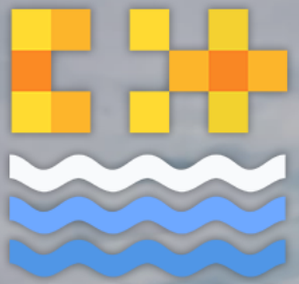
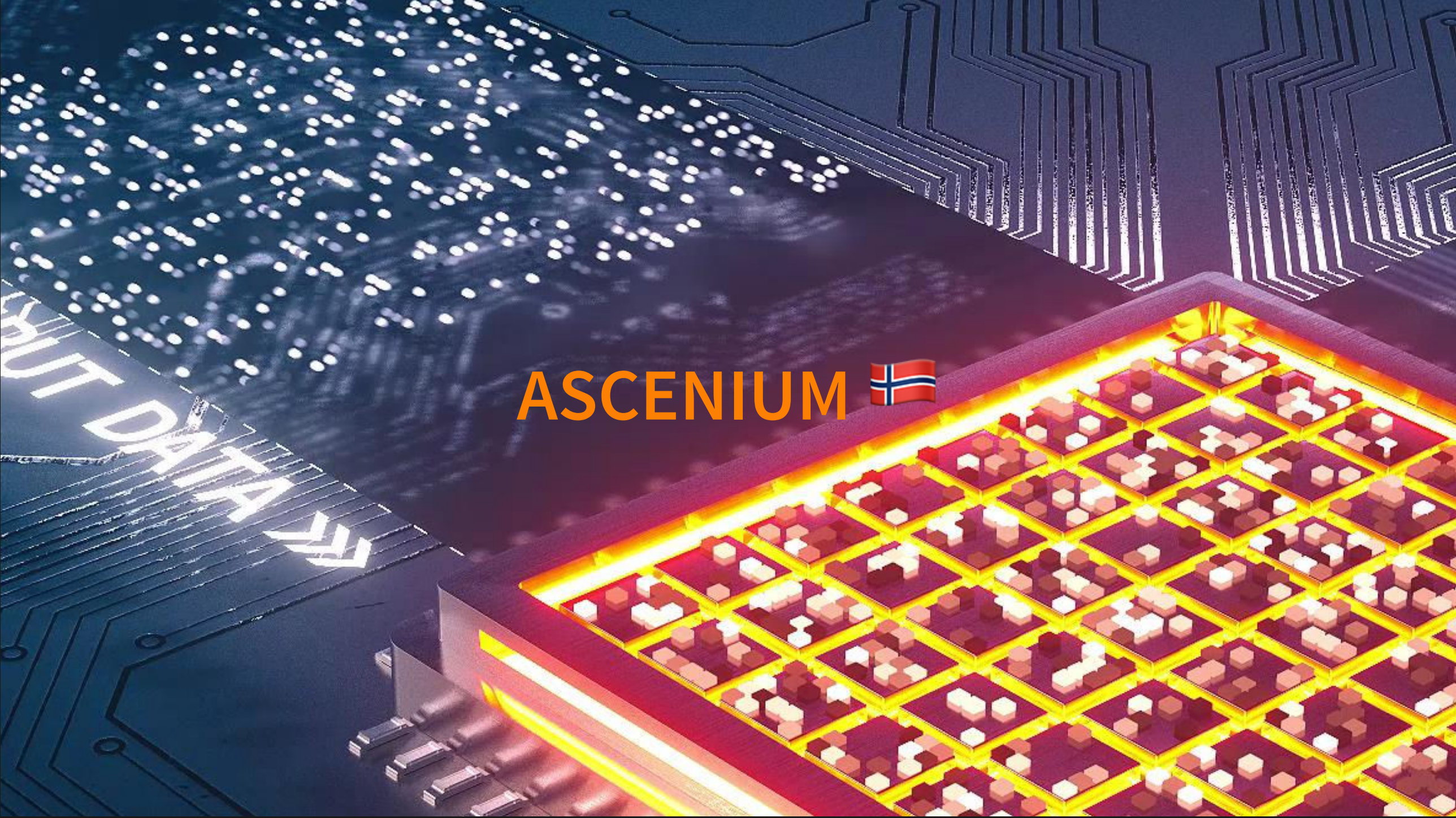
What Happens After the Compiler



Anders Schau Knatten


ASCENIUM

CPPQUIZ.ORG

C++ Quiz

You've answered 1 of 138 questions correctly. (Clear)

Question #197 Difficulty: ●○○

According to the C++17 standard, what is the output of this program?

```
#include <iostream>

int j = 1;

int main() {
  int& i = j, j;
  j = 2;
  std::cout << i << j;
}
```

Answer: The program is guaranteed to output: [ ] Answer

Problems? View a hint or try another question.

I give up, show me the answer (make 3 more attempts first).

Mode : Training

You are currently in training mode, answering random questions. Why not Start a new quiz? Then you can boast about your score, and invite your friends.

Contribute

Create your own!

Android app

Get Sergey Vasilchenko's CppQuiz Android app.

@knatten / @cppquiz / @AffectiveCpp

BEFORE WE START

- Linux Only, Windows is similar
- Ask (most) questions during the presentation

A SIMPLE FUNCTION

```c
int main()
{
    return 1;
}
```

```c
int main()
{
    return 1;
}
```

```asm
main:
    push    rbp
    mov     rbp, rsp
    mov     eax, 1
    pop     rbp
    ret
```

```c
int main()
{
    return 1;
}
```

```asm
main:
    push    rbp
    mov     rbp, rsp
    mov     eax, 1
    pop     rbp
    ret
```

```c
int main()
{
    return 1;
}
```

```asm
main:
    push    rbp
    mov     rbp, rsp
    mov     eax, 1
    pop     rbp
    ret
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```asm
compute():
        push    rbp
        mov     rbp, rsp
        mov     eax, 1
        pop     rbp
        ret
main:
        push    rbp
        mov     rbp, rsp
        call    compute()
        nop
        pop     rbp
        ret
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```asm
compute():
        push    rbp
        mov     rbp, rsp
        mov     eax, 1
        pop     rbp
        ret
main:
        push    rbp
        mov     rbp, rsp
        call    compute()
        nop
        pop     rbp
        ret
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```asm
compute():
        push    rbp
        mov     rbp, rsp
        mov     eax, 1
        pop     rbp
        ret
main:
        push    rbp
        mov     rbp, rsp
        call    compute()
        nop
        pop     rbp
        ret
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```asm
compute():
        push    rbp
        mov     rbp, rsp
        mov     eax, 1
        pop     rbp
        ret
main:
        push    rbp
        mov     rbp, rsp
        call    compute()
        nop
        pop     rbp
        ret
```

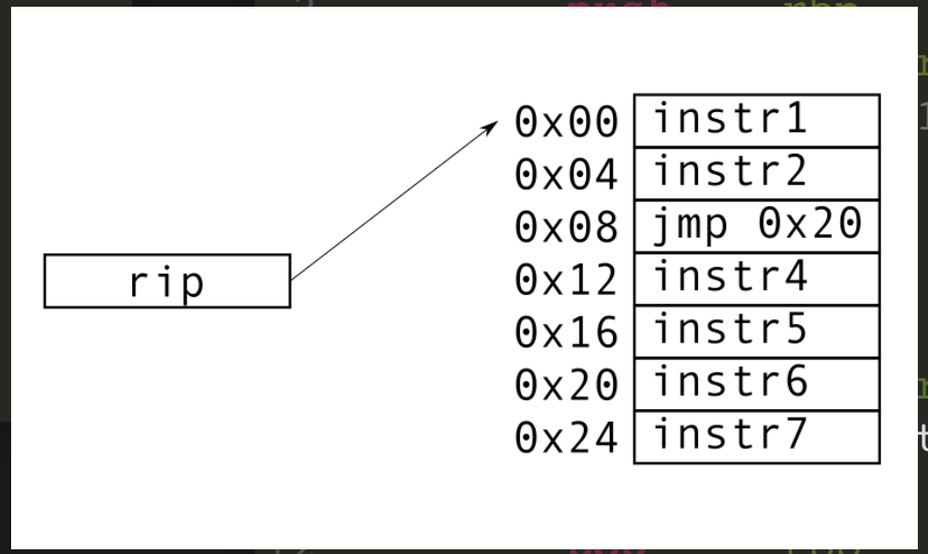
```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```asm
compute():
        push    rbp
        mov     rbp, rsp
        mov     eax, 1
        pop     rbp
        ret
main:
        push    rbp
        mov     rbp, rsp
        call    compute()
        nop
        pop     rbp
        ret
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```asm
compute():
        push    rbp
        mov     rbp, rsp
        mov     eax, 1
        pop     rbp
        ret
main:
        push    rbp
        mov     rbp, rsp
        call    compute()
        nop
        pop     rbp
        ret
```

