Curs 5

5. Programarea microcontrolerelor

5.1 Limbajul de asamblare al microcontrolerului 8051

- Microcontrolerul 80C51 posedă un set de instrucţiuni orientat pe aplicaţii de monitorizare şi control;
- Există instrucţiuni la nivel de bit care permit comanda individuală a rangurilor din registre şi din porturi, fără afectarea restului de ranguri;
- Pentru accesul rapid în cadrul unor structuri de date de dimensiuni mici, există mai multe moduri de adresare şi mai multe tipuri de instrucţiuni de transfer;

Sisteme Încorporate

- Moduri de adresare
  - **Adresarea directă:** operandul este specificat printr-o adresă pe 8 biţi în cadrul instrucţiunii; se foloseşte doar pentru adresarea memoriei RAM interne şi a registrelor cu funcţiuni speciale; exemplu: MOV A,09H;
  - **Adresarea indirectă:** în cadrul instrucţiunii se specifică un registru care conţine adresa operandului; exemplu: ADD A,@Ri;
  - **Adresarea de registru:** este folosită pentru adresarea unui operand aflat în unul din registrele R0 - R7; cod eficient;
  - **Adresarea implicită:** este folosită de instrucţiuni care au un operand într-un registru predeterminat, de exemplu acumulatorul sau registrul DPTR; în aceste cazuri nu mai este necesară specificarea operandului; exemplu: ADD A,Rn;
  - **Adresarea imediată:** în cadrul instrucţiunii, după cod urmează o constantă; exemplu: ADD A,#64 sau ADD A,#64H;
  - **Adresarea indexată:** poate fi folosită doar pentru citirea memoriei de program; este foarte utilă la citirea tabelelor de căutare din memoria de program; un registru de bază de 16 biţi, DPTR sau PC, indică baza tabelei iar acumulatorul indică intrarea în tabelă; exemplu: MOVC A,@A + DPTR; adresarea indexată se foloseşte şi la instrucţiunile de salt cu selecţie: adresa destinaţie a unei instrucţiuni de salt se obţine adunând conţinutul registrului de bază cu cel al acumulatorului; exemplu: JMP @A + DPTR.

Sisteme Încorporate

- Setul de instrucţiuni
  - Sintaxa:

    etic: mnemonică destinaţie, sursă ; comentariu
  - Etic este eticheta instrucţiunii, mnemonica denumeşte instrucţiunea, destinaţia şi sursa se obţin prin moduri de adresare iar comentariul este util programatorului pentru a urmări modul de realizare al programului; eticheta şi comentariul sunt opţionale.
- Instrucţiuni aritmetice
  - Instrucţiuni cu 2 operanzi: ADD, ADDC, SUBB, MUL şi DIV; utilizează registrul A;
  - Instrucţiuni cu 1 operand: INC, DEC;
  - Instrucţiune cu operand implicit: DAA;
  - Utilizează mai multe moduri de adresare:
    - ADD A,10H - este adresare directă,
    - ADD A,#80 - este adresare implicită (operandul este în zecimal),
    - ADD A,R0 - este adresare de registru,
    - ADD A,@R1 - este adresare indirectă.

- Instrucţiuni logice:
  - Instrucţiuni care implementează funcţiile logice de bază: CPL, ANL, ORL şi XRL;
  - Operanzii se găsesc în registrul A sau pot fi furnizaţi prin adresare directă, ceea ce înseamnă că se referă la memoria RAM internă şi la SFR, prin adresare imediată sau prin adresare indirectă;
  - Instrucţiuni de rotire a acumulatorului: RR, RRC, RL şi RLC;
  - Instrucţiune de interschimbare a celor 2 jumătăţi ale acumulatorului: SWAP; o aplicaţie este conversia unui număr din binar în BCD;
    Exemplu:

    ```
    MOV   B,#10
    DIV    AB
    SWAP  A
    ADD   A,B.
    ```
  - Instrucţiunea de divizare oferă câtul în A şi restul în B; câtul este cifra zecilor, ca urmare trebuie să ajungă în jumătatea mai semnificativă a lui A; operaţia este realizată de SWAP iar instrucţiunea de adunare duce în jumătatea mai puţin semnificativă a lui A, cifra unităţilor; rezultatul, adică numărul convertit în BCD, se va afla în registrul A;

Sisteme Încorporate

- Instrucţiuni de transfer:
  - Transfer cu memoria RAM internă;
  - Transfer cu memoria externă;
  - Transfer cu memoria RAM internă:
    - De transfer,
    - De lucru cu stiva,
    - De interschimbare;
  - Instrucţiunea de transfer are mnemonica MOV iar operanzii pot fi acumulatorul sau furnizaţi în mod direct, imediat, indirect sau prin registru;
  - Primii 128 octeţi din memoria RAM internă pot fi accesaţi prin adresare directă şi indirectă iar următorii 128 octeţi din memoria RAM internă, la microcontrolerele la care sunt implementaţi, pot fi accesaţi prin adresare indirectă; adresarea directă în acest spaţiu va duce la SFR;
  - Există şi o instrucţiune de transfer pe 16 biţi care încarcă o valoare imediată în registrul DPTR;

