Napredni C kurs

Namenski računarski sistemi
Čas 1, 2021/2022

Struktura C programa

- Konstante i tipovi podataka
- Promenljive (lokalne, globalne, eksterne...)
- Izrazi (if, while, for...)
- Operatori (+, -, *, /,~...)
- Funkcije sa argumentima
  - Prenos po vrednosti
  - Prenos po adresi
  - Prenos po referenci (C#)
- Makro izrazi
- Preprocesorske direktive
- Biblioteke
- Komentari

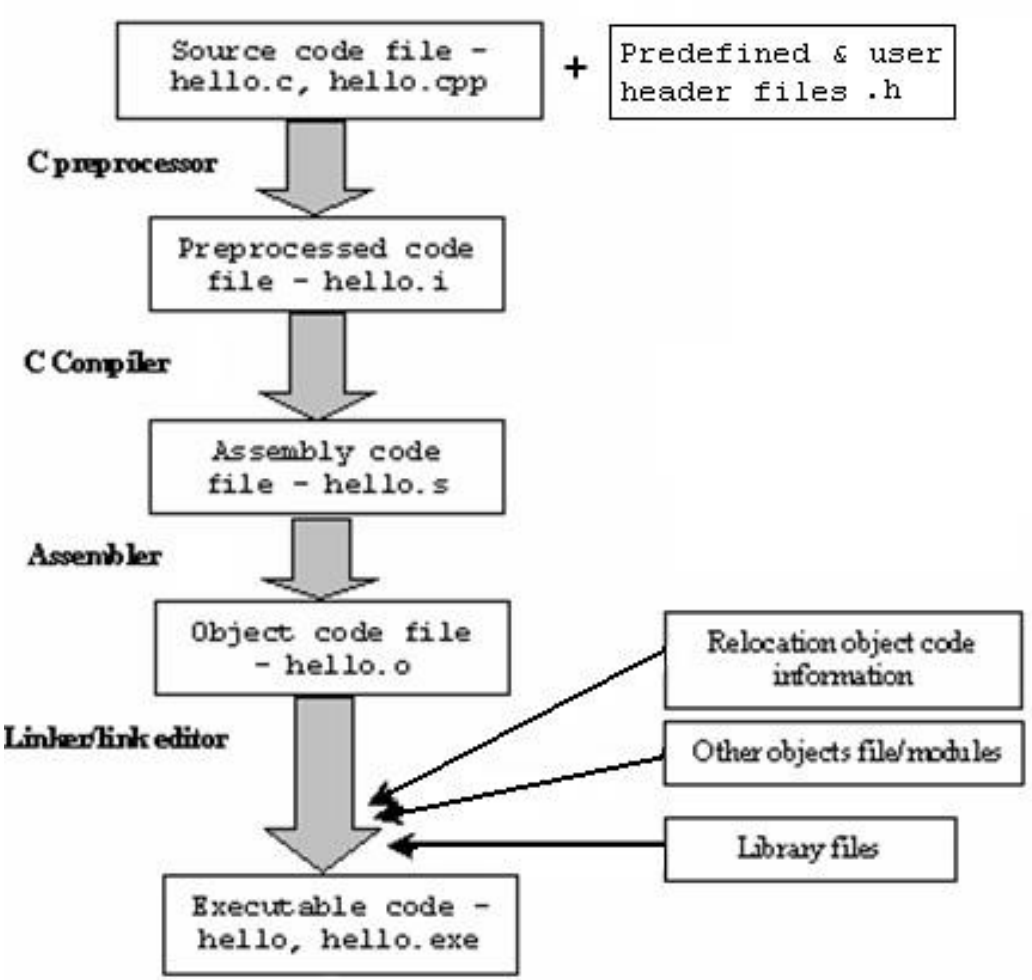
```
#include <stdio.h>
#include "mydef.h"

#define ....          /* lokalne definicije */
int gvInt;            // globalna promenljiva
extern ...            /* referenca na gvar */
int func1(...);       /* redeklaracija */

void main( void )
{
    int i;            /* lokalna promenljiva */
    izraz 1;          /* komentar */
    func1(i, &i);     /* poziv funkcije */
    izraz 2;
    .....
}
int func( int k, int *pk )
{
    izraz3;           /* telo funkcije */
    return 0;
}
```

