrel Shifter

TABLE 2-9
Function Table for 4-Bit Barrel Shifter

Select		**Output**				
S_1	S_0	Y_3	Y_2	Y_1	Y_0	**Operation**
0	0	D_3	D_2	D_1	D_0	No rotation
0	1	D_2	D_1	D_0	D_3	Rotate one position
1	0	D_1	D_0	D_3	D_2	Rotate two positions
1	1	D_0	D_3	D_2	D_1	Rotate three positions

Table 2.9. Function Table for 4-Bit Barrel Shifter

TABLE 2-10
G Select, H Select, and MF Select Codes Defined in Terms of FS Codes

FS FS4 FS3 FS2 FS1 FS0	MF Select	G Select S2 S1 S0 Ci	H Select S1 S0	Microoperation
00000	0	0000	00	$F = A$
00001	0	0001	00	$F = A + 1$
00010	0	0010	00	$F = A + B$
00011	0	0011	00	$F = A + B + 1$
00100	0	0100	01	$F = A + \overline{B}$
00101	0	0101	01	$F = A + \overline{B} + 1$
00110	0	0110	01	$F = A - 1$
00111	0	0111	01	$F = A$
01000	0	1000	10	$F = A \wedge B$
01010	0	1010	10	$F = A \vee B$
01100	0	1100	10	$F = A \oplus B$
01110	0	1110	10	$F = \overline{A}$
10000	1	0000	00	$F = B$
10100	1	0100	01	$F = \text{sr } B$
11000	1	1000	10	$F = \text{sl } B$

FS

FS4 ------------------- MF

FS3 ------------------- G3 (S2) ------ H1

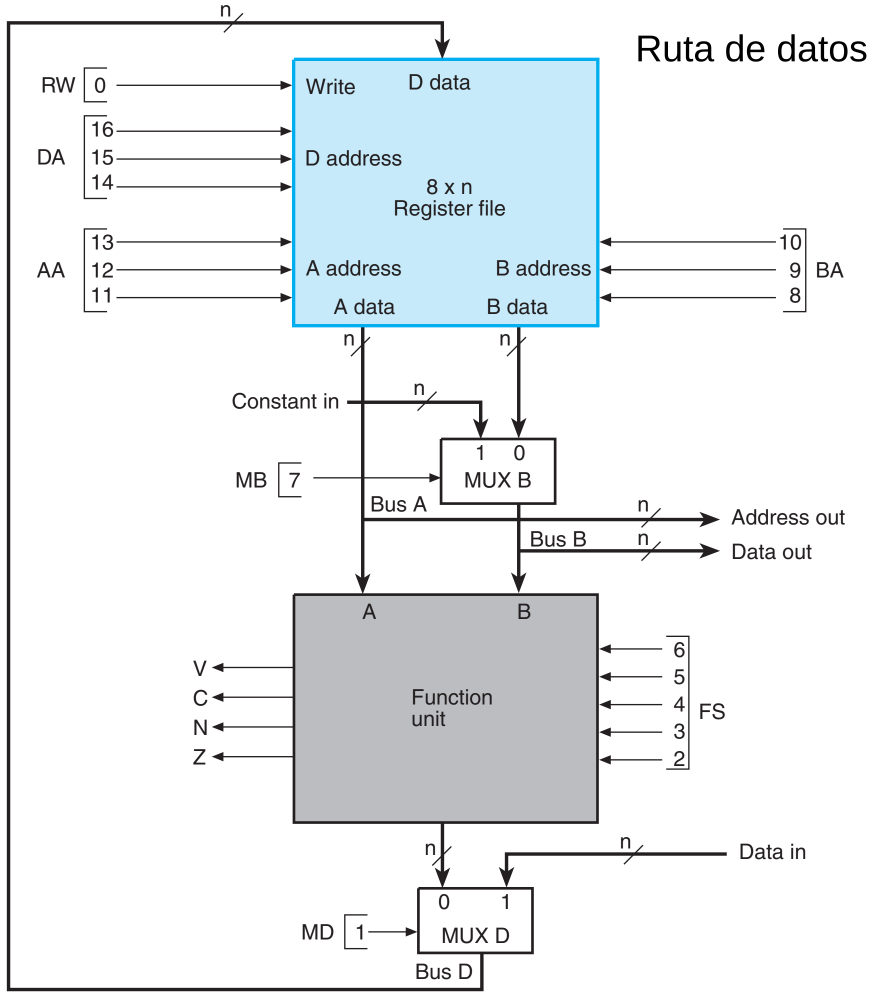
FS2 ------------------- G2 (S1) ------ H0

