Izen-abizenak .....

NAN zenbakia .....

Oharra:

Azterketa honekin lor daitekeen notarik altuena 7 da. Beste 3 puntuak laborategi-praktiken ebaluazioari dagozkie.

Galdera teorikoak

(a) Azaldu xehetasunez ondoz-ondoko hurbilketan oinarrutako A/D bihuragiluaren funtzionamendua, dagokion zirkuitu elektronikoa marraztuz (0.5 puntu).

Ondoz-ondoko hurbilketan oinarritutako A/D bihurgailuaren funtzionamenduaren azalpena klaseko gardenkietan (periferikoak gaia) edota 80c552 mikrokontrolagailuaren manualean (50. eta 51. orrialdeak) aurki daiteke.

(b) Ondoz-ondoko hurbilketan oinarritutako A/D bihurgailu bat duen eta 5 V-etara elikatuko den mikrokontroladore bat diseinatu nahi dugu. 0°C eta 50°C bitartean temperatura neurtzeko gaitasuna izan behar du bihurgailuak. Zein izan beharko litzateke bihurgailuaren bit-kopuru minimoa, temperatura neurketan gutxienez 1°C-ko bereizmena izateko? (0.5 puntu).

A/D bihurgailuaren irismena hurrengoa da:

$$irismena = V_{in,max} - V_{in,min}, \tag{1}$$

non, kasu partikular honetan, $V_{in,max} = 5V$ y $V_{in,min} = 0V$.

Bestalde,honelaxe definitzen da bihurgailuaren bereizmena:

$$bereizmena = \frac{irismena}{2^N}, \tag{2}$$

no N A/D bihurgailuaren bit-kopurua den. Tenperaturan 1°C-tako erresoluzioak tentsioan 0.1 V-etako erresoluzioa dakarrenez, beharrezkoa den bit kopurua honelaxe kalkulatzen da:

$$N = log_2(\frac{5}{0.1}) = 5.64. \tag{3}$$

Beraz eta A/D bihurgailuaren bit-kopurua zenbaki oso bat izan behar denez, N=6 izan behar da, gutxienez, eskatutako espezifikazioak betetzeko.

(c) Zeintzuk dira RET eta RETI instrukzioen arteko ezberdintasun nagusiak? (0.5 puntu).

Bi instrukzioek pilan zegoen memoria berreskuratzen dute programaren exekuzioarekin jarraitzeko. Hala ere, ezberdintasun nabarmen bat dago. Eten bat gertatzean, etenak gahitzeko bita desgaitu egiten da (EA = 0).

Mikrokontrolagailua berriz eten ahal izateko, beharrezkoa da bit hori habilitatzea (EA = 1) behin etenari erantzuten dion azpierrutina exekutatu ondoren. RETI instrukzioak funtzio gehigarri hori betetzen du; aldiz, RET instrukzikoak bakarrik pilatik berreskuratzen du itzulerako helbidea.

(d) Hurrengo irudian azaltzen den *timer*-zirkuitua dugu.

- Sarrerako erlojuaren mahiztasuna 60 MHz-etakoa da.
- Aurreeskalatzaileak mahiztasuna biko multiploetan zati dezake, hori da, zati $1, 2, 4, ...$ edo 256.
- Posteskalatzailea finkoa da, eta bere sarrerako maiztasuna bigatik zatitzen du beti.

Zehaztu etenen artean tenporizadore horrek sortu ditzakeen denbora maximoa eta minimoa. Definitu denbora maximoa eta minimoa lortzeko beharrezkoak lirateekeen erregistroen balioak (0.5 puntu).

Tenporizadoreak maiztasun maximoarekin (denbora minimoarekin) etengo du bloke bakoitzak ahalik eta gutxien zatitzen duenean sarrerako seinaleko maiztasuna. Kasu honetan, aurreeskalatzaileak maiztasuna zati bat eginez lortzen da, eta baita ere 8 biteko kontagailua FFh baliora hasieratuz. Era horretara eta kontuan hartuta posteskalatzailearen mahiztasun-zatiketa konstantea dela (sarrerako maiztasuna zati bi), eten bat sortuko da 30 MHz-etara, hori da, $3.3.10^{-8}$ s-ro.