Prevođenje C programa

- file_name.c
  - C izvorni kod
- file_name.h
  - C header file
- file_name.i
  - Preprocesirani kod
- file_name.s, .asm
  - Asemblerski kod
  - Asemblerski kod koji se mora preprocesirati
- file_name.o, .obj
  - Objektni kod
- file_name.exe
  - Izvršni kod



Faze prevođenja C programa

- Izvorni kod -> Asemblerski kod -> Objektni kod -> Izvršni kod

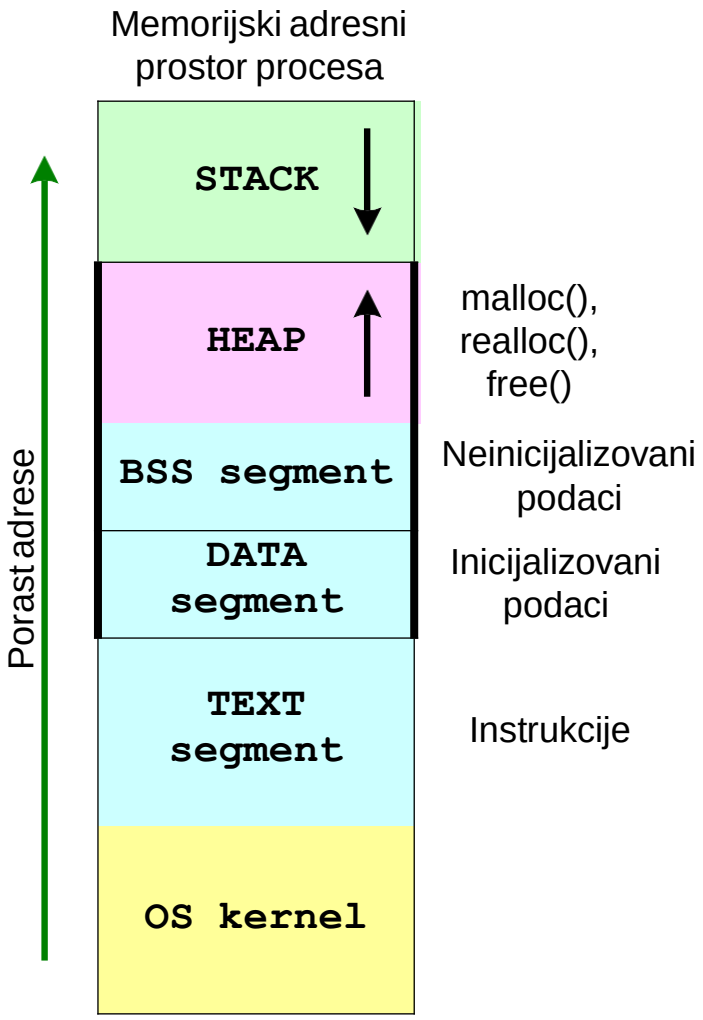
- Preprocesor – prvi prolaz prevođenja C koda
- Kompajler – drugi prolaz, generiše asemblerski kod
- Asembler – generiše objektni kod i asemblerski listing
- Linker – final faza gde se
  - Više .obj i .lib modula kombinuju u jedan izvršni (.exe)
  - Rešavaju reference na spoljne simbole
  - Vrši konačno dodeljivanje adresa funkcijama/promenljivim (relokacija)
- U IDE kompajlerima ove procedure su integrisane

Programski model

- U krajnjoj posledici programski model je
  - Organizacija promenljivih i funkcija, i načina međusobnog povezivanja, u okviru izvršnog programa
  - Iako zavisan od HW platforme, sličan u većini implementacija
- Postaje značajan u krajnjim fazama C prevođenja, zato pod programskim modelom podrazumevamo
  - Konvencije generisanja asemblerskog koda
    - Memorijska alokacija promenljivih
    - Registarske konvencije i korišćenje stack-a
  - Garancija međusobne kompatibilnosti
    - Povezivanje programa različitih kompajlera
    - Aplikativnih programa i operativnih sistema

