CAPÍTULO 5: INDALO 1.0

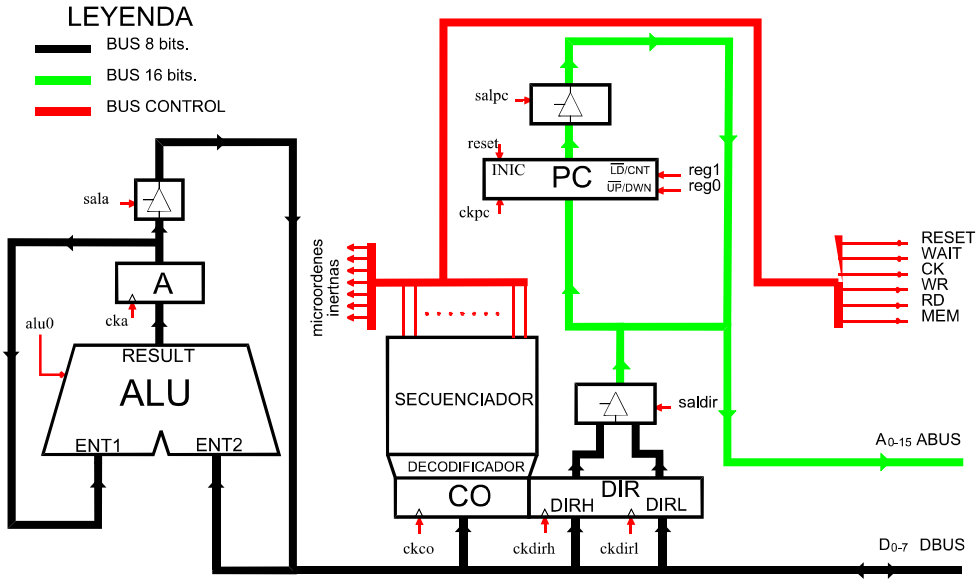


Fig.5.1. Unidad Central de Proceso (CPU) de Indalo 1.0.

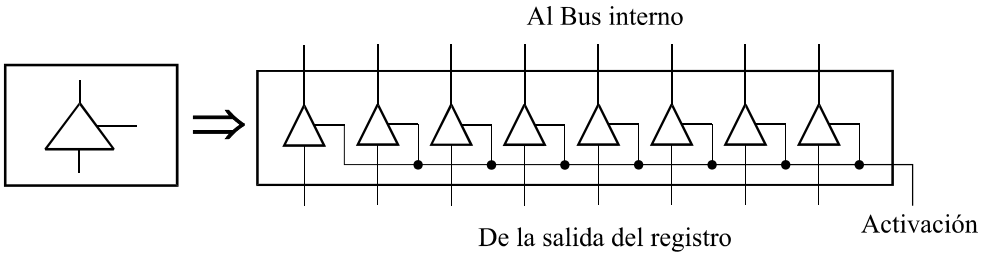
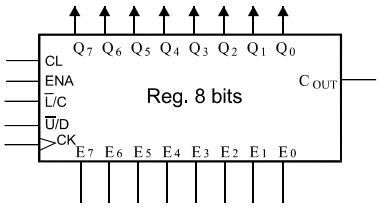


Fig.5.2. Contenido de un bloque triestado de 8 bits.



CL	CK	L/C	ENA	U/D	Operación
1	X	X	X	X	Pone a cero el contador
0	Inact.	X	X	X	No varía el contador
0	↑	0	X	X	Carga paralelo del cont.
0	↑	1	0	X	No varía contador
0	↑	1	1	0	Incrementa contador
0	↑	1	1	1	Decrementa contador

L/C= Load/Count: carga en paralelo si es 0, y actúa como contador si es 1.
U/D= Up/Down: si L/C=1, selecciona funcionamiento ascendente (U/D=0) o descendente(U/D=1) del contador
COUT= Salida de acarreo: si ENA=1, COUT se activa al pasar de 0FFH a 0 en cuenta ascendente, y de 0 a 0FFH en cuenta descendente.
ENA= Permite el funcionamiento como contador: si es 0 y L/C=1, ignora los pulsos de reloj. Está diseñada para conectarse con COUT de la etapa anterior.
CL= Clear: entrada asíncrona de puesta a 0 del contador.

Fig.5.3. Contador ascendente/descendente de 8 bits con carga paralela síncrona.

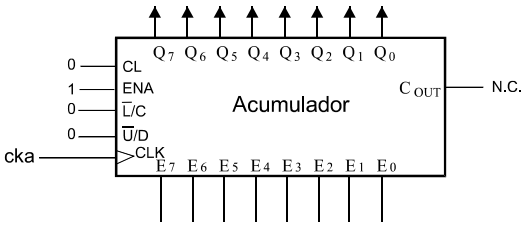
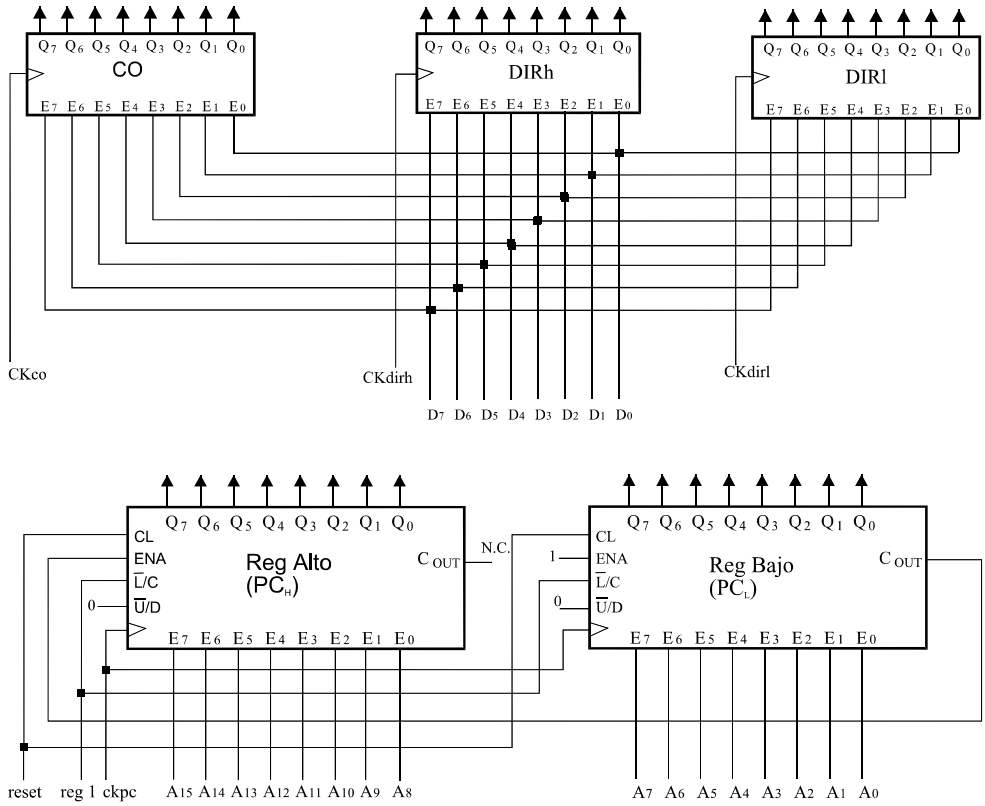


Fig.5.4 Registro Acumulador en el Indalo 1.0



reset	ckpc	reg1	Operación
1	X	X	Inicializa a FFF0 Hexa.
0	Inact.	X	No varía el registro
0	↑	0	Carga paralelo
0	↑	0	Incrementa registro

Fig.5.5 Registro de Instrucción y PC en el Indalo 1.0

	Código máquina	
Ensamblador	Binario	Hexadecimal
ADD A, (address)	0000 0000	00H
MOV A, (address)	0001 0000	10H
MOV (address), A	0010 0000	20H
JMP address	0011 0000	30H

Tabla 5.1. Set de instrucciones de Indalo 1.0.