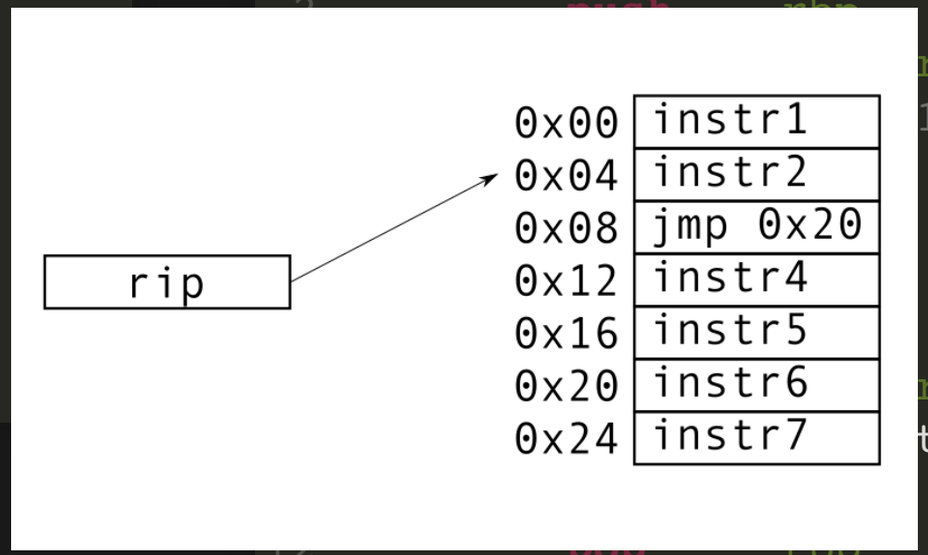
```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```




```
1  compute():
                          rsp
                          1
                          rsp
                          te()
12          pop     rbp
13          ret
```

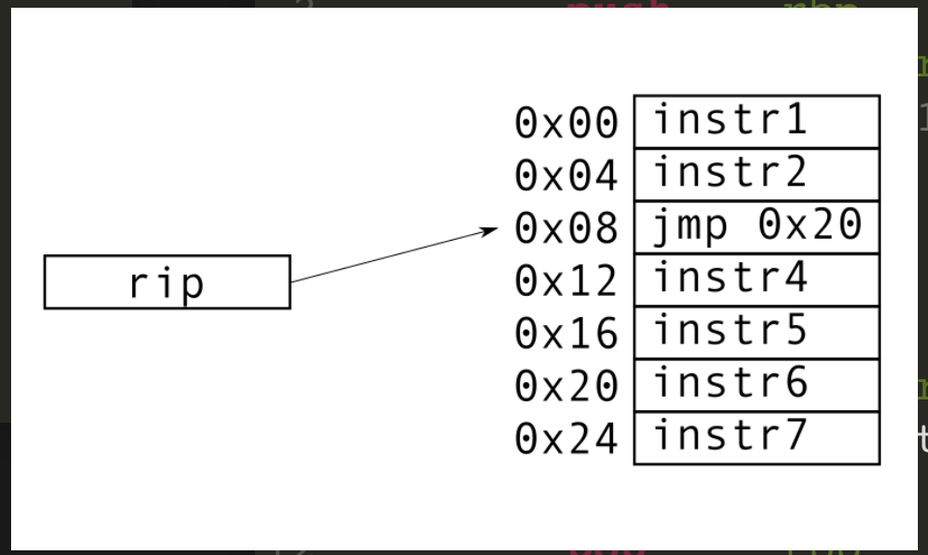
```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```
1  compute():
                    rsp
                    1
                    rsp
                    te()
13          ret
```

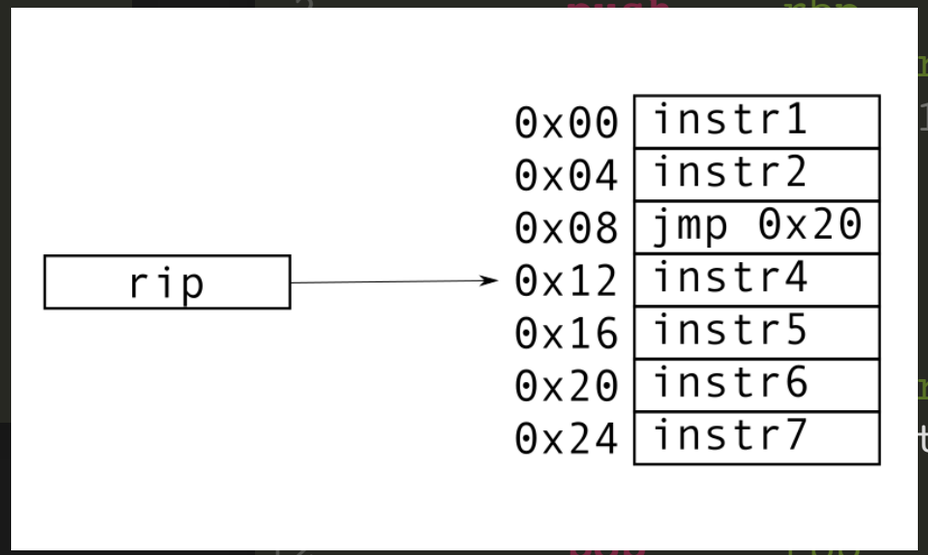


```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```
1  compute():
                    rsp
                    1
                    rsp
                    te()
13          ret
```

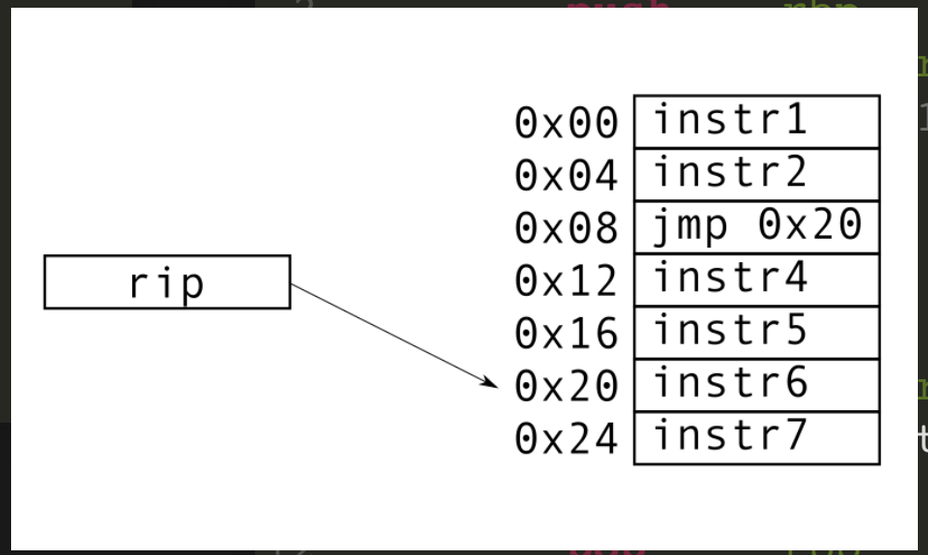