- Instrucţiunile de lucru cu stiva sunt:
  - PUSH pentru depunere în stivă şi
  - POP pentru extragere din stivă;
  - PUSH incrementează indicatorul de stivă şi apoi depune un octet iar POP extrage un octet şi apoi decrementează SP;
  - La iniţializare microcontrolerul încarcă valoarea 07H în SP ca urmare prima locaţie din stivă la care se va face depunerea este 08H;
  - De remarcat că, dat fiind faptul că depunerile în stivă duc la creşterea conţinutului SP, există riscul ca stiva să ocupe şi zona din RAM – ul intern alocată utilizatorului, ca urmare se recomandă ca începutul stivei să fie deplasat spre partea superioară a spaţiului ocupat de RAM – ul intern;
  - Instrucţiunile de interschimbare sunt XCH la care acumulatorul este interschimbat cu un alt operand şi XCHD la care interschimbarea se va realiza doar la nivelul jumătăţii inferioare;

- Transfer cu memoria externă:
  - Instrucţiuni de transfer cu memoria externă de date şi
  - Instrucţiuni de transfer cu memoria externă de program.
  - În prima categorie intră instrucţiunea MOVX care are un operand în acumulator iar celălalt se obţine prin adresare indirectă;
  - Dacă memoria externă de date cere adrese pe 16 biţi, atunci registrul folosit pentru adresarea indirectă va fi DPTR iar dacă memoria externă de date cere doar adrese pe 8 biţi, atunci registrele implicate vor fi Ri;
  - Avantajul celei de-a doua variante este cod mai compactat;
  - Instrucţiunea MOVX este singura cale prin care programatorul poate avea acces la memoria externă de date;
  - A doua categorie este alcătuită din instrucţiunea MOVC;
  - Spre deosebire de MOVX, MOVC nu permite decât citire din memoria externă de cod;
  - Foloseşte un operand în A iar celălalt este accesat prin adresare indirectă, regiştrii implicaţi fiind A + DPTR sau A + PC;

Sisteme Încorporate

- Instrucţiunea este foarte utilă la implementarea tabelelor de căutare în memoria externă de program prin adresare indexată;
- Adresa de început a tabelei se va afla în DPTR sau PC iar indexul, care va indica intrarea în tabel dorită, se va afla în A;
- Fie următorul exemplu: se consideră un şir de 10 octeţi ASCII aflaţi în memorie începând cu adresa ADR_BAZĂ şi se doreşte trimiterea lor la o interfaţă serie; există subrutina EMISIE care preia conţinutul acumulatorului şi îl trimite la interfaţa serie; rutina este:

```
        MOV     DPTR,#ADR_BAZA
        MOV     B,#10; se iniţializează contorul care indică numărul
                ;de octeţi de trimis
        MOV     R0,#0; se iniţializează indexul
ETIC:   MOV     A,R0; se transferă în A indexul
        MOVC    A,@A+DPTR; se transferă în A octetul ce trebuie
                        ;trimis
        CALL    EMISIE; se transferă octetul la interfaţa serie
        INC     R0; se actualizează indexul
        DJNZ    B,ETIC; se repetă secvenţa pînă ce sunt trimişi toţi
                ;cei 10 octeţi
```

Sisteme Încorporate

- Instrucţiuni la nivel de bit:
  - Sunt instrucţiuni foarte utile în aplicaţii de control, în care se realizează comenzi prin intermediul unor ranguri individuale ale porturilor;
  - Operanzii sunt fie indicatorul de condiţii C fie biţi accesaţi prin adresare directă;
  - Adresele sub 80H se referă la biţi din memoria RAM internă iar adresele peste 80H, inclusiv, se referă la biţi din SFR;
  - Instrucţiuni:
    - CLR anulează indicatorul C sau un bit accesat prin adresare indirectă,
    - SETB pune la 1 aceiaşi operanzi,
    - CPL complementează aceiaşi operanzi,
    - ANL şi ORL realizează operaţia ŞI, respectiv SAU, între indicatorul C şi un bit accesat prin adresare directă,
    - MOV realizează transfer între aceeaşi operanzi;

Sisteme Încorporate

- Transferul unui bit din memorie sau dintr-un SFR la un port (de exemplu la portul P1, rangul 7):

  MOV C,BIT; se transferă bitul în indicatorul C

  MOV P1.7,C; se transferă C la port;

- Se pot realiza toate operaţiile logice asupra unor biţi individuali cu excepţia operaţiei SAU EXCLUSIV;
- Operaţia poate fi realizată prin următoarea rutină:

  MOV C,BIT1; se transferă în C bitul 1

  JNB BIT2,CONT; dacă al 2-lea bit este 0 atunci se face saltul la
  ; CONT

  CPL C; bitul 2 a fost 1, ca urmare rezultatul trebuie să fie 0

  CONT: ____

  ____

- Rezultatul operaţiei SAU EXCLUSIV între BIT1 şi BIT2 se va afla în C;

- Instrucţiuni de salt:
  - Există un subset puternic de instrucţiuni de salt care confirmă orientarea microcontrolerelor spre aplicaţii de monitorizare, comandă şi control;
  - Există 4 variante de instrucţiuni de salt necondiţionat:
    - LJMP (“Long JMP”): este o instrucţiune pe 3 octeţi din care ultimii 2 sunt adresa destinaţie; cu această instrucţiune poate fi accesată orice locaţie din spaţiul de memorie de 64 ko;
    - SJMP (“Short JMP”): este o instrucţiune pe 2 octeţi din care al 2 – lea este adresa destinaţie şi ea este relativă la conţinutul lui PC; spaţiul de memorie care poate fi accesat este limitat la – 128 până la +127 faţă de adresa instrucţiunii următoare;
    - AJMP: este o instrucţiune pe 2 octeţi care conţin adresa destinaţie pe 11 biţi (3 biţi în primul octet şi cei 8 mai puţin semnificativi în al 2 – lea octet); spaţiul care poate fi adresat constă din pagina de 2 ko în care se află şi instrucţiunea AJMP;
    - JMP @A+DPTR este o instrucţiune complexă care funcţionează ca un comutator, similar cu instrucţiunea CASE din limbajul C; adresa destinaţie va fi conţinutul registrului A plus cel al registrului DPTR ca urmare poate fi implementată uşor o tabelă de salturi;

Sisteme Încorporate

- De exemplu, se consideră că în memorie, la adresa TABELA_SALT se află o tabelă cu mai multe adrese destinaţie, fiecare ocupând 2 octeţi; următoarea rutină realizează saltul la una din aceste adrese destinaţie în funcţie de valoarea unui index care îi este furnizat rutinei:

```
MOV     DPTR,TABELA_SALT; se încarcă în DPTR adresa de
                        ; început a tabelei
MOV     A,INDEX; se încarcă în A indexul
RL      A; indexul este înmulţit cu 2 fiindcă o adresă ocupă 2 octeţi
JMP     @A+DPTR; se realizează saltul;
```

- Grupa conţine şi instrucţiuni de salt condiţionat; adresele lor destinaţie sunt relative la conţinutul lui PC:
  - JZ şi JNZ realizează saltul dacă acumulatorul este 0, respectiv diferit de 0;
  - JC şi JNC realizează saltul dacă indicatorul C este 1, respectiv 0;
  - JB (Jump if Bit Set) şi JNB realizează saltul dacă bitul indicat prin adresare directă este 1, respectiv 0; cu aceste instrucţiuni se pot lua decizii în funcţie de starea unor biţi individuali din memoria internă sau din registre, inclusiv porturi;
  - JBC realizează saltul dacă bitul este 1 şi apoi anulează bitul, indicat prin adresare directă;

Sisteme Încorporate

- Grupa mai conţine 2 instrucţiuni complexe;
- Prima este CJNE (“Compare and Jump if Not Equal”): instrucţiunea are 2 operanzi şi se realizează saltul doar dacă cei 2 operanzi nu sunt egali; este utilă în controlul executării buclelor;
- Un alt efect al instrucţiunii este acela că poziţionează indicatorul C: dacă primul operand este mai mic ca al 2 – lea atunci C va conţine pe 1 iar în caz contrar îl va conţine pe 0; acest efect este folosit pentru a realiza operaţiile “mai mare ca” sau “mai mic ca”;
- A doua este DJNZ (“Decrement and Jump if Not Zero”): este decrementat un registru sau o locaţie de memorie din RAM – ul intern şi dacă nu s-a ajuns la 0, se face saltul;
- Principala utilizare a instrucţiunii este pentru controlul executării buclelor; în operandul instrucţiunii se încarcă contorul care va arăta de câte ori să se execute o buclă iar aceasta va trebui să se încheie cu DJNZ;
- Adresa destinaţie pentru ambele instrucţiuni este relativă la PC;
- La toate instrucţiunile de salt, destinaţia este dată de programator sub formă simbolică, de etichetă, programului asamblor revenindu-i sarcina de a da o valoare concretă etichetei;