Tenporizadoreak maiztasun minimoarekin (denbora maximoarekin) etengo du bloke bakoitzak ahalik eta gehien zatitzen duenean sarrerako seinaleko maiztasuna. Kasu honetan, aurreeskalatzaileak maiztasuna zati 256 eginez lortzen da, eta baita ere 8 biteko kontagailua 00h baliora hasieratuz. Era horretara eta kontuan hartuta posteskalatzailearen mahiztasun-zatiketa konstantea dela (sarrerako maiztasuna zati bi), eten bat sortuko da 457.7 Hz-etara, hori da, 2.18 ms.

1. ariketa

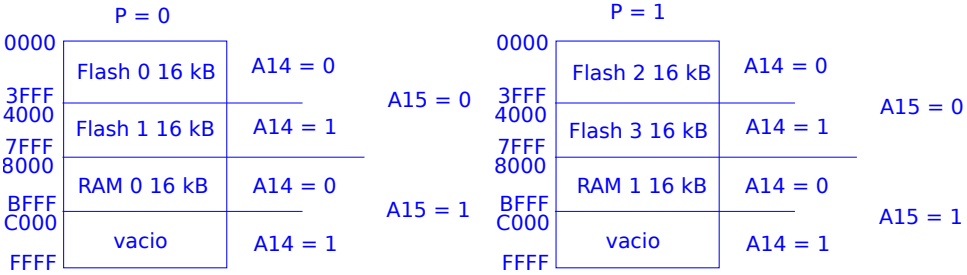
Hurrengo ezaugarriak dituen Von Neuman motako arkitektura duen memoria diseinatu nahi da:

- 8 biteko datu-busa eta 16 biteko helbide-busa.
- Reset-helbidea jatorrian (0x0000 helbidean).
- 64 kB-etako programa-memoria eta 32 kB datu-memoria izan behar ditu sistemak. Beraz, beharrezkoa da bi orri erabiltzea.

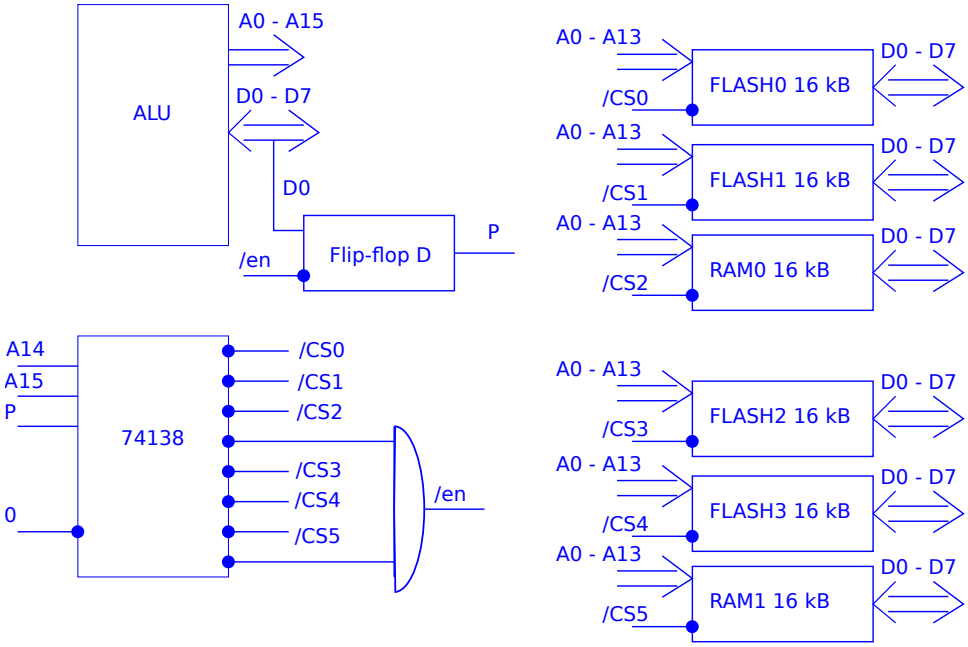
Helburu horiek lortzeko, hurrengo hardware elementuak ditugu eskuragarri: 16 kB-etako FLASH memoriak, 16 kB-etako RAM memoriak, 3tik 8rako multiplexoreak, D-motako biegonkorrak eta ate logikoak.