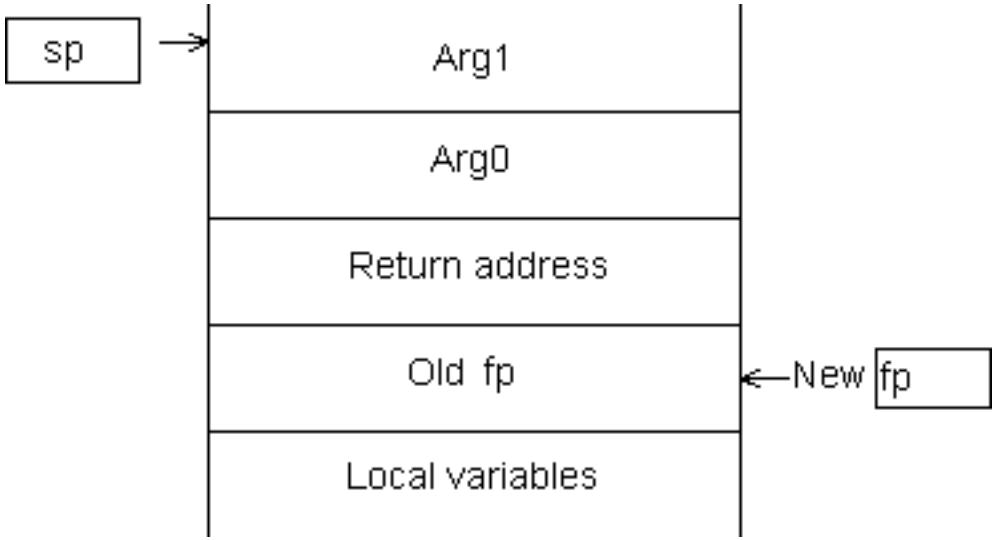
Memorijski segmenti programa

- Svaki program dobija svoj prostor pri punjenju
- TEXT sekcija može biti deljena
  - reentrant code – programs
- DATA – non-zero init global i static
  - .rdata – read only (const)
- BSS (Block Started by Symbol) – u izvršavanju DATA, u .exe ga nema
- HEAP – dinamička memorija
  - Pristup samo poreko pokazivača
  - Kontrola opsega (kompajler, run-time)
- STACK – lokalne promenljive, poziv funkcija i prenos argumenata
  - Stack frame, stacksize



Korišćenje stack-a pri pozivu funkcija

- Stack frame – osnovna struktura
  - Formira se i briše pre/po svakom pozivu funkcije na tekućoj lokaciji stack-a
  - Sadrži argumente, povratnu adresu i lokalne promenljive



