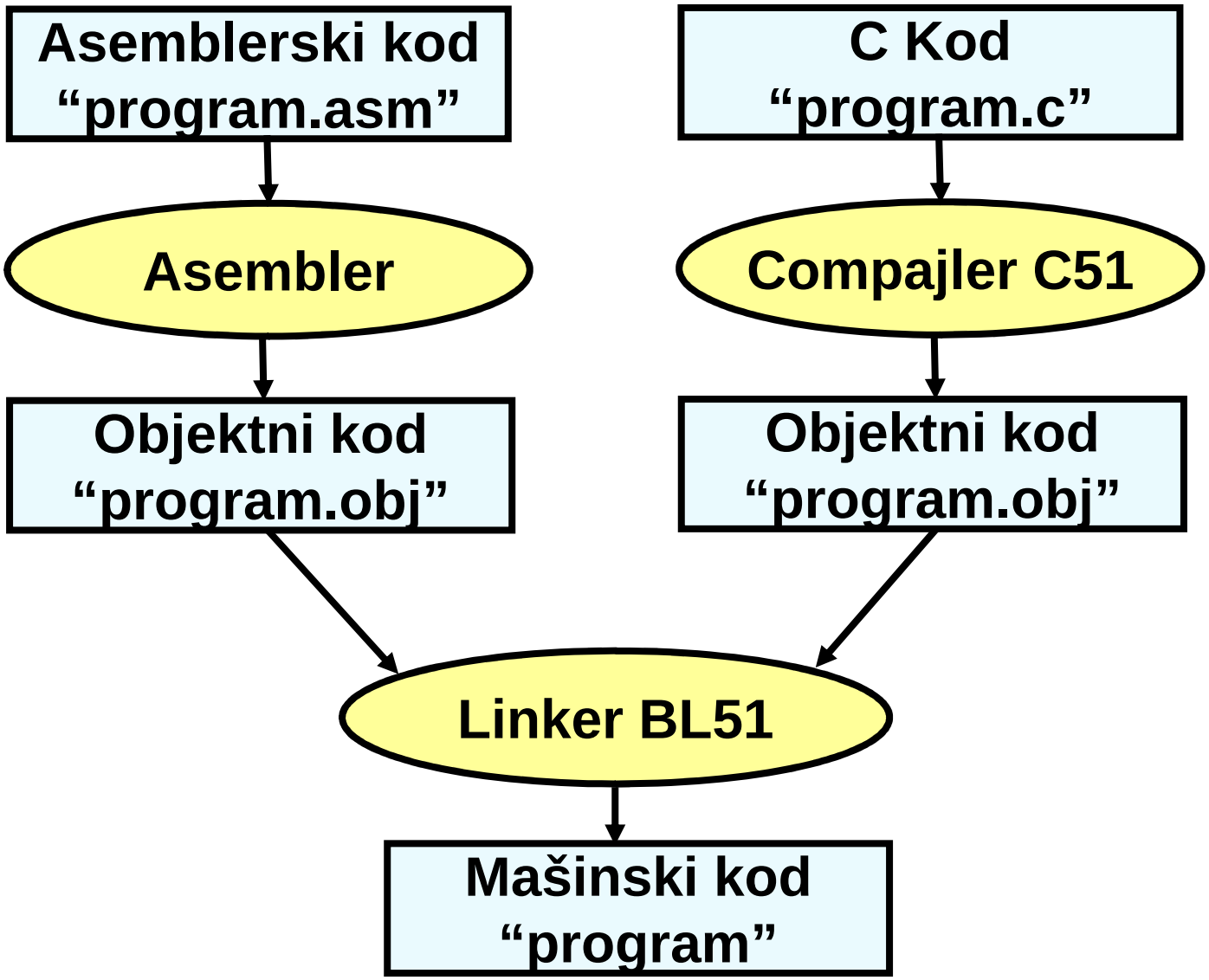
Mikrokontroleri

Mikrokontroleri

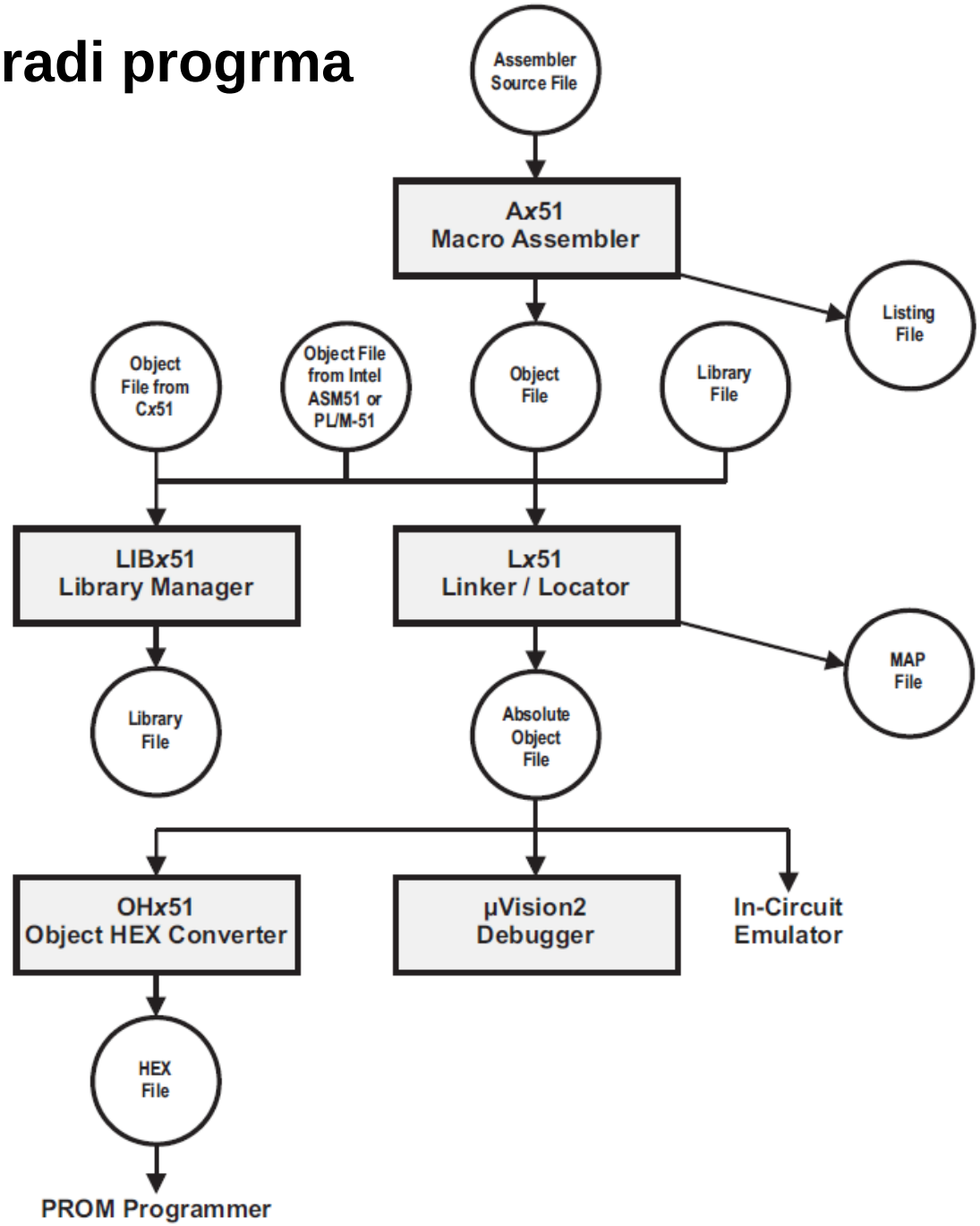
Poglavlje V

Programiranje va asembleru

Tok izrade programa



Koraci u izradi progrma



Tok izrade programa

Extension	Content and Description
.A51 **.ASM** **.SRC**	Source code file: contains ASCII text that is the input for the **A*x*51** assembler.
.C **.C51**	C source code file: contains ASCII text that is the input for the **C*x*51** ANSI C compiler.
.INC **.H**	Include file: contains ASCII text that is merged into an source code file with the include directive. Also these files are input files for **A*x*51** or **C*x*51**.
.OBJ	Relocatable object file: is the output of the **A*x*51** or **C*x*51** that contains the program code and control information. Several relocatable object files are typically input files for the **L*x*51** Linker/Locater.
.LST	Listing object file: is generated by **A*x*51** or **C*x*51** to document the translation process. A listing file typically contains the ASCII program text and diagnostic information about the source module. Appendix F describes the format of the **A*x*51** listing file.
. *(none)* **.ABS**	Absolute object file: is the output of the **L*x*51**. Typically it is a complete program that can be executed on the *x*51 CPU.
.M51 **.MAP**	Linker map file: is the listing file generated from **L*x*51**. A map file contains information about the memory usage and other statistic information.
.HEX **.H86**	Hex file: is the output file of the **OH*x*51** object hex converter in Intel HEX file format. HEX files are used as input file for PROM programmers or other utility programs.

Uvod

Šta je asembler?