Hurrengoa eskatzen da:

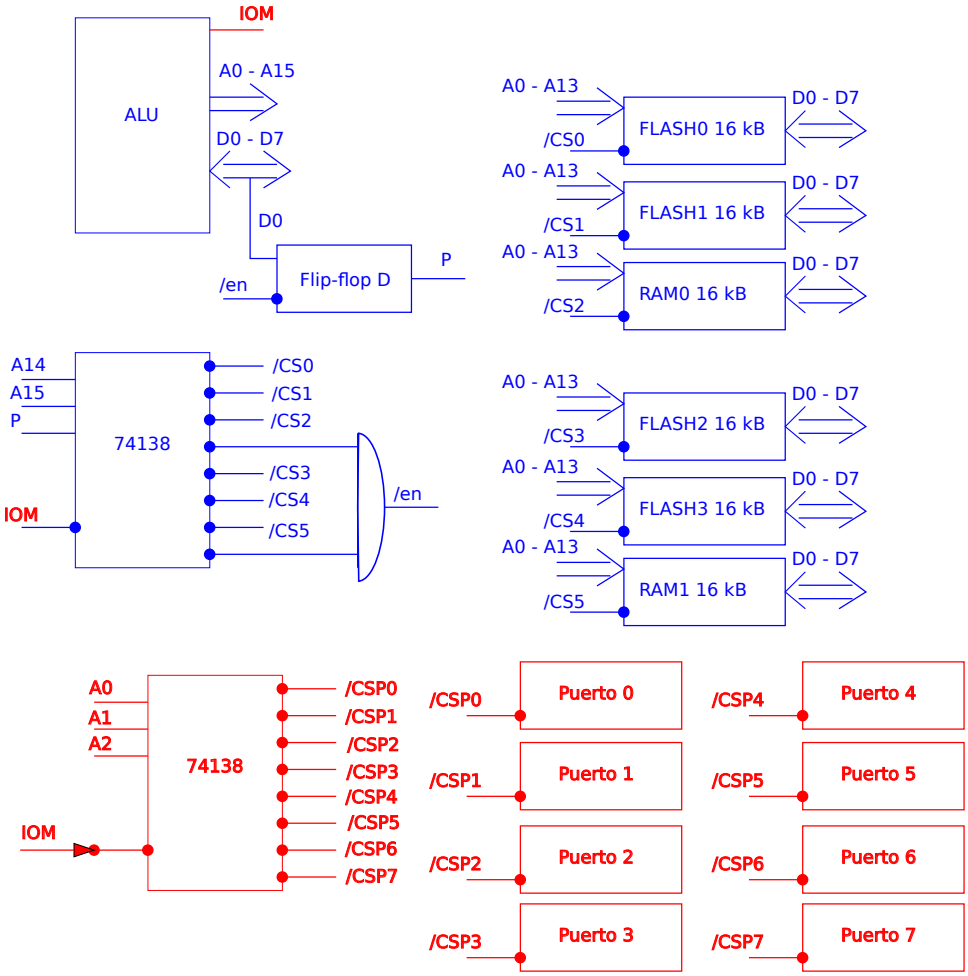
(a) Konputagailuaren memoria-mapa marraztu, erabilitako guneak eta hutsik dauden guneak zehaztuz, eta baita ere guneen hasierako eta amaierako helbideak eta horien aktibaziorako erabilitako seinaleak (puntu 1).



(b) Memoriaren hardware zirkuitua irudikatu, beharrezkoak diren hardware elementuak, mikroprozesadorea eta datu- eta helbide-busen lerroak kontuan hartuta. Erabili beharreko hardware elementu kopurua minimizatu egin behar da (puntu 1).

