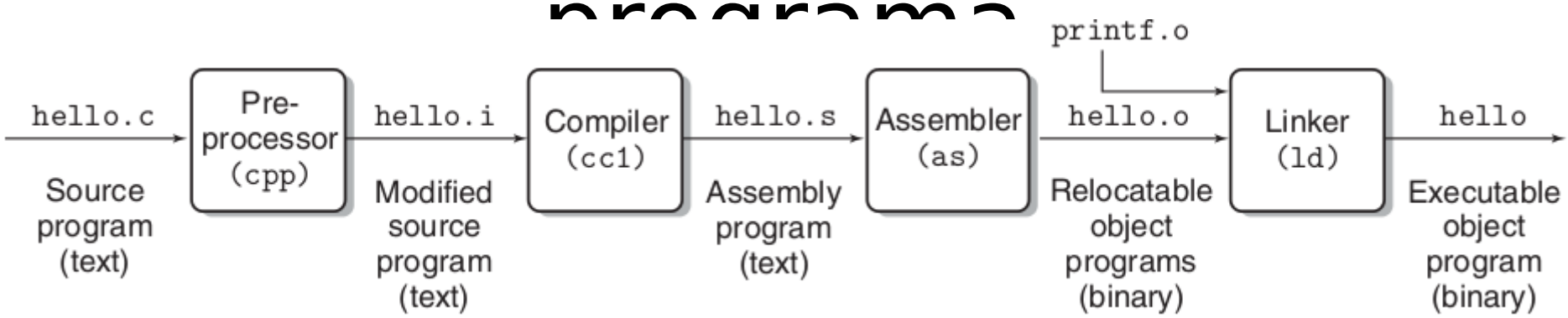
GNU Programski alati

Proces prevođenja programa



- Komandom

  gcc –save-temps -hello.c –o hello

  Pokreće se proces prevođenja izvornog C programa u izvršni program na mašinskom jeziku (save temps opcija ne briše radne fajlove procesa).
- Proces obuhvata faze makroprocesiranja, kompajliranja (prevođenja), asembliranja i povezivanja
- gcc je *compiler driver* koji orkestrira pojedine alate

Proces prevođenja programa

hello.c

```
#include <stdio.h>

#define PORUKA "Zdravo svete.\n"

Int main() {
  printf(PORUKA);
}
```

Proces prevođenja programa

- **Faza makroprocesiranja.** Pretprocesor (cpp) menja originalni C program prema direktivama koje počinju # znakom.
- Na primer, #include komanda u liniji 1 hello.c govori pretprocesoru da pročita sadržaj sistemskog stdio.h zaglavlja datoteke i ubaci ga direktno u programski tekst.
- #define vrši zamenu makroimena PORUKA tekstom zamene “Zdravo svete.”
- Rezultat je još jedan C program hello.i
- (Eksplicitan poziv: **cpp hello.c –o hello.i**)

Proces prevođenja programa

hello.i

```
....
# 1 "hello.c"
# 1 "/usr/include/stdio.h" 1 3 4
....
extern int printf (const char *__restrict __format, ...);
....
# 2 "hello.c" 2

# 5 "hello.c"
int main() {
  printf("Zdravo svete.\n");
}
```

Proces prevođenja programa

- **Faza kompilacije**. Prevodilac (cc1) prevodi tekst datoteke *hello.i* u tekstualni fajl *hello.s*, koji sadrži program u asemblerskom jeziku.
- Svaka naredba asemblerskog programa opisuje jednu naredbu u mašinskom jeziku. Asemblerski jezik je zajednički izlaz za različite kompajlere (na primer, C kompajler i Fortran kompajler).
- Direktan poziv:
- gcc -print-prog-name=cc1

  /usr/lib/gcc/x86_64-linux-gnu/7/cc1

- **/usr/lib/gcc/x86_64-linux-gnu/7/cc1 hello.i –o hello.s**

- (dodatne opcije za čistiji hello.s)
  -Og -fno-pie -masm=intel -fno-asynchronous-unwind-tables

Proces prevođenja programa (hello.s)