- **Program koji prevodi simbolički kod (asemblerski jezik) u izvršni objektni kod.**
- **Objektni kod se može izvršiti na 8051 kompatibilnom MC**
- **Pisanje programa na mašinskom jeziku!**
- **Simbolički prepoznatljive instrukcije**
- **Tri osnovna dela asemblerskog programa**
  - Mašinske instrukcije – kod koji se izvršava
  - Asemblerske direktive – definišu strukturu programa i simbole i generišu kod koji se ne izvršava (podatke, poruke i sl.)
  - Asemblerske kontrole – upravljaju tokom asembliranja

Uvod

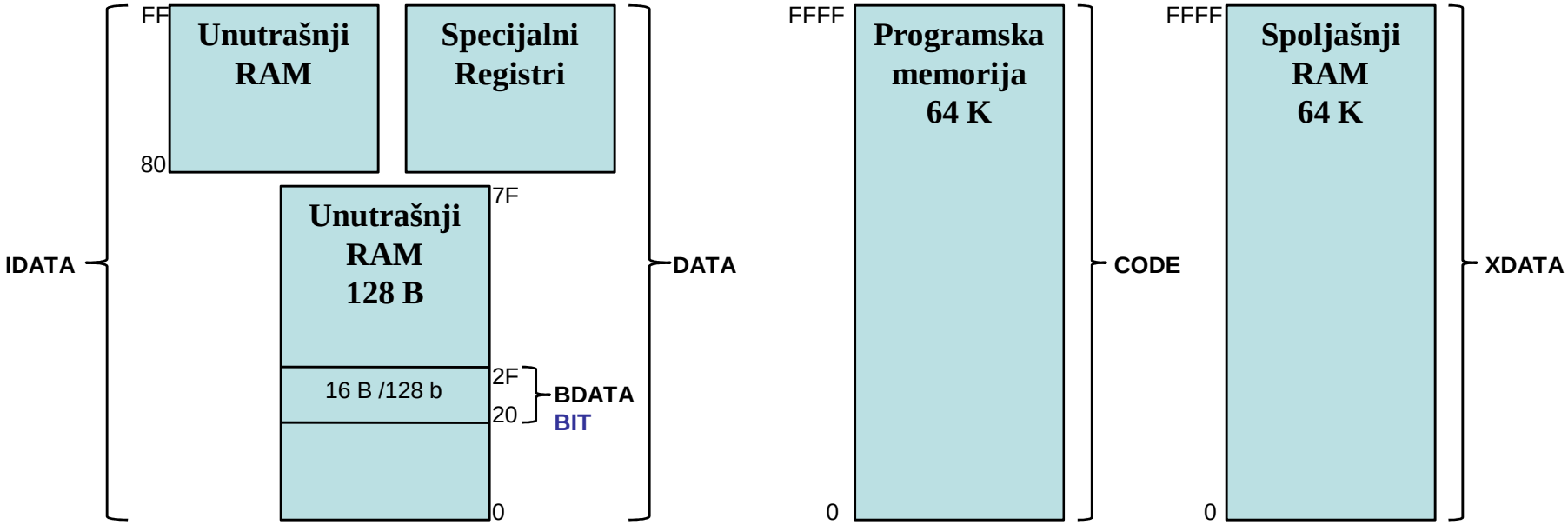
Modularno programiranje?

- **Složenost i veličina programa – pisanje programa u jednom ili više delova (modula)**
- **Pristup modularnom programiranju kao i pri projektovanju hardvera. Crna kutija sa ulazima i izlazima**
- **Pogodnosti modularnog programiranja**
  - Efikasan razvoj programa. Manji programi se lakše razumevaju, projektuju i testiraju.
  - Višestruka upotreba podprograma.
  - Jednostavnije debagiranja i izmena programa
  - **Segment** kao osnovni gradivni elemenat

Uvod

Segment

- **Blok programa ili podataka u memoriji.**
- **Apsolutni ili relokatibilni.**
- **Relokatibilni ima ime, tip i druge atribute. Segmenti sa istim imenom iz različitih modula (parcijalni segmenti) se kombinuju u jedan segment pomoću linkera.**
- **Direktna prezentacija memorijskih prostora. Bit segmenti**



Uvod

Modul

- **Sadrži više segmenta ili parcijalnih segmenata.**
  - **Parcijalni – ako postoji segment sa istim imenom u nekom drugom modulu**
- **Izvorni kod koji se može prevoditi nezavisno.**
- **Sadrže definicije svih simbola koji se koriste u modulu.**
- **Može biti napisan kao jedan ASCII tekstualni fajl ili kao više fajlova korišćenjem direktive `INCLUDE`.**
- **Elementi modula:**
  - **Asemblerske instrukcije**
  - **Direktive**
  - **Kontrole**
  - **Komentari**
  - **Simboli**
  - **Labele**

Uvod

Porces prevođenja programa sa jednim modulom

- Keil asembler **A51.EXE**. Razvojno okruženje, IDE.