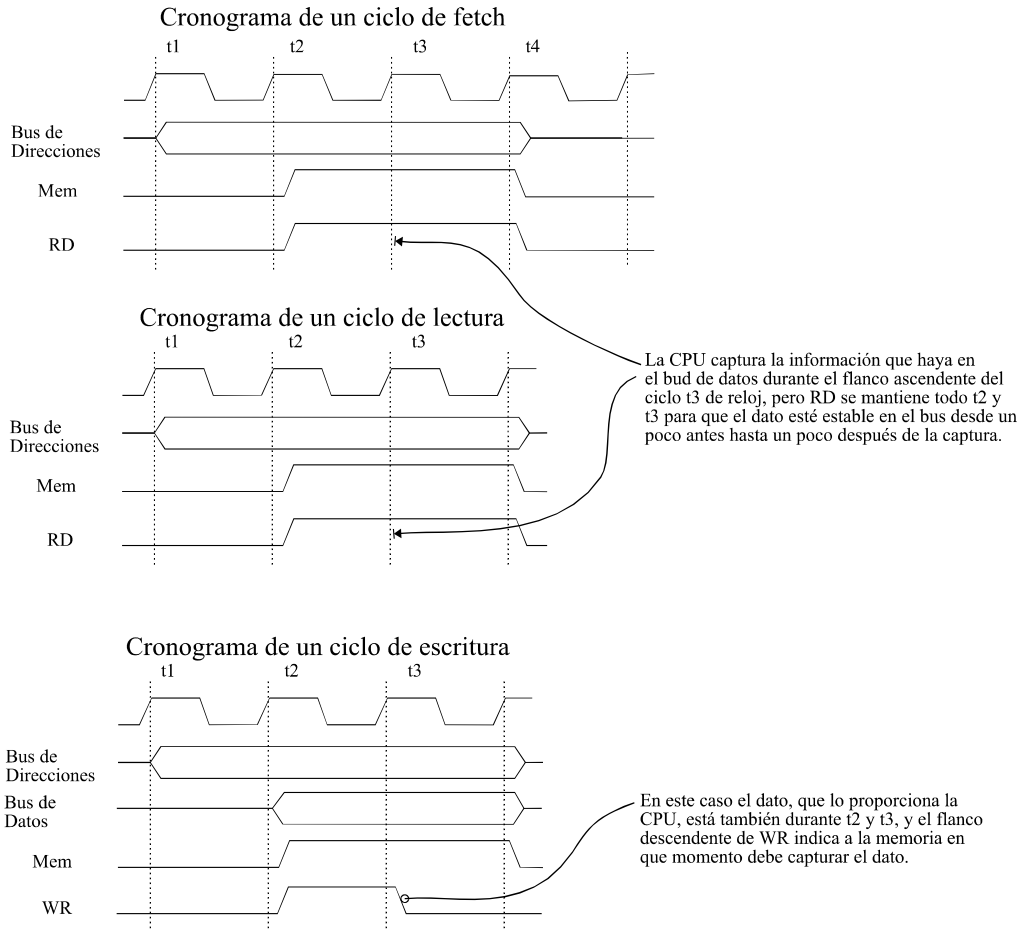


Fig.5.6. Cronogramas de los ciclos de máquina en Indalo 1.0

Operación	Nivel RT			Microórdenes
Primer ciclo de máquina (fetch1)	PC	→	ABus	*salpc*
	(ABus)	→	DBus	*mem*, *rd*
	Dbus	→	CO	*ckco*
	PC + +	→	PC	*reg1, ckpc*
2º ciclo de máquina (fetch2)	PC	→	ABus	*salpc*
	(ABus)	→	DBus	*mem*, *rd*
	DBus	→	DIRL	*ckdirl*
	PC + +	→	PC	*reg1, ckpc*
Tercer ciclo de máquina (fetch3)	PC	→	ABus	*salpc*
	(ABus)	→	DBus	*mem*, *rd*
	DBus	→	DIRH	*ckdirh*
	PC ++	→	PC	*reg1,ckpc*

Tabla 5.2 Ciclos de fetch comunes a las 4 instrucciones.

Operación	Nivel RT			Microórdenes
Ciclo de lectura	DIR	→	ABus	*saldir*
De memoria	(ABus)	→	DBus	*mem*, *rd*
A+(DIR)→A	DBus + A	→	A	*alu0*, *cka*

Tabla 5.3 Ejecución de la instrucción ADD A, (address)

Operación	Nivel RT			Microórdenes
Ciclo de lectura	DIR	→	ABus	*saldir*
De memoria:	(ABus)	→	DBus	*mem*, *rd*
(DIR)→A	DBus	→	A	*cka*

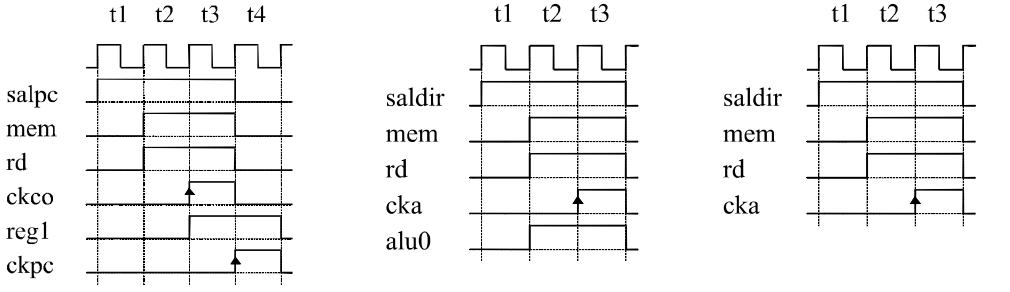
Tabla 5.4.Ejecución de la instrucción MOV A, (address)

Operación	Nivel RT			Microórdenes
Ciclo de escritura	DIR	→	ABus	*saldir*
En memoria	A	→	DBus	*sala*
A→(DIR)	DBus	→	(ABus)	*mem*, *wr*

Tabla 5.5.Ejecución de la instrucción MOV (address), A

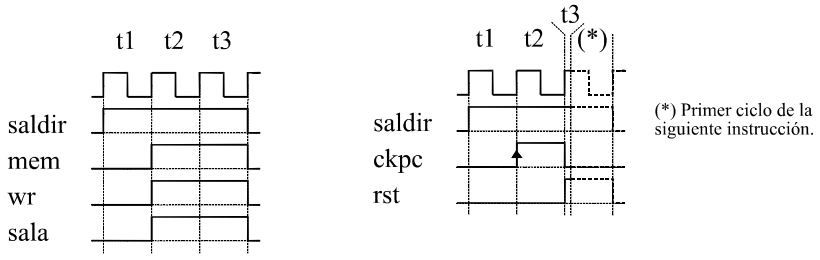
Operación	Nivel RT			Microórdenes
Transferencia interna:	DIR	→	ABus	*saldir*
DIR→PC	ABus	→	PC	*mem*, *wr*

Tabla 5.6. Ejecución de la instrucción JMP address



Ciclo 1 de fetch [*] ***Ejecución de ADD A,(addr)*** ***Ejecución de MOV A,(addr)***

[*] Los ciclos 2º y 3º de fetch son idénticos sustituyendo **ckco** por **ckdirl** y **ckdirh** respectivamente.



Ejecución de MOV (addr),A ***Ejecución de JMP addr***

Fig.5.6. Cronogramas de los ciclos de máquina del INDALO 1.0

	Ciclo fetch 1				**Ciclo fetch 2**				**Ciclo fetch 3**				**Ciclo de ejec**		
estado:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
mem	0	1	1	0	0	1	1	0	0	1	1	0	0	$\overline{b_4} \vee \overline{b_5}$	$\overline{b_4} \vee \overline{b_5}$
ckco	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
reg1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0
ckdirl	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
ckdirh	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
saldir	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
salpc	1	1	1	0	1	1	1	0	1	1	1	0	0	0	0
cka	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\overline{b_5}$
alu0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\overline{b_4} \wedge \overline{b_5}$	$\overline{b_4} \wedge \overline{b_5}$
wr	0	0	0	0	0	0	0	0	0	0	0	0	0	$\overline{b_4} \wedge b_5$	0
sala	0	0	0	0	0	0	0	0	0	0	0	0	0	$\overline{b_4} \wedge b_5$	$\overline{b_4} \wedge b_5$
ckpc	0	0	0	1	0	0	0	1	0	0	0	1	0	$b_4 \wedge b_5$	0
rd	0	1	1	0	0	1	1	0	0	1	1	0	0	$\overline{b_5}$	$\overline{b_5}$
rst	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$b_4 \wedge b_5$

