PROCESADOR PDUA

Versión 0.0	Arquitectura básica	Mauricio Guerrero H.	04/2007
Revisión 0.1	Microprograma y Memoria RAM doble Puerto	Mauricio Guerrero H. Diego Méndez Ch.	11/2007
Revisión 0.2	ALU Bit-Slice	Mauricio Guerrero H.	03/2008

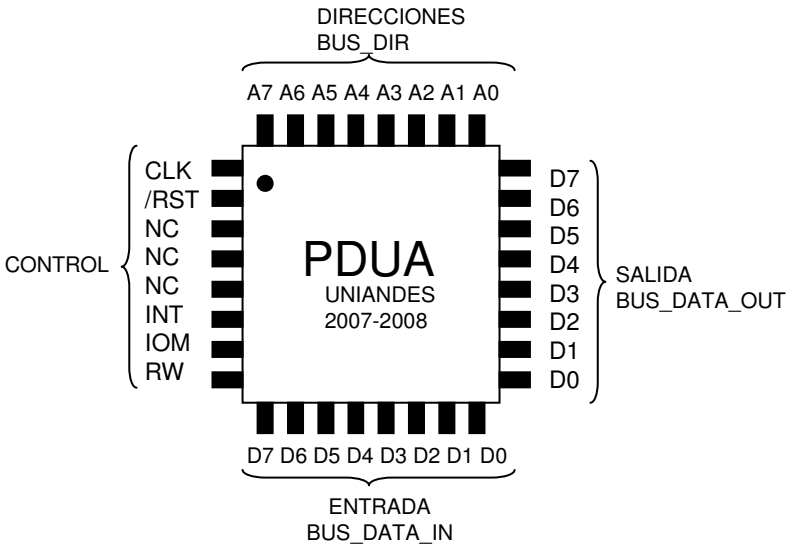
ARQUITECTURA BASICA

Características Generales

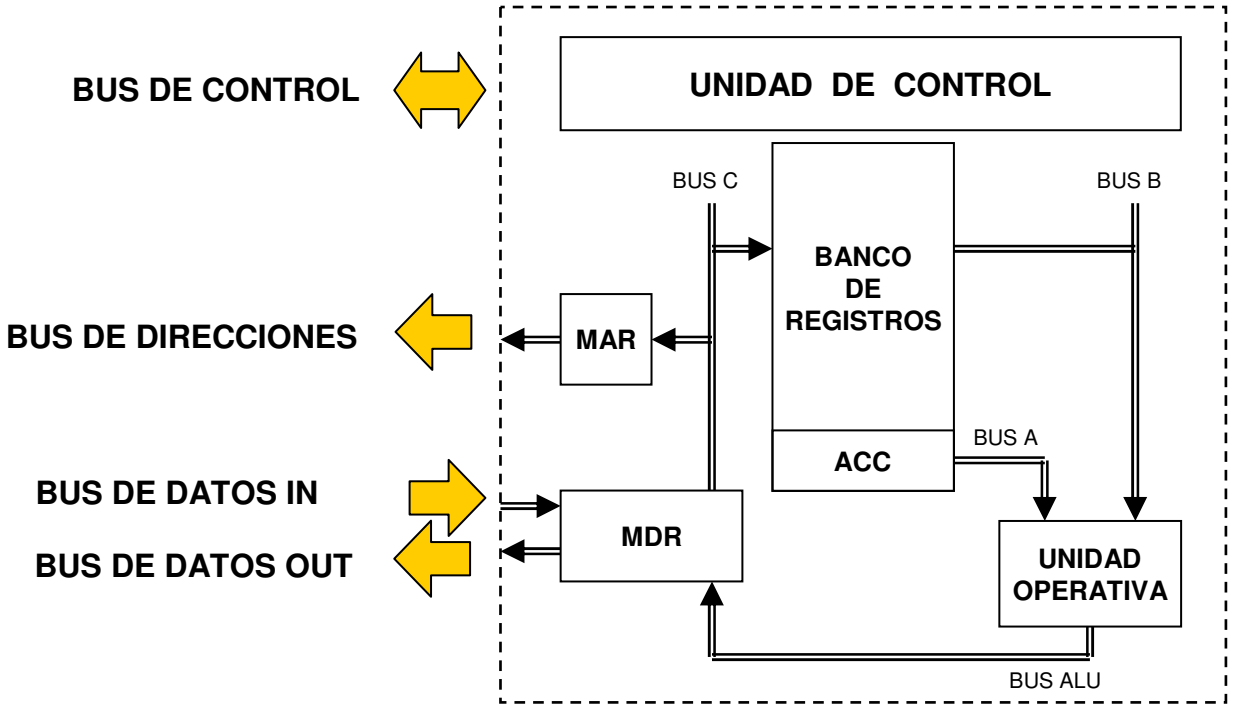
- Procesador de 8 bits
- Espacio de direccionamiento de 256 Bytes (8 bits de dirección)
- Memoria y Periféricos separados
- Control microprogramado
- Manejo de interrupción externa
- Manejo de STACK

Unidades:

- Unidad de control
- Unidad operativa
- Banco de registros
- Interfaz con memoria