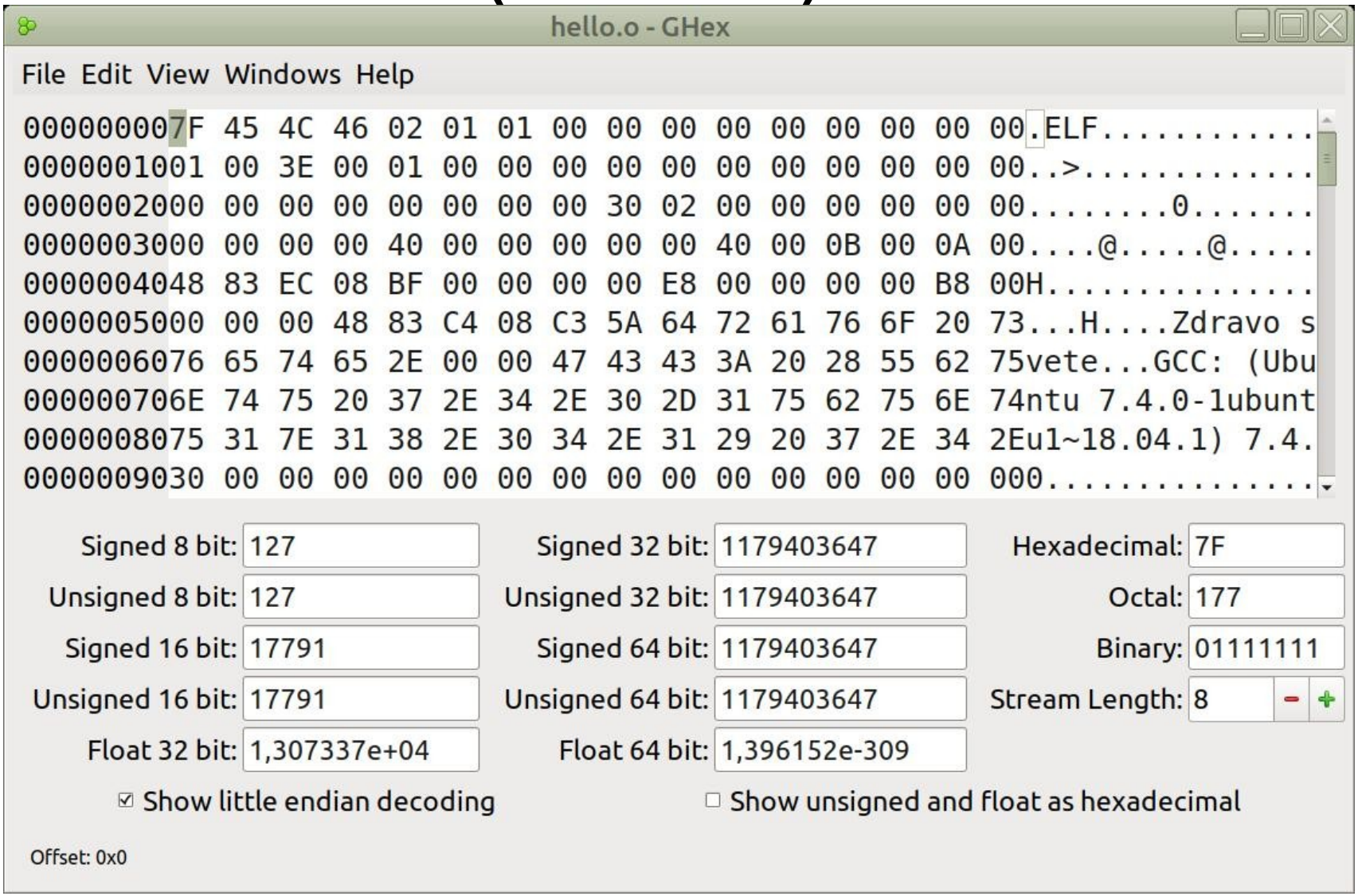
```
    .file "hello.i"
    .intel_syntax noprefix
    .text
    .section    .rodata.str1.1,"aMS",@progbits,1
.LC0:
    .string     "Zdravo svete."
    .text
    .globl      main
    .type       main, @function
main:
    sub rsp, 8
    movedi, OFFSET FLAT:.LC0
    call puts
    moveax, 0
    add rsp, 8
    ret
    .size       main, .-main
    .ident      "GCC: (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0"
    .section    .note.GNU-stack,"",@progbits
```