Tabla 5.7. Estado de las microórdenes durante cada ciclo de reloj

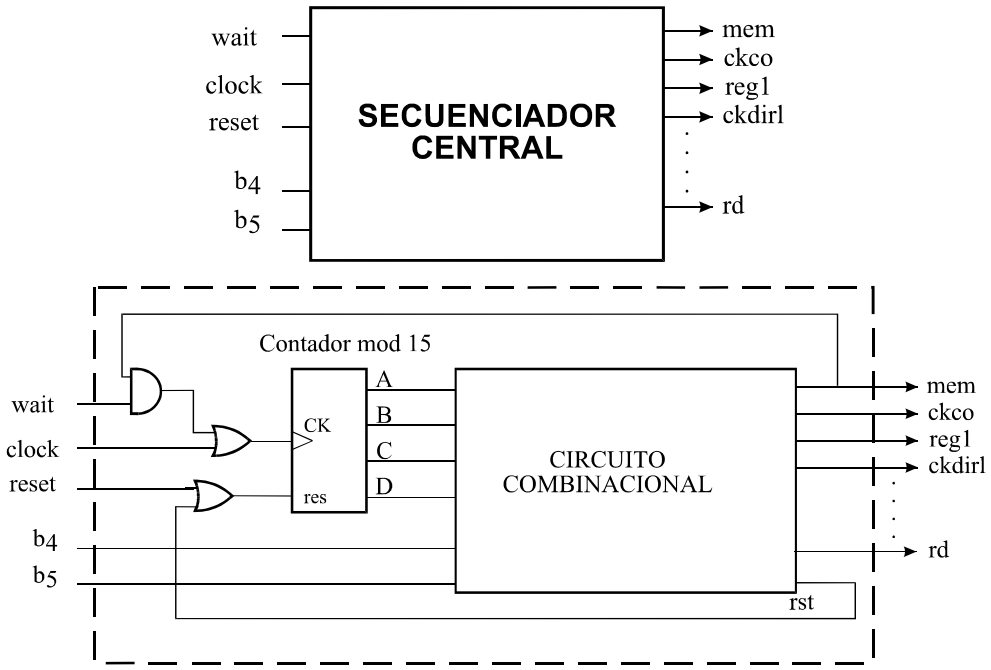


Fig.5.7. Diseño del secuenciador central.

Contador (DCBA)	*Mem*	*ckco*	*reg1*
0000	0	0	0
0001	1	0	0
0010	1	1	1
0011	0	0	1
0100	0	0	0
0101	1	0	0
0110	1	0	1
0111	0	0	1
1000	0	0	0
1001	1	0	0
1010	1	0	1
1011	0	0	1
1100	0	0	0
1101	$\overline{b_4} + \overline{b_5}$	0	0
1110	$\overline{b_4} + \overline{b_5}$	0	0
1111	X	X	X

Simplificando ***mem*** con Karnaugh:

DC \ BA	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	$\overline{b_4}+\overline{b_5}$	X	$\overline{b_4}+\overline{b_5}$
10	0	1	0	1

$$\textbf{mem} = (A + B)(\overline{A} + \overline{B})(\overline{D} + \overline{B} + \overline{b}_4 + \overline{b}_5)$$

ckco se obtiene directamente:

$$\textbf{ckco} = \overline{DCBA}$$

Simplificando ***reg1*** con Karnaugh:

DC \ BA	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	X	0
10	0	0	1	1

$$\textbf{reg1} = \overline{D}B + \overline{C}B = B(\overline{D} + \overline{C})$$

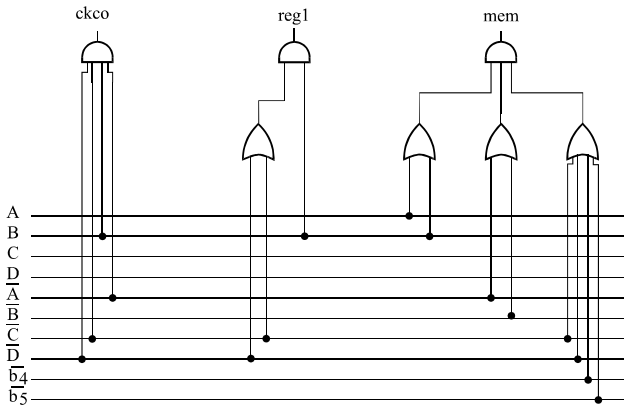


Fig.5.8. Diseño del circuito combinacional para las microórdenes mem, ckco y reg1.

CAPÍTULO 6: INDALO 2.0

JUEGO DE INSTRUCCIONES (por mnemónico)

Instrucción		Descripción	Cod. oper	Flags					Bytes	Cic. Reloj
				C	Z	O	S	P		
ADC	A, op8	A+op8+Fc→A	1001 0op8	X	X	X	X	X	1+A	7+B
ADD	A, op8	A+op8→A	1000 1op8	X	X	X	X	X	1+A	7+B
AND	A, op8	A ∩ op8→A	1001 1op8	0	X	0	X	X	1+A	7+B
CLC		0→Fc	0000 1101	0	N	N	N	N	1	5
CMP	A, op8	A-op8→Act. Flags	1000 0op8	X	X	X	X	X	1+A	7+B
DEC	reg16	reg16--→reg16	0011 00reg16	N	N	N	N	N	1	7
DEC	reg8	reg8--→reg8	1010 00reg8	N	X	X	X	X	1	7
INC	reg16	reg16++→reg16	0011 01reg16	N	N	N	N	N	1	7
INC	reg8	reg8++→reg8	1010 10reg8	N	X	X	X	X	1	7
JC	addr	Si Fc=1 PC+rel8→PC	0001 0001	N	N	N	N	N	2	7/12
JMP	addr	addr→PC	0001 0010	N	N	N	N	N	3	15
JMP	reg16	reg16→PC	0001 11reg16	N	N	N	N	N	1	5
JNC	addr	Si Fc=0 PC+rel8→PC	0001 0011	N	N	N	N	N	2	7/12
JNO	addr	Si Fo=0 PC+rel8→PC	0001 0100	N	N	N	N	N	2	7/12
JNP	addr	Si Fp=0 PC+rel8→PC	0001 0101	N	N	N	N	N	2	7/12
JNS	addr	Si Fs=0 PC+rel8→PC	0001 0110	N	N	N	N	N	2	7/12
JNZ	addr	Si Fz=0 PC+rel8→PC	0001 0111	N	N	N	N	N	2	7/12
JO	addr	Si Fo=1 PC+rel8→PC	0001 1000	N	N	N	N	N	2	7/12
JP	addr	Si Fp=1 PC+rel8→PC	0001 1001	N	N	N	N	N	2	7/12
JS	addr	Si Fs=1 PC+rel8→PC	0001 1010	N	N	N	N	N	2	7/12
JZ	addr	Si Fz=1 PC+rel8→PC	0001 1011	N	N	N	N	N	2	7/12
MOV	reg8, op8	op8→reg8	011reg8 op8	N	N	N	N	N	1+A	7+B
MOV	mem8, reg8	Reg8→mem8	0100 mem8reg8	N	N	N	N	N	1+A	7+B
MOV	reg16,op16	op16→reg16	0101 reg16op16	N	N	N	N	N	1+A	7+B
NEG	A	-A→A	1011 0000	X	X	X	X	X	1	7
NOP		No operar	0000 0000	N	N	N	N	N	1	5
NOT	A	$\overline{A}$ →A	1011 1000	0	X	0	X	X	1	7
OR	A, op8	A ∪ op8→A	1100 0op8	0	X	0	X	X	1+A	7+B
RCL	A	A <rc< →A	1100 1000	X	X	X	X	X	1	7
RCR	A	A >rc> →A	1101 0000	X	X	X	X	X	1	7
ROL	A	A <r< →A	1010 1000	X	X	X	X	X	1	7
ROR	A	A >r> →A	1010 0000	X	X	X	X	X	1	7
SAR	A	A >sa> →A	0100 0000	X	X	0	X	X	1	7
SBB	A, op8	A-(op8+Fc)→A	1101 1op8	X	X	X	X	X	1+A	7+B
SHL	A	A <s< →A	1110 0000	X	X	X	X	X	1	7
SHR	A	A >s> →A	1110 1000	X	X	X	0	X	1	7
STC		1→Fc	0000 1001	1	N	N	N	N	1	5
SUB	A, op8	A-op8→A	1111 0op8	X	X	X	X	X	1+A	7+B
XOR	A, op8	A⊕op8→A	1111 1op8	0	X	0	X	X	1+A	7+B

Nota:
Para los valores de A y B consultar la tabla op8, mem8 y op16
En saltos condicionales JC,... se usan 7 ciclos si no se cumple la condición.

LÍNEAS DE CONTROL DE LA ALU.