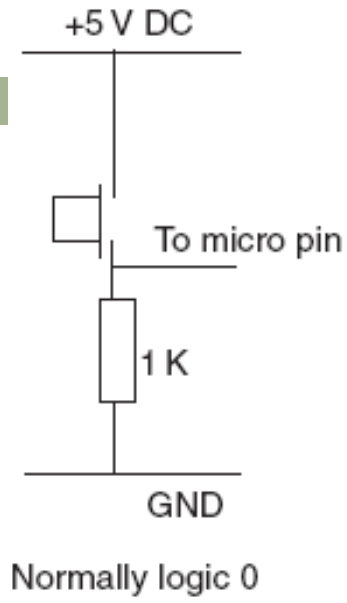
Sisteme Încorporate

- Instrucţiuni de lucru cu subrutinele
  - Instrucţiuni de apel de subrutină:
    - ACALL, la care destinaţia trebuie să se afle în aceeaşi pagină ca şi instrucţiunea de apel (2 octeţi);
    - LCALL la care destinaţia poate fi oriunde în spaţiul de memorie extern de program (3 octeţi);
    - Destinaţiile instrucţiunilor de apel de subrutină pot fi date de programator sub formă simbolică, de etichetă
  - Instrucţiunide revenire din subrutină: RET şi RETI;
  - Instrucţiunea RETI este recomandată la încheierea subrutinelor de tratare a cererilor de întrerupere întrucât, în plus faţă de instrucţiunea RET, reface starea sistemului de întreruperi la ieşirea din rutina de tratare;
- Instrucţiunea NOP
  - Este o instrucţiune fără efect;
  - În mod uzual este folosită pentru întârzieri sau pentru a înlocui instrucţiuni care trebuiesc eliminate din program;

Sisteme Încorporate

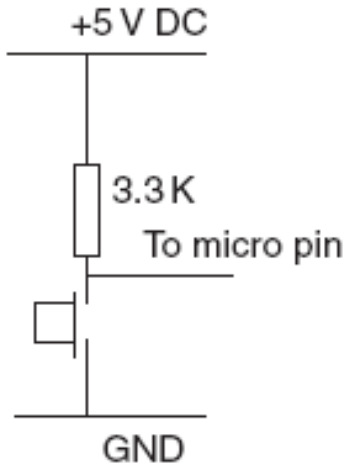
- **Aplicaţii**

1. Testarea unor întrerupătoare (la acţionare să se genereze un semnal dreptunghiular, pin 7, port 1):
   - Cazul 1: dacă întrerupătorul nu este apăsat, la terminalul portului (pin) va fi 0V; dacă întrerupătorul este activat => la pin vor fi 5V (circuit ce produce în mod normal 0V)



Normally logic 0

   - Cazul 2: dacă întrerupătorul nu este apăsat, la terminalul portului (pin) va fi 5V; dacă întrerupătorul este activat => la pin va fi 0V (circuit ce produce în mod normal 5V)



Normally logic 1

Sisteme Încorporate

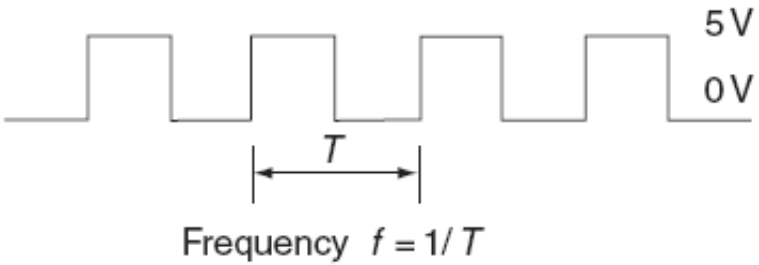
- Cazul 1:

```
ORG 0 ; setează adresa de start la 0
    SJMP START ; short jump la eticheta START
    ORG 0040H ; pune următoarea linie de program la adresa 0040H
    START:
        JB P1.0,PULSE ; sare la PULSE dacă pin 0 port 1 are nivelul logic 1
        CLR P1.7      ; altfel pune la 0 pin 7 port 1
        SJMP START    ; du-te la eticheta START pt verificare întrerupător
    PULSE: ; am apăsat => generează un semnal dreptunghiular :-)
        SETB P1.7     ; setează pin 7 port 1 la 1 logic
        CLR P1.7      ; pune pin 7 port 1 la 0 logic
        AJMP START    ; du-te la eticheta START pt verificare întrerupător
END ; terminare program
```

- Cazul 2:

```
CHECK: JNB P1.0,PULSE ; du-te la etich. PULSE dacă pin 0 port 1 e 0
SJMP CHECK
PULSE: ;.....
```

2. Un microcontroler bazat pe 80C51 (ex. P89C664) are o frecvenţă de clock de 11.0592 Mhz. Care este perioada pentru fiecare ciclu?



