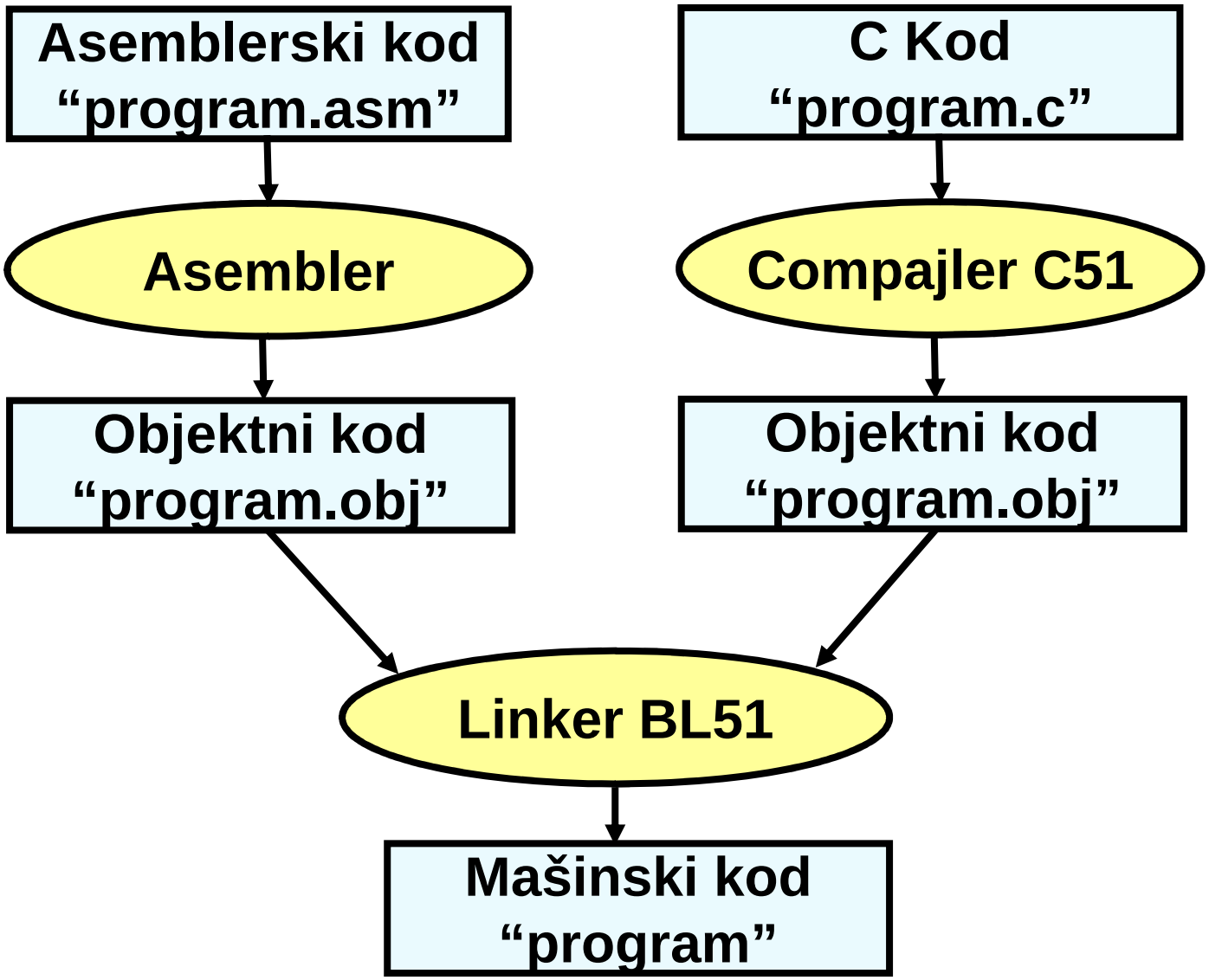
Mikrokontroleri

Mikrokontroleri

Poglavlje VI

Programiranje na C-jeziku

Tok izrade programa



Uvod

Spada u grupu jezika višeg nivoa, blizak asembleru.