Operación	alu3	alu2	alu1	alu0
ENT2	0	0	0	0
No usada	0	0	0	1
ENT1 + ENT2 + Fc	0	0	1	0
ENT1 ∩ ENT2	0	0	1	1
ENT2 --	0	1	0	0
ENT2 ++	0	1	0	1
Ca2(ENT1)	0	1	1	0
Ca1(ENT1)	0	1	1	1
ENT1 ∪ ENT2	1	0	0	0
ENT1 <rc<	1	0	0	1
ENT1 >rc>	1	0	1	0
ENT1 + Ca2(ENT2 + Fc)	1	0	1	1
ENT1 <r<	1	1	0	0
ENT1 >r>	1	1	0	1
ENT1 >sa>	1	1	1	0
ENT1 ⊕ ENT2	1	1	1	1

JUEGO DE INSTRUCCIONES (por código de operación).

Cod. Oper	Instrucción		Bytes
0000 0000	NOP		1
0000 1001	STC		1
0000 1101	CLC		1
0001 0001	JC	addr	2
0001 0010	JMP	addr	3
0001 0011	JNC	addr	2
0001 0100	JNO	addr	2
0001 0101	JNP	addr	2
0001 0110	JNS	addr	2
0001 0111	JNZ	addr	2
0001 1000	JO	addr	2
0001 1001	JP	addr	2
0001 1010	JS	addr	2
0001 1011	JZ	addr	2
0001 11reg16	JMP	reg16	1
0011 00reg16	DEC	reg16	1
0011 01reg16	INC	reg16	1
0100 0000	SAR	A	1
0100 mem8reg8	MOV	mem8, reg8	1+A
0101 reg16op16	MOV	reg16,op16	1+A
011reg8 op8	MOV	reg8, op8	1+A
1000 0op8	CMP	A, op8	1+A
1000 1op8	ADD	A, op8	1+A
1001 0op8	ADC	A, op8	1+A
1001 1op8	AND	A, op8	1+A
1010 0000	ROR	A	1
1010 00reg8	DEC	reg8	1
1010 1000	ROL	A	1
1010 10reg8	INC	reg8	1
1011 0000	NEG	A	1
1011 1000	$\overline{\text{NOT}}$	A	1
1100 0op8	OR	A, op8	1+A
1100 1000	RCL	A	1
1101 0000	RCR	A	1
1101 1op8	SBB	A, op8	1+A
1110 0000	SHL	A	1
1110 1000	SHR	A	1
1111 0op8	SUB	A, op8	1+A
1111 1op8	XOR	A, op8	1+A

OPERANDOS DE 8 BITS.

Cód.	Mnem.	Dirección del operando	A	B
000	(addr)	La dirección del operando aparece en los 2 bytes siguientes al Cod. oper.	2	11
001	A	Operando en el registro A	0	0
010	B	Operando en el registro B	0	0
011	C	Operando en el registro C	0	0
100	dat8	Valor inmediato viene en el byte siguiente al Cod. oper	1	3
101	(BC)	Operando en la dirección apuntada por BC	0	1
110	(X+rel8)	Operando en la dirección apuntada por X+ rel8	1	6

Reg8	
01	A
10	B
11	C

Mem8		A	B
00	(addr)	2	11
01	(BC)	0	1
10	(X+rel8)	1	6

OPERANDOS DE 16 BITS.

Cód.	Mnem.	Dirección del operando	A	B
00	BC	Operando en el registro BC	0	2
01	X	Operando en el registro X	0	2
10	dat16	Valor inmediato viene en los 2 bytes siguientes al Cod. op.	2	8

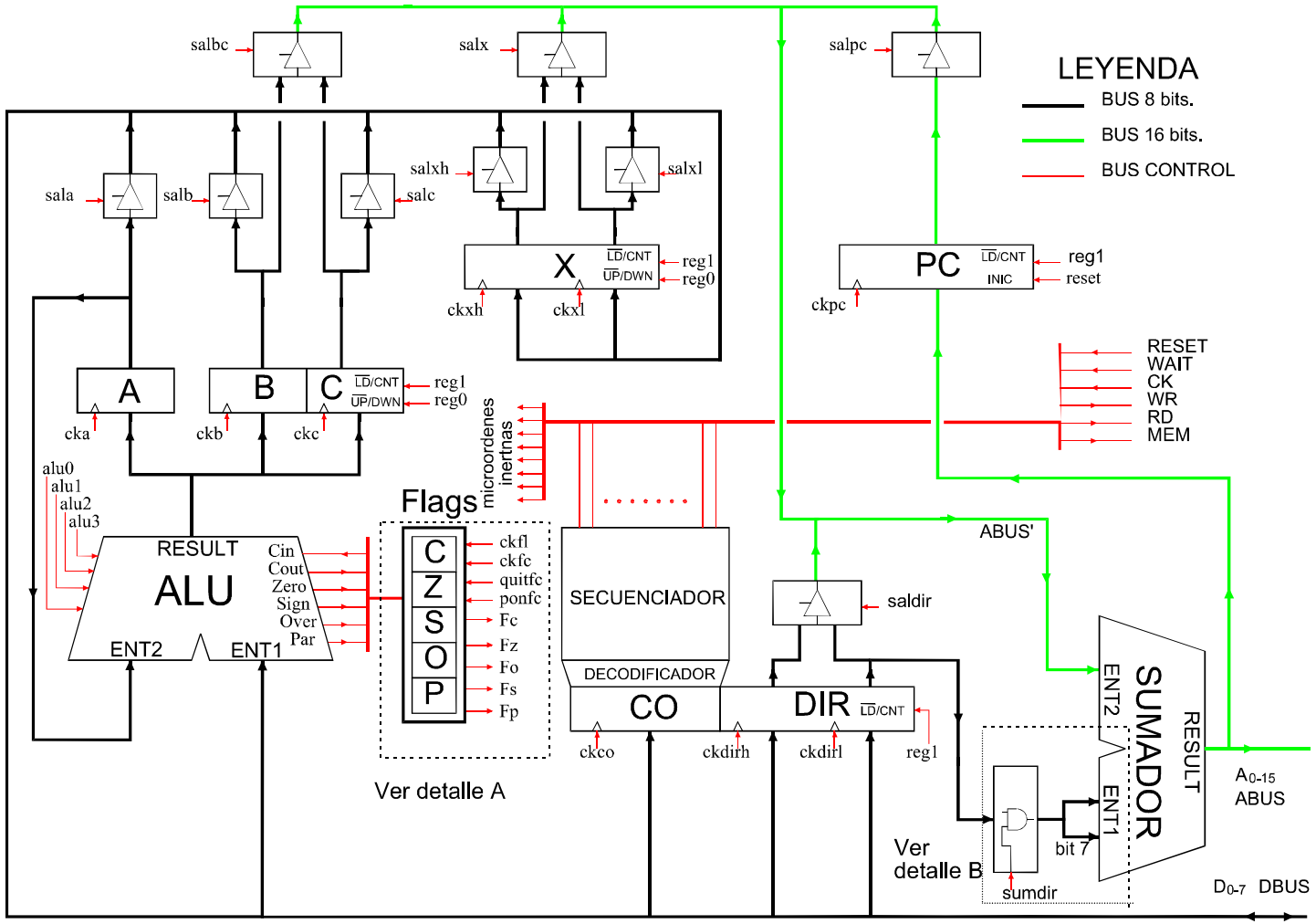
Reg16	
00	BC
01	X

NOTAS Y SÍMBOLOS.

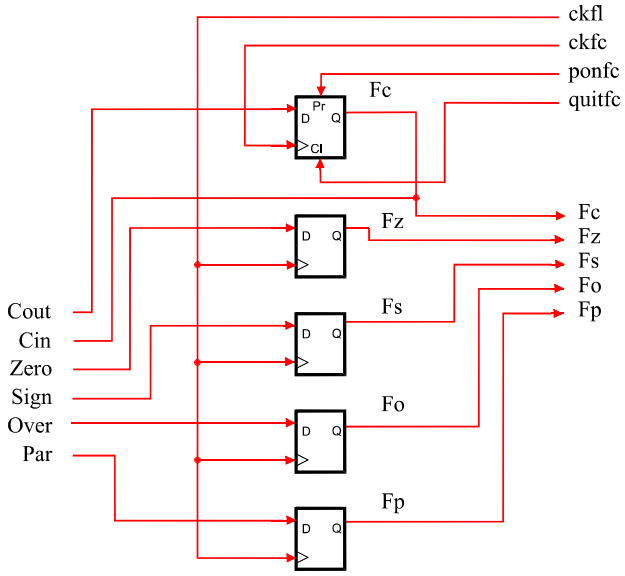
Flags: C=Carry, Z=cero, O=Overflow, P=paridad.
X= Flag se actualiza con la ejecución de la inst.
N=Flag no cambia con la ejecución de la inst.
0: Flag se pone a 0
1: Flag se pone a 1
Fc: Flag de Carry
op8: Operando de 8 bits.
addr: Dirección (Entero sin signo de 16 bits)
rel8: Entero con signo de 8 bits.