  Primer poziva sa komandne linije
  ```
  A51 PROGRAM.A51
  ```

  Izlaz iz asemblera

  ```
  A51 MACRO ASSEMBLER V6.00 ASSEMBLY COMPLETE. 0
  WARNING(S), 0 ERROR(S)
  ```

Linkovanje objektnih modula

- Linker **BL51.EXE** generiše apsolutni objektni fajl i *map* fajl koji sadrži statističke informacije i odgovarajuće poruke.Sadrži više segmenta ili parcijalnih segmenata.

  ```
  BL51 PROGRAM.OBJ
  ```

  Izlaz iz linkera

  ```
  BL51 LINKER/LOCATER V4.00
  LINK/LOCATE RUN COMPLETE. 0 WARNING(S), 0 ERROR(S)
  ```

Uvod

Porces prevođenja programa sa više modula

Primer poziva sa komandne linije

```
A51 MODUL_1.A51
A51 MODUL_2.A51
A51 MODUL_3.A51
```

Ili

```
A51 MODUL_1.A51 MODUL_2.A51 MODUL_3.A51
```

Linkovanje objektnih modula

```
BL51 MODUL_1.OBJ MODUL_2.OBJ MODUL_3.OBJ
```

Direktive

- **Nasuprot instrukcijama, direktive ne generišu mašinski kod**
  - Imaju specijalni kod koji je smešten u asemblerski program i daju mu instrukcije kako da izvrši određene funkcije
  - Mogu biti korišćene za definisanje simbola, rezervaciju i inicijalizaciju memorijskog prostora i kontrolu smeštanja programskog koda

- **Asemblerske direktive su specifične za svaki asembler. Na ovom mestu biće korišćene Keil A51 Asembler**
- **Direktive su grupisane u sledeće kategorije:**
  - Kontrolu segmenta
  - Kontrolu adresa
  - Definisanje simbola
  - Inicijalizaciju i rezervaciju memorijskog prostora

Kontrola segmenta

- Pod segmentom se podrazumeva kontinualni programski blok ili blok podataka
  - Primer:
    - Definicija funkcije (u programskoj memoriji)
    - Polje podataka (u memoriji podataka)

- Dva osnovna tipa segmenta u zavisnosti od osobine relokatibilnosti
  - *Generički ili relokatibilni*
  - *Apsolutni*

- Oba tipa mogu biti specificirana kao jedna od pet memorijskih klasa
  - CODE, DATA/BDATA, IDATA, XDATA, BIT

Generički (relokatibilni) segment

- Kreira se direktivom SEGMENT
- Konačnu memorijsku lokaciju dodeljuje linker
- Format je sledeći:

```
<Symbol>        SEGMENT        <segment_memory_class>
```

- Primer:

```
PODACI      SEGMENT     DATA
```

Direktiva definiše relokatibilni segment sa imenom PODACI, memorijske klase DATA

- *Kada se jednom definiše ime segmenta, potrebno je selektovati segment direktivom* RSEG

```
RSEG   PODACI
```

Generički (relokatibilni) segment

Memory Class	Address Range	Description
DATA	D:00 – D:7F	Direct addressable on-chip RAM.
BIT	D:20 – D:2F	bit-addressable RAM; accessed bit instructions.
IDATA	I:00 – I:FF	Indirect addressable on-chip RAM; can be accessed with @R0 or @R1.
XDATA	X:0000 – X:FFFF	64 KB RAM (read/write access). Accessed with MOVX instruction.
CODE	C:0000 – C:FFFF	64 KB ROM (only read access possible). Used for executable code or constants.
BANK 0 **... BANK 31**	B0:0000 – B0:FFFF B31:0000 – B31:FFFF	Code Banks for expanding the program code space to 32 x 64KB ROM.

Apsolutni segment

- Apsolutni segment označava fiksni memorijski segment. Kreira se direktivama CSEG, DSEG i XSEG.

- Konačna lokacija segmenta poznata je u trenutku kompajliranja

- Format direktive je sledeći:

```
CSEG   AT <address>  ; definiše apsolutni kodni segment
DSEG   AT <address>  ; definiše apsolutni segment podataka
XSEG   AT <address>  ; definiše apsolutni segment podataka u
                     ; spoljašnjem ramu
```

- *Primer*:

```
CSEG   AT   0300H    ;selekcija kodnog segmenta sa
                     ;početkom na adresi 0300H
DSEG   AT   040H     ;selekcija segmenta podataka sa
                     ;početkom na adresi 040H
```

Kontrola adrese - ORG

- Format direktive ORG je:

```
ORG    <izraz>
```

- ORG direktiva se koristi da postavi vrednost lokacionog brojača u tekućem segmentu na ofset specificiran izrazom

- Direktiva ne menja adresu početka segmenta. Adresa segmenta se može promeniti korišćenjem standardnne segment direktive

- **Primer:**

```
ORG    80H    ;Postavi lokacioni brojac na 80H
```

- Koristi se u svim segmentima.