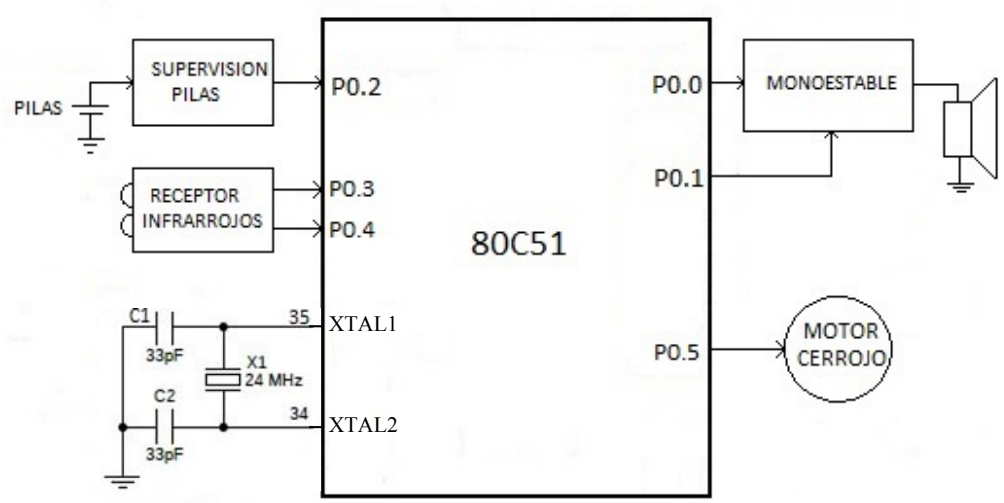


(c) Kontuan hartuta diseinatzen ari garen mikrokontroladorearen arkitektura Intel motakoa dela, gehitu 2(b) atalean sortu den zirkuitu elektronikoari beharrezkoak diren elementuak eta konexioak 8 sarrera/irteerako portu digital gehitu ahal izateko (0.5 puntu).



2. ariketa

Urrutiko kontrol infragorri baten bidez kontrolatutako sarraila elektronikoa dugu. 80c552 mikrokontroladore batekin kontrolatzen dira sarrailaren funtzioak. Errezeptore infragorri bat, pilen egoera monitorizatzen duen zirkuitu bat, bozgorailu baten bidez soinua sortzeko zirkuitu monoegonkor bat eta atea irekitzeko edota ixteko motorra ditu sarrailak. Irudian sistemak dituen gailu guztiak eta horien portu digitalen bidezko konexioak erakusten dira.



Errezeptore infragorritik jasotzen ditu aginduak sarrailak. Agindu bat exekutatu ondoren hots bat sortzen du sarrailak bozgorailuaren bidez. Pilen egoera monitorizatzen duen zirkuituak "pila baxua" egoera adierazten badu, sarraila ireki egin beharko da aurretik itxita balego, eta ez ditu ixteko aginduei jaramonik egingo. Kasu horretan, bi hots sortuko dira bozgorailuaren bidez ixteko aginduari jaramonik egin ez zaiola adierazteko. Hotsen iraupena zirkuitu monoegonkorrak kontrolatzen du.

P0.0 eta P0.1 portuen balioen arabera sortuko ditu zirkuitu monoegonkorrak hots 1 edo 2 (ikus taula). Hurrengo tauletan adierazten dira portuen sarrerako eta irteerako maila logikoak eskatzen diren funtzioak bete ahal izateko.

	"0"	"1"	Trantsizioa ("0" → "1" edo "1" → "0")
P0.0	-	-	1 edo 2 hots, P0.1 ren arabera
P0.1	1 hots	2 hots	-
P0.2	Pila baxua	Pila OK	-
P0.5	Motorrari irekitzeko agindua	Motorrari ixteko agindua	-

P0.3 y P0.4	“00” ó “11” Ez dago agindu berririk	“10” Irekitzeko agindua	“01” Ixteko agindua

Hurrengoa eskatzen da:

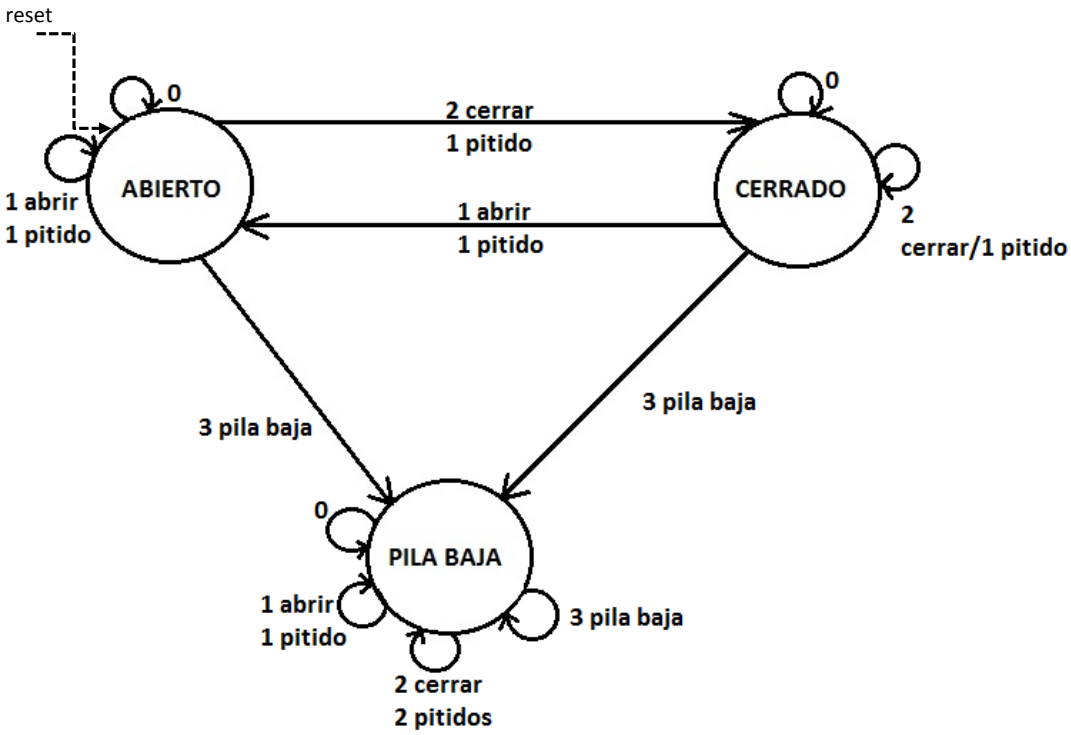
(a) Sarraila kontrolatzen duen egoera, gertaera eta ekintza makina irudikatu (0.5 puntu).

(b) Programaren fluxu-diagrama irudikatu (puntu 1).

(c) Sarraila kontrolatzen duen programa idatzi mihiztatzailean (puntu 1).

Jarraian azaltzen da ariketa honek izan dezakeen soluzio posible bat.

(a)