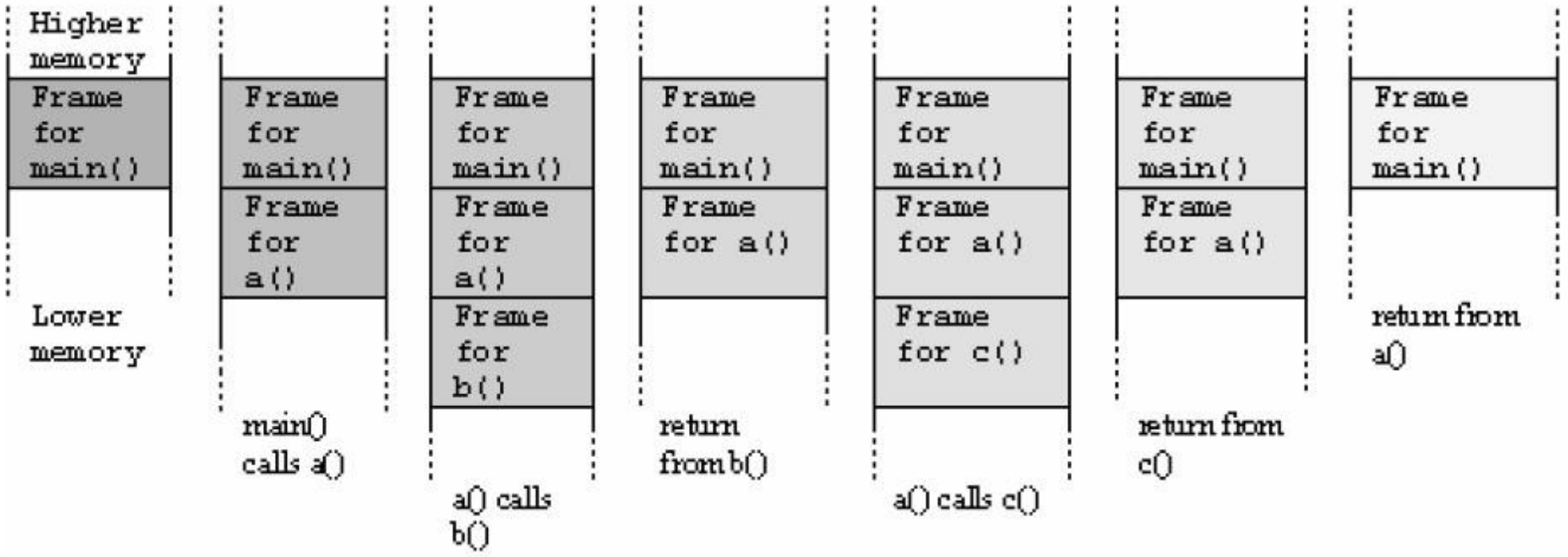
Izgled stack-a pri pozivanju funkcija

```
int main()
{
    a();
    return 0;
}
```

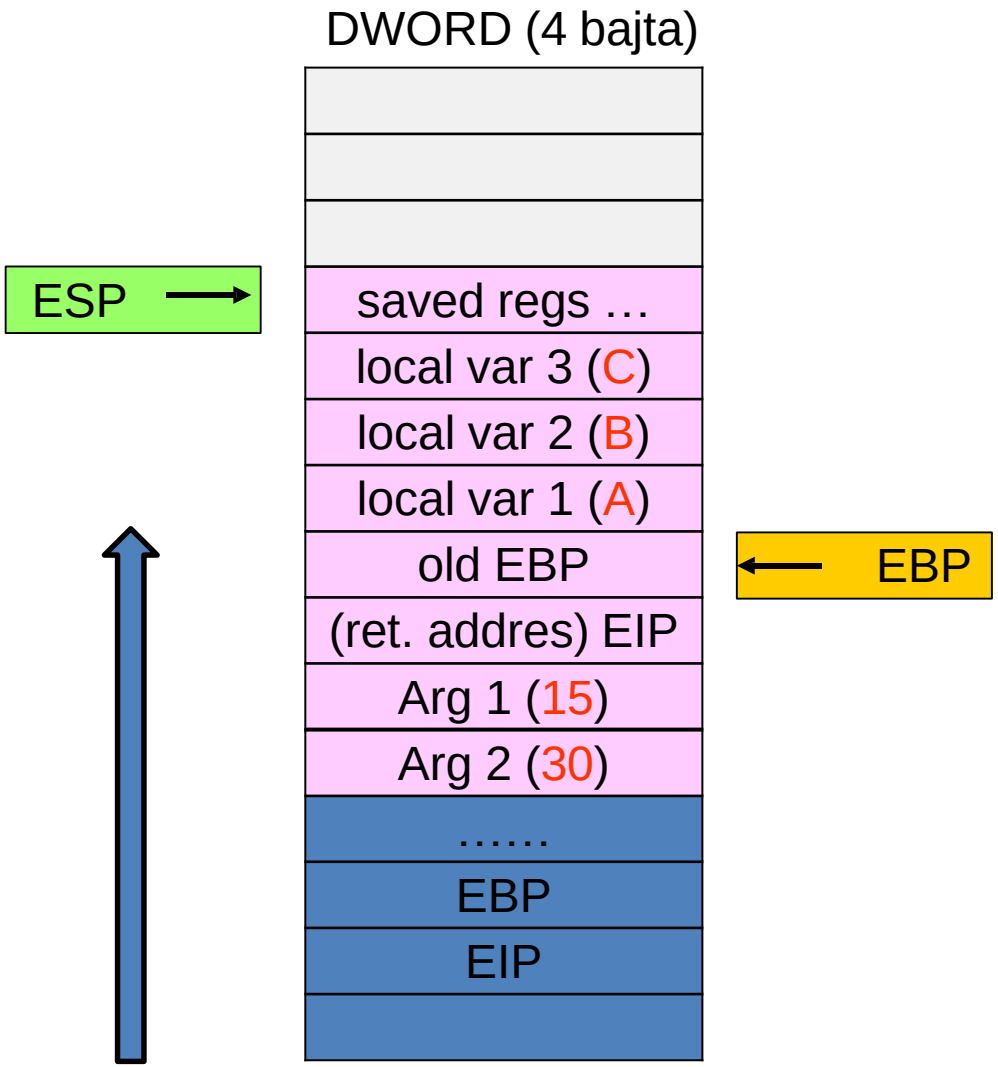
```
int a()
{
    b();
    c();
    return 0;
}
```

```
int b()
{ return 0; }

int c()
{ return 0; }
```