Flags
C: Contiene el acarreo en las operaciones de suma. Funciona como Borrow en las de resta y comparación. En las operaciones de desplazamiento y rotación, contiene el bit saliente.
Z: Vale 1 si el contenido del resultado es =0.
O: Realiza una operación XOR entre los acarreos de los 2 bits mas significativos de los operandos. En Shifts y Rotaciones se pone a 1 si el bit de mayor peso cambia como consecuencia de la rotación.
S: Es una copia del bit de mayor peso del resultado.
P: Es 0 si el número de unos en el resultado es impar.

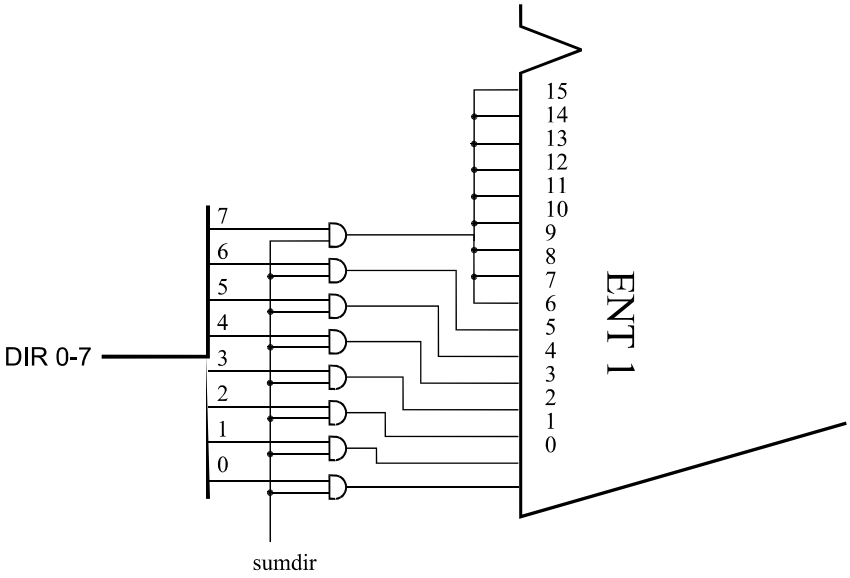
ESQUEMA DEL INDALO 2.0



- **CONFIGURACIÓN DEL REGISTRO DE FLAGS**



- **DETALLE DEL SUMADOR DE DIRECCIÓN**



CAPÍTULO 7: INDALO 3.0

JUEGO DE INSTRUCCIONES (por mnemónico)

Instrucción		Descripción	Cod. oper	Flags						Bytes	Cic. Reloj
				C	Z	O	S	P	I		
ADC	A, op8	A+op8+Fc→A	1001 0op8	X	X	X	X	X	N	1+A	7+B
ADD	A, op8	A+op8→A	1000 1op8	X	X	X	X	X	N	1+A	7+B
AND	A, op8	A ∩ op8→A	1001 1op8	0	X	0	X	X	N	1+A	7+B
CALL	addr	SP-2→SP, PC→ (SP), addr→PC	0000 1011	N	N	N	N	N	N	3	27
CLC		0→Fc	0000 1101	0	N	N	N	N	N	1	5
CLI		0→Fi	0000 1100	N	N	N	N	N	0	1	5
CMP	A, op8	A-op8→Act. Flags	1000 0op8	X	X	X	X	X	N	1+A	7+B
DEC	reg16	reg16-- →reg16	0011 00reg16	N	N	N	N	N	N	1	7
DEC	reg8	reg8-- →reg8	1010 00reg8	N	X	X	X	X	N	1	7
HLT		Paso a Halt	0000 1110	N	N	N	N	N	N	1	5
IN	A, C	(puerto C)→A	0000 1111	N	N	N	N	N	N	1	8
INC	reg16	reg16++→reg16	0011 01reg16	N	N	N	N	N	N	1	7
INC	reg8	reg8++→reg8	1010 10reg8	N	X	X	X	X	N	1	7
INT	vector	SP-2→SP, PC→(SP), (V*2)→PC	0001 0000	N	N	N	N	N	N	2	30
JC	addr	Si Fc=1 PC+rel8→PC	0001 0001	N	N	N	N	N	N	2	7/12
JMP	addr	addr→PC	0001 0010	N	N	N	N	N	N	3	15
JMP	reg16	reg16→PC	0001 11reg16	N	N	N	N	N	N	1	5
JNC	addr	Si Fc=0 PC+rel8→PC	0001 0011	N	N	N	N	N	N	2	7/12
JNO	addr	Si Fo=0 PC+rel8→PC	0001 0100	N	N	N	N	N	N	2	7/12
JNP	addr	Si Fp=0 PC+rel8→PC	0001 0101	N	N	N	N	N	N	2	7/12
JNS	addr	Si Fs=0 PC+rel8→PC	0001 0110	N	N	N	N	N	N	2	7/12
JNZ	addr	Si Fz=0 PC+rel8→PC	0001 0111	N	N	N	N	N	N	2	7/12
JO	addr	Si Fo=1 PC+rel8→PC	0001 1000	N	N	N	N	N	N	2	7/12
JP	addr	Si Fp=1 PC+rel8→PC	0001 1001	N	N	N	N	N	N	2	7/12
JS	addr	Si Fs=1 PC+rel8→PC	0001 1010	N	N	N	N	N	N	2	7/12
JZ	addr	Si Fz=1 PC+rel8→PC	0001 1011	N	N	N	N	N	N	2	7/12
MOV	reg8, op8	op8→reg8	011reg8 op8	N	N	N	N	N	N	1+A	7+B
MOV	mem8, reg8	Reg8→mem8	0100 mem8reg8	N	N	N	N	N	N	1+A	7+B
MOV	reg16,op16	op16→reg16	0101 reg16op16	N	N	N	N	N	N	1+A	7+B
NEG	A	-A→A	1011 0000	X	X	X	X	X	N	1	7
NOP		No operar	0000 0000	N	N	N	N	N	N	1	5
NOT	A	$\overline{A}$ →A	1011 1000	0	X	0	X	X	N	1	7
OR	A, op8	A ∪ op8→A	1100 0op8	0	X	0	X	X	N	1+A	7+B
OUT	C, A	A→(puerto C)	0010 0000	N	N	N	N	N	N	1	8
POP	AF	(SP)→AF, SP+2→SP	0000 0001	X	X	X	X	X	X	1	15
POP	BC	(SP)→BC, SP+2→SP	0000 0010	N	N	N	N	N	N	1	15
POP	X	(SP)→X, SP+2→SP	0000 0011	N	N	N	N	N	N	1	15
PUSH	AF	SP-2→SP, AF→(SP)	0000 0101	N	N	N	N	N	N	1	15
PUSH	BC	SP-2→SP, BC→(SP)	0000 0110	N	N	N	N	N	N	1	15
PUSH	X	SP-2→SP, X→(SP)	0000 0111	N	N	N	N	N	N	1	15
RCL	A	A <rc< →A	1100 1000	X	X	X	X	X	N	1	7
RCR	A	A >rc> →A	1101 0000	X	X	X	X	X	N	1	7
RET		(SP)→PC, SP+2→SP	0000 1000	N	N	N	N	N	N	1	13
ROL	A	A <r< →A	1010 1000	X	X	X	X	X	N	1	7
ROR	A	A >r> →A	1010 0000	X	X	X	X	X	N	1	7
SAR	A	A >sa> →A	0100 0000	X	X	0	X	X	N	1	7
SBB	A, op8	A-(op8+Fc) →A	1101 1op8	X	X	X	X	X	N	1+A	7+B
SHL	A	A <s< →A	1110 0000	X	X	X	X	X	N	1	7
SHR	A	A >s> →A	1110 1000	X	X	X	0	X	N	1	7
STI		1→Fi	0000 1010	N	N	N	N	N	1	1	5
STC		1→Fc	0000 1001	1	N	N	N	N	N	1	5
SUB	A, op8	A-op8→A	1111 0op8	X	X	X	X	X	N	1+A	7+B
XOR	A, op8	A⊕op8→A	1111 1op8	0	X	0	X	X	N	1+A	7+B

Para los valores de A y B consultar las tablas op8, op16 y mem8.
En saltos condicionales 7/12 quiere decir que la duración es de 7 ciclos de reloj si no se

JUEGO DE INSTRUCCIONES
(por código de operación)