Proces prevođenja programa

- **Faza asembliranja**. Asembler (as) prevodi hello.s u mašinski jezik, pakuje ga u obliku poznatom kao relokativni objektni program i čuva rezultat u hello.o objektnoj datoteci. Hello.o fajl je binarni fajl koji pored bajtova mašinskih instrukcija čuva još informacije o globalnim simbolima itd.
- Direktno pozivanje: **as hello.s -o hello.o**
- file hello.o

  hello.o: ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), not stripped

Proces prevođenja programa (hello.o)



Proces prevođenja programa (hello.o)

$ objdump -M intel -f -t -d hello.o

(objdump prikazuje imformacije o objektnim i izvršnim fajlovima

–f hederi, -t simboli, -d disasembler, -M intel sintaksa)

hello.o: file format elf64-x86-64
architecture: i386:x86-64, flags 0x00000011:
HAS_RELOC, HAS_SYMS
start address 0x0000000000000000

Disassembly of section .text:

0000000000000000 <main>:

0:	48 83 ec 08	sub	rsp,0x8
4:	bf 00 00 00 00	mov	edi,0x0
9:	e8 00 00 00 00	call	e <main+0xe>
e:	b8 00 00 00 00	mov	eax,0x0
13:	48 83 c4 08	add	rsp,0x8
17:	c3	ret	

SYMBOL TABLE:

0000000000000000 l	df *ABS*	0000000000000000	hello.i
0000000000000000 l	d .text	0000000000000000	.text
0000000000000000 l	d .data	0000000000000000	.data
0000000000000000 l	d .bss	0000000000000000	.bss
0000000000000000 l	d .rodata.str1.1	0000000000000000	.rodata.str1.1
0000000000000000 l	d .note.GNU-stack	0000000000000000	.note.GNU-stack
0000000000000000 l	d .comment	0000000000000000	.comment
0000000000000000 g	F .text	0000000000000018	main
0000000000000000	*UND*	0000000000000000	puts

Proces prevođenja programa

- **Faza povezivanja**. Primetiti da program *hello* poziva *printf* funkciju, koja je deo standardne C biblioteke koje pruža svaki C kompajler.
- *printf* funkcija nalazi se u posebnom pretkompajliranom objektnom fajlu pod nazivom printf.o, koji mora nekako da se spoji sa našim hello.o programom.
- Linker (ld) obavlja ovo povezivanje. Rezultat je *hello* fajl, koji je izvršni objektni fajl (ili jednostavno izvršni fajl) koji je spreman da se učita u memoriju i izvršava od strane operativnog sistema.

Proces prevođenja programa

- **Faza povezivanja**. Da bismo znali kako tačno pozvati linker (ld) korisno je zadati verbose opciju gcc **-v** hello.c –o hello koja ispisuje kako prednji kraj poziva svaki alat (collect2 je fasada za ld):
- COLLECT_GCC_OPTIONS='-v' '-o' 'hello' '-mtune=generic' '-march=x86-64'
- /usr/lib/gcc/x86_64-linux-gnu/7/collect2 -plugin /usr/lib/gcc/x86_64-linux-gnu/7/liblto_plugin.so -plugin-opt=/usr/lib/gcc/x86_64-linux-gnu/7/lto-wrapper -plugin-opt=-fresolution=/tmp/ccnACcrs.res -plugin-opt=-pass-through=-lgcc -plugin-opt=-pass-through=-lgcc_s -plugin-opt=-pass-through=-lc -plugin-opt=-pass-through=-lgcc -plugin-opt=-pass-through=-lgcc_s --sysroot=/ --build-id --eh-frame-hdr -m elf_x86_64 --hash-style=gnu --as-needed -dynamic-linker /lib64/ld-linux-x86-64.so.2 -pie -z now -z relro -o hello /usr/lib/gcc/x86_64-linux-gnu/7/../../../x86_64-linux-gnu/Scrt1.o /usr/lib/gcc/x86_64-linux-gnu/7/../../../x86_64-linux-gnu/crti.o /usr/lib/gcc/x86_64-linux-gnu/7/crtbeginS.o -L/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/x86_64-linux-gnu/7/../../../x86_64-linux-gnu -L/usr/lib/gcc/x86_64-linux-gnu/7/../../../../lib -L/lib/x86_64-linux-gnu -L/lib/../lib -L/usr/lib/x86_64-linux-gnu -L/usr/lib/../lib -L/usr/lib/gcc/x86_64-linux-gnu/7/../../.. /tmp/ccOim1HV.o -lgcc --push-state --as-needed -lgcc_s --pop-state -lc -lgcc --push-state --as-needed -lgcc_s --pop-state /usr/lib/gcc/x86_64-linux-gnu/7/crtendS.o /usr/lib/gcc/x86_64-linux-gnu/7/../../../x86_64-linux-gnu/crtn.o