```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```



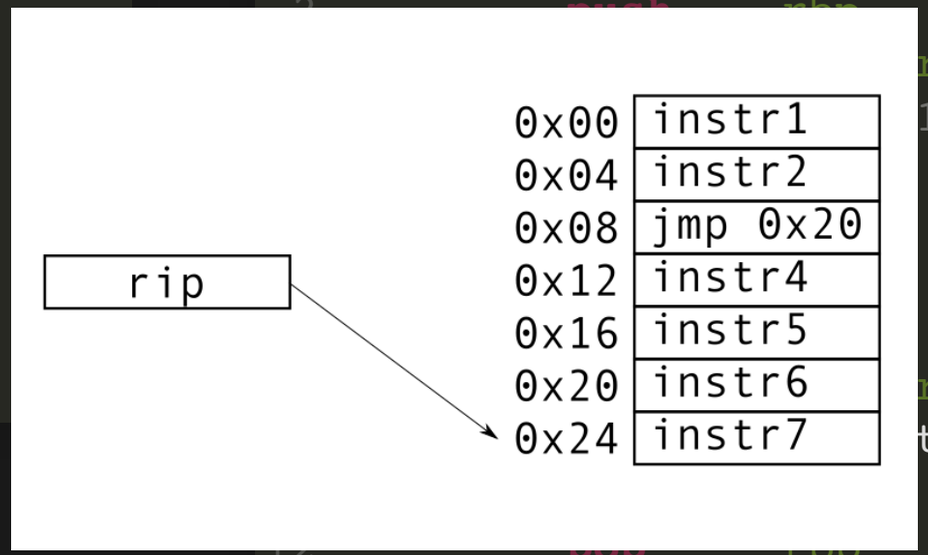

```asm
1  compute():
                    rsp
                    1
                    rsp
                    te()
13          ret
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```