- **Pisanje velikih i kompleksnih programa na asembleru može biti naporno, oduzima puno vremena, lako se prave greške.**
- **Pisanje programa na C jeziku je lakše.**
- **Kod generisan iz C programa izvršava se sporije i zauzima više memorije nego asemblerski.**
- **Asemblerski kod može biti kompaktniji i efikasniji**
- **C jezik nudi bolju strukturnost programa, ima bogatiji skup operacija i lakše je programiranje.**
- **Kad brzina izvršavanja nije kritična, treba se opredeliti za pisanje programa na C jeziku.**

Uvod

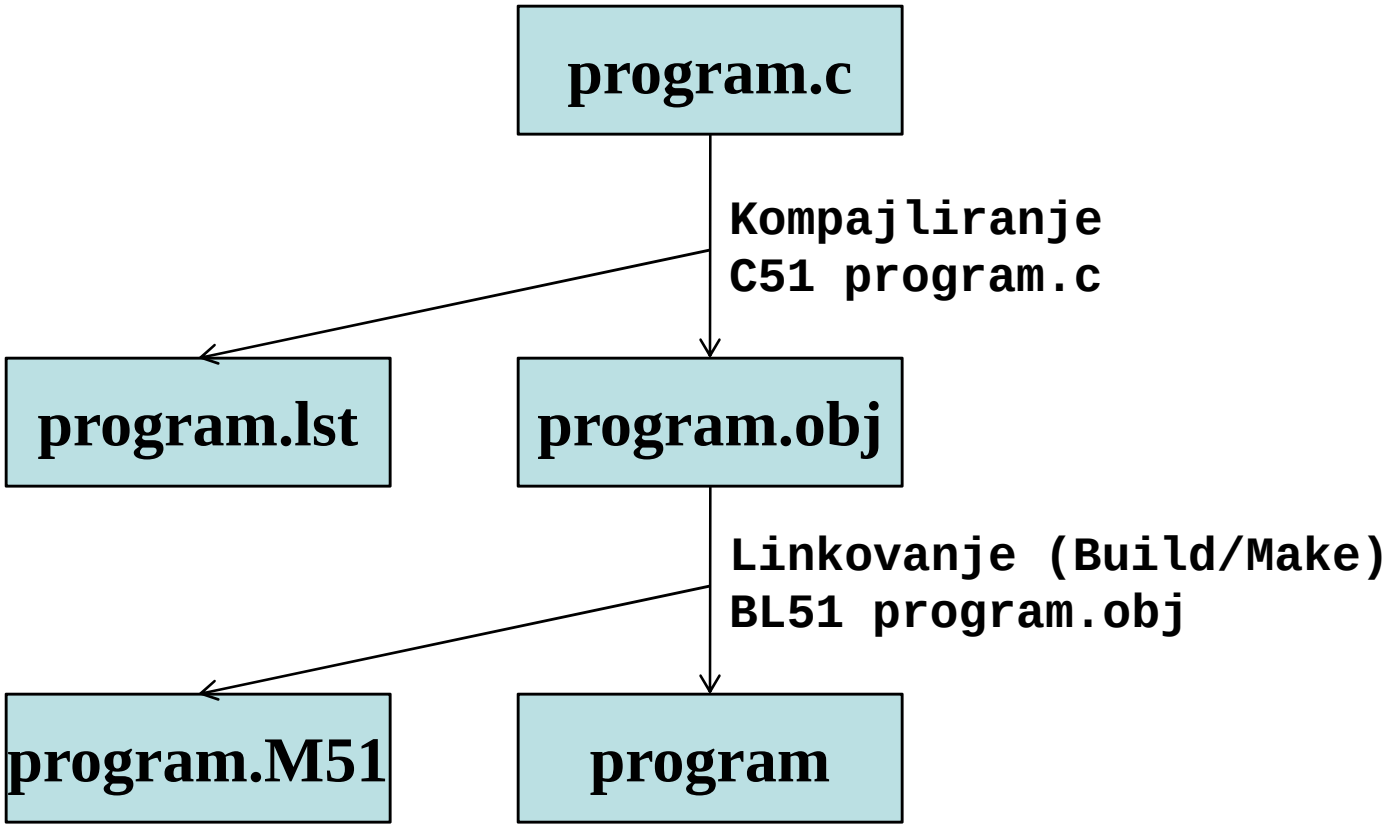
Dostupni kompajleri

- **Keil – integrisan sa razvojnim okruženjem, IDE Keil uVision, komercijalni softver.**