Proces prevođenja programa

- **Faza povezivanja**
- Na komandnoj liniji linkera (ld) redosled zadavanja fajlova je sledeći:

  crt1.o crti.o crtbegin.o [-L paths] [user objects] [gcc libs] [C libs] [gcc libs] crtend.o crtn.o
- Informativno objašnjenje crt (**c run time**) fajlova koji daju podršku izvršavanju prevedenog C programa. To su male sekcije koda koje se ubacuju (ld) i izvršavaju na početku prevedenog C programa i na kraju. Ovaj kod se brine za inicijalizaciju globalnih promenjljivih, heap-a i steka:
- **crt0.o** Older style of the initial runtime code. Usually not generated anymore with Linux toolchains, but often found in bare metal toolchains. Serves same purpose as crt1.o (see below).

Proces prevođenja programa

- **crt1.o** Newer style of the initial runtime code. Contains the _start symbol which sets up the env with argc/argv/libc _init/libc _fini before jumping to the libc main. glibc calls this file 'start.S'.
- **crti.o** Defines the function prolog; _init in the .init section and _fini in the .fini section. glibc calls this 'initfini.c'.
- **crtn.o** Defines the function epilog. glibc calls this 'initfini.c'.
- **Scrt1.o** Used in place of crt1.o when generating Position Independent Executables (PIEs).
- **gcrt1.o** Used in place of crt1.o when generating code with profiling information. Compile with -pg. Produces output suitable for the gprof util.
- **Mcrt1.o** Like gcrt1.o, but is used with the prof utility. glibc installs this as a dummy file as it's useless on linux systems.

Proces prevođenja programa

- **crtbegin.o** GCC uses this to find the start of the constructors.
- **crtbeginS.o** Used in place of crtbegin.o when generating shared objects/PIEs.
- **crtbeginT.o** Used in place of crtbegin.o when generating static executables.
- **crtend.o** GCC uses this to find the start of the destructors.
- **crtendS.o** Used in place of crtend.o when generating shared objects/PIEs.

Proces prevođenja programa

- **Faza povezivanja**.
- **Statičko povezivanje** (sav bibliotečki kod biće ubačen u izvršni fajl):