Stack frame




```
void main(void)
{
……
    ret = funkcija(15, 30);
}

int funkcija(int Arg1, int Arg2)
{
int A, B, C;
    [pristup Arg1 == EBP + 8]
    [pristup Arg2 == EBP + 12]
    [pristup loc.var B == EBP - 8]
……
    return B;
}
```

Direktive C Preprocesora

- Uključivanje drugih datoteka u fajl za prevođenje
  - #include <stdlib.h>, #include “MyDef.h”
  - Koristiti / (ne \) npr. #include “Common/MyDef.h”
- Definicija konstanti, tipova podataka i makroa
  - #define PI 3.14
  - #define AREA(a,b) ((a)*(b))
- Kontrola prevođenja (conditional compilation)

```
#if !defined(NULL)
        #define NULL 0
#endif

#ifdef DEBUG
        printf(“Var x= %d”, x);
#endif
```

Kreiranje makroa

- Prilikom kreiranja makroa, neophodno je obratiti pažnju na korišćenje zagrada, kojim se reguliše prioritet operacija
- Primer nepravilnog definisanja makroa:

```
#define JEDAN 1
#define DVA JEDAN + JEDAN        ; DVA = 1+1
#define CETIRI DVA * DVA         ; CETIRI = 1+1*1+1, što je 3
```

- Primer pravilnog definisanja makroa:

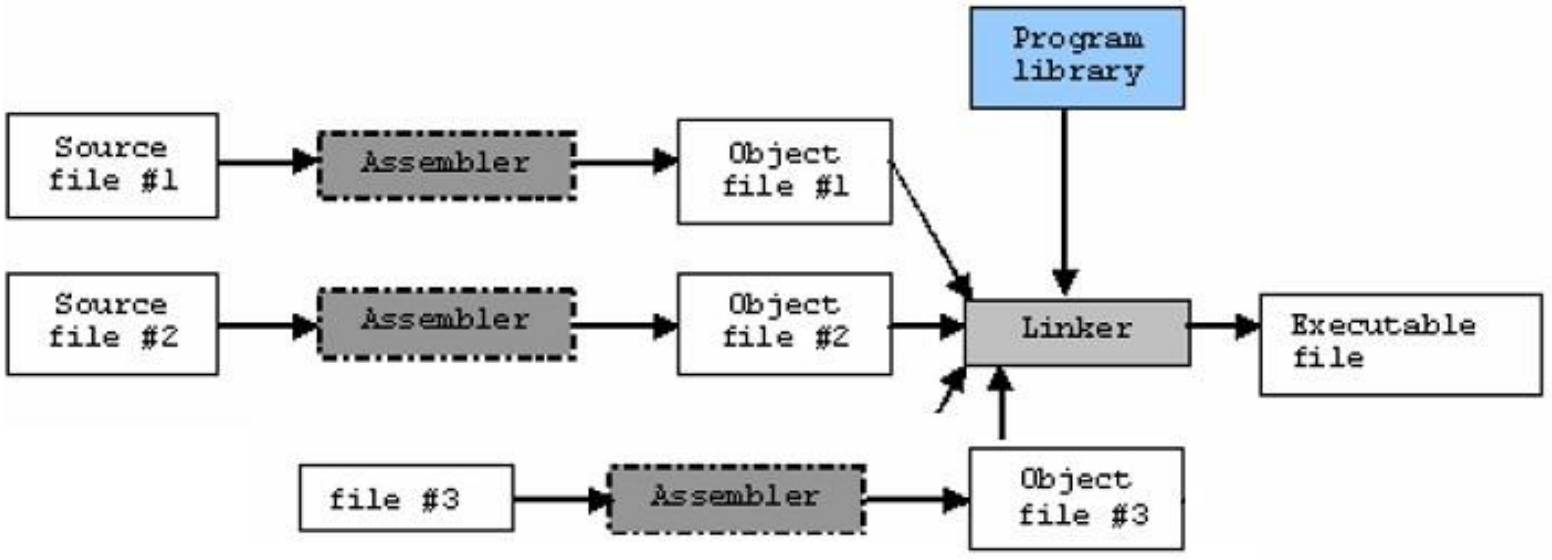
```
#define JEDAN 1
#define DVA (JEDAN + JEDAN)      ; DVA = 1+1
#define CETIRI (DVA * DVA)       ; CETIRI = ((1+1)*(1+1)), što je 4
```