```
1  compute():
                          rsp
                          1
                          rsp
                          te()
12          pop     rbp
13          ret
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```




```
1  compute():
                          rsp
                          1
                          rsp
                          te()
12          pop     rbp
13          ret
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```asm
compute():
        push    rbp
        mov     rbp, rsp
        mov     eax, 1
        pop     rbp
        ret
main:
        push    rbp
        mov     rbp, rsp
        call    compute()
        nop
        pop     rbp
        ret
```

```
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```
gcc -c main.c
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```
gcc -c main.c
```

```
xxd main.o
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```
gcc -c main.c
```

```
xxd main.o
```

```
00000000: 7F45 4C46 0201 0100 0000 0000 0000 0000  .ELF............
00000010: 0100 3E00 0100 0000 0000 0000 0000 0000  ..>.............
00000020: 0000 0000 0000 0000 A001 0000 0000 0000  ................
00000030: 0000 0000 4000 0000 0000 4000 0B00 0A00  ....@.....@.....
00000040: F30F 1EFA 5548 89E5 B801 0000 005D C3F3  ....UH.......]..
00000050: 0F1E FA55 4889 E5B8 0000 0000 E800 0000  ...UH...........
00000060: 005D C300 4743 433A 2028 5562 ...
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```
gcc -c main.c
```

```
objdump -wDr -Mintel main.o
```

```
xxd main.o
```

```
7F454C4602010100000000000000000001003E0001000000000000000000000000000000
00000000A001000000000000000000004000000000000040000B000A00F30F1EFA554889E5

B8010000005DC3F30F1EFA554889E5B800000000E8000000005DC3004743433A20285562...
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```
gcc -c main.c
```

```
xxd main.o
```

```
objdump -wDr -Mintel main.o
```

```
Disassembly of section .text:

0000000000000000 <compute>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   rbp
   5:   48 89 e5                mov    rbp,rsp
   8:   b8 01 00 00 00          mov    eax,0x1
   d:   5d                      pop    rbp
   e:   c3                      ret

000000000000000f <main>:
   f:   f3 0f 1e fa             endbr64
  13:   55                      push   rbp
  14:   48 89 e5                mov    rbp,rsp
  17:   b8 00 00 00 00          mov    eax,0x0
  1c:   e8 00 00 00 00          call   21 <main+0x12>
                                  1d: R_X86_64_PLT32  compute-0x4
  21:   5d                      pop    rbp
  22:   c3                      ret
```

```
7F454C4602010100000000000000000001003E0001000000000000000000000000000000
00000000A001000000000000000000400000000000400000B000A00F30F1EFA554889E5

B8010000005DC3F30F1EFA554889E5B800000000E8000000005DC3004743433A20285562...
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```
gcc -c main.c
```

```
xxd main.o
```

```
objdump -wDr -Mintel main.o
```

```
Disassembly of section .text:

0000000000000000 <compute>:
   0:  f3 0f 1e fa       endbr64
   4:  55                push   rbp
   5:  48 89 e5          mov    rbp,rsp
   8:  b8 01 00 00 00    mov    eax,0x1
   d:  5d                pop    rbp
   e:  c3                ret

000000000000000f <main>:
   f:  f3 0f 1e fa       endbr64
  13:  55                push   rbp
  14:  48 89 e5          mov    rbp,rsp
  17:  b8 00 00 00 00    mov    eax,0x0
  1c:  e8 00 00 00 00    call   21 <main+0x12>
                           1d: R_X86_64_PLT32  compute-0x4
  21:  5d                pop    rbp
  22:  c3                ret
```

```
7F454C4602010100000000000000000001003E0001000000000000000000000000000000
00000000A001000000000000000000400000000000400000B000A00F30F1EFA554889E5

B8010000005DC3F30F1EFA554889E5B800000000E8000000005DC3004743433A20285562...
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```
gcc -c main.c
```

```
xxd main.o
```

```
objdump -wDr -Mintel main.o
```

```
Disassembly of section .text:

0000000000000000 <compute>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   rbp
   5:   48 89 e5                mov    rbp,rsp
   8:   b8 01 00 00 00          mov    eax,0x1
   d:   5d                      pop    rbp
   e:   c3                      ret

000000000000000f <main>:
   f:   f3 0f 1e fa             endbr64
  13:   55                      push   rbp
  14:   48 89 e5                mov    rbp,rsp
  17:   b8 00 00 00 00          mov    eax,0x0
  1c:   e8 00 00 00 00          call   21 <main+0x12>
                                  1d: R_X86_64_PLT32  compute-0x4
  21:   5d                      pop    rbp
  22:   c3                      ret
```

```
7F454C4602010100000000000000000001003E0001000000000000000000000000000000
00000000A001000000000000000000400000000000400000B000A00F30F1EFA554889E5

B8010000005DC3F30F1EFA554889E5B800000000E8000000005DC3004743433A20285562...
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```
gcc -c main.c
```

```
xxd main.o
```

```
objdump -wDr -Mintel main.o
```

```
 1 Disassembly of section .text:
 2
 3 0000000000000000 <compute>:
 4    0:  f3 0f 1e fa       endbr64
 5    4:  55                push   rbp
 6    5:  48 89 e5          mov    rbp,rsp
 7    8:  b8 01 00 00 00    mov    eax,0x1
 8    d:  5d                pop    rbp
 9    e:  c3                ret
10
11 000000000000000f <main>:
12    f:  f3 0f 1e fa       endbr64
13   13:  55                push   rbp
14   14:  48 89 e5          mov    rbp,rsp
15   17:  b8 00 00 00 00    mov    eax,0x0
16   1c:  e8 00 00 00 00    call   21 <main+0x12>
17                            1d: R_X86_64_PLT32  compute-0x4
18   21:  5d                pop    rbp
19   22:  c3                ret
```

```
7F454C4602010100000000000000000001003E0001000000000000000000000000000000
00000000A001000000000000000000400000000000400000B000A00F30F1EFA554889E5

B8010000005DC3F30F1EFA554889E5B800000000E8000000005DC3004743433A20285562...
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```
gcc -c main.c
```

```
xxd main.o
```

```
objdump -wDr -Mintel main.o
```

```
Disassembly of section .text:

0000000000000000 <compute>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   rbp
   5:   48 89 e5                mov    rbp,rsp
   8:   b8 01 00 00 00          mov    eax,0x1
   d:   5d                      pop    rbp
   e:   c3                      ret

000000000000000f <main>:
   f:   f3 0f 1e fa             endbr64
  13:   55                      push   rbp
  14:   48 89 e5                mov    rbp,rsp
  17:   b8 00 00 00 00          mov    eax,0x0
  1c:   e8 00 00 00 00          call   21 <main+0x12>
                                  1d: R_X86_64_PLT32  compute-0x4
  21:   5d                      pop    rbp
  22:   c3                      ret
```

```
7F454C4602010100000000000000000001003E0001000000000000000000000000000000
00000000A00100000000000000000004000000000040000B000A00F30F1EFA554889E5

B8010000005DC3F30F1EFA554889E5B800000000E8000000005DC3004743433A20285562...
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```
gcc -c main.c
```

```
xxd main.o
```

```
objdump -wDr -Mintel main.o
```

```
 1 Disassembly of section .text:
 2
 3 0000000000000000 <compute>:
 4    0:  f3 0f 1e fa       endbr64
 5    4:  55                push   rbp
 6    5:  48 89 e5          mov    rbp,rsp
 7    8:  b8 01 00 00 00    mov    eax,0x1
 8    d:  5d                pop    rbp
 9    e:  c3                ret
10
11 000000000000000f <main>:
12    f:  f3 0f 1e fa       endbr64
13   13:  55                push   rbp
14   14:  48 89 e5          mov    rbp,rsp
15   17:  b8 00 00 00 00    mov    eax,0x0
16   1c:  e8 00 00 00 00    call   21 <main+0x12>
17                            1d: R_X86_64_PLT32  compute-0x4
18   21:  5d                pop    rbp
19   22:  c3                ret
```

```
7F454C4602010100000000000000000001003E0001000000000000000000000000000000
00000000A001000000000000000000400000000000400000B000A00F30F1EFA554889E5

B8010000005DC3F30F1EFA554889E5B800000000E8000000005DC3004743433A20285562...
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```
gcc -c main.c
```

```
xxd main.o
```

```
objdump -wDr -Mintel main.o
```

```
 1 Disassembly of section .text:
 2
 3 0000000000000000 <compute>:
 4    0:  f3 0f 1e fa       endbr64
 5    4:  55                push   rbp
 6    5:  48 89 e5          mov    rbp,rsp
 7    8:  b8 01 00 00 00    mov    eax,0x1
 8    d:  5d                pop    rbp
 9    e:  c3                ret
10
11 000000000000000f <main>:
12    f:  f3 0f 1e fa       endbr64
13   13:  55                push   rbp
14   14:  48 89 e5          mov    rbp,rsp
15   17:  b8 00 00 00 00    mov    eax,0x0
16   1c:  e8 00 00 00 00    call   21 <main+0x12>
17                            1d: R_X86_64_PLT32  compute-0x4
18   21:  5d                pop    rbp
19   22:  c3                ret
```

```
7F454C4602010100000000000000000001003E0001000000000000000000000000000000
00000000A001000000000000000000400000000000400000B000A00F30F1EFA554889E5

B8010000005DC3F30F1EFA554889E5B800000000E8000000005DC3004743433A20285562...
```

```c
int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

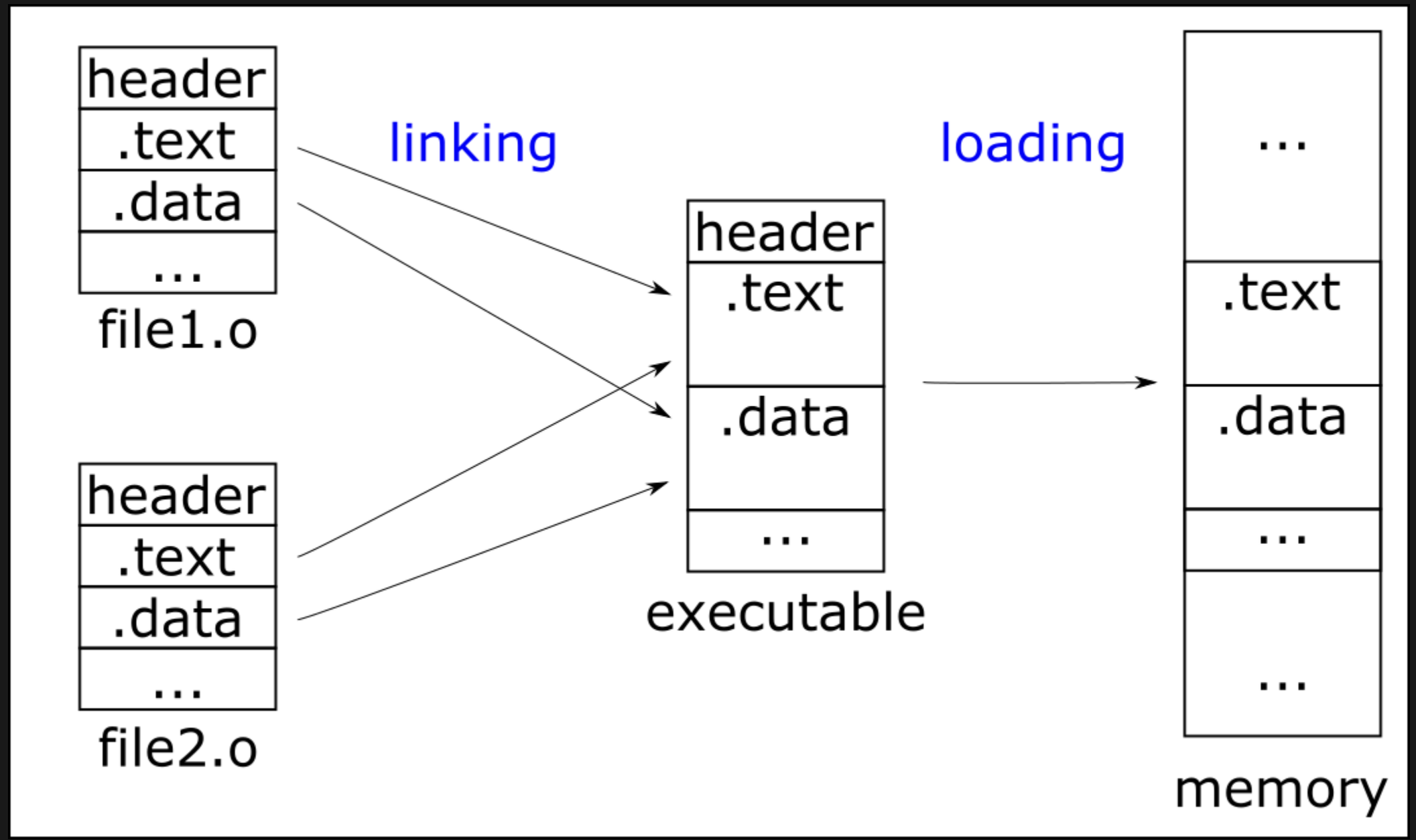
```
gcc -c main.c
```

```
xxd main.o
```

```
objdump -wDr -Mintel main.o
```

```
Disassembly of section .text:

0000000000000000 <compute>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   rbp
   5:   48 89 e5                mov    rbp,rsp
   8:   b8 01 00 00 00          mov    eax,0x1
   d:   5d                      pop    rbp
   e:   c3                      ret

000000000000000f <main>:
   f:   f3 0f 1e fa             endbr64
  13:   55                      push   rbp
  14:   48 89 e5                mov    rbp,rsp
  17:   b8 00 00 00 00          mov    eax,0x0
  1c:   e8 00 00 00 00          call   21 <main+0x12>
                                  1d: R_X86_64_PLT32  compute-0x4
  21:   5d                      pop    rbp
  22:   c3                      ret
```

