ORGANIZAREA MEMORIEI

MEMORIA PROGRAM

Microcontrolerul 8051 are spaţii separate de adresare pentru Memoria Program şi Memoria de Date. Memoria program poate avea maxim 64 K bytes lungime. Primii 4 K pot fi selectaţi de pe cip. Figura de mai jos arată harta memoriei program pentru microcontrolerul 8051.

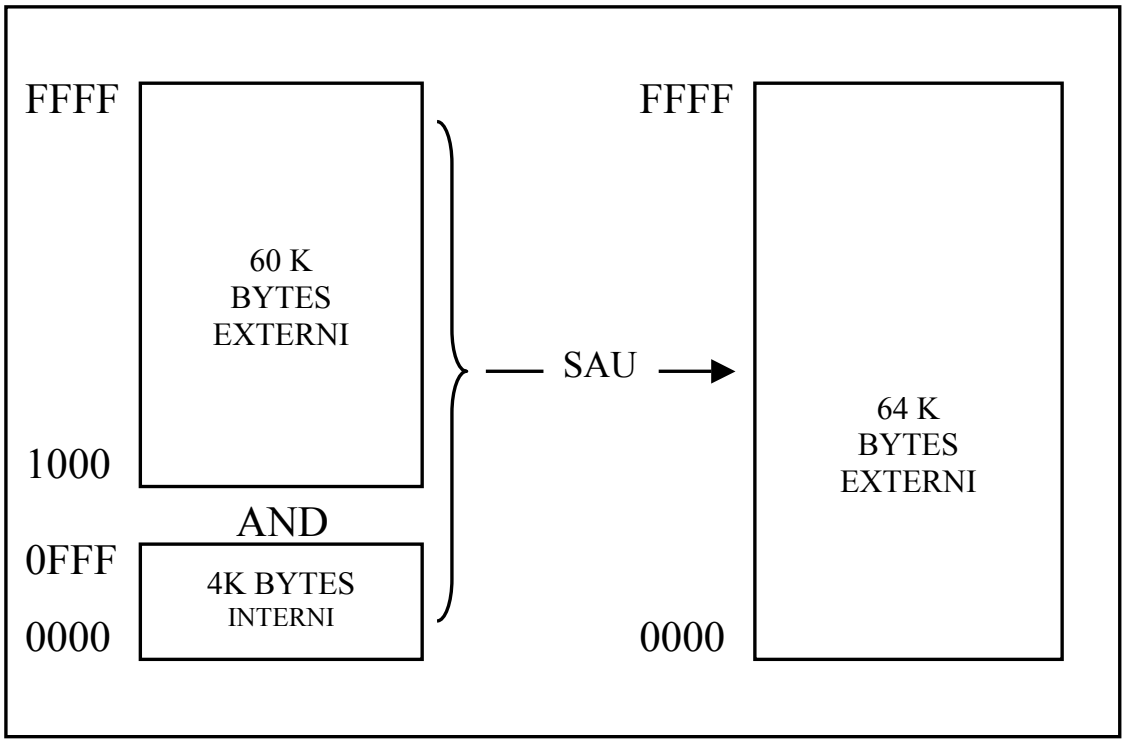


Fig. 1 Memoria Program a circuitului 8051

MEMORIA DE DATE

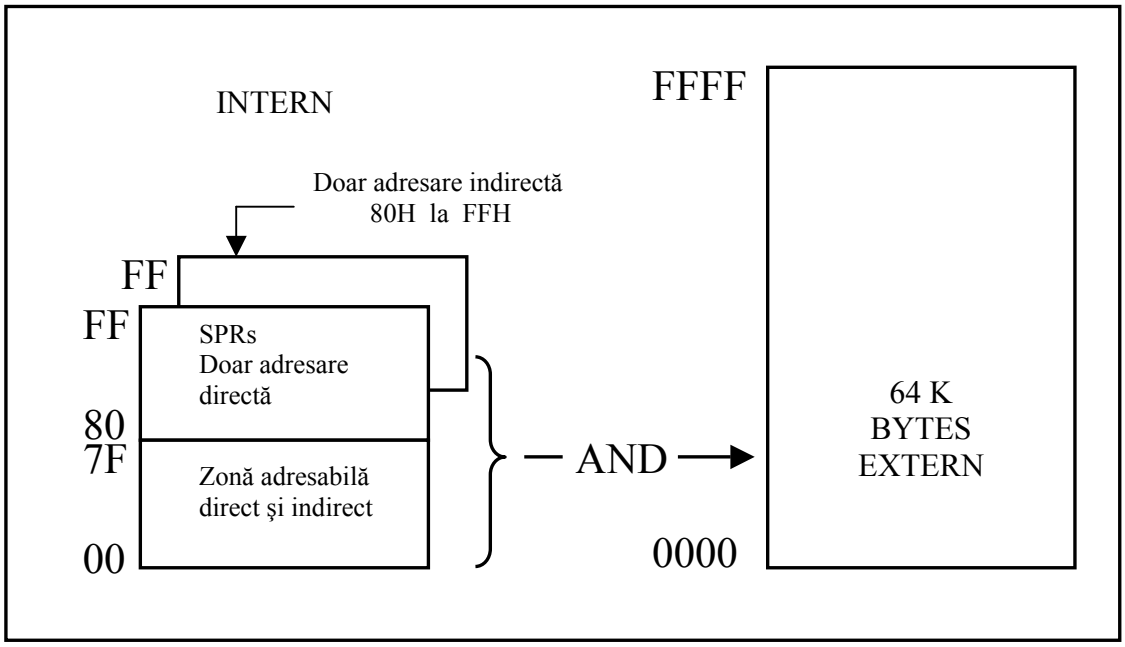


Fig. 2 Memoria de Date a circuitului 8051

Circuitul 8051 poate adresa o Memorie de Date externă de până la 64k bytes. Instrucţiunea "MOVX" este utilizată pentru a accesa memoria externă de date.
Circuitul 8051 are 128 bytes on-chip RAM, plus un număr de „Special Function Registers" (SFRs). Cei 128 bytes pot fi accesaţi prin adresare directă (MOV data addr) sau adresare indirectă (MOV @Ri).

ZONA ADRESATĂ INDIRECT:

Observaţi că în Fig. 2 SFR-urile şi RAM-ul indirect adresat au aceleaşi adrese (80H – 0FFH). Oricum ele sunt două zone separate şi sunt accesate în mod diferit.
Spre exemplu instrucţiunea
MOV 80H, #0AAH