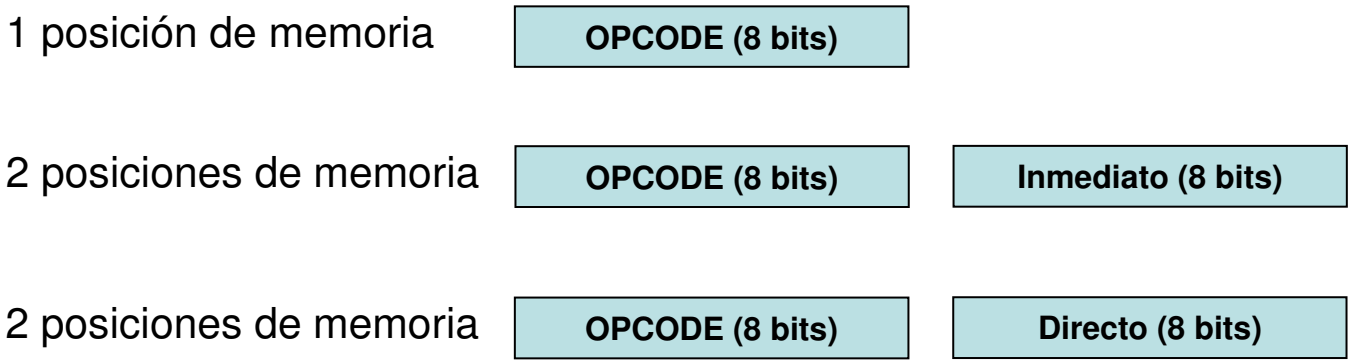


ARQUITECTURA BASICA



- **BUS DE DIRECCIONES: 8 Bits (Espacio de memoria 256 Bytes, I/O de 256)**
- **BUS DE DATOS: 8 Bits**
- **BUS DE CONTROL: CLK, RST, INT, IO/M, R/W**

FORMATO DE INSTRUCCION



CONJUNTO DE INSTRUCCIONES

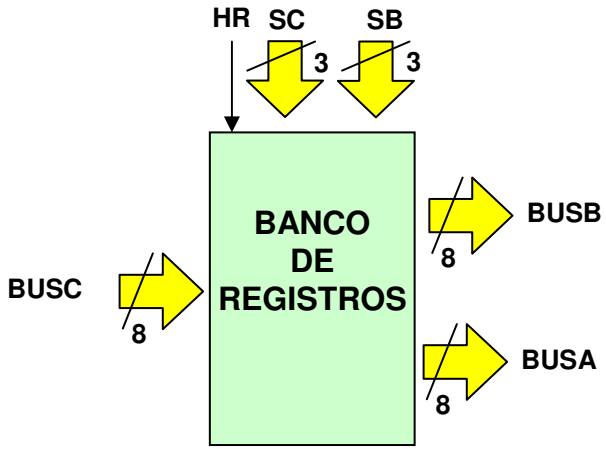
OPCODE	NEMONICO	Bytes	BANDERAS
00000000	RESERVADO	--	--
00001xxx	MOV ACC,A	1	SI
00010xxx	MOV A,ACC	1	SI
00011xxx	MOV ACC,CTE	2	NO
00100xxx	MOV ACC,[DPTR]	1	NO
00101xxx	MOV DPTR,ACC	1	SI
00110xxx	MOV [DPTR],ACC	1	SI
00111xxx	INV ACC	1	SI
01000xxx	AND ACC,A	1	SI
01001xxx	ADD ACC,A	1	SI
01010xxx	JMP CTE	2	NO
01011xxx	JZ CTE	2	NO
01100xxx	JN CTE	2	NO
01101xxx	JC CTE	2	NO
01110xxx	CALL DIR	2	NO
01111xxx	RET	2	NO
1xxxx---	DISPONIBLE	1	??

ENTIDAD

Entidad de la arquitectura completa:

```
entity PDUA is
   Port ( clk            : in     std_logic;
          rst_n          : in     std_logic;
          int            : in     std_logic;
          iom            : out    std_logic;          -- IO=0,M=1
          rw             : out    std_logic;          -- R=0,W=1
          bus_dir        : out    std_logic_vector(7 downto 0);
          bus_data_in    : in     std_logic_vector(7 downto 0);
          bus_data_out   : out    std_logic_vector(7 downto 0));
end PDUA;
```

BANCO DE REGISTROS

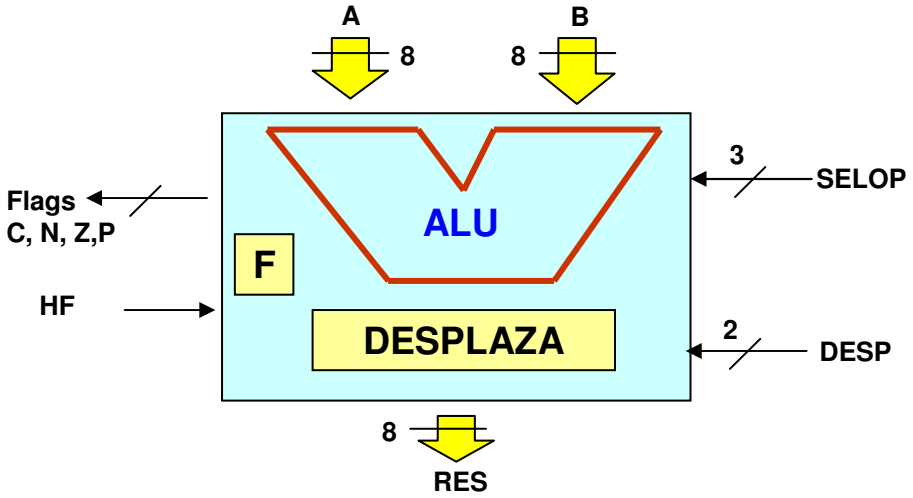


PC
SP
DPTR
A
VI
Temp
Cte1
ACC

Program Counter
Stack Pointer
Data Pointer
Registro A
Vector Interrupción
Temporal (uso interno)
Constante -1
Acumulador

```
entity   BANCO   is
Port (          CLK,Rst_n       :IN STD_LOGIC;
                HR:             :IN STD_LOGIC;                                  -- Habiltador registro de entrada (Activo alto)
                SC, SB          :IN STD_LOGIC_VECTOR (2 downto 0);  -- Selectores de registro (destino, origen)
                BUSC            :IN STD_LOGIC_VECTOR (7 downto 0);  -- Entrada de datos
                BUSA, BUSB      :OUT STD_LOGIC_VECTOR (7 downto 0); -- BUS ACC, BUS B
);
End BANCO;
```