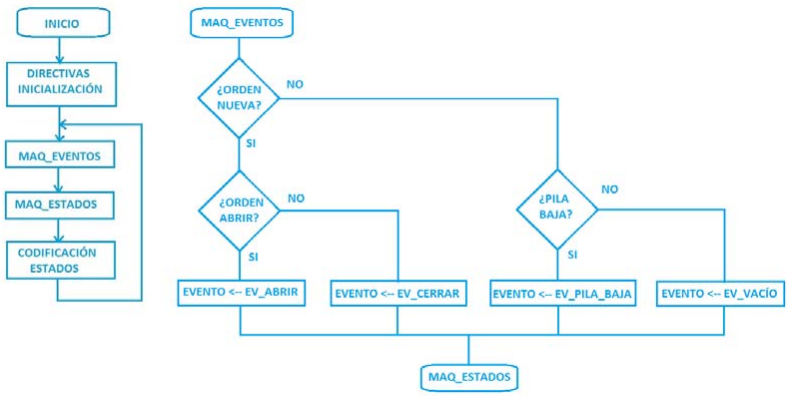


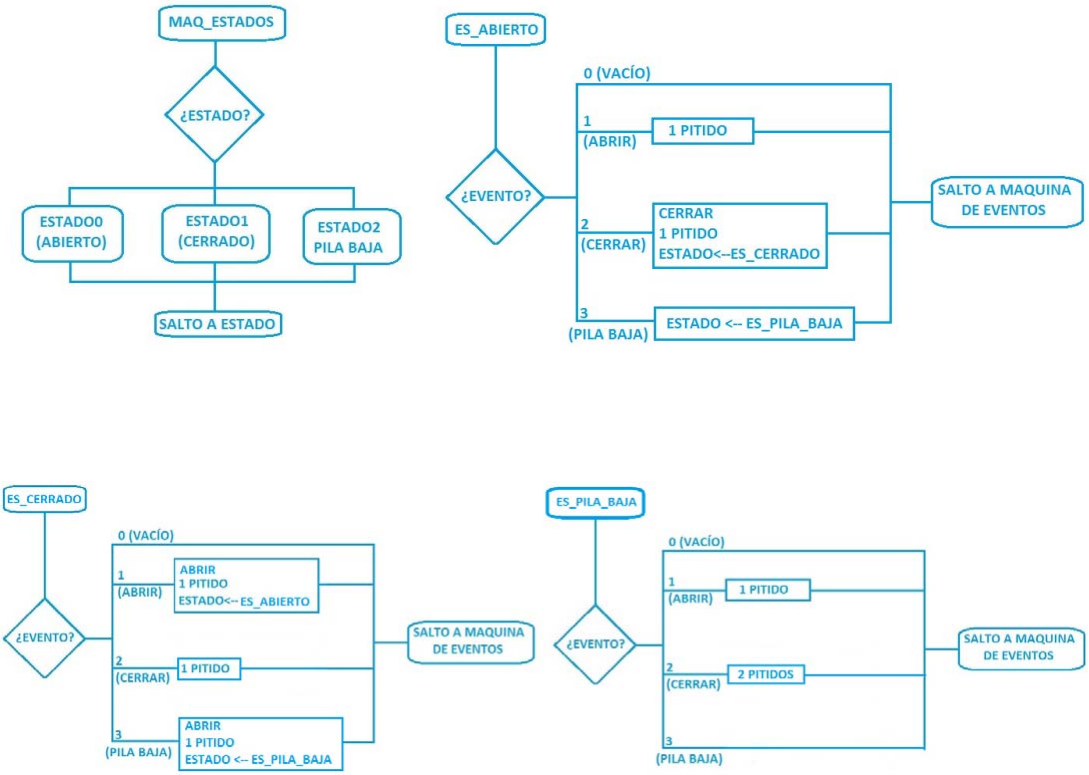
ESTADO	CODIFICACIÓN
0 ABIERTO	00
1 CERRADO	02
2 PILA BAJA	04

EVENTOS
0 VACÍO
1 ORDEN ABRIR
2 ORDEN CERRAR
3 PILA BAJA

ACCIONES
0 ABRIR
1CERRAR
2 UN PITIDO
3 DOS PITIDOS

(b)





(c)

CÓDIGO

```
ES_ABIERTO    EQU   0x00    ;directivas e inicializaciones
ES_CERRADO    EQU   0x02
ES_PILA_BAJA  EQU   0x04
EV_VACIO      EQU   0x00
EV_ABRIR      EQU   0x01
EV_CERRAR     EQU   0x02
EV_PILA_BAJA  EQU   0x03
ESTADO        DATA  0x00          ;empezamos en estado abierto
EVENTO        DATA  0x00          ;empezamos en evento vacio

              ORG   0x00          ;reset
              SJMP  INICIALIZACION
              ORG   0x20          ; comienzo del código

INICIALIZACION:

              CLR P0.5            ; empezamos con cerrojo abierto
              AJMP  MAQ_EVENTOS
```

MÁQUINA DE EVENTOS

```
MAQ_EVENTOS:
                JB P0.3, loop0          ;inicio comprobación órdenes
                JNB P0.4, no_ord
                SJMP cerrar             ;el evento es cerrar
loop0:          JB P0.4, no_ord         ;fin comprobación órdenes
                MOV EVENTO, EV_ABRIR    ;evento abrir
                SJMP salida1
cerrar:         MOV EVENTO, EV_CERRAR   ;evento cerrar
                SJMP salida1
no_ord:         JB P0.2, evvacio
                MOV EVENTO, EV_PILA_BAJA ;evento pila baja
                SJMP salida1
evvacio:        MOV EVENTO, EV_VACIO    ;no hay evento
salida1:        AJMP MAQ_ESTADOS
```

MÁQUINA DE ESTADOS

```
MAQ_ESTADOS:
                MOV A, ESTADO           ;no es necesario multiplicar por 2
                MOV DPTR, #LIST_EST
                JMP @A+DPTR
LIST_EST:
                AJMP ES_ABIERTO
                AJMP ES_CERRADO
                AJMP ES_PILA_BAJA
```

ESTADO ABIERTO

```
ES_ABIERTO:
                MOV A, EVENTO
                CJNE A, EV_ABRIR, loop1 ;saltar si no es abrir
                CLR P0.1                ;evento abrir 1 pitido
                CPL P0.0                ;flanco
                SJMP salida2
loop1:          CJNE A, EV_CERRAR, loop2 ;saltar si no es cerrar
                SETB P0.5               ;cerramos
                CLR P0.1                ;evento cerrar 1 pitido
                CPL P0.0                ;flanco
                MOV ESTADO, ES_CERRADO  ;pasamos a estado cerrado
                SJMP salida2
loop2:          CJNE A, EV_PILA_BAJA, salida2 ;saltar si no es pila baja
                MOV ESTADO, ES_PILA_BAJA ;pasamos a estado pila baja
salida2:        AJMP MAQ_EVENTOS
```