Scrie 0Aah în Portul 0 care este unul din SFR-uri şi instrucţiunea
MOV R0, #80H
MOV @R0, #0BBH
Scrie 0BBH în locaţia 80H din RAM-ul de date. Astfel după execuţia celor două instrucţiuni anterioare Portul 0 va conţine 0AAH şi locaţia 80 din RAM va conţine 0BBH.

ZONA ADRESATĂ DIRECT ŞI INDIRECT:

Cei 128 de bytes ai RAM-ului care pot fi adresaţi direct şi indirect pot fi împărţiţi în trei segmente, aşa cum urmează şi se arată în Fig. 3.

1. Register Banks 0-3: Locaţiile 0 până la 1FH (32 bytes). ASM-51 şi dispozitivul după reset sunt setate implicit registrelor din Bank-ul 0. Pentru a folosi alt bank de registre utilizatorul trebuie să-l selecteze în program. Fiecare register bank conţine 8 registre de câte un byte, de la 0 până la 7.
Reset iniţializează Stack Pointer-ul la locaţia 07H şi este incrementat odată cu startul de la locaţia 08H care este primul registru (R0) al celui de al doilea bank.de registre. Astfel, în vederea folosirii mai mult de un bank de registre, SP-ul ar trebui iniţializat cu o locaţie diferită din RAM care nu e folosită pentru păstrare de date (de exemplu, partea mai semnificativă a RAM-ului).