```
7F454C4602010100000000000000000001003E0001000000000000000000000000000000
00000000A001000000000000000000400000000000400000B000A00F30F1EFA554889E5

B8010000005DC3F30F1EFA554889E5B800000000E8000000005DC3004743433A20285562...
```


header
.text
.data
...
file1.o
header
.text
.data
...
file2.o
linking
header
.text
.data
...
executable
loading
...
.text
.data
...
...
memory

RELOCATIONS

```
objdump -wDr -Mintel main.o
```

```
0000000000000000 <compute>:
   0:  f3 0f 1e fa      endbr64
   4:  55               push   rbp
   5:  48 89 e5         mov    rbp,rsp
   8:  b8 01 00 00 00   mov    eax,0x1
   d:  5d               pop    rbp
   e:  c3               ret

000000000000000f <main>:
   f:  f3 0f 1e fa      endbr64
  13:  55               push   rbp
  14:  48 89 e5         mov    rbp,rsp
  17:  b8 00 00 00 00   mov    eax,0x0
  1c:  e8 00 00 00 00   call   21 <main+0x12>
                          1d: R_X86_64_PLT32  compute-0x4
  21:  5d               pop    rbp
  22:  c3               ret
```

```
objdump -wDr -Mintel main.o
```

```
0000000000000000 <compute>:
   0:  f3 0f 1e fa      endbr64
   4:  55               push   rbp
   5:  48 89 e5         mov    rbp,rsp
   8:  b8 01 00 00 00   mov    eax,0x1
   d:  5d               pop    rbp
   e:  c3               ret

000000000000000f <main>:
   f:  f3 0f 1e fa      endbr64
  13:  55               push   rbp
  14:  48 89 e5         mov    rbp,rsp
  17:  b8 00 00 00 00   mov    eax,0x0
  1c:  e8 00 00 00 00   call   21 <main+0x12>
                          1d: R_X86_64_PLT32  compute-0x4
  21:  5d               pop    rbp
  22:  c3               ret
```

```
readelf -rW main.o
```

```
objdump -wDr -Mintel main.o
```

```
0000000000000000 <compute>:
   0:  f3 0f 1e fa      endbr64
   4:  55               push   rbp
   5:  48 89 e5         mov    rbp,rsp
   8:  b8 01 00 00 00   mov    eax,0x1
   d:  5d               pop    rbp
   e:  c3               ret

000000000000000f <main>:
   f:  f3 0f 1e fa      endbr64
  13:  55               push   rbp
  14:  48 89 e5         mov    rbp,rsp
  17:  b8 00 00 00 00   mov    eax,0x0
  1c:  e8 00 00 00 00   call   21 <main+0x12>
                          1d: R_X86_64_PLT32  compute-0x4
  21:  5d               pop    rbp
  22:  c3               ret
```

```
readelf -rW main.o
```

```
Relocation section '.rela.text' at offset 0x198 contains 1 entry:
    Offset             Info             Type       Symbol's Value  Symbol's Name + A
000000000000001d 0000000300000004 R_X86_64_PLT32 0000000000000000 compute - 4
```

```
gcc -o main main.o
objdump -wDr -Mintel main
```

```
gcc -o main main.o
objdump -wDr -Mintel main
```

```
 1 Disassembly of section .text:
 2
 3 (...)
 4
 5 0000000000001129 <compute>:
 6     1129: f3 0f 1e fa           endbr64
 7     112d: 55                    push   rbp
 8     112e: 48 89 e5              mov    rbp,rsp
 9     1131: b8 01 00 00 00        mov    eax,0x1
10     1136: 5d                    pop    rbp
11     1137: c3                    ret
12
13 0000000000001138 <main>:
14     1138: f3 0f 1e fa           endbr64
15     113c: 55                    push   rbp
16     113d: 48 89 e5              mov    rbp,rsp
17     1140: b8 00 00 00 00        mov    eax,0x0
18     1145: e8 df ff ff ff        call   1129 <compute>
19     114a: 5d                    pop    rbp
20     114b: c3                    ret
21     114c: 0f 1f 40 00           nop    DWORD PTR [rax+0x0]
```

```
gcc -o main main.o
objdump -wDr -Mintel main
```

```
 1 Disassembly of section .text:
 2
 3 (...)
 4
 5 0000000000001129 <compute>:
 6     1129: f3 0f 1e fa           endbr64
 7     112d: 55                    push   rbp
 8     112e: 48 89 e5              mov    rbp,rsp
 9     1131: b8 01 00 00 00        mov    eax,0x1
10     1136: 5d                    pop    rbp
11     1137: c3                    ret
12
13 0000000000001138 <main>:
14     1138: f3 0f 1e fa           endbr64
15     113c: 55                    push   rbp
16     113d: 48 89 e5              mov    rbp,rsp
17     1140: b8 00 00 00 00        mov    eax,0x0
18     1145: e8 df ff ff ff        call   1129 <compute>
19     114a: 5d                    pop    rbp
20     114b: c3                    ret
21     114c: 0f 1f 40 00           nop    DWORD PTR [rax+0x0]
```

```
gcc -o main main.o
objdump -wDr -Mintel main
```

```
df ff ff ff
```

```
 1 Disassembly of section .text:
 2
 3 (...)
 4
 5 0000000000001129 <compute>:
 6     1129: f3 0f 1e fa           endbr64
 7     112d: 55                    push   rbp
 8     112e: 48 89 e5              mov    rbp,rsp
 9     1131: b8 01 00 00 00        mov    eax,0x1
10     1136: 5d                    pop    rbp
11     1137: c3                    ret
12
13 0000000000001138 <main>:
14     1138: f3 0f 1e fa           endbr64
15     113c: 55                    push   rbp
16     113d: 48 89 e5              mov    rbp,rsp
17     1140: b8 00 00 00 00        mov    eax,0x0
18     1145: e8 df ff ff ff        call   1129 <compute>
19     114a: 5d                    pop    rbp
20     114b: c3                    ret
21     114c: 0f 1f 40 00           nop    DWORD PTR [rax+0x0]
```

```
gcc -o main main.o
objdump -wDr -Mintel main
```

```
 1 Disassembly of section .text:
 2
 3 (...)
 4
 5 0000000000001129 <compute>:
 6     1129: f3 0f 1e fa           endbr64
 7     112d: 55                    push   rbp
 8     112e: 48 89 e5              mov    rbp,rsp
 9     1131: b8 01 00 00 00        mov    eax,0x1
10     1136: 5d                    pop    rbp
11     1137: c3                    ret
12
13 0000000000001138 <main>:
14     1138: f3 0f 1e fa           endbr64
15     113c: 55                    push   rbp
16     113d: 48 89 e5              mov    rbp,rsp
17     1140: b8 00 00 00 00        mov    eax,0x0
18     1145: e8 df ff ff ff        call   1129 <compute>
19     114a: 5d                    pop    rbp
20     114b: c3                    ret
21     114c: 0f 1f 40 00           nop    DWORD PTR [rax+0x0]
```

```
df ff ff ff
```

```
ff ff ff df
```

```
gcc -o main main.o
objdump -wDr -Mintel main
```

```
 1 Disassembly of section .text:
 2
 3 (...)
 4
 5 0000000000001129 <compute>:
 6     1129: f3 0f 1e fa           endbr64
 7     112d: 55                    push   rbp
 8     112e: 48 89 e5              mov    rbp,rsp
 9     1131: b8 01 00 00 00        mov    eax,0x1
10     1136: 5d                    pop    rbp
11     1137: c3                    ret
12
13 0000000000001138 <main>:
14     1138: f3 0f 1e fa           endbr64
15     113c: 55                    push   rbp
16     113d: 48 89 e5              mov    rbp,rsp
17     1140: b8 00 00 00 00        mov    eax,0x0
18     1145: e8 df ff ff ff        call   1129 <compute>
19     114a: 5d                    pop    rbp
20     114b: c3                    ret
21     114c: 0f 1f 40 00           nop    DWORD PTR [rax+0x0]
```