- Na primer, za rezervisanje po jednog bajta memorije podataka za promenljive SEKUNDE i MINUTI možemo pisati:

```
                DSEG          ;data segment
                ORG 30H
SEKUNDE:        DS    1
MINUTI: DS      1
```

Kontrola adrese - END

- Format direktive je:

  END

- END direktiva označava kraj izvornog fajla

- Informiše asembler da zafrši sa asembliranjem programa.
  - Saki tekst koji se pojavi iza direktive END se ignoriše od strane asemblera

- Neophodna je u svakom izvornom fajlu
  - Ukoliko se ne nađe na kraju programa, asembler prijavljuje grešku

Definisanje simbola

- Direktiva definisanja simbola dodeljuje simboličko ime nekom iskazu ili registru
  - Iskaz može biti konstanta, referenca na adresu ili drugo simboličko ime
- Značaj primene simboličkih imena u čitljivosti programa
- Izmena programa jednostavnija – dovoljno promeniti samo u iskazu direktive
  - Ostali iskazi koji se referenciraju na simbol automatski se podešavaju

Definisanje simbola – EQU, SET

- Format ovih direktiva je sledeći:

  Symbol EQU <*expression*>
  Symbol EQU <*register*>
  Symbol SET <*expression*>
  Symbol SET <*register*>

- Slično makro definiciji #define u C jeziku

- Iskaz može da sadrži jednostavne matematičke operatore
  ‘+’, ’-‘, ‘ * ‘, ‘/’, MOD

- Registri koji se mogu koristit
  A, R0, R1, R2, R3, R4, R5, R6 i R7

Definisanje simbola – EQU, SET

Binarni operatori

Operator	Syntax	Description
NOT	NOT *expression*	Bit–wise complement
SHR	*expression* SHR *count*	Shift right
SHL	*expression* SHL *count*	Shift left
AND	*expression* AND *expression*	Bit–wise AND
OR	*expression* OR *expression*	Bit–wise OR
XOR	*expression* XOR *expression*	Bit–wise exclusive OR

Relacioni operatori

Operator	Syntax	Result
GTE	*expression1* GTE *expression2*	True if *expression1* is greater than or equal to *expression2*
LTE	*expression1* LTE *expression2*	True if *expression1* is less than or equal to *expression2*
NE	*expression1* NE *expression2*	True if *expression1* is not equal to *expression2*
EQ	*expression1* EQ *expression2*	True if *expression1* is equal to *expression2*
LT	*expression1* LT *expression2*	True if *expression1* is less than *expression2*
GT	*expression1* GT *expression2*	True if *expression1* is greater than *expression2*
>=	*expression1* >= *expression2*	True if *expression1* is greater than or equal to *expression2*
<=	*expression1* <= *expression2*	True if *expression1* is less than or equal to *expression2*
<>	*expression1* <> *expression2*	True if *expression1* is not equal to *expression2*
=	*expression1* = *expression2*	True if *expression1* is equal to *expression2*
<	*expression1* < *expression2*	True if *expression1* is less than *expression2*
>	*expression1* > *expression2*	True if *expression1* is greater than *expression2*

Definisanje simbola – EQU, SET

- *Primer*:

```
COUNT     EQU   R3    ;simb. ime registra
TOTAL     EQU   200   ;simb. ime constante
AVERG     SET   TOTAL/5
TABLE     EQU   10
VALUE     SET   TABLE*TABLE
```

Definisanje simbola – EQU, SET

- Svaka od ovih direktiva dodeljuje vrednost adrese simbolu. Format direktive je:

Symbol	BIT	<*bit_address*>
Symbol	CODE	<*code_address*>
Symbol	DATA	<*data_address*>
Symbol	IDATA	<*idata_address*>
Symbol	XDATA	<*xdata_address*>

bit_address	Adresa bit lokacije koja je na raspolaganju od lokacije 00H do 7FH kao ofset od bajt lokacije 20H
code_address	Opseg adresa od 0000H do 0FFFFH
data_address	Adrese od 00H to 7FH (unutrašnja memorija) i specijalni registri od 80H do 0FFH
idata_address	Adrese od 00H do 0FFH
xdata_address	Spoljašnji ram prostor od 0000H do 0FFFFH

Definisanje simbola – EQU, SET

- *Primer*:

```
MARKER   BIT   2EH   ;Lokacija 2EH kao bit
                     ;MARKER
Port2    DATA  A0H   ;Simbolicko ime portu P2
```