Predefinisani makroi

Simbolička konstanta	Opis
__DATE__	Datum prevođenja
__LINE__	Broj linije u .c datoteci
__FILE__	Ime datoteke izvornog koda
__TIME__	Vreme prevođenja
__STDC__	Označava ANSI C kompatibilnost

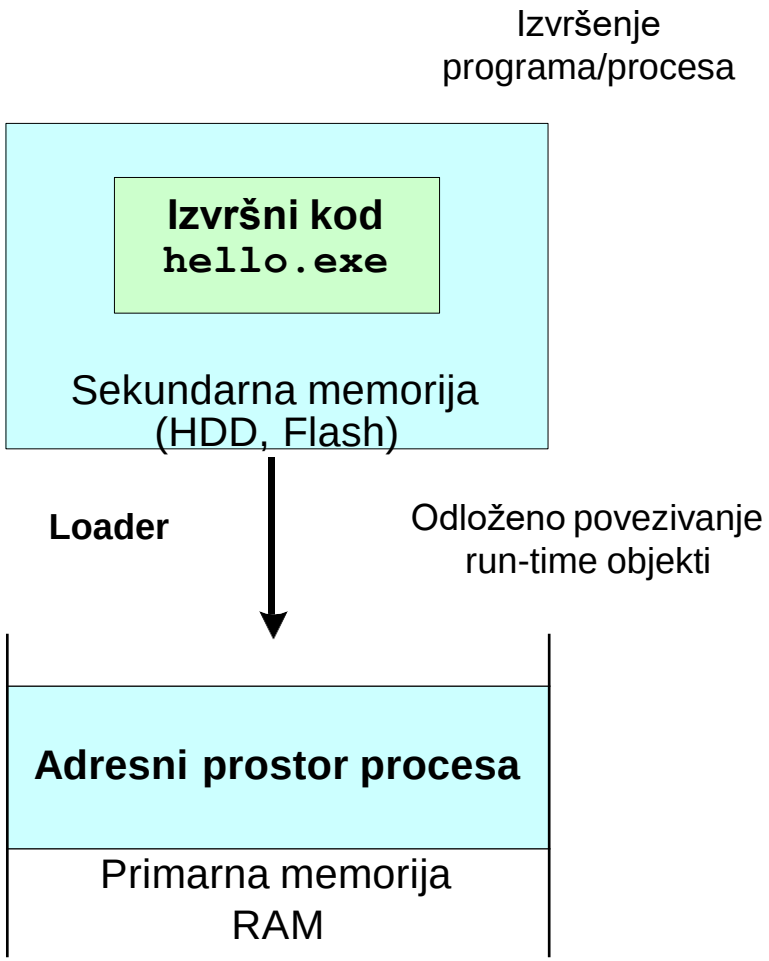
Linker

- Povezuje u izvršni kod, rešavanjem međusobnih adresnih referenci
  - Statičko – sve potrebno u izvršni fajl
  - Dinamičko – deljeni izvršni moduli (.dll)