Cod. Oper	Instrucción		Bytes
0000 0000	NOP		1
0000 0001	POP	AF	1
0000 0010	POP	BC	1
0000 0011	POP	X	1
0000 0101	PUSH	AF	1
0000 0110	PUSH	BC	1
0000 0111	PUSH	X	1
0000 1000	RET		1
0000 1001	STC		1
0000 1010	STI		1
0000 1011	CALL	addr	3
0000 1100	CLI		1
0000 1101	CLC		1
0000 1110	HLT		1
0000 1111	IN	A, C	1
0001 0000	INT	vector	2
0001 0001	JC	addr	2
0001 0010	JMP	addr	3
0001 0011	JNC	addr	2
0001 0100	JNO	addr	2
0001 0101	JNP	addr	2
0001 0110	JNS	addr	2
0001 0111	JNZ	addr	2
0001 1000	JO	addr	2
0001 1001	JP	addr	2
0001 1010	JS	addr	2
0001 1011	JZ	addr	2
0001 11reg16	JMP	reg16	1
0010 0000	OUT	C, A	1
0011 00reg16	DEC	reg16	1
0011 01reg16	INC	reg16	1
0100 0000	SAR	A	1
0100 mem8reg8	MOV	mem8,reg8	1+A
0101 reg16op16	MOV	reg16,op16	1+A
011reg8 op8	MOV	reg8,op8	1+A
1000 0op8	CMP	A,op8	1+A
1000 1op8	ADD	A,op8	1+A
1001 0op8	ADC	A,op8	1+A
1001 1op8	AND	A,op8	1+A
1010 00reg8	DEC	reg8	1
1010 1000	ROL	A	1
1010 10reg8	INC	reg8	1
1011 0000	NEG	A	1
1011 1000	NOT	A	1
1100 0op8	OR	A,op8	1+A
1100 1000	RCL	A	1
1101 0000	RCR	A	1
1101 1op8	SBB	A,op8	1+A
1110 0000	SHL	A	1
1110 1000	SHR	A	1
1111 0op8	SUB	A,op8	1+A
1111 1op8	XOR	A,op8	1+A

op8 : Operando de 8 bits.
Los códigos de operación para las diferentes formas de direccionamiento de op8 son:

Cód.	Mnem.	Dirección del operando	A	B
000	(addr)	La dirección del operando aparece en los 2 bytes siguientes al Cod. oper.	2	11
001	A	Operando en el registro A	0	0
010	B	Operando en el registro B	0	0
011	C	Operando en el registro C	0	0
100	dat8	Valor inmediato viene en el byte siguiente al Cod. Oper	1	3
101	(BC)	Operando en la dirección apuntada por BC	0	1
110	(X+rel8)	Operando en la dirección apuntada por X+ rel8.	1	6

Reg8	
01	A
10	B
11	C

mem8		A	B
00	(addr)	2	11
01	(BC)	0	1
10	(X+rel8)	1	6

op16 ✚ Operando de 16 bits.
Los códigos de operación para las diferentes formas de direccionamiento de op16 son:

Cód.	Mnem.	Dirección del operando	A	B
00	BC	Operando en el registro BC	0	2
01	X	Operando en el registro X	0	2
10	dat16	Valor inmediato viene en los 2 bytes siguientes al Cod.oper	2	8
11	SP	Operando en el registro SP	0	2

Reg16	
00	BC
01	X
10	SP

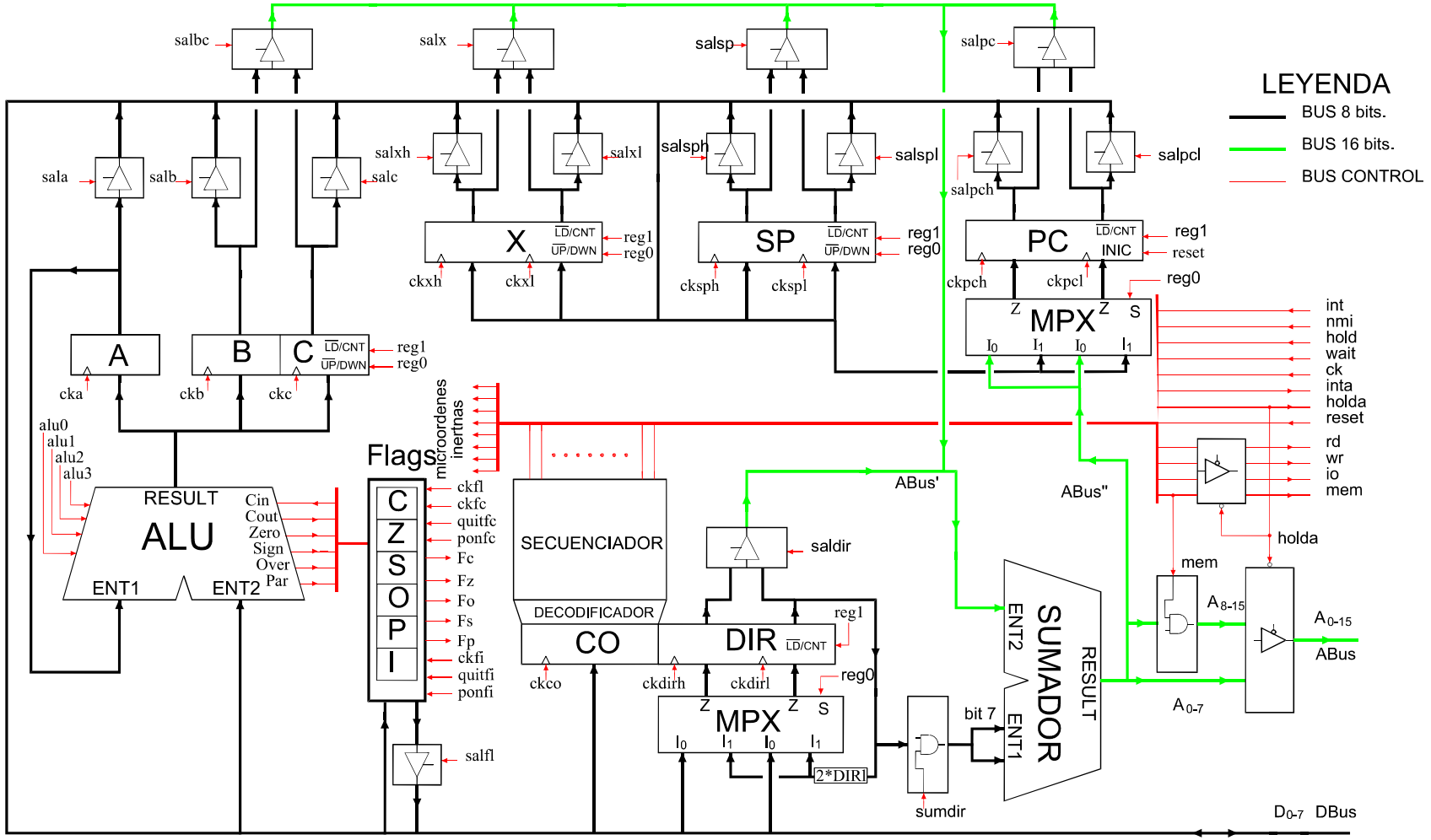
addr: Direc. (Entero sin signo de 16 bits)
rel8: Entero con signo de 8 bits.
Flags:
0: Flag se pone a 0
1: Flag se pone a 1
X= Flag se actualiza con la ejecución de la inst.
N=Flag no cambia con la ejecución de la inst.C:
Es el acarreo en las sumas, el borrow en las restas, y en desplaz. y rotaciones, es el bit saliente.
Z: Vale 1 si el contenido del resultado es =0.
O: Realiza una operación XOR entre los acarreos de los 2 bits mas significativos de los operandos. En Shifts y Rotaciones se pone a 1 si el bit de mayor peso cambia como consecuencia de la operación.
S: Es una copia del bit de mayor peso del resultado.
P: Es 0 si el número de unos en el resultado es impar.