2. Bit Addressable Area: 16 bytes au fost asignaţi pentru acest segment, 20H-2FH. Fiecare din cei 128 biţi ai acestui segment pot fi accesaţi direct (0-7FH).
Biţii pot fi referiţi în două moduri, ambele acceptate de ASM-51. Un mod este de a ne referi la adresa lor de exemplu, de la 0 la 7FH. Celălalt este să ne referim la biţii 20H - 2FH. Astfel biţii 0-7 pot fi referiţi şi ca biţii 20.0-20.7, şi biţii 8-FH sunt la fel cu 21.0-21.7 şi aşa mai departe.
Fiecare din cei 16 bytes din acest segment pot fi adresaţi şi ca un byte.

3. Scratch Pad Area: Bytes 30H la 7FH sunt disponibili utilizatorului ca date din RAM. Oricum, dacă pointerul de stivă a fost iniţializat spre această zonă, un număr destul de bytes ar trebui lăsaţi ca să prevină distrugerea datelor din SP.

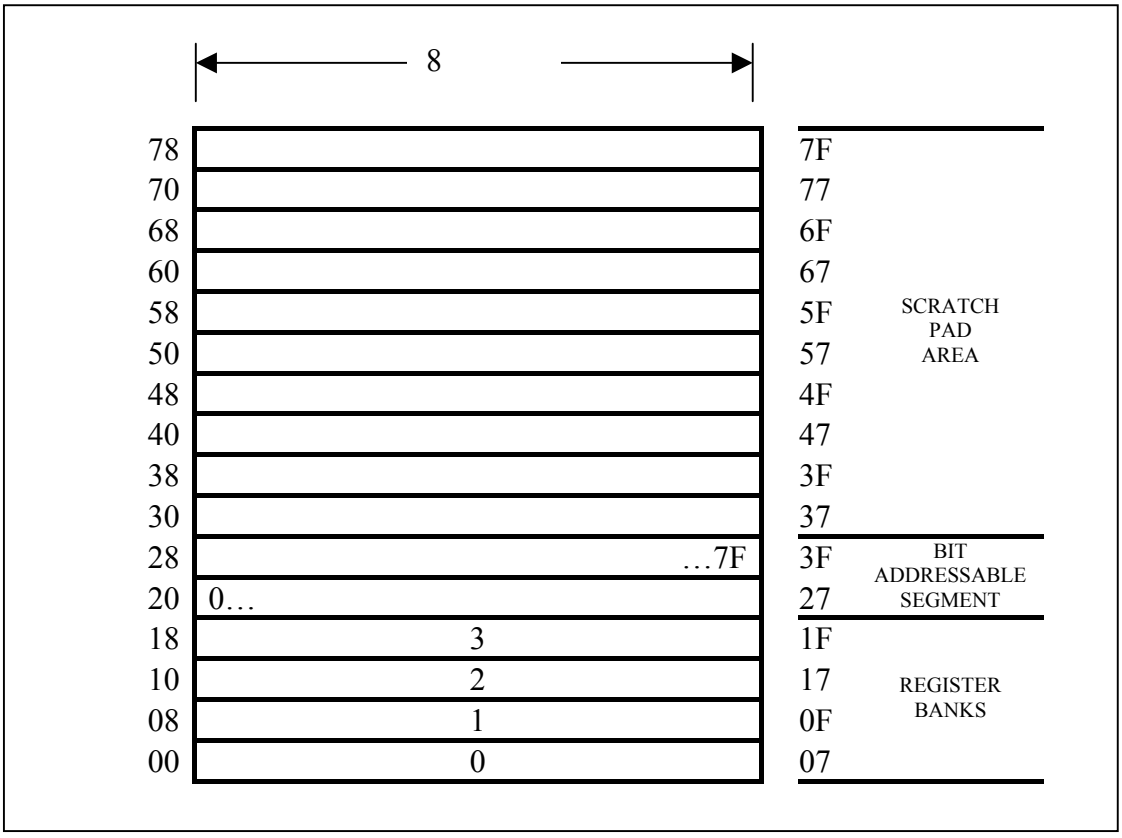


Fig. 3 arată diferitele segmente ale memoriei RAM on-chip.