Definisanje simbola – EQU, SET

Specijalni asemblerski simboli

Register	Description
A	Represents the 8051 Accumulator. It is used with many operations including multiplication and division, moving data to and from external memory, Boolean operations, etc.
DPTR	The DPTR register is a 16-bit data pointer used to address data in XDATA or CODE memory.
PC	The PC register is the 16-bit program counter. It contains the address of the next instruction to be executed.
C	The Carry flag; indicates the status of operations that generate a carry bit. It is also used by operations that require a borrow bit.
AB	The A and B register pair used in MUL and DIV instructions.
R0 – R7	The eight 8-bit general purpose 8051 registers in the currently active register bank. A Maximum of four register banks are available.
AR0 – AR7	Represent the absolute data addresses of R0 through R7 in the current register bank. The absolute address for these registers will change depending on the register bank that is currently selected. These symbols are only available when the USING directive is given. Refer to the USING directive for more information on selecting the register bank. These representations are suppressed by the NOAREGS directive. Refer to the NOAREGS directive for more information.

Inicijalizacija/rezervacija memorije

- Direktive za inicijalizaciju i rezervisanje:
  DB – bajt, **DW – reč** i **DD – dupla reč**

- Ove direktive inicijalizuju i rezervišu memorijski prostor u formi bajta, reči ili duple reči u kodnom prostoru

- Direktive za rezervisanje memorije bez inicijalizacije:
  DS i **DBIT**

  Ove direktive rezervišu specificirani broj bajtova ili bitova u tekućem segmenu

DB (*Define Byte*)

- Inicijalizacija kodne memorije bajt vrednostima

- Format:

  <*label*>: DB <*expression*>, <*expression*>, …

  label

  Simboličko ime adrese gde se smeštaju bajt vrednosti

  expression

  vrednost bajta, može biti string karaktera, simbol, ili 8-bitna konstanta

DB (*Define Byte*)

- *Primer*:

```
        CSEG AT 200H
MSG:    DB ‘Please enter your password’, 0
ARRAY:  DB 10H,20H,30H,40H,50H
```

- Prvi string karaktera smešta se kao ASCII string počev od lokacije 200H u programskoj memoriji ([200H]=50H, [201H]=6CH i t.d.)

- DB direktiva može biti deklarisana samo u kodnom segmentu
  - Ako se definiše u drugom segmentu, asembler generiše grešku

DBIT (*Define Bit*)

- Rezerviše prostor u bit adresabilnom delu RAM memorije.

- *Primer*:

```
            BSEG AT   20H
MARKERI     DBIT 32   ;rezervisanje 32 bita
```

- DBIT direktiva može biti deklarisana samo u BSEG segmentu koji je aktivan
  - Ako se definiše u drugom segmentu, asembler generiše grešku

DW (*Define Word*)

- DW direktiva inicijalizuje kodnu memoriju 16-bitnim rečima
- Ima format:

  <*label*>: DW <*expression*>, <*expression*>, …
- *Primer*:

```
;alociranje 2 reci
CNTVAL:  DW   1025H, 2340H
;10 vrednosti 1234H pocev od lokacije XLOC
XLOC:    DW   10 DUP (1234H)
```

- DUP operator se koristi za dupliranje sekvence
- DW se koristi samo u kodnom segmentu
  - Ako se definiše u drugom segmentu, asembler generiše grešku

DD (*Define Double Word*)

- DD direktiva inicijlizira kodnu memoriju duplim rečima ili 32-bitnim vrednostima

- Direktiva ima format:

  <*label*>: DD <*expression*>, <*expression*>, …

- *Primer*:

```
ADDR:DD   820056EFH, 10203040H
EMPT:DD   3 DUP ( 0 )
```

- Kao i DB i DW direktive, DD može biti specificirana samo u kodnom segmentu

DS (*Define Storage*)

- DS direktiva rezerviše specificirani broj bajtova u tekućem segmentu

- Može biti korišćena u tekućem aktivnom segmentu

  CSEG, ISEG, DSEG or XSEG

- Ima format:

  <label>: **DS** ***<expression>***

- Iskaz nesme sadržati “forward” reference (što još uvek nije definisano), relokatibilne simbole ili eksterne simbole

DS (*Define Storage*)

- *Primer*:

```
XSEG  AT 1000H     ;selekcija memorijskog bloka
                   ; u spoljnoj memoriji pocev od
                   ; adrese 1000H
Ulaz:        DS    16     ; rezervisi 16 bajtova
TipSignala:  DS    1      ; reservisi 1 bajt
```

- Lokacioni brojač tekućeg segmenta se inkrementira odgovarajući broj puta kad god se u programu nađe nardedba DS

- Programe treba da vodi računa da počev od lokacije Ulaz ne upisuje više od 16 bajtova

- Napomena da bajtovi nisu inicijalizovani, već je samo rezervisan prostor

USING direktiva

- Ova direktiva definiše koja će registarska banka (registri R0 do R7) biti korišćena u programu.