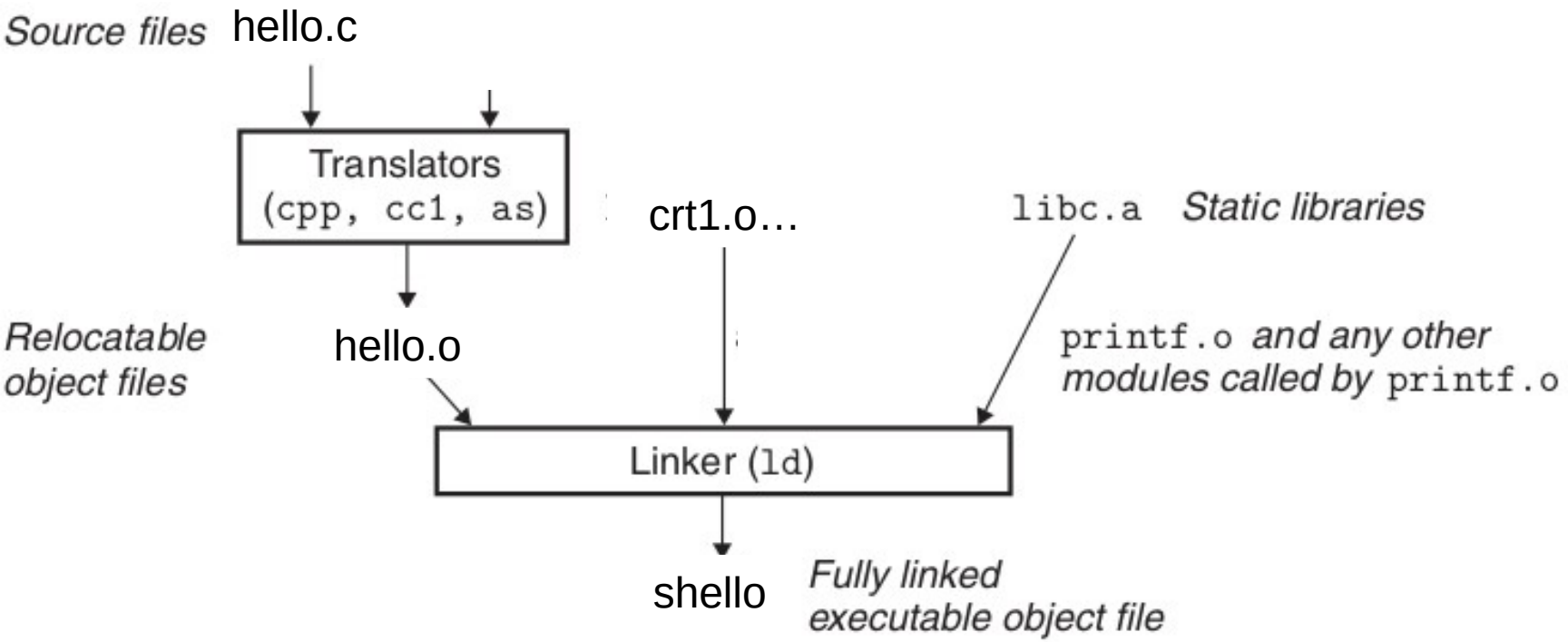
  gcc –static hello.c –o shello

- Komplikovanija varijanta sa pozivanjem ld:

```
export L=/usr/lib/x86_64-linux-gnu
export L1=/usr/lib/gcc/x86_64-linux-gnu/7
ld  -L$L1 -static  $L/crt1.o $L/crti.o
   $L1/crtbeginT.o hello.o  -lc  -lgcc -lgcc_eh
   -lc   $L1/crtend.o $L/crtn.o -o shello
```

Proces prevođenja programa

- **Faza povezivanja**.
- **Statičko povezivanje** (sav bibliotečki kod biće ubačen u izvršni fajl):