```
df ff ff ff
```

```
ff ff ff df
```

```
-21
```

```
gcc -o main main.o
objdump -wDr -Mintel main
```

```
 1 Disassembly of section .text:
 2
 3 (...)
 4
 5 0000000000001129 <compute>:
 6     1129: f3 0f 1e fa           endbr64
 7     112d: 55                    push   rbp
 8     112e: 48 89 e5              mov    rbp,rsp
 9     1131: b8 01 00 00 00        mov    eax,0x1
10     1136: 5d                    pop    rbp
11     1137: c3                    ret
12
13 0000000000001138 <main>:
14     1138: f3 0f 1e fa           endbr64
15     113c: 55                    push   rbp
16     113d: 48 89 e5              mov    rbp,rsp
17     1140: b8 00 00 00 00        mov    eax,0x0
18     1145: e8 df ff ff ff        call   1129 <compute>
19     114a: 5d                    pop    rbp
20     114b: c3                    ret
21     114c: 0f 1f 40 00           nop    DWORD PTR [rax+0x0]
```

```
df ff ff ff
```

```
ff ff ff df
```

```
-21
```

```
114a - 21 = 1129
```

```
gcc -o main main.o
objdump -wDr -Mintel main
```

```
 1 Disassembly of section .text:
 2
 3 (...)
 4
 5 0000000000001129 <compute>:
 6     1129: f3 0f 1e fa           endbr64
 7     112d: 55                    push   rbp
 8     112e: 48 89 e5              mov    rbp,rsp
 9     1131: b8 01 00 00 00        mov    eax,0x1
10     1136: 5d                    pop    rbp
11     1137: c3                    ret
12
13 0000000000001138 <main>:
14     1138: f3 0f 1e fa           endbr64
15     113c: 55                    push   rbp
16     113d: 48 89 e5              mov    rbp,rsp
17     1140: b8 00 00 00 00        mov    eax,0x0
18     1145: e8 df ff ff ff        call   1129 <compute>
19     114a: 5d                    pop    rbp
20     114b: c3                    ret
21     114c: 0f 1f 40 00           nop    DWORD PTR [rax+0x0]
```

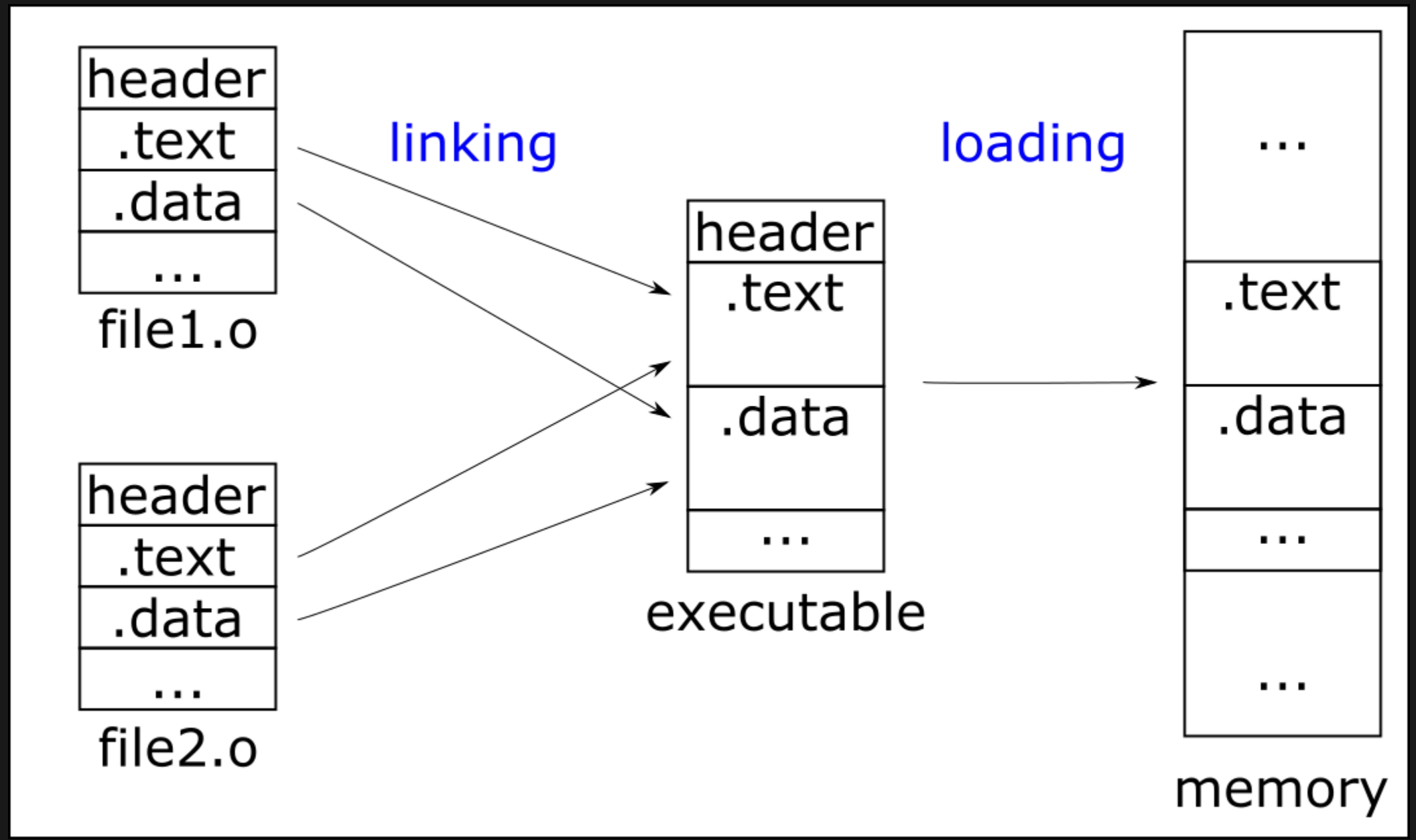
```
df ff ff ff
```

```
ff ff ff df
```

```
-21
```

```
114a - 21 = 1129
```

```
$ ./main; echo $?
1
```


header
.text
.data
...
file1.o
header
.text
.data
...
file2.o
linking
header
.text
.data
...
executable
loading
...
.text
.data
...
...
memory

```c
static int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```c
static int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```
gcc -c main.c
objdump -wDr -Mintel main.o
```

```c
static int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```
gcc -c main.c
objdump -wDr -Mintel main.o
```

```
Disassembly of section .text:

0000000000000000 <compute>:
   0: f3 0f 1e fa           endbr64
   4: 55                    push   rbp
   5: 48 89 e5              mov    rbp,rsp
   8: b8 01 00 00 00        mov    eax,0x1
   d: 5d                    pop    rbp
   e: c3                    ret

000000000000000f <main>:
   f: f3 0f 1e fa           endbr64
  13: 55                    push   rbp
  14: 48 89 e5              mov    rbp,rsp
  17: b8 00 00 00 00        mov    eax,0x0
  1c: e8 df ff ff ff        call   0 <compute>
  21: 5d                    pop    rbp
  22: c3                    ret
```

```c
static int compute()
{
    return 1;
}

int main()
{
    return compute();
}
```

```
gcc -c main.c
objdump -wDr -Mintel main.o
```

```
Disassembly of section .text:

0000000000000000 <compute>:
   0: f3 0f 1e fa           endbr64
   4: 55                    push   rbp
   5: 48 89 e5              mov    rbp,rsp
   8: b8 01 00 00 00        mov    eax,0x1
   d: 5d                    pop    rbp
   e: c3                    ret

000000000000000f <main>:
   f: f3 0f 1e fa           endbr64
  13: 55                    push   rbp
  14: 48 89 e5              mov    rbp,rsp
  17: b8 00 00 00 00        mov    eax,0x0
  1c: e8 df ff ff ff        call   0 <compute>
  21: 5d                    pop    rbp
  22: c3                    ret
```

Phew!

MORE FILES!

main.c