OPERACIONES BÁSICAS DE INDALO 3.0

Operación	Nro. ciclos reloj	Transferencias	Instrucciones y notas
F	5	(PC)→CO; PC++	Casi todas. (1)
		(PC)→DIR_l; PC++	(2)
		(PC)→DIR_h; PC++	(3)
		(PC)→X_l, X_h, SP_l o SP_h; PC++	MOV X,*dat16*; MOV SP,*dat16*
F_{alu}	5	A-(PC)→Act. Flags; PC++	CMP A,*dat8*
		A *oper2* (PC)→A; PC++	XXX A,*dat8*
		(PC)→*reg8*; PC++	MOV *reg8*,dat8
F_{flag}	5	(PC)→CO; PC++; 0→F_C o F_I	CLC; CLI; ADD A,*op8*; SUB A,*op8*; CMP A,*op8*; SHR A; SHL A
		(PC)→CO; PC++; 1→F_C o F_I	STC; STI
F_{noinc}	3	(PC)→CO	JMP *reg16*; RET
		(PC)→DIR_h	JMP *addr*
R	3	(SP)→F, X_l, X_h, PC_l o PC_h	POP AF; POP X; RET
		(DIR)→PC_l ó PC_h	INT *vector* (6)
R_{alu}	3	A-*mem8*→Act. Flags	CMP A,*mem8*
		A *oper2 mem8*→A	XXX A,*mem8* (4)
		mem8→*reg8*	MOV *reg8*,*mem8*
		(SP)→*reg8*	POP AF; POP BC
W	3	*reg8*→*mem8*	MOV *mem8*,*reg8*
		reg8→(SP)	PUSH AF; PUSH BC
		F, X_l, X_h, PC_l o PC_h→(SP)	PUSH AF; PUSH X; CALL *addr*; INT *vector* (6)
IP	3	(puerto C)→A	IN A,C
OP	3	A→(puerto C)	OUT C,A
RI	2	*Vector*→DIR_l	Interrupciones hardware
I_{t8}	2	*reg16l*→X_l o SP_l; *reg16h*→X_h o SP_h	MOV X,*reg16* MOV SP,*reg16*
I_{talu}	2	A-*reg8*→Act. Flags	CMP A,*reg8*
		A *oper2 reg8*→A	XXX A,*reg8* (4)
		reg8→*reg8*	MOV *reg8*,*reg8*
		A *oper1*→A	YYY A (5)
		reg8++ ó *reg8*--	INC *reg8*; DEC *reg8*
		reg16l→C o *reg16h*→B	MOV BC,*reg16*
I_{t16}	2	PC+DIR_l→PC	J*cond addr* (si se cumple la condición)
		DIR→PC	JMP *addr*; CALL *addr*
		reg16→PC	JMP *reg16*
I_{id16}	2	*reg16*++ o *reg16*--	INC *reg16*; DEC *reg16*
		PC++	Jcond addr (si no se cumple la condición)
		SP--	PUSH AF; PUSH BC; PUSH X; CALL *addr*;INT *vector* (6)
		SP++	POP AF; POP BC; POP X; RET
		DIR++	INT *vector* (6)
I_{2dirl}	2	2*DIR_l→DIR	INT *vector* (6)

NOTAS:
(1) Se exceptúan:
- CLC, CLI, ADD, SUB, CMP, SHR y SHL que leen el código de operación mediante F_{flag}.
- JMP *reg16* y RET que lo hacen por medio de F_{noinc}.
(2) Producen esta transferencia:
- Las instrucciones que contienen el operando *rel8*, excepto J*cond addr* cuando no se cumple la condición.
- Las instrucciones que contienen el operando *addr*.
- INT *vector*.
(3) Producen esta transferencia las instrucciones que contienen el operando *addr*, excepto JMP *addr* que emplea F_{noinc}. Nótese que el operando de J*cond addr* no es *addr* sino *rel8*.
(4) *Oper2* es cualquiera de las operaciones binarias que puede realizar la CPU: suma con carry, resta con borrow, AND, OR u OR exclusivo; XXX es cualquiera de los mnemónicos de las instrucciones que realizan operaciones binarias: ADD, ADC, SUB, SBB, AND, OR o XOR.
(5) *Oper1* es cualquiera de las operaciones unarias que puede realizar la CPU: rotaciones (circulares y a través de carry; a derecha y a izquierda); shift aritmético a la derecha, y complementos (a 1 y a 2). YYY es cualquiera de los mnemónicos de las instrucciones que realizan las operaciones anteriores: ROR, ROL, RCR, RCL, SHL, SAR, SAL, NOT y NEG.
(6) También se producen estas operaciones en el proceso de atención a las interrupciones hardware.

ESQUEMA DEL INDALO 3.0



CAPÍTULO 8: INTERFACES DE ENTRADA/SALIDA

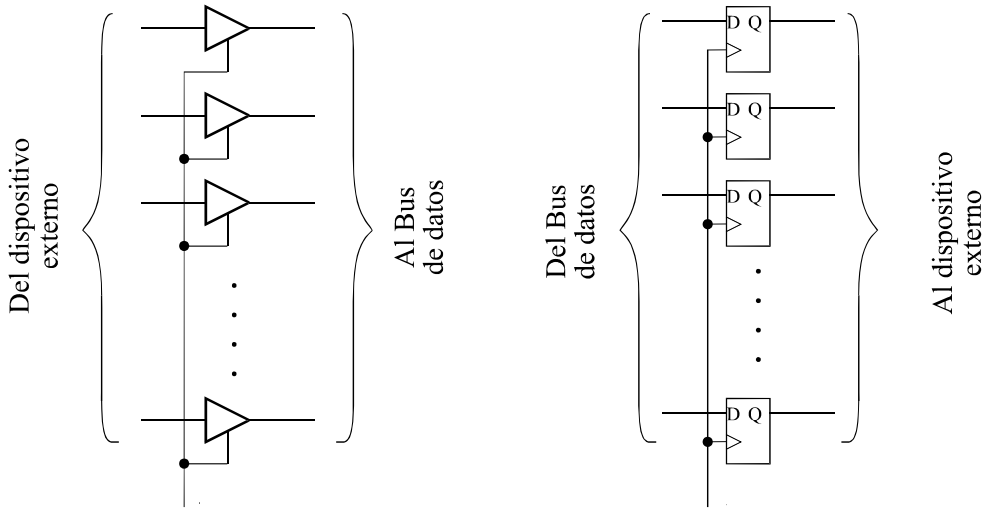


Fig.8.1. Buffer triestado y Biestables como elementos básicos de E/S.

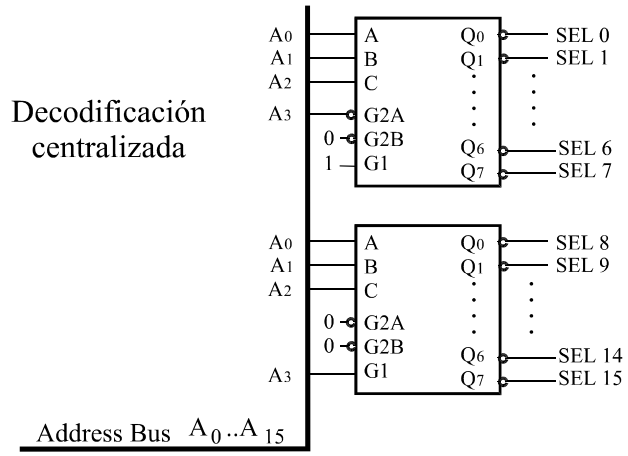


Fig.8.2 Decodificador centralizado (varias líneas de selección con un circuito).

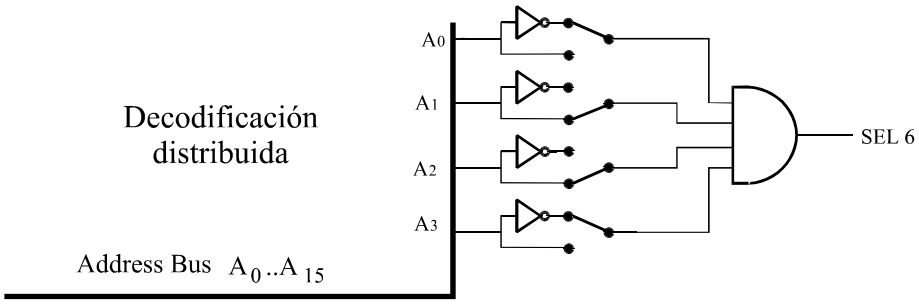


Fig.8.3 Decodificación de direcciones de E/S distribuida.