Punjač (Loader)

- Pre izvršavanja se puni u operativnu memoriju
  - Funkcija OS
  - Sa diska
    - Sve se kopira
  - Iz Flash-a
    - Sve se kopira (max.var)
    - Samo inicijalizovane promenljive (min.var)
- Dinamičko povezivanje
  - Deffered linking
  - Run-time moduli/biblioteke
- Bootstrap loader
  - Nalazi se u ROM ili EPROM čipu i zadatak mu je da pri paljenju računara učita OS



Proces punjenja

- Verifikacija izvršnog programa i računanje memorijskih zahteva
- Verifikacija raspoložive memorije i prava pristupa
- Alokacija potrebne memorije i prenos sa sekundarne memorije
- Formiranje sekcija podataka (DATA + BSS + Heap sekcija)
- Inicijalizacija stack-a i argumenata za main() funkciju
- Pozivanje main() funkcije

Tipovi podataka u C

Type Name	Bytes	Other Names	Range of Values
int	4	signed	−2,147,483,648 to 2,147,483,647
unsigned int	4	unsigned	0 to 4,294,967,295
__int8	1	char	−128 to 127
unsigned_int8	1	unsigned char	0 to 255
__int16	2	short, short int, signed short int	−32,768 to 32,767
unsigned_int16	2	unsigned short, unsigned short int	0 to 65,535
__int32	4	signed, signed int, int	−2,147,483,648 to 2,147,483,647
unsigned_int32	4	unsigned, unsigned int	0 to 4,294,967,295
__int64	8	long long, signed long long	−9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned_int64	8	unsigned long long	0 to 18,446,744,073,709,551,615
bool	1	none	false or true
char	1	none	−128 to 127 by default, 0 to 255 when compiled by using /J
signed char	1	none	−128 to 127
unsigned char	1	none	0 to 255
short	2	short int, signed short int	−32,768 to 32,767
unsigned short	2	unsigned short int	0 to 65,535
long	4	long int, signed long int	−2,147,483,648 to 2,147,483,647
unsigned long	4	unsigned long int	0 to 4,294,967,295
long long	8	none (but equivalent to_int64)	−9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned long long	8	none (but equivalent to unsigned_int64)	0 to 18,446,744,073,709,551,615
enum	varies	none	/
float	4	none	3.4E +/- 38 (7 digits)
double	8	none	1.7E +/- 308 (15 digits)
long double	same as double	none	Same as double
wchar_t	2	__wchar_t	0 to 65,535

Promenljive

- Deo memorije koji čuva određeni tip podataka
  - tip_pod ImePromenljive;
  - sizeof(ImePromenljive) == sizeof(tip_pod)
- Tipovi promenljivih:
  - lokalne – vidljive unutar jedne funkcije
    - Smeštene na stack-u
  - globalne – definisane iznad tela funkcije
  - extern – spoljne (drugi modul)
  - static – dostupne samo funkcijama u jednom modulu
  - const – konstante (read only)
  - register – čuvaju se u registru
  - volatile – podložna promeni iz prekida

Izvedeni tipovi podataka

- Direktive: struct, union, typedef, enum
- Sve izvedene tipove treba označiti typedef-om
- Prvi član enum-a uvek mora biti inicijalizovan
- Strukture su zgodan način za grupisanje i prenos podataka između funkcija
  - sizeof(struct) == Σ_i sizeof(član $_i$), za pack(1)
  - Pristup članu: ImeStruct.član, PtrStruct->član

```
typedef struct {
   uint16_t       x;
   uint16_t       y;
} gdiPoint_t;

typedef enum { ILL = -1, GOOD, ...} boxStatus_t;

typedef struct listNode_t {
   struct listNode_t  *next;
   int                data;
} listNode_t;
```