FS1 ------------------- G1 (S0)

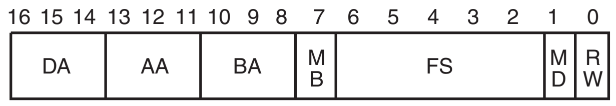
FS0 ------------------- G0 (Cin)

Table 2.10. *G* Select, *H* Select, and *MF* Select Codes Defined in Terms of *FS* Codes

Ruta de datos con archivo de 8 registros



(a) Block Diagram



(b) Control word

Fig. 2.20. Datapath with Control Variables

TABLE 2-11
Encoding of Control Word for the Datapath

DA, AA, BA		MB		FS		MD		RW	
Function	**Code**	**Function**	**Code**	**Function**	**Code**	**Function**	**Code**	**Function**	**Code**
$R0$	000	Register	0	$F = A$	00000	Function	0	No write	0
$R1$	001	Constant	1	$F = A + 1$	00001	Data In	1	Write	1
$R2$	010			$F = A + B$	00010				
$R3$	011			$F = A + B + 1$	00011				
$R4$	100			$F = A + \overline{B}\quad 1$	00100				
$R5$	101			$F = A + \overline{B} + 1$	00101				
$R6$	110			$F = A - 1$	00110				
$R7$	111			$F = A$	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 2.11. Encoding of Control Word for the Datapath

TABLE 2-12
Examples of Microoperations for the Datapath, Using Symbolic Notation

Micro-operation	DA	AA	BA	MB	FS	MD	RW
$R1 \leftarrow R2 + \overline{R3} + 1$	$R1$	$R2$	$R3$	Register	$F = A + \overline{B} + 1$	Function	Write
$R4 \leftarrow$ sl R6	$R4$	—	$R6$	Register	$F =$ sl B	Function	Write
$R7 \leftarrow R7 + 1$	$R7$	$R7$	—	Register	$F = A + 1$	Function	Write
$R1 \leftarrow R0 + 2$	$R1$	$R0$	—	Constant	$F = A + B$	Function	Write
Data out $\leftarrow R3$	—	—	$R3$	Register	—	—	No Write
$R4 \leftarrow$ Data in	$R4$	—	—	—	—	Data in	Write
$R5 \leftarrow 0$	$R5$	$R0$	$R0$	Register	$F = A \oplus B$	Function	Write (Reset)

Table 2.12. Examples of Microoperations for the Datapath, Using Symbolic Notation

☐ TABLE 2-13

Micro-operation	DA	AA	BA	MB	FS	MD	RW
$R1 \leftarrow R2 - R3$	001	010	011	0	00101	0	1
$R4 \leftarrow$ sl R6	100	000	110	0	11000	0	1
$R7 \leftarrow R7 + 1$	111	111	000	0	00001	0	1
$R1 \leftarrow R0 + 2$	001	000	000	1	00010	0	1
Data out $\leftarrow R3$	000	000	011	0	00000	0	0
$R4 \leftarrow$ Data in	100	000	000	0	00000	1	1
$R5 \leftarrow 0$	101	000	000	0	01100	0	1

Table 2.13. Examples of Microoperations from Table 2-12, Using Binary Control Words