UNIDAD OPERATIVA



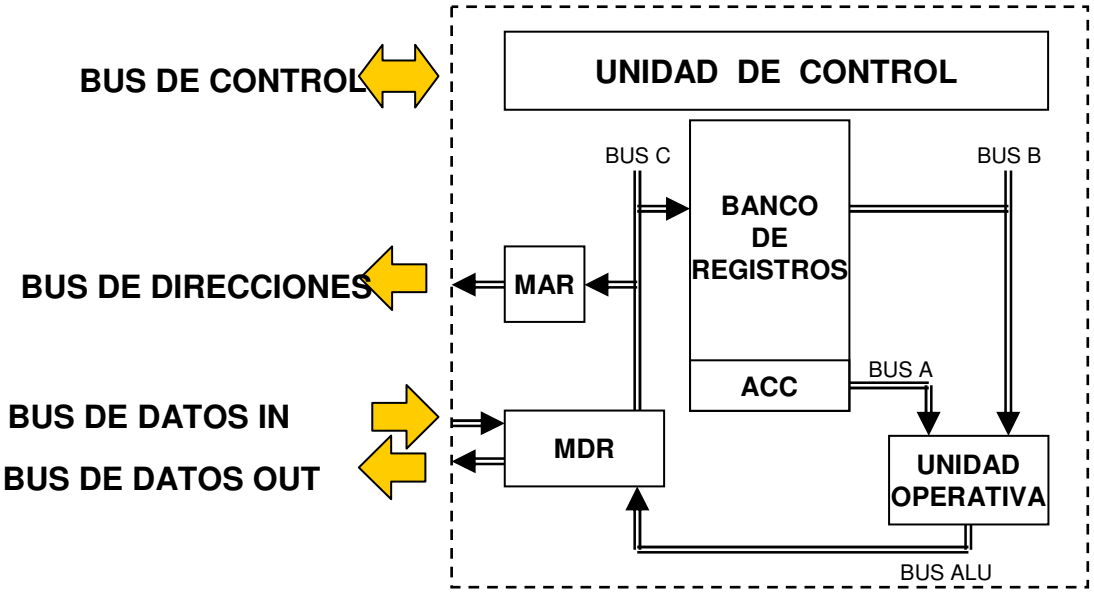
SELOP			
B	0	0	0
Not B	0	0	1
A and B	0	1	0
A or B	0	1	1
A xor B	1	0	0
A add B	1	0	1
B + 1	1	1	0
Not B + 1	1	1	1

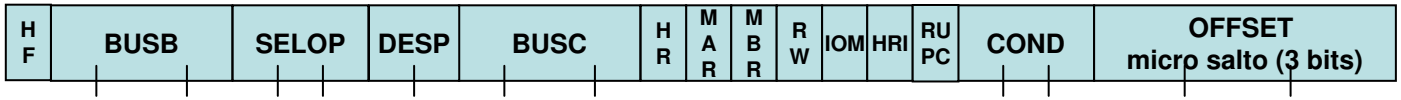
DESP		
No desplaza	0	0
Desplaza 1 bit a la derecha	0	1
Desplaza 1 bit a la izquierda	1	0
NU	1	1

```
Entity   ALU   is
Port (    HF          :IN      STD_LOGIC;                      -- Hab. banderas
          A,B         :IN      STD_LOGIC(7 downto 0);          -- Operandos
          SELOP       :IN      STD_LOGIC(2 downto 0);          -- Selección operación
          DESP        :IN      STD_LOGIC_VECTOR (1 downto 0);-- Desplazamiento
          S           :OUT     STD_LOGIC_VECTOR(7 downto 0); -- Resultado
          C,N,Z,P     :OUT     STD_LOGIC                       -- Banderas
         );
End ALU;
```