- Direktiva ne vrši izmenu sadržaja PSW registra, već se to mora definisati programski

- *Primer*:

```
USING 0      ;Izbor RB 0
PUSH AR0     ;smestanje u stek R0 iz RB0
PUSH AR1     ;smestanje u stek R1 iz RB0

USING 1
PUSH AR0     ;smestanje u stek R0 iz RB1
PUSH AR1     ;smestanje u stek R1 iz RB1
```

- **AR0, AR1, AR2, AR3, AR4, AR5, AR6 i AR7 direktne adrese registara R0 do R7 aktivne registarske banke**

Direktive uslovnog asembliranja

- Ove direktive se koriste da definišu takozvane uslovne blokove programa. Blok počine IF direktivom i završava direktivom ENDIF, ELSEIF ili ELSE. Iskaz ili simbol (u zagradi) koji sledi iza direktive IF je uslov pod koji određuje da li se deo programa biti preveden u mašinski kod.
  - Ako je iskaz istinit ili simbol jednak jedinice, vrši se prevođenje programskog bloka sve do ELSE, ELSEIF ili ENDIF direktive.
  - Ako je iskaz neistinit ili simbol jednak nule ceo blok se zanemaruje

- *Primer*:

```
IF (VERZIJA = 1)
        LCALL program1
ELSE
        LCALL program2
ENDIF
```

Kontrolne direktive

- Ove direktive se prepoznaju po simbolu $ koji se nalazi na početku. Predstavljaju komande koje definišu koje se sve fajlovi koriste u toku kompajliranja programa, gde će biti smešteni izvršni fajlovi kao konačni izgled kompajliranog programa

- $INCLUDE

  Govori asembleru da koristi podatke smeštene u drugom fajlu

- *Primer*:

```
$INCLUDE (TABELA.ASM)
```

Kontrolni iskazi (*statement*)

- EXTRN
  Može se pojaviti bilo gde u programu, definiše simbol koji tekuće fajl koristi ali je definisan u drugom objektnom modulu

- *Primer*:

```
EXTRN VELICINA (TABSIZE)
```

- PUBLIC
  Definiše simbole koji se mogu koristiti u drugim modulima

- *Primer*:

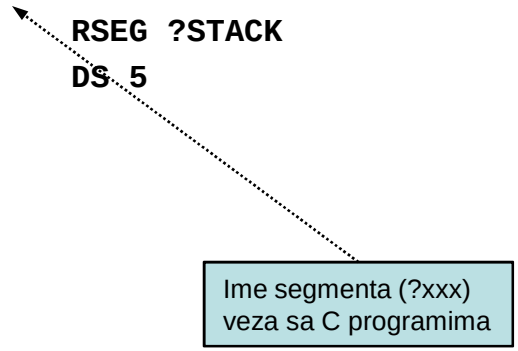
```
PUBLIC TABSIZE
```

Primer programskog šablona – *template*

```
;------------------------------------------------------------------------------
; Eksportovanje simbola koji su definisani u ovom modulu
;------------------------------------------------------------------------------
PUBLIC data_variable
PUBLIC code_entry
PUBLIC typeless_number
PUBLIC xdata_variable
PUBLIC bit_variable

;------------------------------------------------------------------------------
; Mogucnost ukljucivanja vise simbola u PUBLIC iskaz
;------------------------------------------------------------------------------
PUBLIC data_variable1, code_table, typeless_num1, xdata_variable1

;------------------------------------------------------------------------------
; Definisanje STACK segmenta u glavnom modulu
;------------------------------------------------------------------------------
?STACK SEGMENT IDATA          ; ?STACK se smesta u IDATA RAM
    RSEG ?STACK               ; aktiviranje ?STACK segmenta
    DS 5                      ; rezervisanje prostora za stek, 5 bajtova
```