Prezentare generală a asamblorului ASM51

1. Introducere

Asamblorul Asm51 primeşte o „sursă" scrisă în limbaj de asamblare creată cu un editor de text şi o translează într-un fişier obiect. Această translare este făcută în două treceri prin fişierul sursă. În cele ce urmează va fi prezentată sintaxa cerută de asamblor pentru a genera fişierul obiect fără erori.

2. Simboluri

Simbolurile sunt reprezentări alfanumerice ale constantelor, adreselor, macrourilor, etc. Setul legal de caractere este format din setul de litere (atât mici, cât şi mari), numerele zecimale (0..9) şi caractere speciale, semnul întrebării (?) şi underscore (_). Pentru a nu fi confundat un simbol cu un număr, toate simbolurile trebuie să înceapă cu o literă sau un caracter special. Asamblorul nu face distincţie între literele mari şi literele mici, astfel următoarele simboluri reprezintă acelaşi lucru pentru asamblor: PORT1 şi port1.

Simbolurile pot fi definite numai o singură dată. Acestea pot avea o lungime maximă de 255 caractere, deşi numai primele 32 sunt semnificative.

Există anumite simboluri rezervate care nu pot fi definite de utilizator. Aceste simboluri reprezintă directivele asamblor, instrucţiunile 8051, simbolurile pentru operanzii impliciţi şi următorii operatori din timpul asamblării care au simboluri alfanumerice: EQ, NE, GT, GE, LT, LE, HIGH, LOW, MOD, SHR, SHL, NOT, AND, OR şi XOR.

Simbolurile pentru operanzii impliciţi include simbolurile: A, AB, C, DPTR, PC, R0, R1, R2, R3, R4, R5, R6, R7, AR0, AR1, AR2, AR3, AR4, AR5, AR6 şi AR7.

3. Etichete

Etichetele sunt un caz special de simboluri. Acestea sunt folosite numai înaintea expresiilor care au adrese fizice asociate cu ele. Etichetele trebuie să respecte aceleaşi reguli ca şi la crearea simbolurilor, numai că trebuie să fie urmate de caracterul ":". Un exemplu de etichetă urmată de o instrucţiune este:

```
START:    MOV A,#23
```

4. Elemente de control ale asamblorului

Aceste elemente de control sunt folosite pentru a specifica de unde asamblorul îşi ia fişierul sursă, unde pune fişierul obiect şi cum formatează fişierul de listing. În tabelul de mai jos sunt prezentate elementele de control ale asamblorului.

$DATE(date)	Pune data în headerul de pagină
$INCLUDE(file)	Inserează fişierul în sursă
$NOLIST	Nu mai creează listingul de ieşire
$MOD52	Foloseşte simbolurile predefinite 8052
$NOMOD	Nu se folosesc simboluri predefinite
$NOOBJECT	Nu se generează fişier obiect
$NOPAGING	Listing fără paginare
$PAGEWIDTH(n)	Numărul de coloane pe pagina de listing
$NOPRINT	Listingul nu va fi generat la ieşire
$NOSYMBOLS	Tabla de simboluri nu va fi generată
$LIST	Permite listingului să fie generat la ieşire
$MOD51	Foloseşte simbolurile predefinite 8051
$MOD44	Foloseşte simbolurile predefinite 8044
$OBJECT(file)	Plasează ieşirea obiect în fişier
$PAGING	Împarte listingul de ieşire în pagini
$PAGELENGTH(n)	Numărul de linii pe o pagină de listing
$PRINT(file)	Plasează ieşirea listingului în fişier
$SYMBOLS	Ataşează tabela de simboluri la listing
$TITLE(string)	Plasează un şir în headerul de pagină

După cum se poate vedea din tabel, toate elementele de control ale asamblorului sunt prefixate de semnul dolar ($). Este permis numai un singur element de control pe linie. Totuşi, comentariile se pot afla pe aceeaşi linie cu controlul.