- **Reads51**
  http://www.rigelcorp.com/reads51.htm

- **SDCC – Small Device C Compiler**
  http://sdcc.sourceforge.net/

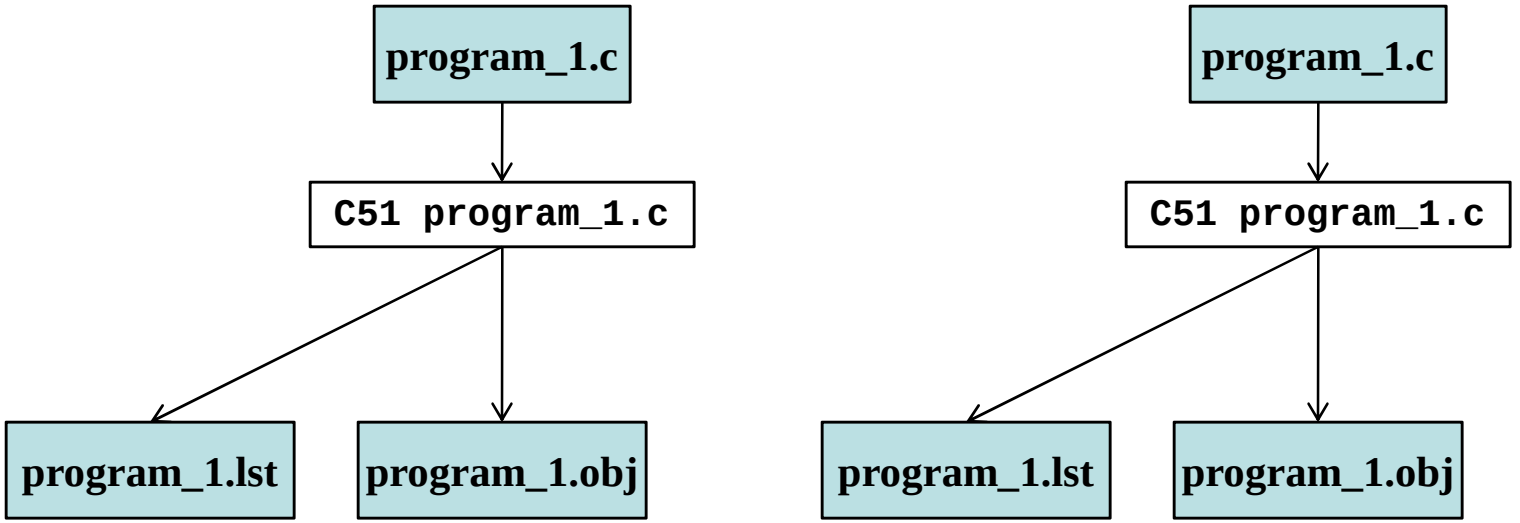
Koraci u izradi progrma



Modularno programiranje

- **Omobućava modularno programiranje**
- **Nije objektno orjentisani jezik**
- **Svaki zadatak može biti zatvoren u obliku funkcije**
- **Ceo program mora biti zatvoren u funkciju “main”**

Koraci u izradi programa sa više modula



Osnovna struktura C programa

1. Direktive i *include* fajlovi
2. Deklaracija globalnih promenljivih i konstanti
3. Deklaracija funkcija
4. “Main” funkcija
5. Sub-funkcije
6. Prekidne rutine (ISR- *Interrupt Service Routines*)

Osnove C jezika

- **Svaki C program se sastoji nod:**
  - Promenljive *Variables*
  - Funkcije (jedna mora biti “main”)
    - Iskazi (*Statements*)