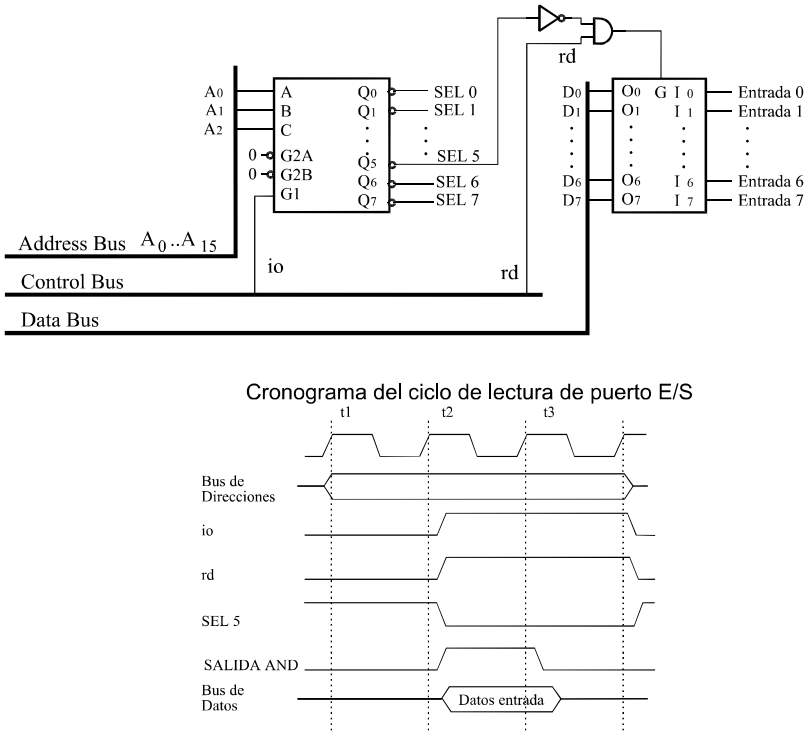


Fig.8.4 Circuito de selección del puerto 5 para entrada y cronograma de funcionamiento.

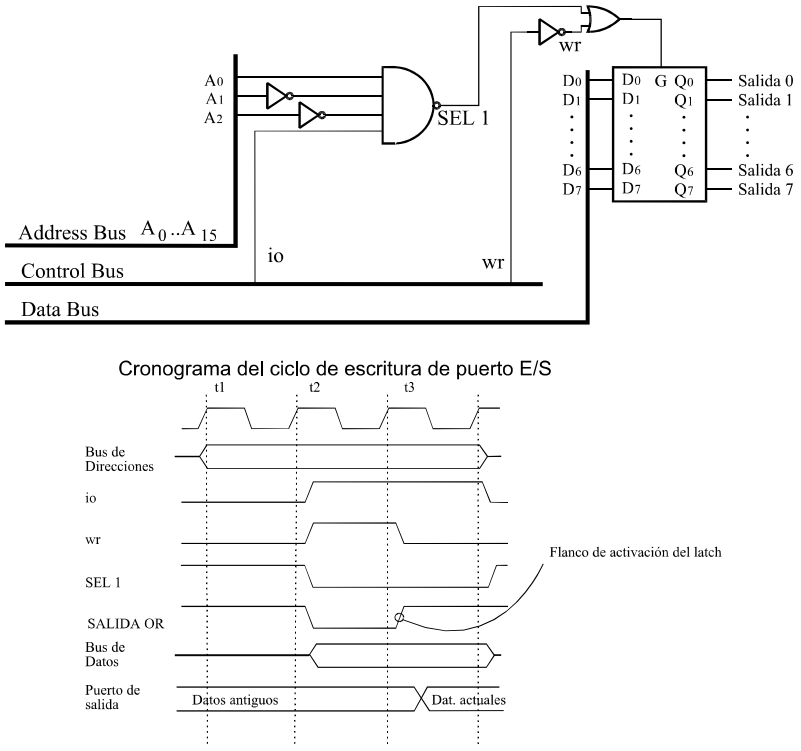


Fig.8.5 Circuito de selección del puerto 1 para escritura, y cronográma de funcionamiento.

Tipo	Nombre	Direcc.	Descripción
Datos	DATA 1	Salida	Líneas de datos . En ellas se coloca el carácter a imprimir. Deben contener el dato al menos 0,5 µS antes de activar STROBE y 0,5 µS después de desactivarlo.
Datos	DATA 2	Salida	
Datos	DATA 3	Salida	
Datos	DATA 4	Salida	
Datos	DATA 5	Salida	
Datos	DATA 6	Salida	
Datos	DATA 7	Salida	
Datos	DATA 8	Salida	
Control	$\overline{\text{STROBE}}$	Salida	Indica a la impresora que se envía un byte
Control	$\overline{\text{AUTO FEED}}$	Salida	Indica a la impresora que avance una línea después caráct. CR
Control	$\overline{\text{INIT}}$	Salida	Resetea o inicializa a la impresora.
Control	$\overline{\text{SLCT IN}}$	Salida	(Select In) Indica a la impresora que está seleccionada
Estado	$\overline{\text{ACK}}$	Entrada	Indica que el último carácter ha sido recibido y procesado
Estado	BUSY	Entrada	Indica que la impresora está ocupada y no puede aceptar datos
Estado	PE	Entrada	(Paper End) Indica que la impresora no tiene papel.
Estado	SLCT	Entrada	(Select) Indica que la impresora está seleccionada
Estado	$\overline{\text{ERROR}}$	Entrada	Indica que se ha producido algún error en la impresora.

Tabla 8.1. Líneas del Interface Centronics.

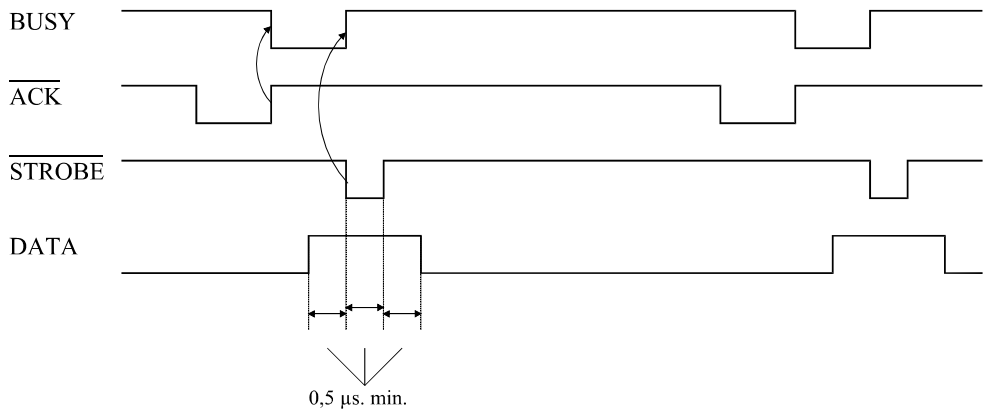


Fig.8.6 Conexión de cuatro displays usando un único latch.

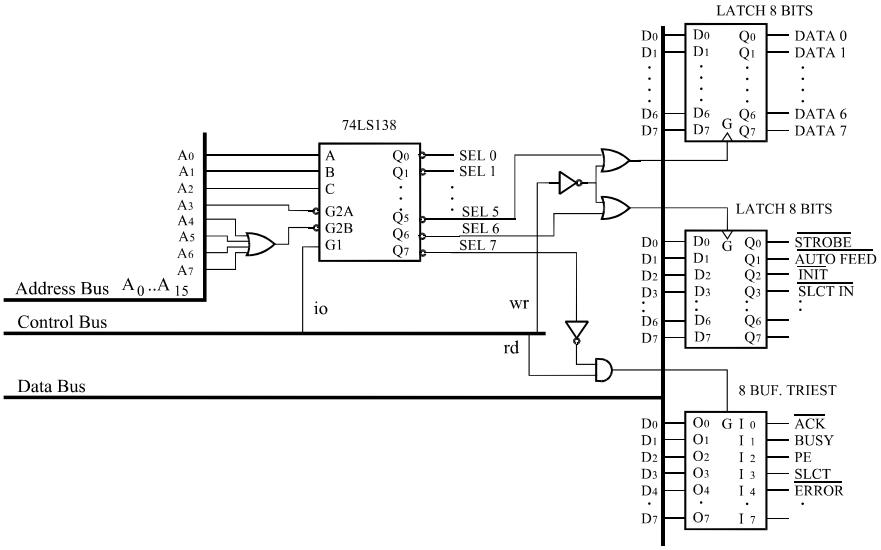


Fig.8.7 Circuito de control del interface Centronics.

Valor en A	Causa	BUSY	PE	$\overline{\text{ERROR}}$
-	Impresora preparada	0	0	1
1	Impresora ocupada (generalmente cuando está procesando el último byte recibido o tiene el buffer lleno)	1	0	1
2	Impresora sin papel	1	1	0
3	Off Line u otros errores	1	0	0

Tabla 8.2. Estados posibles en el interfaz Centronics.

Fig. 8.8. Ordinograma comprobación estado impresora.