```c
#include "library.h"

int main()
{
    return compute();
}
```

library.c

```c
int compute()
{
    return 1;
}
```

library.h

```c
int compute();
```

main.c

```c
#include "library.h"

int main()
{
    return compute();
}
```

library.c

```c
int compute()
{
    return 1;
}
```

library.h

```c
int compute();
```

```
gcc -c library.c
objdump -Dr -Mintel library.o
```

main.c

```c
#include "library.h"

int main()
{
    return compute();
}
```

library.c

```c
int compute()
{
    return 1;
}
```

library.h

```c
int compute();
```

```
gcc -c library.c
objdump -Dr -Mintel library.o
```

```
0000000000000000 <compute>:
   0: f3 0f 1e fa           endbr64
   4: 55                    push   rbp
   5: 48 89 e5              mov    rbp,rsp
   8: b8 01 00 00 00        mov    eax,0x1
   d: 5d                    pop    rbp
   e: c3                    ret
```

main.c

```c
#include "library.h"

int main()
{
    return compute();
}
```

library.c

```c
int compute()
{
    return 1;
}
```

library.h

```c
int compute();
```

```
gcc -c library.c
objdump -Dr -Mintel library.o
```

```
0000000000000000 <compute>:
   0: f3 0f 1e fa           endbr64
   4: 55                    push   rbp
   5: 48 89 e5              mov    rbp,rsp
   8: b8 01 00 00 00        mov    eax,0x1
   d: 5d                    pop    rbp
   e: c3                    ret
```

main.c

```c
#include "library.h"

int main()
{
    return compute();
}
```

library.c

```c
int compute()
{
    return 1;
}
```

library.h

```c
int compute();
```

```
gcc -c library.c
objdump -Dr -Mintel library.o
```

```
0000000000000000 <compute>:
   0: f3 0f 1e fa           endbr64
   4: 55                    push   rbp
   5: 48 89 e5              mov    rbp,rsp
   8: b8 01 00 00 00        mov    eax,0x1
   d: 5d                    pop    rbp
   e: c3                    ret
```

```
gcc -c main.c
objdump -Dr -Mintel main.o
```

main.c

```c
#include "library.h"

int main()
{
    return compute();
}
```

library.c

```c
int compute()
{
    return 1;
}
```

library.h

```c
int compute();
```

```
gcc -c library.c
objdump -Dr -Mintel library.o
```

```
0000000000000000 <compute>:
   0: f3 0f 1e fa           endbr64
   4: 55                    push   rbp
   5: 48 89 e5              mov    rbp,rsp
   8: b8 01 00 00 00        mov    eax,0x1
   d: 5d                    pop    rbp
   e: c3                    ret
```

```
gcc -c main.c
objdump -Dr -Mintel main.o
```

```
0000000000000000 <main>:
   0: f3 0f 1e fa           endbr64
   4: 55                    push   rbp
   5: 48 89 e5              mov    rbp,rsp
   8: b8 00 00 00 00        mov    eax,0x0
   d: e8 00 00 00 00        call   12 <main+0x12>
            e: R_X86_64_PLT32 compute-0x4
  12: 5d                    pop    rbp
  13: c3                    ret
```

main.c

```c
#include "library.h"

int main()
{
    return compute();
}
```

library.c

```c
int compute()
{
    return 1;
}
```

library.h

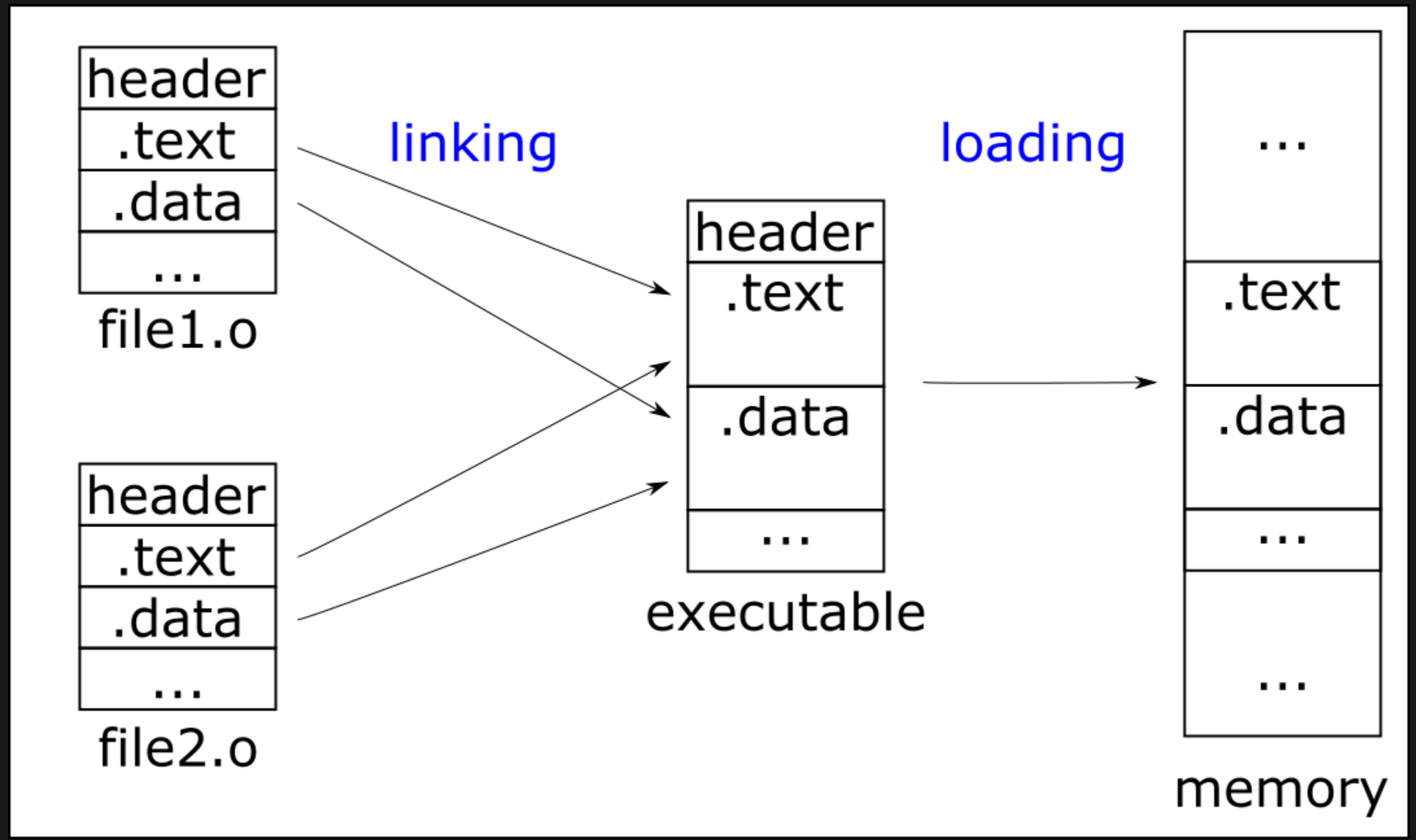
```c
int compute();
```

```
gcc -c library.c
objdump -Dr -Mintel library.o
```

```
0000000000000000 <compute>:
   0: f3 0f 1e fa           endbr64
   4: 55                    push   rbp
   5: 48 89 e5              mov    rbp,rsp
   8: b8 01 00 00 00        mov    eax,0x1
   d: 5d                    pop    rbp
   e: c3                    ret
```