ESTADO CERRADO

```
ES_CERRADO:
                    MOV A, EVENTO
                    CJNE A, EV_ABRIR, loop3     ;saltar si no es abrir
                    CLR P0.5                    ;abrimos
                    CLR P0.1                    ;1 pitido
                    CPL P0.0                    ;flanco
                    MOV ESTADO, ES_ABIERTO      ;pasamos a estado abierto
                    SJMP salida3
loop3:              CJNE A, EV_CERRAR, loop4    ;saltar si no es cerrar
                    CLR P0.1                    ;1 pitido
                    CPL P0.0                    ;flanco
                    SJMP salida3
loop4:              CJNE A, EV_PILA_BAJA, salida3 ;saltar si no es pila baja
                    CLR P0.5                    ;abrimos
                    CLR P0.1                    ;1 pitido también valen 2
                    CPL P0.0                    ;flanco
                    MOV ESTADO, ES_PILA_BAJA    ;pasamos a estado pila baja
salida3:            AJMP MAQ_EVENTOS
```

ESTADO PILA BAJA

```
ES_PILA_BAJA:
                    MOV A, EVENTO
                    CJNE A, EV_ABRIR, loop5     ;saltamos si no es abrir
                    CLR P0.1                    ;1 pitido
                    CPL P0.0                    ;flanco
                    SJMP salida4
loop5:              CJNE A, EV_CERRAR, salida4  ;saltamos si no es cerrar
                    SETB P0.1                   ;2 pitidos
                    CPL P0.0                    ;flanco
salida4:            AJMP MAQ_EVENTOS
```

80C51 FAMILY INSTRUCTION SET

Table 7. 80C51 Instruction Set Summary

Interrupt Response Time: Refer to Hardware Description Chapter.

Instructions that Affect Flag Settings(1)

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	0	X		ANL C,/bit	X		
DIV	0	X		ORL C,bit	X		
DA	X			ORL C,/bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

(1)Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.

Notes on instruction set and addressing modes:

Rn	Register R7-R0 of the currently selected Register Bank.
direct	8-bit internal data location's address. This could be an Internal Data RAM location (0-127) or a SFR [i.e., I/O port, control register, status register, etc. (128-255)].
@Ri	8-bit internal data RAM location (0-255) addressed indirectly through register R1 or R0.
#data	8-bit constant included in the instruction.
#data 16	16-bit constant included in the instruction
addr 16	16-bit destination address. Used by LCALL and LJMP. A branch can be anywhere within the 64k-byte Program Memory address space.
addr 11	11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2k-byte page of program memory as the first byte of the following instruction.
rel	Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is –128 to +127 bytes relative to first byte of the following instruction.
bit	Direct Addressed bit in Internal Data RAM or Special Function Register.

MNEMONIC		DESCRIPTION	BYTE	OSCILLATOR PERIOD
ARITHMETIC OPERATIONS				
ADD	A,Rn	Add register to Accumulator	1	12
ADD	A,direct	Add direct byte to Accumulator	2	12
ADD	A,@Ri	Add indirect RAM to Accumulator	1	12
ADD	A,#data	Add immediate data to Accumulator	2	12
ADDC	A,Rn	Add register to Accumulator with carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with carry	2	12
ADDC	A,@Ri	Add indirect RAM to Accumulator with carry	1	12
ADDC	A,#data	Add immediate data to A_{CC} with carry	2	12
SUBB	A,Rn	Subtract Register from A_{CC} with borrow	1	12
SUBB	A,direct	Subtract direct byte from A_{CC} with borrow	2	12
SUBB	A,@Ri	Subtract indirect RAM from A_{CC} with borrow	1	12
SUBB	A,#data	Subtract immediate data from A_{CC} with borrow	2	12
INC	A	Increment Accumulator	1	12
INC	Rn	Increment register	1	12

Table 7. 80C51 Instruction Set Summary (Continued)