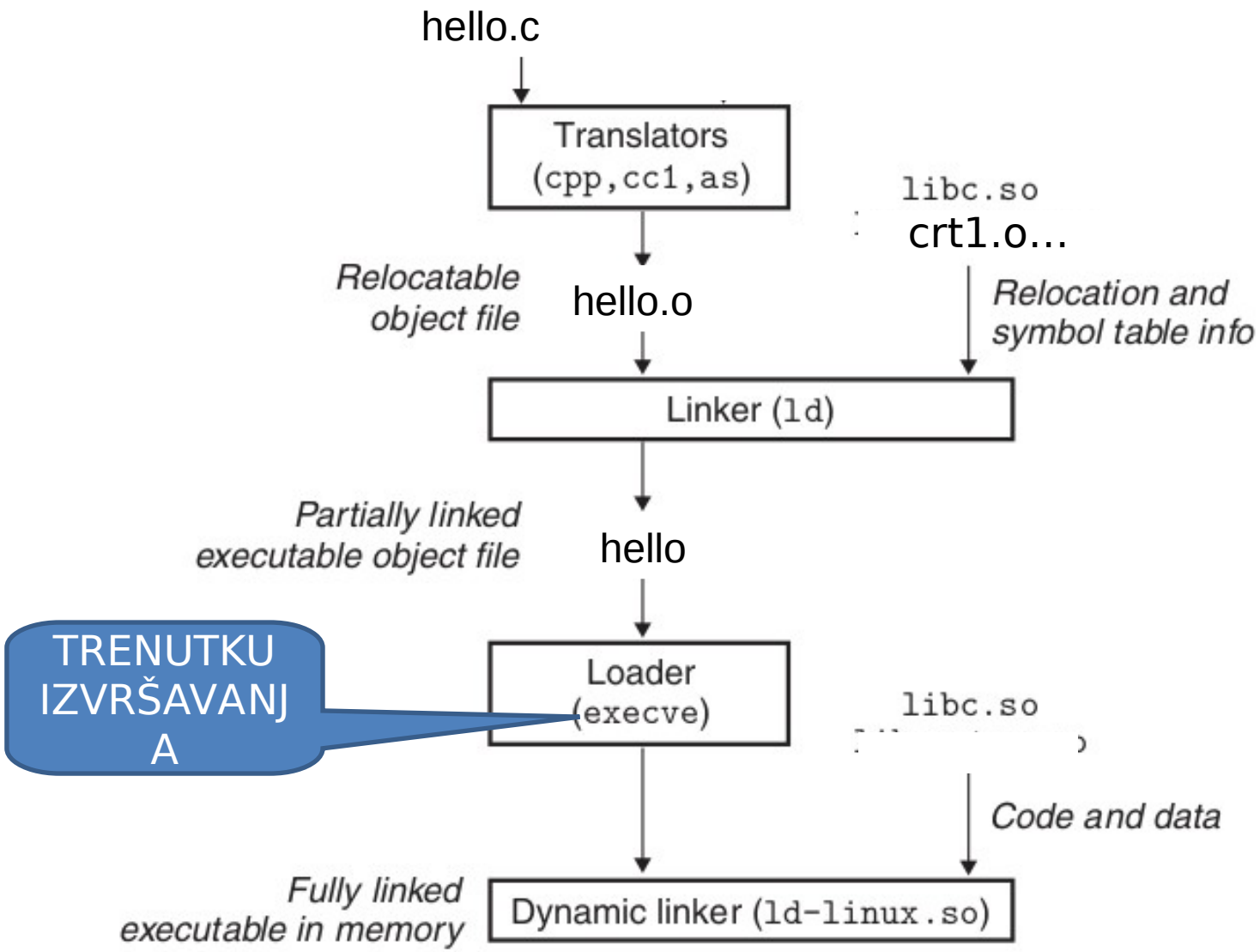


Proces prevođenja programa

- **Faza povezivanja**.
- **Dinamičko povezivanje** (dinamički povezivač će u trenutku startovanja povezati kod našeg programa sa deljenom bibliotekom):
- **gcc hello.c –o hello**
- Komplikovanija varijanta sa pozivanjem ld:

```
export L=/usr/lib/x86_64-linux-gnu
export L1=/usr/lib/gcc/x86_64-linux-gnu/7
ld  -dynamic-linker /lib64/ld-linux-x86-64.so.2
   $L/crt1.o $L/crti.o $L1/crtbegin.o hello.o -lc
   $L1/crtend.o $L/crtn.o -o hello
```

Proces prevođenja programa (dinamičko povezivanje)



Proces prevođenja programa

- Statičko vs dinamičko povezivanje – veličina izvršnog programa:

```
$ ls –l *
-rwxrwxr-x 1 db db   8064 феб  9 10:18 hello
-rw-rw-r-- 1 db db     88 феб  9 08:57 hello.c
-rw-rw-r-- 1 db db  17927 феб  9 09:15 hello.i
-rw-rw-r-- 1 db db   1552 феб  9 10:31 hello.o
-rw-rw-r-- 1 db db    362 феб  9 09:19 hello.s
-rwxrwxr-x 1 db db 844592 феб  9 10:38 shello
```

Proces prevođenja

```
objdump -M intel -f  -d hello

hello:     file format elf64-x86-64
architecture: i386:x86-64, flags 0x00000112:
EXEC_P, HAS_SYMS, D_PAGED
start address 0x00000000004003d0
Disassembly of section .text:


00000000004003d0 <_start>:
  4003d0:       31 ed                   xor    ebp,ebp
  4003d2:       49 89 d1                mov    r9,rdx
  4003d5:       5e                      pop    rsi
  4003d6:       48 89 e2                mov    rdx,rsp
  4003d9:       48 83 e4 f0             and    rsp,0xfffffffffffffff0
  4003dd:       50                      push   rax
  4003de:       54                      push   rsp
  4003df:       49 c7 c0 40 05 40 00    mov    r8,0x400540
  4003e6:       48 c7 c1 d0 04 40 00    mov    rcx,0x4004d0
  4003ed:       48 c7 c7 b7 04 40 00    mov    rdi,0x4004b7
  4003f4:       ff 15 f6 0b 20 00       call   QWORD PTR
    [rip+0x200bf6]        # 600ff0 <__libc_start_main@GLIBC_2.2.5>
  4003fa:       f4                      hlt
```

```
0000000000004004b7 <main>:
  4004b7:       48 83 ec 08             sub    rsp,0x8
  4004bb:       bf 54 05 40 00          mov    edi,0x400554
  4004c0:       e8 fb fe ff ff          call   4003c0 <puts@plt>
  4004c5:       b8 00 00 00 00          mov    eax,0x0
  4004ca:       48 83 c4 08             add    rsp,0x8
  4004ce:       c3                      ret
  4004cf:       90                      nop
```