MICROINSTRUCCIONES

FORMATO DE LA MICROINSTRUCCION



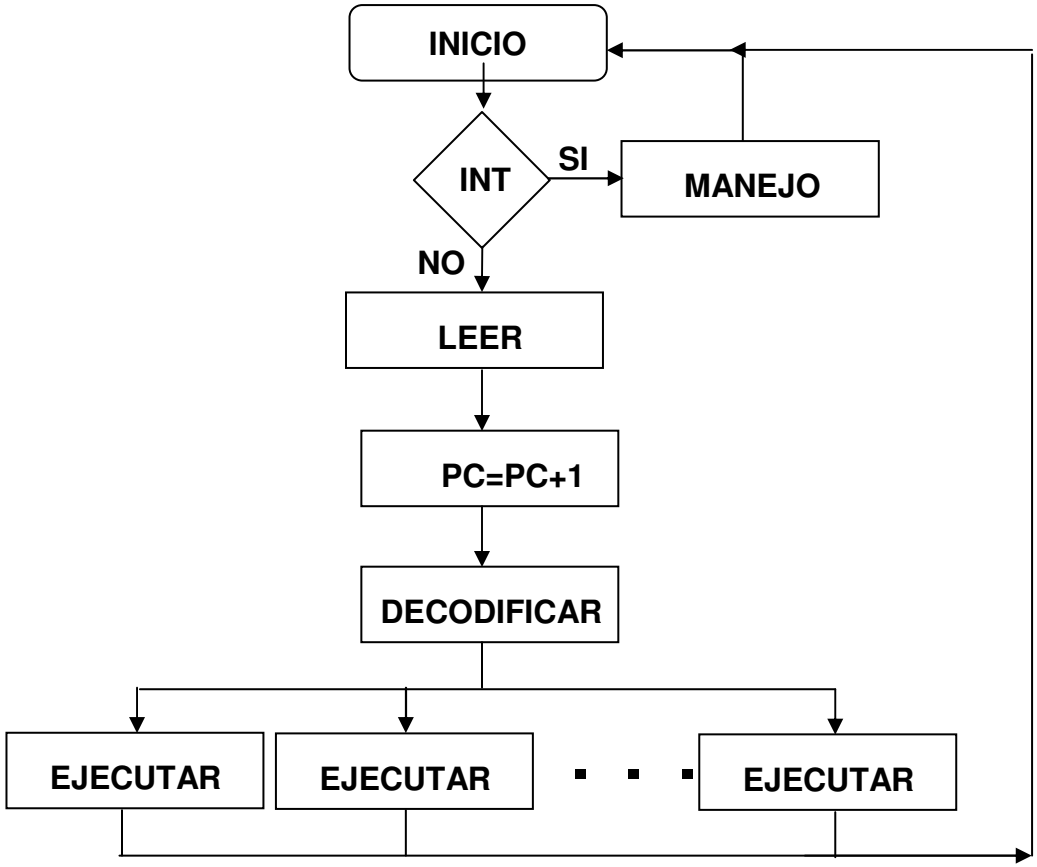


HF:	Habilitador de almacenamiento de banderas
BUSB:	Selector registro Origen (operando B de la ALU)
SELOP:	Selección de operación en la ALU
DESP:	Desplazamiento
BUSC:	Selector registro DESTINO
HR:	Habilitador de registro destino
MAR:	Habilitador de registro de dirección
MBR:	Selector de Datos (ALU o MEMORIA) (depende de RW)
RW:	0 = Lectura, 1= Escritura (de memoria)
IOM:	0 = Acceso a IO
HRI:	Habilitador de carga del registro de Instrucción
RUPC:	Reinicio microPC (Nuevo ciclo de fetch)
COND:	Condición de salto
Offset:	Microsalto de decodificación a nivel microprograma

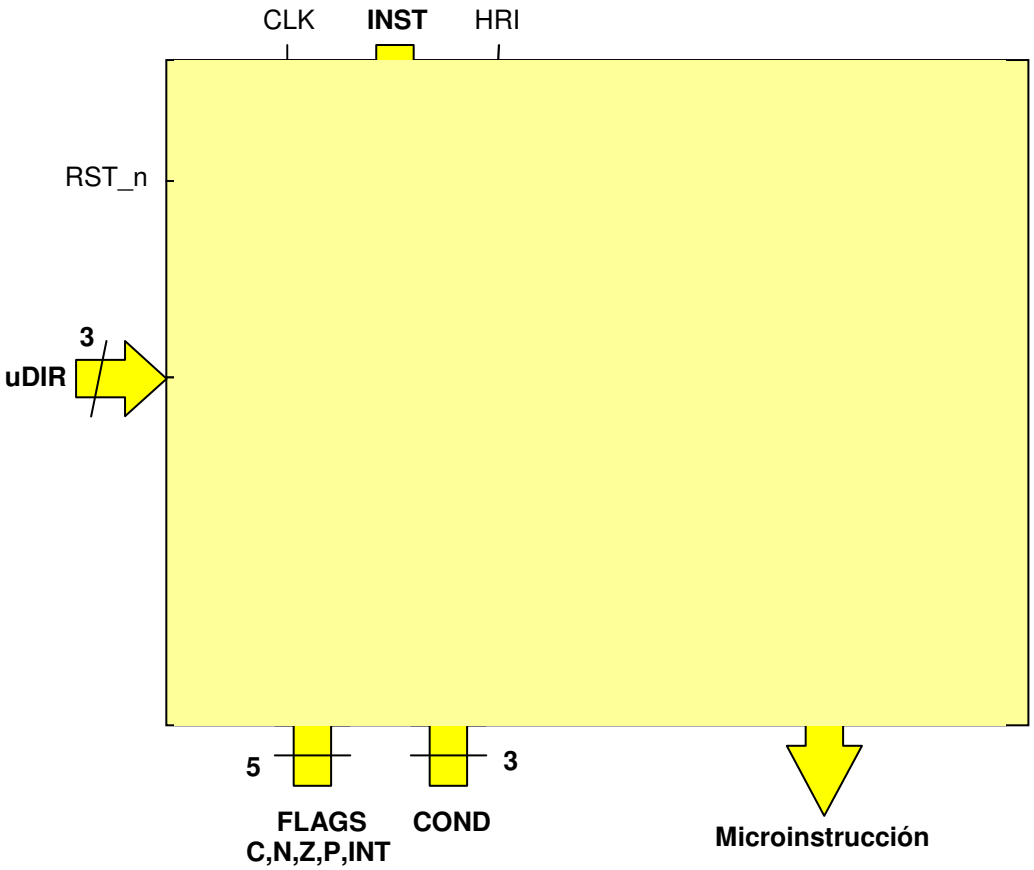
CONDICION	A	B	C
No salta	0	0	0
Salto incondicional	0	0	1
Salto si cero	0	1	0
Salto si negativo	0	1	1
Salto si carry	1	0	0
Salto si paridad	1	0	1
Salto si interrupción	1	1	0
Disponible	1	1	1

FETCH

- **INT** : Revisar si hay interrupciones
- **MANEJO** : Manejo de interrupciones
- **LEER** : Leer instrucción de memoria
- **DECODIFICAR** : Que instrucción es?
- **EJECUTAR** : Realizar el microprograma que ejecuta la instrucción)