Primer programskog šablona – *template*

```
; Deklaracija segmenata i promenljivih.
;-------------------------------------------------------------------------
; DATA SEGMENT–Rezervisanje prostora u DATA RAM–Obrisati ako se ne koristi
;-------------------------------------------------------------------------
Podaci    SEGMENT DATA        ; segment za DATA RAM.
          RSEG Podaci         ; aktiviraj segment
var1:     DS 1                ; rezervacija 1 bajta za var1
var2:     DS 2                ; rezervacija 2 bajta za var2
;-------------------------------------------------------------------------
; XDATA SEGMENT--Rezervisanje prostora u XDATA RAM–Obrisati ako se ne koristi
;-------------------------------------------------------------------------
Extram    SEGMENT XDATA       ; segment za XDATA RAM
          RSEG EXTRAM         ; aktiviraj segment
xvar1:    DS 1                ; rezervisi 1 bajt za xvar1
xvar2:    DS 500              ; rezervisi 500 bajtova za xvar2
;-------------------------------------------------------------------------
; INPAGE XDATA SEGMENT–Rezervacija prostora u XDATA RAM stranici (page size: 256 Bytes)
; INPAGE segmenti korisni kada se adresiranje vrsi sa @R0 ili @R1
;-------------------------------------------------------------------------
Page1     SEGMENT XDATA INPAGE ; segment koji je “inpage”
          RSEG Page1           ; aktiviraj segment
xvar1:    DS 1                 ; rezervisi 1 bajt za xvar1
xvar2:    DS 200               ; rezervisi 200 bajtova za xvar2
```

Sve u okviru jedne stranice

Primer programskog šablona – *template*

```
;-------------------------------------------------------------------------
; ABSOLUTE XDATA SEGMENT–Rezervacija prostora u XDATA RAM na apsolutnim adresama
; Korisno za memorijski mapirani U/I. Obrisati ako se ne koristi.
;-------------------------------------------------------------------------
          XSEG AT 8000H       ; kreiraj i ukljuci apsolutni segment na adresi 8000H
Izlaz:    DS 1                ; rezervisi 1 bajt za port Izlaz
Ulaz:     DS 1                ; rezervisi 1 bajt za port Ulaz
;-------------------------------------------------------------------------
; BIT SEGMENT–Rezervacija prostora u bit adrsabilnom ramu–Obrisati ako se ne koristi
;-------------------------------------------------------------------------
Bitvar    SEGMENT BIT         ; segment for BIT RAM.
          RSEG Bitvar         ; switch to this bit segment
bitvar1:  DBIT 1              ; rezervisi 1 bit za promenljivu bitvar1
bitvar2:  DBIT 4              ; rezervisi 4 bita za promenljivu bitvar2
;-------------------------------------------------------------------------
; Definisanje konstanti (typeless)
;-------------------------------------------------------------------------
CR        EQU 0DH             ; dodeli simbolu CR vrednost 0D hex
LF        EQU 0AH             ; dodeli simbolu LF vrednost 0A hex
Baud      EQU 9600
Xtal      EQU 12000
```

Primer programskog šablona – *template*

```
;-------------------------------------------------------------------------------
; Na reset adresi LJMP na start u glavnom programu
;-------------------------------------------------------------------------------
          CSEG AT 0           ; absolutni segment na adresi 0
          LJMP start          ; reset lokcija (skok na pocetak programa)
;-------------------------------------------------------------------------------
; CODE SEGMENT—Rezervacija prostora za program
;-------------------------------------------------------------------------------
Program   SEGMENT CODE
          RSEG Program        ; aktiviraj segment
          USING 0             ; oznaci aktivnu RB
start:    MOV SP,#?STACK-1    ; dodeli korektnu vrednost stek pointeru
;-------------------------------------------------------------------------------
; Ovde smestiti programski kod. (Navedeni program nije funkcionalan)
;-------------------------------------------------------------------------------
          ORL IE,#82H         ; enable interrupt system (timer 0)
          SETB TR0            ; enable timer 0
label:    MOV A,data_symbol
          ADD A,#typeless_symbol
          CALL Procedura1
          MOV DPTR,#xdata_symbol
          MOVX A,@DPTR
          MOV R1,A
          PUSH AR1
          CALL Procedura2
          POP AR1
          ADD A,R1
          JMP label

Proc1:    RET
Proc2:    RET
```

Primer programskog šablona – *template*

```
;-------------------------------------------------------------------------
; Prekidne rutine
;-------------------------------------------------------------------------
          CSEG AT 0BH ; 0BH is address for Timer 0 interrupt
          LJMP Tmr0int
;-------------------------------------------------------------------------
; Dati svakom prekidu sopstveni kodni segment
;-------------------------------------------------------------------------
Int0_seg  SEGMENT CODE        ; segment prekidne rutine
          RSEG int0_seg       ; aktiviranje segmenta
          USING 1             ; registarska banka za prekidnu rutinu
Tmr0int:  PUSH PSW
          MOV PSW,#08H        ; aktiviranje registarske banke 1
          PUSH ACC
          MOV R1,data_variable
          MOV DPTR,#xdata_variable
          MOVX A,@DPTR
          ADD A,R1
          MOV data_variable1,A
          CLR A
          ADD A,#0
          MOV data_variable1+1,A
          POP ACC
          POP PSW
          RETI
;-------------------------------------------------------------------------
; END directiva je uvek obavezna
;-------------------------------------------------------------------------
          END                 ; Kraj fajla
```