- **Definisanje SFR-a kao promenljivih:**

#include <8052.h>

Promenljive

- Sve promenljive moraju biti deklarisane na početku programa, pre prvog iskaza

- Deklaracija sadrži tip (*type*) i listu promenljivih

Primer:

```
void main (void) {
int var1, var2;
...
}
```

Tipovi promenljivih

ANSI C

8051 Compiler Specific

Data Type	Bits	Bytes	Value Range
signed char	8	1	-128 to +127
unsigned char	8	1	0 to 255
enum	8/16	1 or 2	-128 to +127 or -32768 to +32767
signed short	16	2	-32768 to +32767
unsigned short	16	2	0 to 65535
signed int	16	2	-32768 to +32767
unsigned int	16	2	0 to 65535
signed long	32	4	-2147483648 to 2147483647
unsigned long	32	4	0 to 4294967295
float	32	4	±1.175494E-38 to ±3.402823E+38
pointer	8-24	1-3	Adresa promenljive
bit	1	-	0 to 1
sbit	1	-	0 to 1
sfr	8	1	0 to 255
sfr16	16	2	0 to 65535

Kod nekih kompajlera 4 bajta

Memorijski model

- Određuje podrazumevani tip memorije koja se koristi za argumente funkcije, automatske promenljive, i deklaraciju bez eksplicitne memorijske specifikacije
  SMALL, COMPACT i LARGE

- SMALL model smešta promenljive u DATA prostor
  COMPACT model smešta promenljive u IDATA prostor
  LARGE model smešta promenljive u XDATA prostor

- Specificiranje memorijskog modela u comandnoj liniji
  C51 program.c SMALL
  Ili u izvornom fajlu
  #pragma SMALL

Memorijski model

- Eksplicitno određivanje tipa memorije za podatak **data, idata** i **bdata**

Primer:

```
unsigned char data name;
int idata count;
int bdata status;
```

Kako se prevode iskazi:

name = ‘A’;
count = 25;
status = 0x2501;

Bit adresabilni podaci

- Bit adresabilni podaci se smeštaju u bit adresabilni memorijski prostor adrese 0x20 do 0x2F
- Deklaracija sa **bdata, bit** ili **sbit**

```
int bdata X;          // 16-bit bit-addressable variable X
bit flag;             // bit-valued variable flag

sbit X7_Flag = X^7;  // bit 7 of X (bit variable)
sbit Red_LED = P0^1; // bit 1 of Port P0 (bit-addressable SFR)

int bdata status;
bit s2 = status^5;
```

Aritmetičke operacije

- Osnovne aritmetičke operacije

Operator	Opis
+	Sabiranje
–	Oduzimanje
*	Množenje
/	Deljenje
%	Deljenje po modulu
–	Negacija (unary minus)

- **Primer:**

```
unsigned int count = 0x0F;
TMR2RL = -count;    // TMR2RL = 0xFFFF-0x0F+1 = 0xFFF1
```

Relacioni operatori

- Upoređuju podatke i kao izlaz daju:
  “True” ili “False”

- **if** iskazi, **for** petlje i **while** petlje često koristi relacione operatore

Operator	Description
==	Equal to
!=	Not Equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Logički operatori

- Logike operacije nad binarnim podacima (jednobitne veličine)

Operator	Description
&&	Logical AND
||	Logical OR
!	Logical NOT

“Bitwise” operatori

Operator	Description
&	Bitwise AND
|	Bitwise OR
~	Bitwise NOT (1’s Compliment)
^	Bitwise Exclusive OR
<<	Shift Left
>>	Shift Right

- Logičke operacije bit po bit
- Primer:

```
Result = Var1 & Var2;
```

  - Ako je **`Var1`** = 00100100b i **`Var2`** = 10100000b, tada je
  - **`Value1 & Value2`** :

    00100100b & 10100000b = 00100000b

Pointeri

- Jedna od najvećih snaga C jezika, ali može biti i najveća slabost.

- Nepravilna upotreba (zloupotreba) često daje C jezku epitet “opasnog” jezika.

Pointeri u asembleru

- Pointer na asembleru odgovara indirektnom adresiranju.

```
MOV R0,#40      ;Put on-chip address to be indirectly
MOV A,@RO       ;addressed in R0 (ili R1)

MOV R0,#40      ;Adresa podatka u spoljašnjoj memoriji
MOVX A,@RO      ;kao indirektna aresa u R0 (ili R1)

MOV DPTR,#1234 ;Adresa podatka u spoljašnjoj memoriji
MOVX A,@DPTR   ;kao indirektna aresa u DPTR

CLR A
MOV DPTR,#0040 ;Adresa podatka u programskoj memoriji indirectly
MOVC A,@A+DPTR ;kao indeksirana adresa u DPTR i A
```