MNEMONIC		DESCRIPTION	BYTE	OSCILLATOR PERIOD
ARITHMETIC OPERATIONS (Continued)				
INC	direct	Increment direct byte	2	12
INC	@Ri	Increment indirect RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	Rn	Decrement Register	1	12
DEC	direct	Decrement direct byte	2	12
DEC	@Ri	Decrement indirect RAM	1	12
INC	DPTR	Increment Data Pointer	1	24
MUL	AB	Multiply A and B	1	48
DIV	AB	Divide A by B	1	48
DA	A	Decimal Adjust Accumulator	1	12
LOGICAL OPERATIONS				
ANL	A,Rn	AND Register to Accumulator	1	12
ANL	A,direct	AND direct byte to Accumulator	2	12
ANL	A,@Ri	AND indirect RAM to Accumulator	1	12
ANL	A,#data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
ANL	direct,#data	AND immediate data to direct byte	3	24
ORL	A,Rn	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@Ri	OR indirect RAM to Accumulator	1	12
ORL	A,#data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct,#data	OR immediate data to direct byte	3	24
XRL	A,Rn	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL	A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12
RL	A	Rotate Accumulator left	1	12
RLC	A	Rotate Accumulator left through the carry	1	12
RR	A	Rotate Accumulator right	1	12
RRC	A	Rotate Accumulator right through the carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER				
MOV	A,Rn	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@Ri	Move indirect RAM to Accumulator	1	12

Table 7. 80C51 Instruction Set Summary (Continued)

MNEMONIC		DESCRIPTION	BYTE	OSCILLATOR PERIOD
DATA TRANSFER (Continued)				
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	Rn,A	Move Accumulator to register	1	12
MOV	Rn,direct	Move direct byte to register	2	24
MOV	RN,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,Rn	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@Ri	Move indirect RAM to direct byte	2	24
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@Ri,A	Move Accumulator to indirect RAM	1	12
MOV	@Ri,direct	Move direct byte to indirect RAM	2	24
MOV	@Ri,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to A_{CC}	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to A_{CC}	1	24
MOVX	A,@Ri	Move external RAM (8-bit addr) to A_{CC}	1	24
MOVX	A,@DPTR	Move external RAM (16-bit addr) to A_{CC}	1	24
MOVX	A,@Ri,A	Move A_{CC} to external RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move A_{CC} to external RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,Rn	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@Ri	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@Ri	Exchange low-order digit indirect RAM with A_{CC}	1	12
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to carry	2	24
ANL	C,/bit	AND complement of direct bit to carry	2	24
ORL	C,bit	OR direct bit to carry	2	24
ORL	C,/bit	OR complement of direct bit to carry	2	24
MOV	C,bit	Move direct bit to carry	2	12
MOV	bit,C	Move carry to direct bit	2	24
JC	rel	Jump if carry is set	2	24
JNC	rel	Jump if carry not set	2	24

Table 7. 80C51 Instruction Set Summary (Continued)

MNEMONIC		DESCRIPTION	BYTE	OSCILLATOR PERIOD
BOOLEAN VARIABLE MANIPULATION (Continued)				
JB	rel	Jump if direct bit is set	3	24
JNB	rel	Jump if direct bit is not set	3	24
JBC	bit,rel	Jump if direct bit is set and clear bit	3	24
PROGRAM BRANCHING				
ACALL	addr11	Absolute subroutine call	2	24
LCALL	addr16	Long subroutine call	3	24
RET		Return from subroutine	1	24
RETI		Return from interrupt	1	24
AJMP	addr11	Absolute jump	2	24
LJMP	addr16	Long jump	3	24
SJMP	rel	Short jump (relative addr)	2	24
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is zero	2	24
JNZ	rel	Jump if Accumulator is not zero	2	24
CJNE	A,direct,rel	Compare direct byte to A_{CC} and jump if not equal	3	24
CJNE	A,#data,rel	Compare immediate to A_{CC} and jump if not equal	3	24
CJNE	RN,#data,rel	Compare immediate to register and jump if not equal	3	24
CJNE	@Ri,#data,rel	Compare immediate to indirect and jump if not equal	3	24
DJNZ	Rn,rel	Decrement register and jump if not zero	2	24
DJNZ	direct,rel	Decrement direct byte and jump if not zero	3	24
NOP		No operation	1	12