UNIDAD DE CONTROL

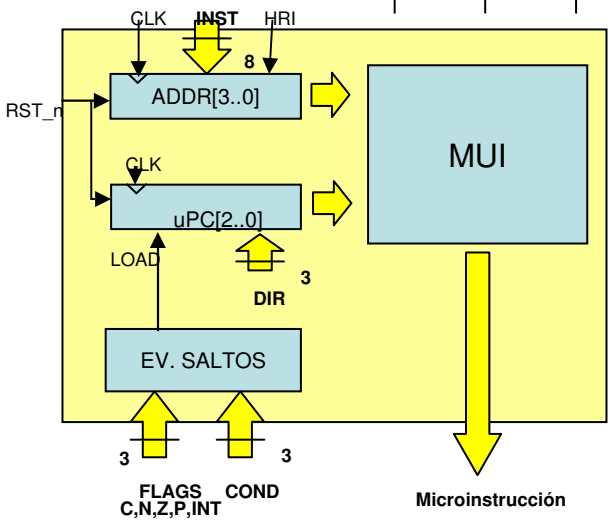


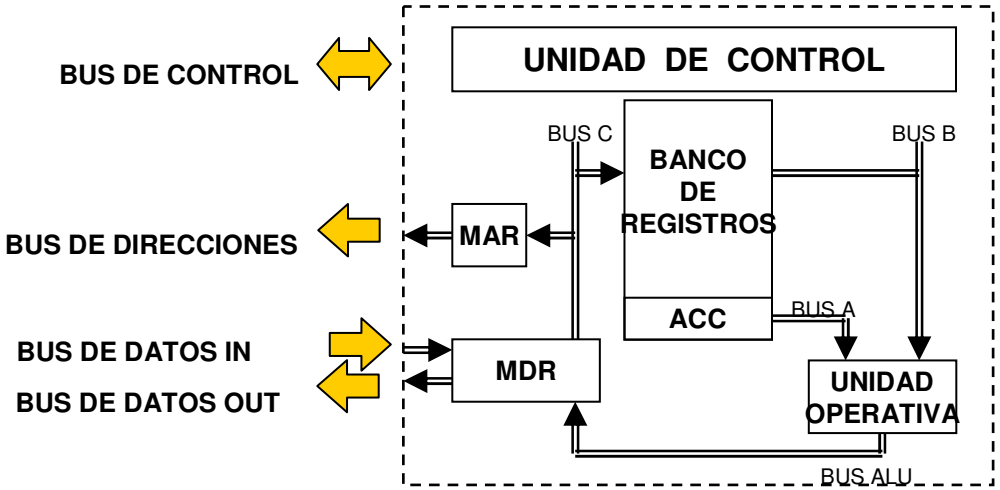
FETCH

	PC
	SP
	DPTR
	A
	B
	Temp
	Cte 1
	ACC

SELOP			
B	0	0	0
Not B	0	0	1
A and B	0	1	0
A or B	0	1	1
A xor B	1	0	0
A add B	1	0	1
B + 1	1	1	0
Not B + 1	1	1	1

		HF	BUSB (3)	SELOP (3)	DESP (2)	BUSC (3)	HR	MAR	MBR	RW	IOM	HRI	RUPC	COND (3)	OFFS (3)	
when	x"00"	=> UI <= "0	XXX	XXX	XX	XXX	0	0	0	0	0	0	1	110	100";	-- JINT 100
when	x"01"	=> UI <= "0	000	000	00	XXX	0	1	0	0	1	0	1	000	XXX";	-- MAR=PC, RD MREQ
when	x"02"	=> UI <= "0	000	110	00	000	1	0	0	0	1	0	1	000	XXX";	-- PC=PC+1, RD MREQ
when	x"03"	=> UI <= "0	000	000	00	000	0	0	1	0	1	1	0	000	XXX";	-- MDR=DEX,
														--	INST=MDR,RDMREQ,rst upc	
-- INT																
when	x"04"	=> UI <= "0	000	000	00	000	0	0	0	0	0	0	0	000	000";	
when	x"05"	=> UI <= "0	000	000	00	000	0	0	0	0	0	0	0	000	000";	
when	x"06"	=> UI <= "0	000	000	00	000	0	0	0	0	0	0	0	000	000";	
when	x"07"	=> UI <= "0	000	000	00	000	0	0	0	0	0	0	0	000	000";	