Durata unui ciclu (T) $= \dfrac{1}{11.0592 \times 10^6} = 90.423\ \text{ns} \qquad (n = 10^{-9})$

Analiza exemplului Testarea unor întrerupătoare:

SETB durează 6 cicluri de clock a microcontrolerului

CLR durează 6 cicluri de clock a microcontrolerului

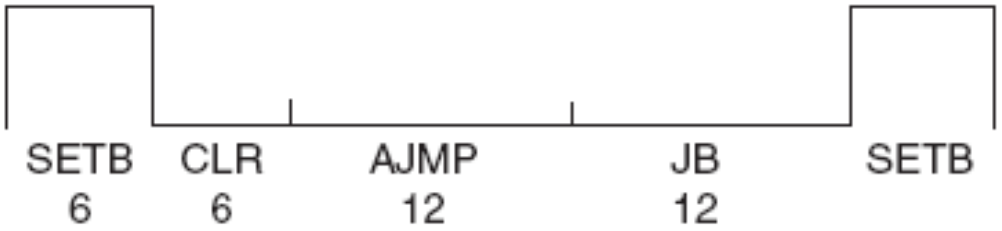
AJMP durează 12 cicluri de clock a microcontrolerului

JB durează 12 cicluri de clock a microcontrolerului

Sisteme Încorporate

=>Obţinem un semnal inegal ON/OFF, 1/0 (deci nu un semnal dreptunghiular egal pe durata 1/0)

- SETB durează 6 cicluri de clock
- CLR durează în cazul nostru 30 de cicluri de clock deoarece (!) o analiză mai atentă arată că această instrucţiune este activată şi pe durata lui AJMP şi JB



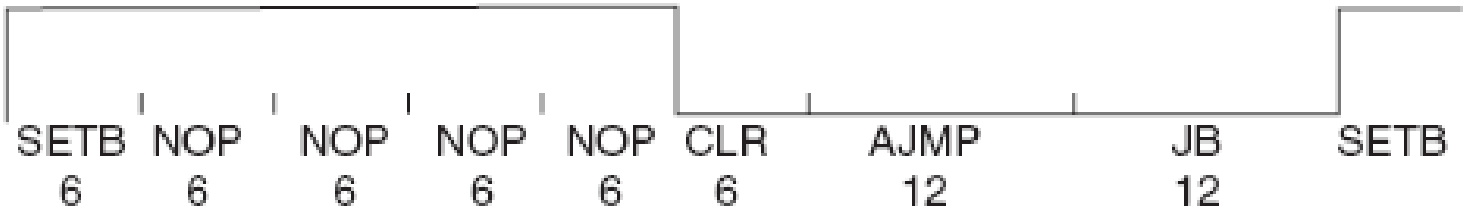
SOLUŢIE:

- Folosirea instrucţiunii NOP (**N**o **OP**eration)
- NOP durează 6 cicluri de clock ale microcontrolerului

```
ORG 0 ; setează adresa de start la 0
    SJMP START ; short jump la eticheta START
    ORG 0040H ; pune următoarea linie de program la adresa 0040H
    START:
        JB P1.0,PULSE ; sare la PULSE dacă pin 0 port 1 are nivelul logic 1
        CLR P1.7      ; altfel pune la 0 pin 7 port 1
        SJMP START    ; du-te la eticheta START pt verificare întrerupător
    PULSE: ; am apăsat => generează un semnal dreptunghiular :-)
        SETB P1.7     ; setează pin 7 port 1 la 1 logic
        NOP           ; menţine 1 logic pe pinul 7, port 1
        NOP
        NOP
        NOP
        CLR P1.7      ; pune pin 7 port 1 la 0 logic
        AJMP START    ; du-te la eticheta START pt verificare întrerupător
END ; terminare program
```

Sisteme Încorporate

- Semnalul obţinut: 60 cicluri de clock, adică 60 x 90.423 ns = 5.43 us, deci frecvenţa acestui semnal este 1/5.43 = 184 kHz => limitare la frecvenţa maximă a microcontrolerului



- Alte aplicaţii la care poate fi folosit NOP?

3. Generarea unor întârzieri (time delays):

- Metodă: decrementarea valorilor din registre

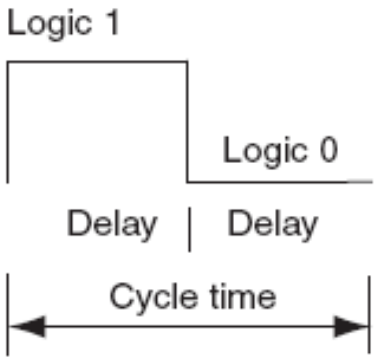
```
DELAY: MOV R0,#number ; mută un număr într-un registru pe 8 biţi
                      ; registrul R0
TAKE: DJNZ R0,TAKE ; decrementează R0 până când devine 0

RET                ; întoarcere din subrutina DELAY
```

- Analiza codului:
- MOV durează 6 cicluri de clock; DJNZ, RET, ACALL câte 12 cicluri de clock;
- Când variabila “number” are o valoare redusă => introducere de NOPs (un total de 24 cicluri de clock):

$$\text{Delay time} = (24 + 12 + 6 + (\text{number} \times 12) + 12) \text{ clock cycles}$$
$$\text{Delay time} = (54 + (12 \times \text{number})) \text{ clock cycles}$$