Pointeri u C51

- C pointer čuva adresu promenljive ili konstante
- Deklaracija


```
unsigned char *pointer ;
```

Prefiks u obliku zvezdice (asterisk) označava da se radi o adresi a ne o znaku za množenje

```
/* 1 – Defininisanje promenljive koj ce cuvati adresu */
unsigned char *pointer ;
/* 2 – Upis adrese promenljive u pointer */
pointer = &c_variable ;
/* 3 – Upis vrednosti ‘0xff’ u indirektno adresiranu promenljivu */
*pointer = 0xff ;
```

Pointeri u C51

- Memorijski specifični pointeri koji ukazuju na **data**, **idata** i **pdata** imaju veličinu jedan bajt.

```
unsigned char data *pointer
```

- Memorijski specifični pointeri koji ukazuju na **xdata** i **code** imaju veličinu dva bajta.

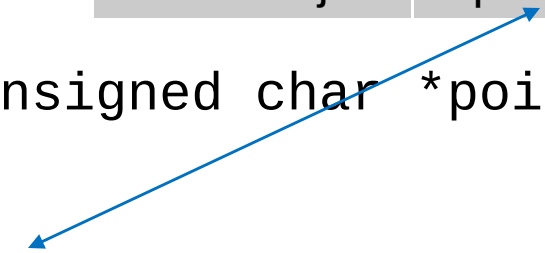
```
unsigned char xdata *pointer
```

Pointeri u C51

- Generički pointeri nemaju eksplicitno definisani tip memorije na koji ukazuju. U memoriji zauzimaju tri bajta

adresa	+0	+1	+2
sadržaj	Tip memorije	Viš deo adr.	Niži deo adr.

```
unsigned char *pointer
```



Tip memorije	data / idata / bdata	xdata	pdata	code
sadržaj	0x00	0x01	0xFE	0xFF

Primeri korišćenja pointera

```
/* Primer 1 – korišćenje pointera */
unsigned char c_variable ;    // Declaracija promenljive
unsigned char *ptr ;          // Declaracija pointera

main() {
c_variable = 0xff ; // Direktno postavljanje vrednosti promenljive
ptr = &c_variable ; // Definisanje vrednosti pointera
*ptr = 0xff ;       // Indirektni upis 0xff u promeljivu
}

/* Primer 2 – korišćenje pointera */
unsigned char c_variable ;        // Declaracija promenljive
unsigned char *ptr = &c_variable ;// Declaracija i inicijalicija
                                  // pointera

main() {
c_variable = 0xff ; // Direktno postavljanje vrednosti promenljive
*ptr = 0xff ;       // Indirektni upis 0xff u promeljivu
}
```

Primeri korišćenja pointera

Pointer sa zvezdicom se moze koristiti kao obican podatak

Iskaz:

```
x = y + 3 ;
```

Može se izvršiti i pomoću pointera

```
char x, y ;
char *x_ptr = &x ;
char *y_ptr = &y ;
*x_ptr = *y_ptr + 3 ;
```

ili

```
x = y * 25 ;
*x_ptr = *y_ptr * 25 ;
```

Primeri korišćenja pointera

Važno za razumevanje pointera:

```
*ptr = var ;
```

Znači da se u lokaciju na koju ukazuje `ptr` upisuje vrednost promenljive `var`

```
ptr = &var ;
```

Znači da se pointeru dodeljuje vrednost adrese na kojoj se nalayi promenljiva `var`

Pointeri na apsolutne adrese

- **U *embedded* sistemima ROM, RAM i periferije su na fiksnim adresama**
- **Kako definisati pointer na apsolutnu adresu, nasuprot adresi promenljive koja je najčešće nepoznata (i nebitna)?**

Definisanje pointera koji ukazuje na unapred poznatu adresu:

```
char *abs_ptr = 0x8000 ;
```

Ako je adresa poznata u izvršavanja programa (*run time*)

```
char *abs_ptr ;
abs_ptr = (char *) 0x8000 ;  // inicijalizacija pointera
*abs_ptr = 0xFF ;            // upis 0xFF u lokaciju 0x8000
*abs_ptr++ ;                 // inkrementiranje pointera
```