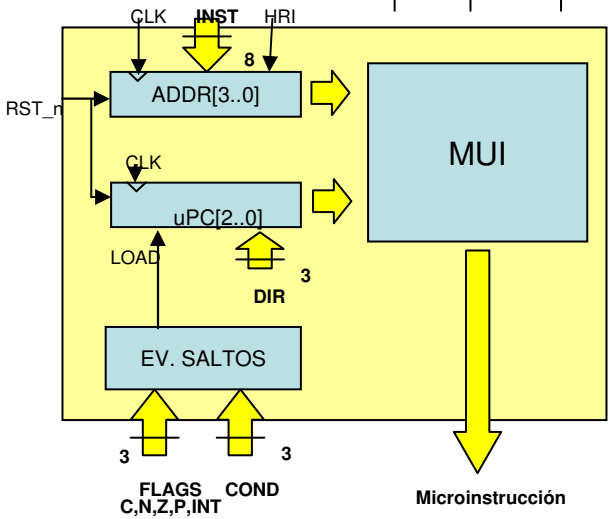


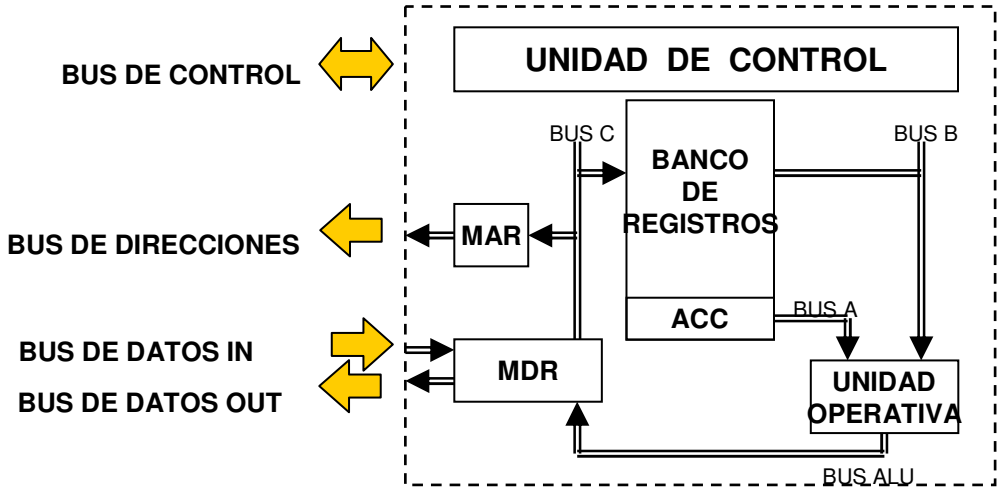
MOV ACC,A

	HF	BUSB (3)	SELOP (3)	DESP (2)	BUSC (3)	HR	MAR	MBR	RW	IOM	HRI	RUPC	COND (3)	OFFS (3)	
when x"08"=> UI <=	"1	011	000	00	111	1	0	0	0	0	0	0	000	XXX";	--ACC = A,Reset UPC
when x"09"=> UI <=	"0	XXX	XXX	XX	XXX	X	X	X	X	X	X	X	XXX	XXX";	

PC
SP
DPTR
A
B
Temp
Cte 1
ACC

SELOP			
B	0	0	0
Not B	0	0	1
A and B	0	1	0
A or B	0	1	1
A xor B	1	0	0
A add B	1	0	1
B + 1	1	1	0
Not B + 1	1	1	1



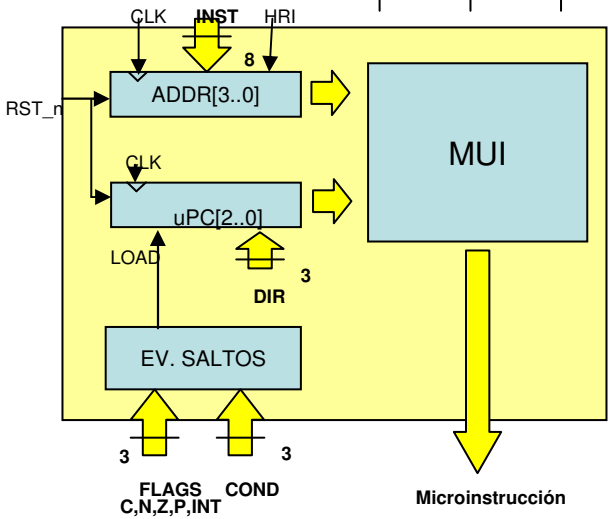


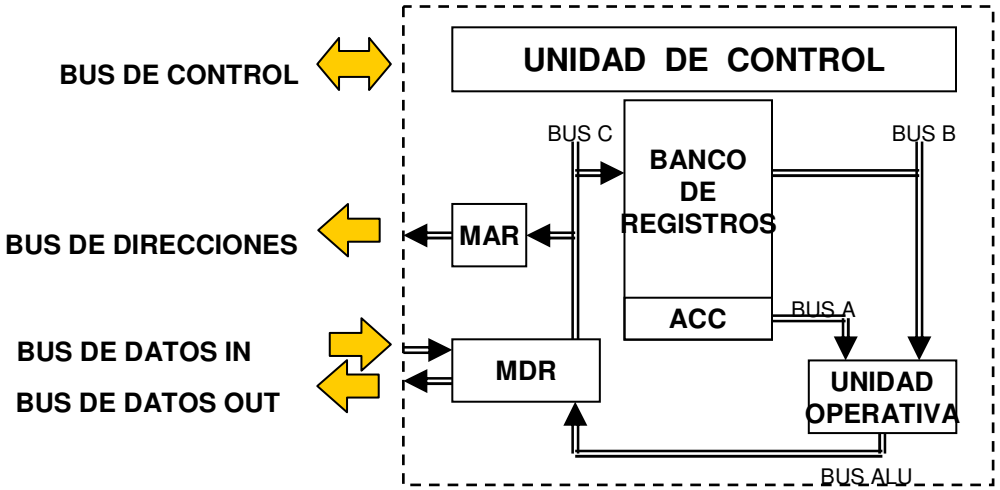
MOV ACC,CTE

HF	BUSB (3)	SELOP (3)	DESP (2)	BUSC (3)	HR	MAR	MBR	RW	IOM	HRI	RUPC	COND (3)	OFFS (3)	
when x"18"=> UI <= "0	000	000	00	XXX	0	1	0	0	1	0	1	000	XXX";	--MAR=PC,RD MREQ
when X"19"=> UI <= "0	000	110	00	000	1	0	0	0	1	0	1	000	XXX";	--PC=PC+1,RD MREQ
when X"1A"=> UI <= "0	000	000	00	111	1	0	1	0	1	0	0	000	XXX";	--ACC=DATA,RD --MREQ, Reset UPC
when x"1B"=> UI <= "X	XXX	XXX	XX	XXX	X	X	X	X	X	X	X	XXX	XXX";	