4. Pentru un microcontroler 80C51 care are un semnal de clock controlat de un cristal de 11.0592 Mhz, să se scrie un program în limbaj de asamblare care va genera un semnal dreptunghiular de 5kHz la pinul 7 al portului 1, în momentul în care un întrerupător cauzează un 1 logic la pinul 0 al portului 1.
   - Frecvenţa semnalului de clock: 11.0592 Mhz
   - Perioada unui ciclu de clock: 1/11.0592 = 90.423 ns
   - Frecvenţa semnalului: 5 kHz
   - Perioada unui ciclu a semnalului: 1/5 kHz = 200 us
   - Întârzierea cerută este jumătate din această valoare pt. că semnalul dreptunghiular are o durată egală pe 1 şi pe 0;



- Prin urmare:
  - Delay = 100 us = (54 + (12 x number)) x 90.423 ns
  - ⇒ **Number = ((100 us / 90.423 ns) – 54)/12 = aprox. 88 (zecimal)**

```
 ORG 0 ; setează adresa de start la 0
    SJMP START ; short jump la eticheta START
    ORG 0040H ; pune următoarea linie de program la adresa 0040H
    START:
        JB P1.0,PULSE          ; sare la PULSE dacă pin 0 port 1 are nivelul logic 1
        CLR P1.7               ; altfel pune la 0 pin 7 port 1
        SJMP START  ; du-te la eticheta START pt verificare întrerupător
    PULSE: ; am apăsat => generează un semnal dreptunghiular :-)
        SETB P1.7      ; setează pin 7 port 1 la 1 logic
        ACALL DELAY
        NOP                    ; menţine 1 logic pe pinul 7, port 1
        NOP
        NOP
        NOP
        CLR P1.7               ; pune pin 7 port 1 la 0 logic
        ACALL DELAY
        AJMP START  ; du-te la eticheta START pt verificare întrerupător
DELAY: MOV R0, #88
TAKE: DJNZ R0, TAKE
            RET
        END ; terminare program
```

5. Delay în buclă dublă (domeniul milisecundelor): folosind tehnica prezentată anterior, presupunând că frecvenţa de clock este de 11.0592 MHz, să se scrie un program care generează un puls de 20 kHz pe pinul 7 al portului 1 al microcontrolerului.

```
DELAY:
    MOV R1,#number1
INNER:
    MOV R0,#number2
TAKE:
    DJNZ R0,TAKE
    DJNZ R1,INNER
    RET
```

- Întârzierea aproximativă = (number 1) x (number 2) x 12 cicluri de clock
- Exemplu: number 1 = 200, number 2 = 240, 1 ciclu de clock = 90.423 ns

=> Delay = 200 x 240 x 12 x 90.423ns = 52.1 ms

6. Delay în buclă triplă (domeniul secundelor):

```
DELAY:
    MOV R2,#number1
OUTER:
    MOV R1, #number2
INNER:
    MOV R0,#number3
TAKE:
    DJNZ R0,TAKE
    DJNZ R1,INNER
    DJNZ R2,OUTER
    RET
```

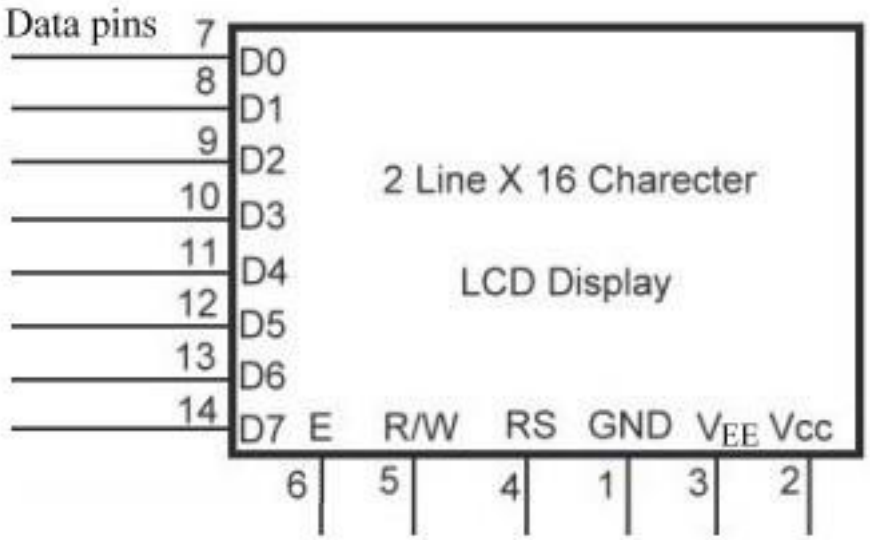
- Întârzierea aproximativă = (number 1) x (number 2) x (number 3) x 12 cicluri de clock
- Exemplu: number 1 = 40, number 2 = 200, number 3 = 240, 1 ciclu de clock = 90.423 ns

=> Delay = 40 x 200 x 240 x 12 x 90.423ns = aprox. 2s (suficient pt a observa aprinderea/stingerea unui LED)