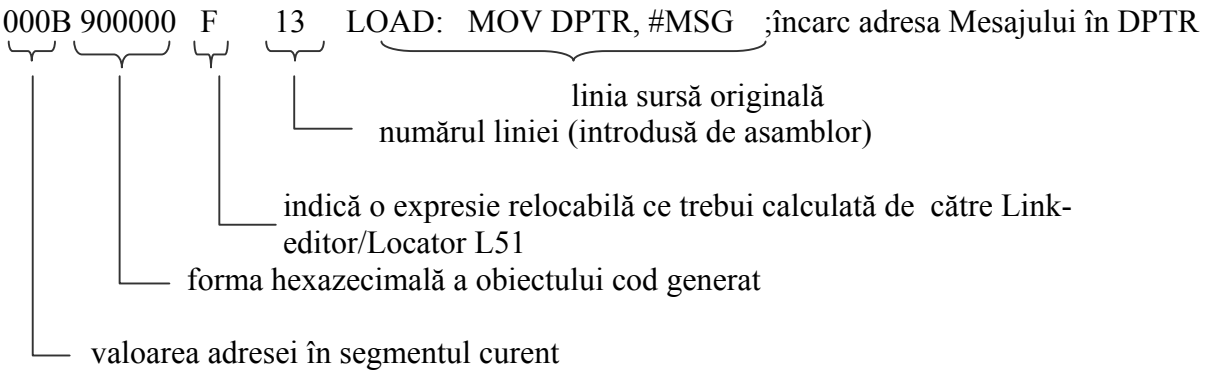
5. Directive asamblor

Sunt folosite pentru a defini simboluri, pentru a rezerva spaţiu de memorie, pentru a păstra valori în memoria program şi a schimba diferite spaţii de memorie. Tabelul următor prezintă un sumar al acestor directive.

EQU	Defineşte un simbol
IDATA	Defineşte un simbol adresat indirect în memoria internă
CODE	Defineşte un simbol în memoria program
DBIT	Rezervă biţi în memoria de lucru la nivel de bit
DW	Păstrează cuvinte în memoria program
END	Marchează sfârşitul fişierului sursă
DSEG	Selectează spaţiul din memoria internă de date
ISEG	Selectează spaţiul adresat indirect din memoria internă de date
IF	Începutul unui bloc de asamblare condiţionată
ENDIF	Sfârşitul unui bloc de asamblare condiţionată
DATA	Defineşte un simbol în memoria internă
XDATA	Defineşte un simbol în memoria externă
BIT	Defineşte un bit intern
DS	Rezervă octeţi în memoria de date
DB	Păstrează valori pe octet în memoria program
ORG	Setează valoarea numărătorului de segment
CSEG	Selectează un spaţiu de memorie program
XSEG	Selectează un spaţiu de memorie de date externă
BSEG	Selectează un spaţiu de memorie adresabilă la nivel de bit
USING	Selectează bacul de regiştri
ELSE	Bloc alternativ de asamblare condiţionată

Este permisă numai o singură directivă pe linie, pe aceeaşi linie fiind posibilă includerea comentariilor.

Înţelesul câmpurilor din fişierul listing este următorul:



Exemplu program:

```
$INCLUDE (REG552.INC)     ;în acest program sunt folosite "trăsăturile" microcontrolerului 80C552
ORG 8000H                 ;memoria program din uC va fi ocupată începând de la adresa 8000H
LED EQU P4.4              ;definesc pinul de Port P4.4 = „LED"
START:
        CLR LED           ;P4.4 = „0" logic          (sting LED – ul)
        CALL DELAY        ;apelez o întârziere
        SETB LED          ;P4.4 = „1" logic (aprind LED – ul)
        CALL DELAY        ;apelez o întârziere
        JMP START         ;salt la Eticheta START

DELAY:                    ;rutină pentru realizarea unei întârzieri de 400ms
        MOV R7,#4
        MOV R6,#14
        MOV R5,#42
DELAY1:
        DJNZ R5,DELAY1
        DJNZ R6,DELAY1
        DJNZ R7,DELAY1
        RET               ;întoarcere din apelare

END                       ;sfârşit de program
```