PC
SP
DPTR
A
B
Temp
Cte 1
ACC

SELOP			
B	0	0	0
Not B	0	0	1
A and B	0	1	0
A or B	0	1	1
A xor B	1	0	0
A add B	1	0	1
B + 1	1	1	0
Not B + 1	1	1	1



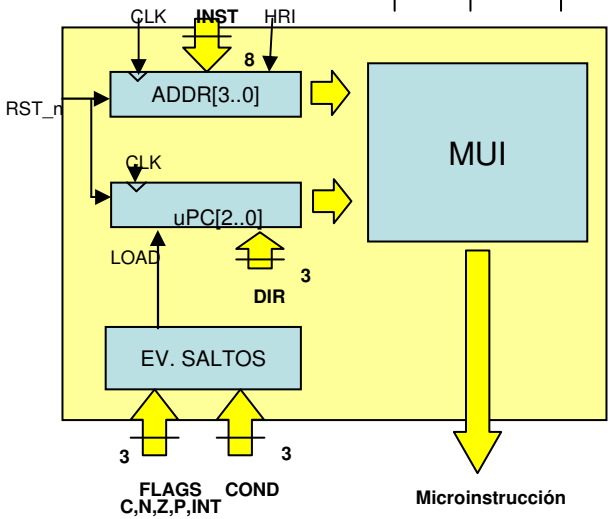


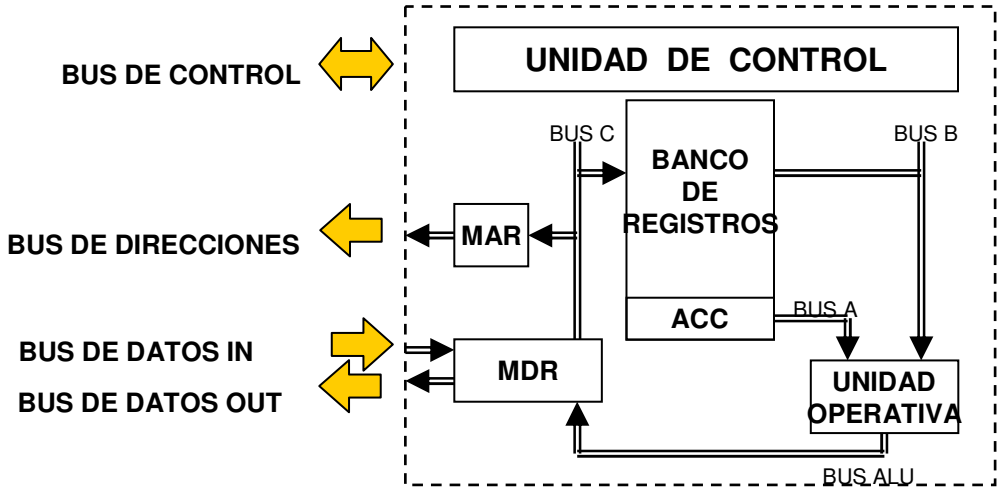
MOV ACC,[DPTR]

PC
SP
DPTR
A
B
Temp
Cte 1
ACC

SELOP			
B	0	0	0
Not B	0	0	1
A and B	0	1	0
A or B	0	1	1
A xor B	1	0	0
A add B	1	0	1
B + 1	1	1	0
Not B + 1	1	1	1

	HF	BUSB (3)	SELOP (3)	DESP (2)	BUSC (3)	HR	MAR	MBR	RW	IOM	HRI	RUPC	COND (3)	OFFS (3)	
when x"20"=> UI <= "0	010	000	00	XXX	0	1	0	0	1	0	1	000	XXX";	--MAR=DPTR, RD MREQ
when x"21"=> UI <= "0	XXX	XXX	XX	111	1	0	1	0	1	0	0	000	XXX";	--MDR=DEX,RD MREQ, ACC=MDR,RST UPC
when x"22"=> UI <= "X	XXX	XXX	XX	XXX	X	X	X	X	X	X	X	XXX	XXX";	