Sisteme Încorporate

7. Conectarea unui circuit LCD LM016L la microcontrolerul 80C51

- 14 terminale



Terminal	Simbol	I/O	Descriere
1	GND	-	Masă
2	Vcc	-	+5V
3	V_{EE}	-	Control contrast
4	RS	I	Selecţie registru comandă/date
5	R/W	I	Selecţie write/read
6	E	I/O	Activare
7	D0	I/O	Magistrala de date pe 8 biţi
8	D1	I/O	Magistrala de date pe 8 biţi
9	D2	I/O	Magistrala de date pe 8 biţi
10	D3	I/O	Magistrala de date pe 8 biţi
11	D4	I/O	Magistrala de date pe 8 biţi
12	D5	I/O	Magistrala de date pe 8 biţi
13	D6	I/O	Magistrala de date pe 8 biţi
14	D7	I/O	Magistrala de date pe 8 biţi

7. Conectarea unui circuit LCD LM016L la microcontrolerul 80C51

Vcc, VEE

Vcc este utilizat pentru a se conecta alimentarea la LCD, iar VEE este folosit pentru controlul contrastului LCD-ului.

RS – selecţie registru

Circuitul LCD dispune de două registre importante. Terminalul RS este folosit pentru selecţia acestor registre, după cum urmează: dacă RS = 0, este selectat registrul de comandă al instrucţiunii, permiţând utilizatorului să trimită o comandă, ca de exemplu “clear display”, “cursor at home” etc. Dacă RS = 1, este selectat registrul de date, permiţând utilizatorului să trimită datele care vor fi afişate pe LCD.

R/W – read/write

Acest terminal permite ca utilizatorul să scrie informaţie la LCD sau să citească informaţie de la acesta. R/W = 1 când se citeşte; R/W = 0 când se scrie.

E – enable

Acest terminal este folosit de LCD pentru a reţine informatia furnizată la terminalele de date. Când datele sunt furnizate către terminalele de date, trebuie aplicat un puls “high-to-low” pentru ca LCD-ul să reţină informaţia de la aceste terminale.

D0 – D7

Terminalele de date pe 8 biţi, D0-D7 sunt folosite pentru a trimite informatia la LCD sau pentru a trimite conţinuturile regiştrilor interni ai LCD-ului

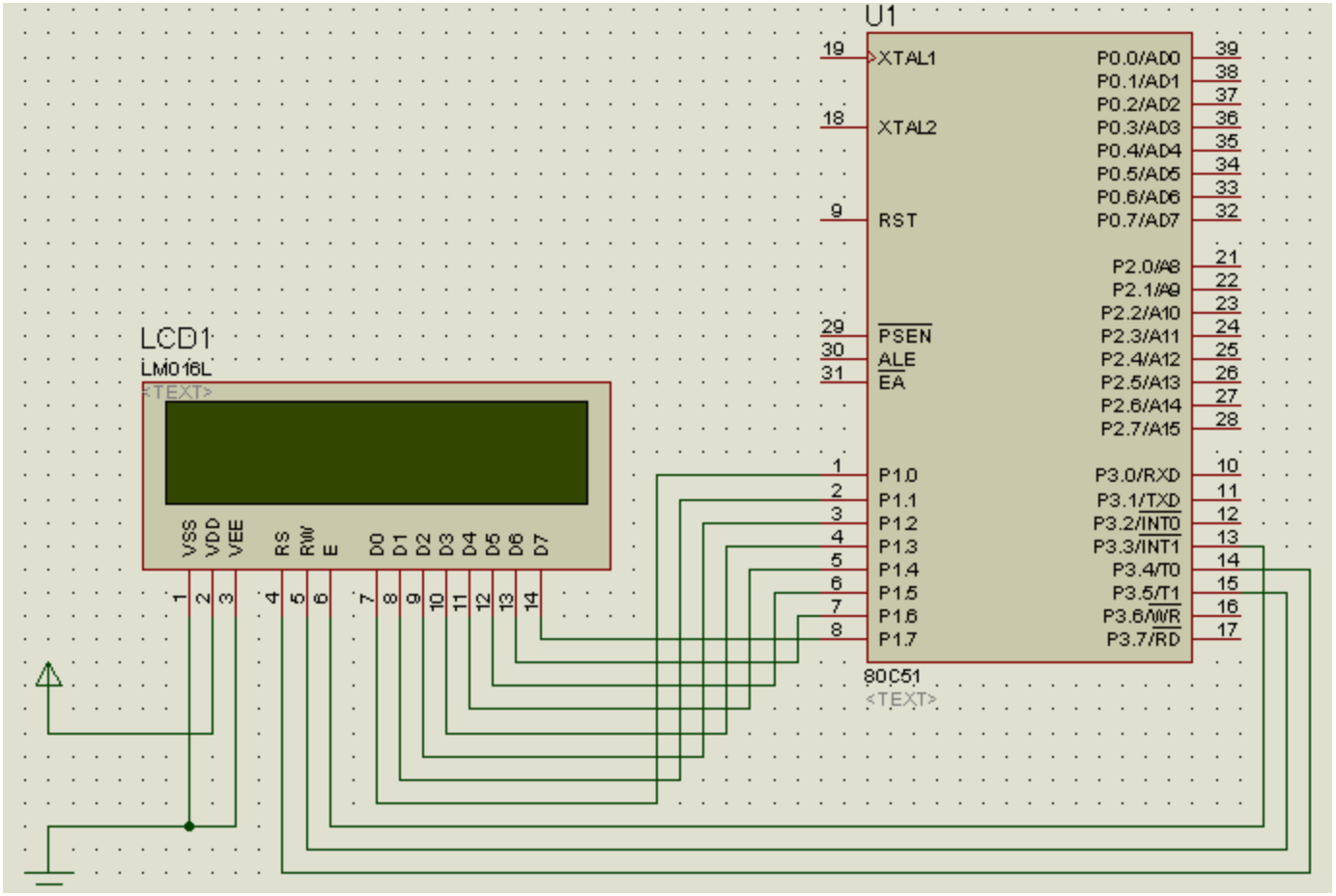
7. Conectarea unui circuit LCD LM016L la microcontrolerul 80C51