Cuando la direccion de los operandos no es relevante se coloca a 000

Ejemplo, segunda microoperacion R4 <- sl R6, campo AA = 000

Inicialmente registros contienen su numero
i.e. R3 contiene 03

Microoperations

$R1 \leftarrow R2 - R3$
$R4 \leftarrow \text{sl R6}$
$R7 \leftarrow R7 + 1$
$R1 \leftarrow R0 + 2$
$\text{Data out} \leftarrow R3$
$R4 \leftarrow \text{Data in}$
$R5 \leftarrow 0$

Escritura Registros
en el ciclo siguiente

bits estado y salida Unidad de
Funciones en el mismo ciclo

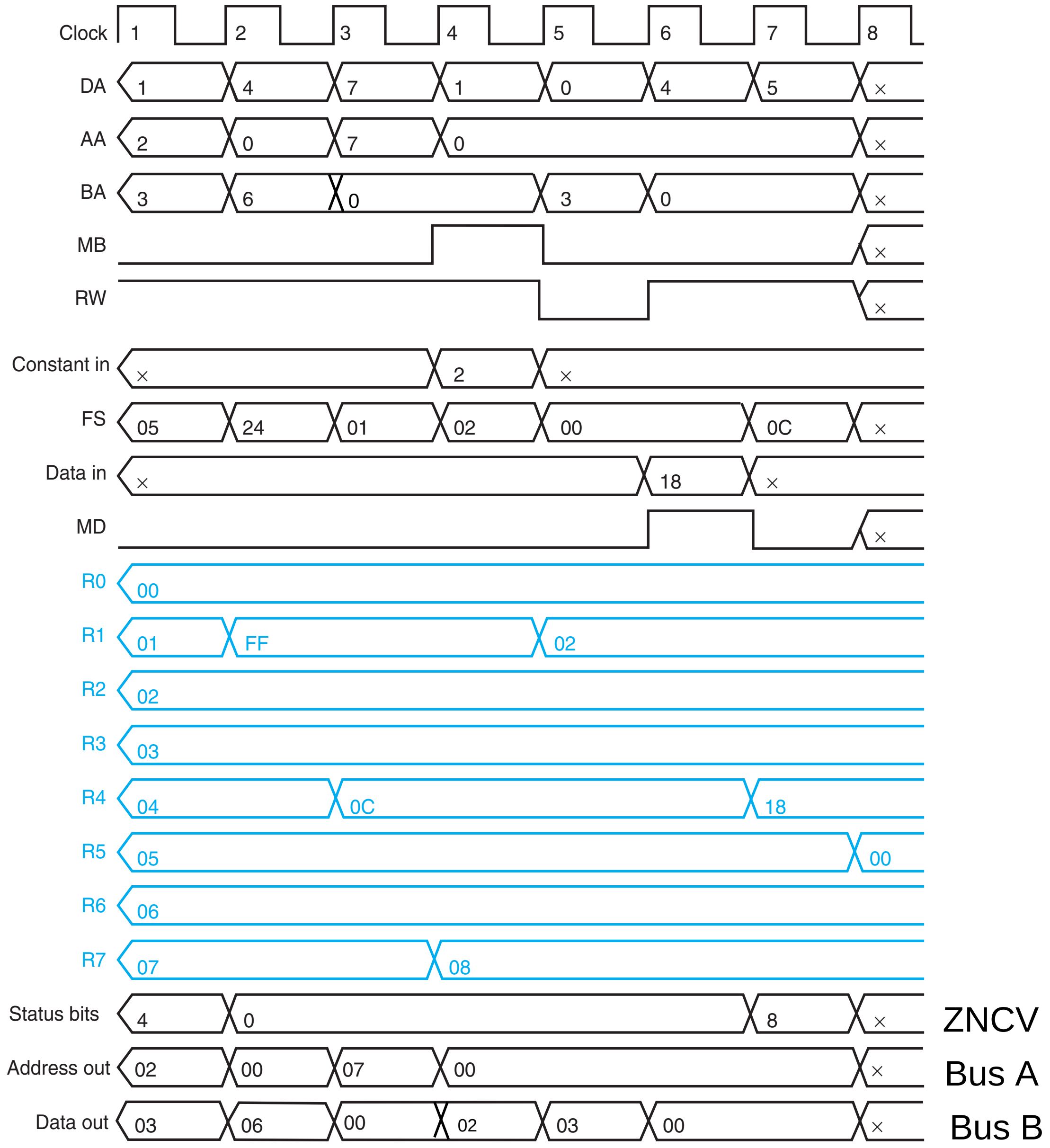


Fig. 2.21. Simulation of the Microoperation Sequence in Table 3-13

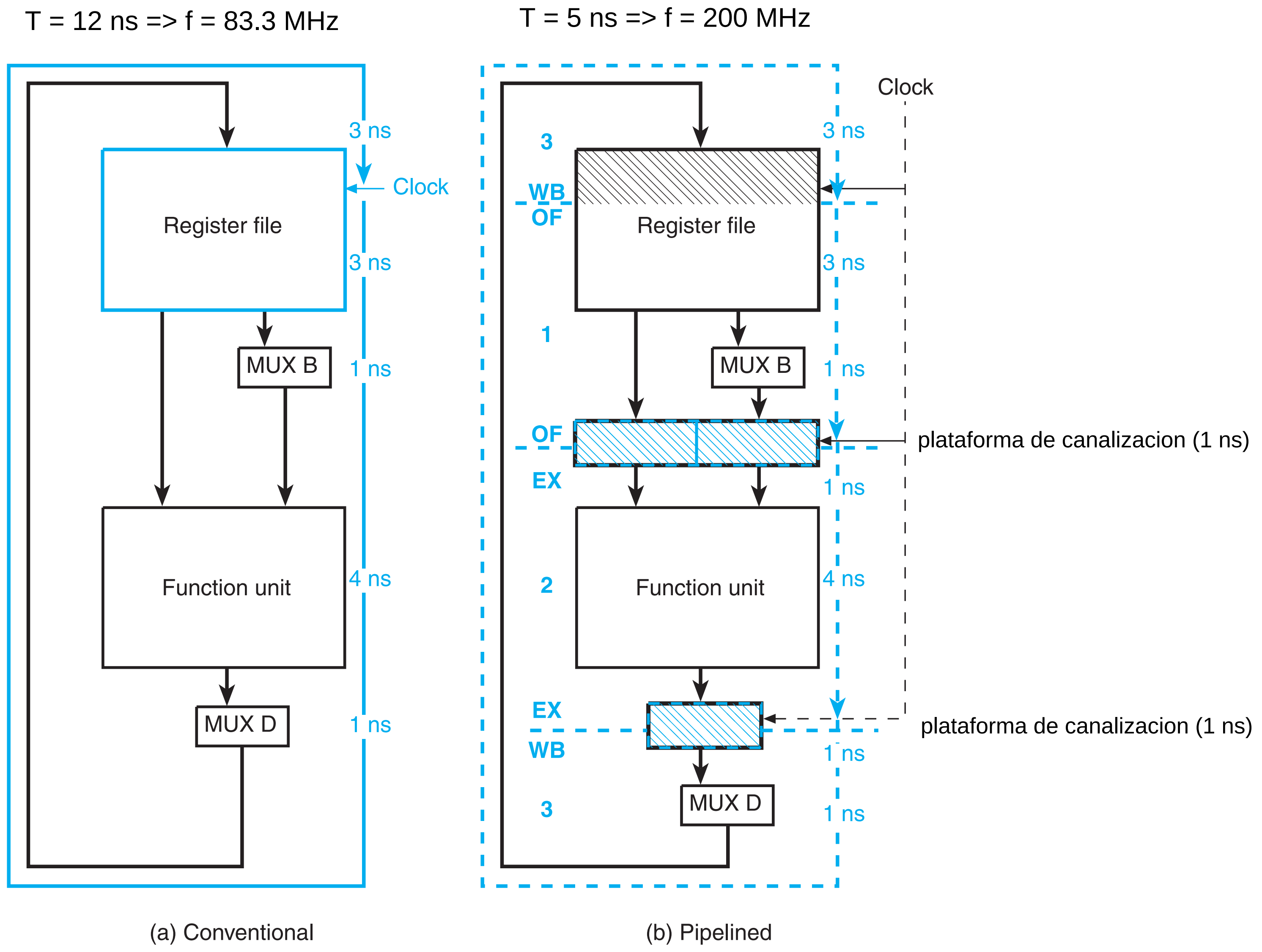


Fig.2.22. Datapath Timing

1 operario: 1 pieza en 1 minuto

3 operarios: cada etapa 20 s => 3 piezas en 1 minuto!