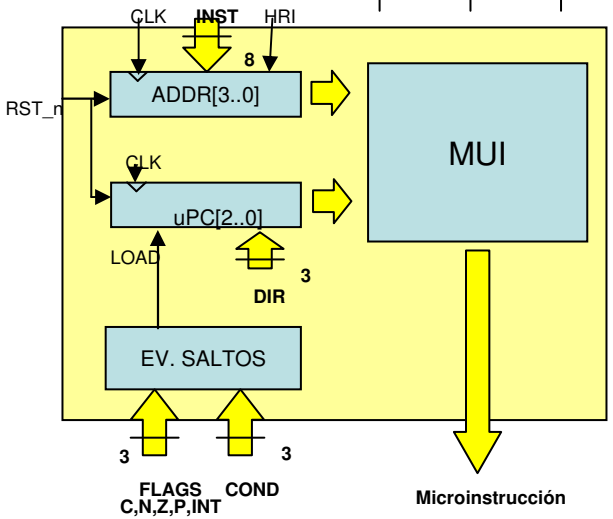


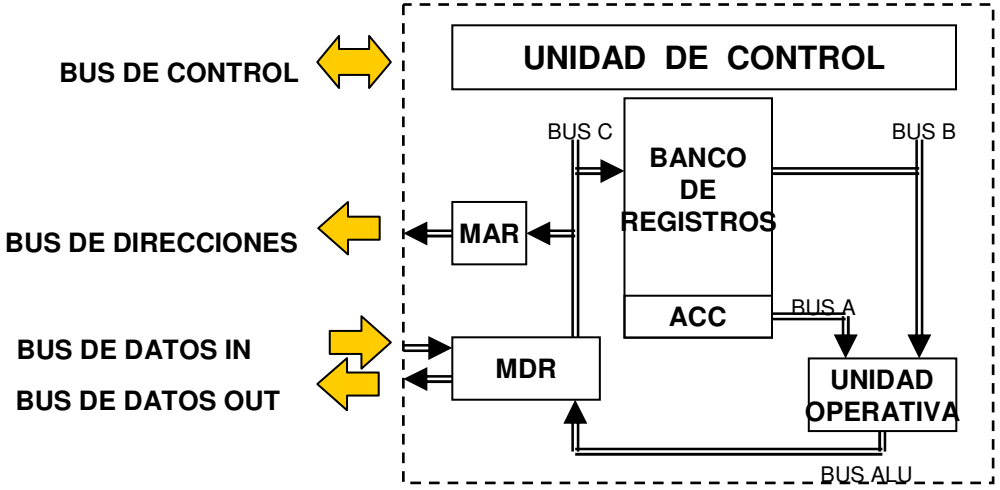
MOV DPTR,ACC

PC
SP
DPTR
A
B
Temp
Cte 1
ACC

SELOP			
B	0	0	0
Not B	0	0	1
A and B	0	1	0
A ur B	0	1	1
A xor B	1	0	0
A add B	1	0	1
B + 1	1	1	0
Not B + 1	1	1	1

	HF	BUSB (3)	SELOP (3)	DESP (2)	BUSC (3)	HR	MAR	MBR	RW	IOM	HRI	RUPC	COND (3)	OFFS (3)	
when x"28"=> UI <= "	1	111	000	00	010	1	0	0	0	0	0	0	000	XXX";	-- DPTR = ACC , Reset UPC
when x"29"=> UI <= "	X	XXX	XXX	XX	XXX	X	X	X	X	X	X	X	XXX	XXX";	



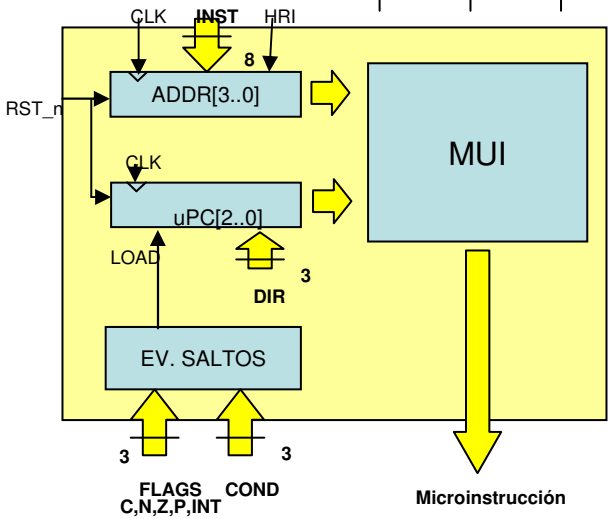


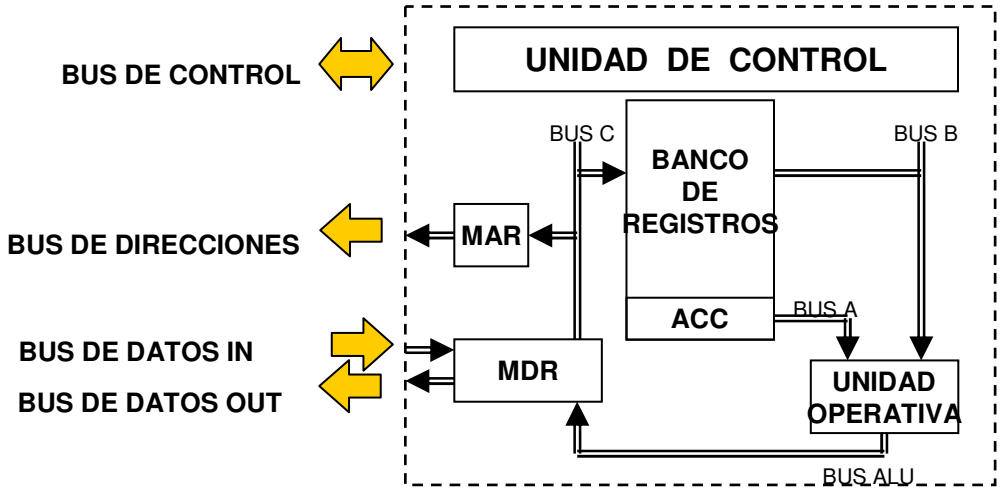
ADD ACC,A

PC
SP
DPTR
A
B
Temp
Cte 1
ACC

SELOP			
B	0	0	0
Not B	0	0	1
A and B	0	1	0
A or B	0	1	1
A xor B	1	0	0
A add B	1	0	1
B + 1	1	1	0
Not B + 1	1	1	1

```
                          HF  BUSB (3)  SELOP (3)  DESP (2)  BUSC (3)  HR  MAR  MBR  RW  IOM  HRI  RUPC  COND (3)  OFFS (3)
when x"48"=> UI <= "1  011  101  00  111  1  0  0  0  0  0  0  000  XXX"; -- ACC = ACC + A,
                                                                              Reset UPC
when x"49"=> UI <= "X  XXX  XXX  XX  XXX  X  X  X  X  X  X  X  XXX  XXX";
```





EJERCICIO

PC
SP
DPTR
A
B
Temp
Cte 1
ACC

SELOP			
B	0	0	0
Not B	0	0	1
A and B	0	1	0
A or B	0	1	1
A xor B	1	0	0
A add B	1	0	1
B + 1	1	1	0
Not B + 1	1	1	1

HF | BUSB (3) | SELOP (3) | DESP (2) | BUSC (3) | HR | MAR | MBR | RW | IOM | HRI | RUPC | COND (3) | OFFS (3)