- Pentru a afişa litere şi numere, trimitem coduri ASCII pentru literele de la A la Z, a la z, şi cifrele 0-9 la aceşti pini în timp ce RS = 1
- coduri de comandă de instrucţiuni care pot fi trimise la LCD pentru a şterge informaţia prezentă pe display sau pentru a forţa cursorul la poziţia iniţială sau pentru a face să clipească cursorul etc.
- Folosim, de asemenea RS=0 pentru a verifica bit-ul ce reprezintă busy flag, pentru a vedea dacă LCD-ul este pregătit să primească informaţia.
- Busy flag-ul este D7 si poate fi citit când R/W = 1 şi RS=0. Când D7 = 1 (busy flag = 1), LCD-ul este ocupat cu diferite operaţii interne şi prin urmare nu va accepta informaţie nouă. Când D7 = 0, LCD-ul este gata să primească o nouă informaţie.

Cod (hexa)	Comandă pentru registrul de instrucţiune
1	Clear display screen
2	Return home
4	Shift cursor to left
5	Shift display right
6	Shift cursor to right
7	Shift display left
8	Display off, Cursor off
A	Display off, Cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning of 1st line
C0	Force cursor to beginning of 2nd line
38	2 lines, 5x7 matrix

7. Conectarea unui circuit LCD LM016L la microcontrolerul 80C51

- Terminalele de date D0-D7 sunt conectate la portul P1.0- P1.7 ale microcontrolerului 80C51.
- Terminalul Enable şi RS la P3.3, respectiv P3.4, iar R/W la P3.5.



7. Conectarea unui circuit LCD LM016L la microcontrolerul 80C51

```
org 0000
    MOV A,#38H              ; initializare LCD 2 linii, matrice
      5x7
    ACALL COMMAND           ; apeleaza subrutina command
    ACALL DELAY             ; apeleaza subrutina delay

    MOV A,#0EH              ; display on, cursor on
    ACALL COMMAND
    ACALL DELAY
    MOV A,#01H              ; Clear LCD
    ACALL COMMAND
    ACALL DELAY
    MOV A,#06H              ; shift cursor right
    ACALL COMMAND
    ACALL DELAY
    MOV A,#80H              ; cursorul la inceputul primei linii
    ACALL COMMAND
    ACALL DELAY

    MOV A,#'L'              ; afiseaza litera L
    ACALL data1
    ACALL DELAY

COMMAND:  MOV P1,A          ; Trimite comanda la LCD

          MOV C,0
          MOV P3.4,C
          CLR P3.4          ; RS=0 pentru comanda
          CLR P3.5          ; R/W=0 pentru scriere
          SETB P3.3         ; E=1 pentru puls H
          CLR P3.3          ; E=0 pentru puls H-to-L
          RET

data1:    MOV P1,A          ; Scrie data1 la LCD
          SETB P3.4         ; RS=1 pentru data1
          CLR P3.5          ; R/W=0 pentru scriere
          SETB P3.3         ; E=1 pentru puls H
          CLR P3.3          ; E=0 for H-to-L pulse
          RET

DELAY:    MOV R2,#50
D1:       MOV R3,#0FFH
D2:       DJNZ R3,D2
          DJNZ R2,D1
          RET

END
```

7. Conectarea unui circuit LCD LM016L la microcontrolerul 80C51

Exerciţiul 1: Să se construiască un program care să afişeze pe LCD textul LCD

Exerciţiul 2: Să se construiască un program care să afişeze pe primul rând al LCD-ului textul LCD, iar pe rândul următor LM016L.