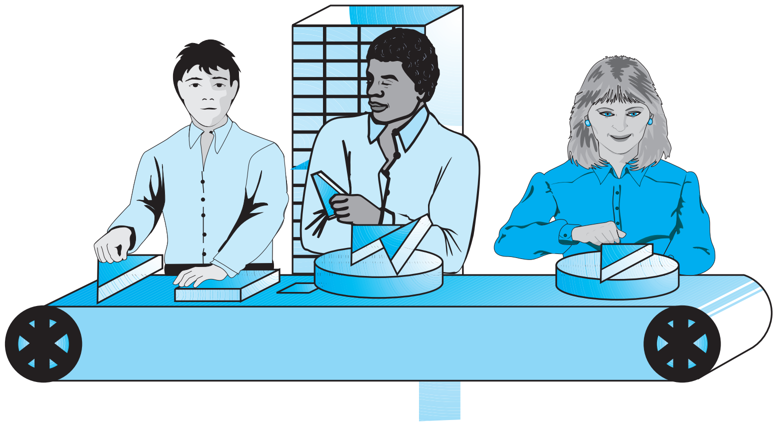


Fig. 2.23. Assembly Line Analogy to Datapath Pipeline

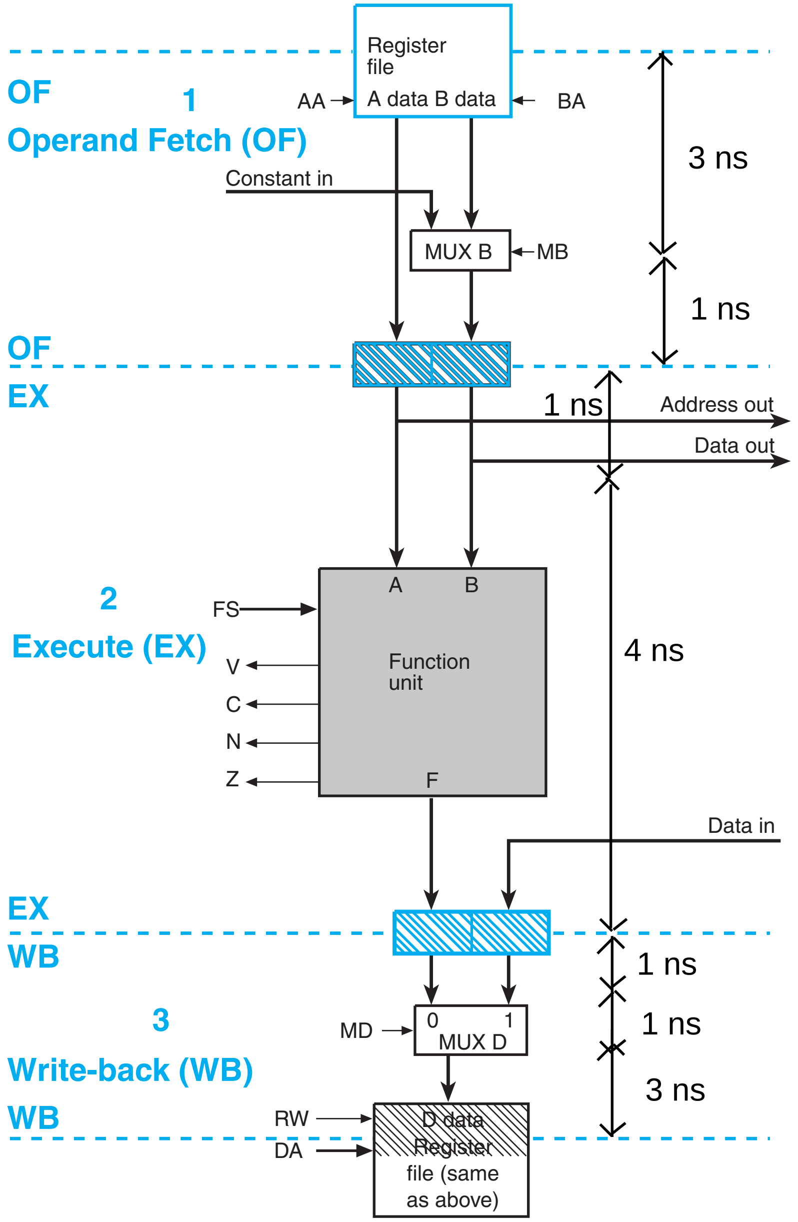


Fig. 2.24. Block Diagram of Pipelined Datapath

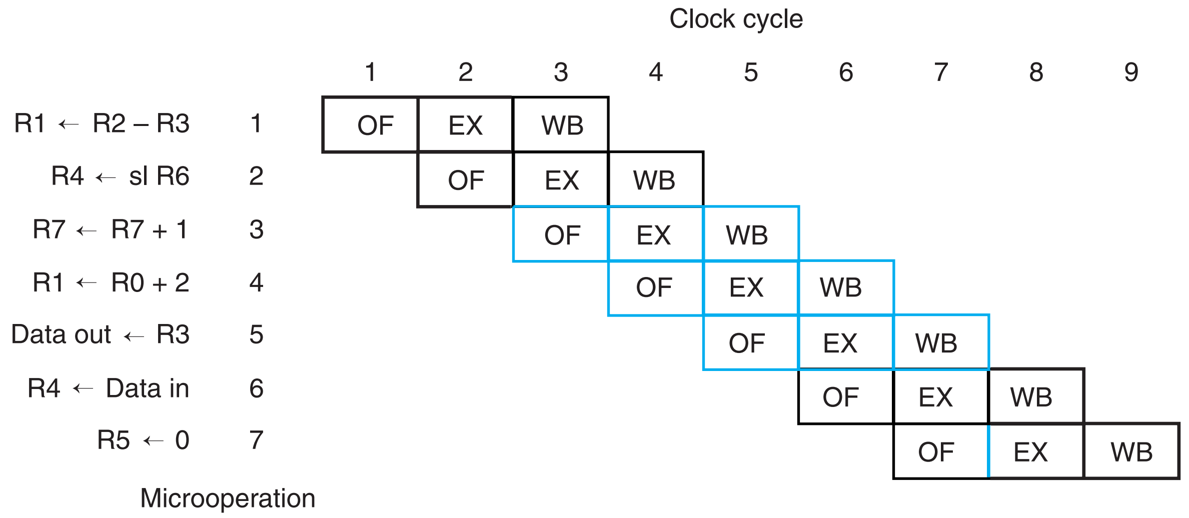


Fig. 2.25. Pipeline Execution Pattern for Microoperation Sequence in Table 3-13

$$\left(\begin{array}{l}\text{sin pipeline: 7 operaciones en 7 ciclos (12 ns): 7x12 = 84 ns}\\ \text{con pipeline: 7 operaciones en 9 ciclos (5 ns): 9x5 = 45 ns}\end{array}\right) \quad 84 / 45 = 1.87$$

$$\left(\begin{array}{l}\text{sin pipeline: 5 operaciones en 5 ciclos (12 ns): 5x12 = 60 ns}\\ \text{con pipeline: 5 operaciones en 5 ciclos (5 ns): 5x5 = 25 ns}\end{array}\right) \quad 60 / 25 = 2.4$$