Izvedeni tipovi podataka

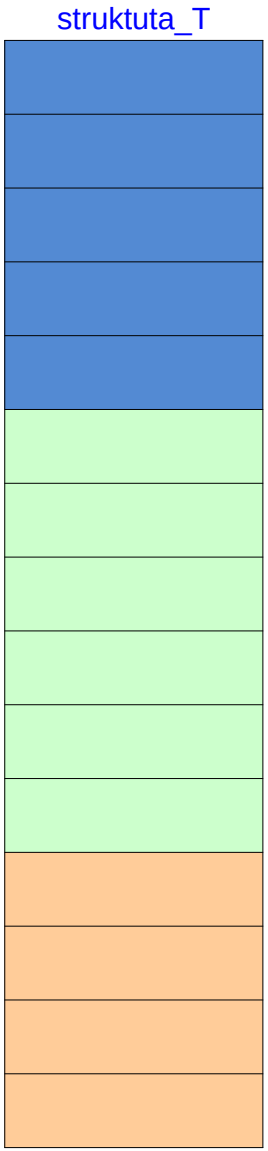
- union direktiva definiše tip podataka gde članovi dele isti prostor za smeštanje (storage space)
  - sizeof(union) = sizeof(najduži član)

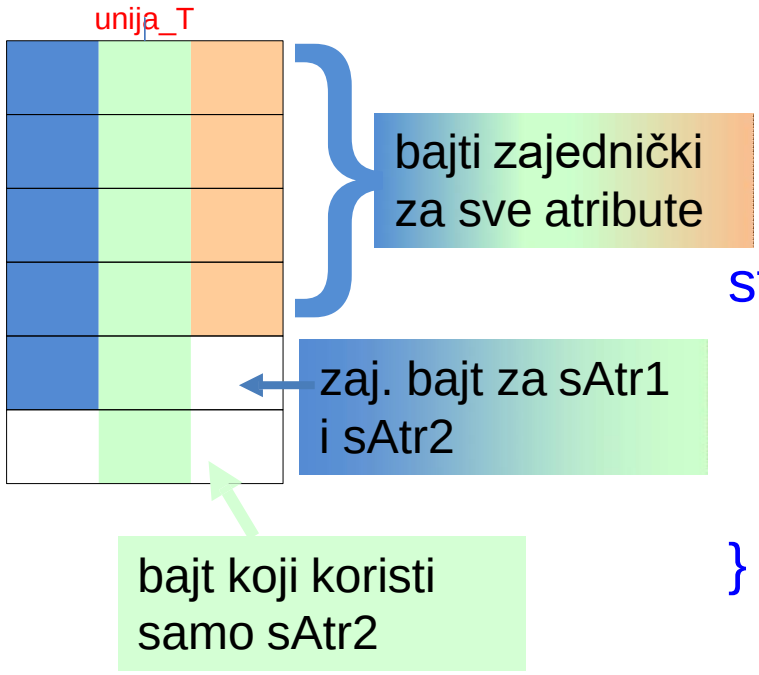
```
union sample
{
int p;
float q;
};
…
union sample content = {234};
  ili
union sample content = {24.67};

  pristip clanovima unije

content.p = 37;
content.q = 1.2765;
```

Razlika između strukture i unije






```
struct {
    char   sAtr1[5];
    short  sAtr2[3];
    long   sAtr3;
} struktura_T;
```

```
union {
    char   sAtr1[5];
    short  sAtr2[3];
    long   sAtr3;
} unija_T;
```

Primeri korišćenja struktura i unija

- Definisati tip podataka **boja** koji sadrži 4 iRGB (intenzitet, crvena, zelena, plava) komponente
  - Boja može da se prikaže kao jedan broj od 0 do 4.2 milijarde
  - Boja može da se razloži na 4 komponente: i, r, g i b tipa bajt (brojevi iz intervala od 0 do 255)

```
typedef union {
    unsigned int color;
    struct {
       byte i, r, g, b;
    }
} boja;
```

Main funkcija

- Ulazna tačka u programu
- int main(int argc, char **argv)

```
C:\>echo This is command line argument
This is command line argument
```