```
gcc -c main.c
objdump -Dr -Mintel main.o
```

```
0000000000000000 <main>:
   0: f3 0f 1e fa           endbr64
   4: 55                    push   rbp
   5: 48 89 e5              mov    rbp,rsp
   8: b8 00 00 00 00        mov    eax,0x0
   d: e8 00 00 00 00        call   12 <main+0x12>
            e: R_X86_64_PLT32 compute-0x4
  12: 5d                    pop    rbp
  13: c3                    ret
```


header
.text
.data
...
file1.o
header
.text
.data
...
file2.o
linking
header
.text
.data
...
executable
loading
...
.text
.data
...
...
memory

```
gcc -o main main.o library.o
objdump -rD -Mintel main
```

```
gcc -o main main.o library.o
objdump -rD -Mintel main
```

```
 1 0000000000001129 <main>:
 2     1129: f3 0f 1e fa           endbr64
 3     112d: 55                    push   rbp
 4     112e: 48 89 e5              mov    rbp,rsp
 5     1131: b8 00 00 00 00        mov    eax,0x0
 6     1136: e8 02 00 00 00        call   113d <compute>
 7     113b: 5d                    pop    rbp
 8     113c: c3                    ret
 9
10 000000000000113d <compute>:
11     113d: f3 0f 1e fa           endbr64
12     1141: 55                    push   rbp
13     1142: 48 89 e5              mov    rbp,rsp
14     1145: b8 01 00 00 00        mov    eax,0x1
15     114a: 5d                    pop    rbp
16     114b: c3                    ret
17     114c: 0f 1f 40 00           nop    DWORD PTR [rax+0x0]
```

```
gcc -o main main.o library.o
objdump -rD -Mintel main
```

```
 1 0000000000001129 <main>:
 2     1129: f3 0f 1e fa           endbr64
 3     112d: 55                    push   rbp
 4     112e: 48 89 e5              mov    rbp,rsp
 5     1131: b8 00 00 00 00        mov    eax,0x0
 6     1136: e8 02 00 00 00        call   113d <compute>
 7     113b: 5d                    pop    rbp
 8     113c: c3                    ret
 9
10 000000000000113d <compute>:
11     113d: f3 0f 1e fa           endbr64
12     1141: 55                    push   rbp
13     1142: 48 89 e5              mov    rbp,rsp
14     1145: b8 01 00 00 00        mov    eax,0x1
15     114a: 5d                    pop    rbp
16     114b: c3                    ret
17     114c: 0f 1f 40 00           nop    DWORD PTR [rax+0x0]
```

DATA

```
int params(int a, int b)
{
    return a+b;
}
```

```c
int params(int a, int b)
{
    return a+b;
}
```

```
gcc -c params.c
objdump -wDr -Mintel params.o
```

```c
int params(int a, int b)
{
    return a+b;
}
```

```
gcc -c params.c
objdump -wDr -Mintel params.o
```

```
1 push   rbp
2 mov    rbp,rsp
3 mov    DWORD PTR [rbp-0x4],edi
4 mov    DWORD PTR [rbp-0x8],esi
5 mov    edx,DWORD PTR [rbp-0x4]
6 mov    eax,DWORD PTR [rbp-0x8]
7 add    eax,edx
8 pop    rbp
9 ret
```

```c
int params(int a, int b)
{
    return a+b;
}
```

```
gcc -c params.c
objdump -wDr -Mintel params.o
```

```
1 push   rbp
2 mov    rbp,rsp
3 mov    DWORD PTR [rbp-0x4],edi
4 mov    DWORD PTR [rbp-0x8],esi
5 mov    edx,DWORD PTR [rbp-0x4]
6 mov    eax,DWORD PTR [rbp-0x8]
7 add    eax,edx
8 pop    rbp
9 ret
```

```c
int params(int a, int b)
{
    return a+b;
}
```

```
gcc -c params.c
objdump -wDr -Mintel params.o
```

```
1 push   rbp
2 mov    rbp,rsp
3 mov    DWORD PTR [rbp-0x4],edi
4 mov    DWORD PTR [rbp-0x8],esi
5 mov    edx,DWORD PTR [rbp-0x4]
6 mov    eax,DWORD PTR [rbp-0x8]
7 add    eax,edx
8 pop    rbp
9 ret
```

```c
int params(int a, int b)
{
    return a+b;
}
```

```
gcc -c params.c
objdump -wDr -Mintel params.o
```

```
1 push   rbp
2 mov    rbp,rsp
3 mov    DWORD PTR [rbp-0x4],edi
4 mov    DWORD PTR [rbp-0x8],esi
5 mov    edx,DWORD PTR [rbp-0x4]
6 mov    eax,DWORD PTR [rbp-0x8]
7 add    eax,edx
8 pop    rbp
9 ret
```

```
int f(int a, struct BigThing b)
{
    return a + b.i;
}
```

```c
int f(int a, struct BigThing b)
{
    return a + b.i;
}
```

```
gcc -c big_args.c
objdump -wD -Mintel big_args.o
```

```c
int f(int a, struct BigThing b)
{
    return a + b.i;
}
```

```
gcc -c big_args.c
objdump -wD -Mintel big_args.o
```

```
1 push   rbp
2 mov    rbp,rsp
3 mov    DWORD PTR [rbp-0x4],edi
4 mov    edx,DWORD PTR [rbp+0x30]
5 mov    eax,DWORD PTR [rbp-0x4]
6 add    eax,edx
7 pop    rbp
8 ret
```

```c
int f(int a, struct BigThing b)
{
    return a + b.i;
}
```

```
gcc -c big_args.c
objdump -wD -Mintel big_args.o
```

```
1 push   rbp
2 mov    rbp,rsp
3 mov    DWORD PTR [rbp-0x4],edi
4 mov    edx,DWORD PTR [rbp+0x30]
5 mov    eax,DWORD PTR [rbp-0x4]
6 add    eax,edx
7 pop    rbp
8 ret
```

```
int local()
{
    int result = 1;
    return result;
}
```

```
int local()
{
    int result = 1;
    return result;
}
```

```
gcc -c local.cpp
objdump -wD -Mintel local.o
```

```cpp
int local()
{
    int result = 1;
    return result;
}
```

```
gcc -c local.cpp
objdump -wD -Mintel local.o
```

```
1 push   rbp
2 mov    rbp,rsp
3 mov    DWORD PTR [rbp-0x4],0x1
4 mov    eax,DWORD PTR [rbp-0x4]
5 pop    rbp
6 ret
```

```cpp
int local()
{
    int result = 1;
    return result;
}
```

```
gcc -c local.cpp
objdump -wD -Mintel local.o
```

```
1 push   rbp
2 mov    rbp,rsp
3 mov    DWORD PTR [rbp-0x4],0x1
4 mov    eax,DWORD PTR [rbp-0x4]
5 pop    rbp
6 ret
```

NO

```c
int global = 2;

int main()
{
    return global;
}
```

```c
int global = 2;

int main()
{
    return global;
}
```

```
gcc -c main.c
objdump -Dr -Mintel main.o
```

```c
int global = 2;

int main()
{
    return global;
}
```

```
gcc -c main.c
objdump -Dr -Mintel main.o
```

```
1  Disassembly of section .text:
2
3  0000000000000000 <main>:
4   0: f3 0f 1e fa          endbr64
5   4: 55                   push rbp
6   5: 48 89 e5             mov  rbp,rsp
7   8: 8b 05 00 00 00 00    mov  eax,DWORD PTR [rip+0x0] # e <main+0xe>
8            a: R_X86_64_PC32  global-0x4
9   e: 5d                   pop  rbp
10  f: c3                   ret
11
12 Disassembly of section .data:
13
14 0000000000000000 <global>:
15  0: 02 00                add  al,BYTE PTR [rax]
16   ...
```

```c
int global = 2;

int main()
{
    return global;
}
```

```
gcc -c main.c
objdump -Dr -Mintel main.o
```

```
Disassembly of section .text:

0000000000000000 <main>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   rbp
   5:   48 89 e5                mov    rbp,rsp
   8:   8b 05 00 00 00 00       mov    eax,DWORD PTR [rip+0x0] # e <main+0xe>
                        a: R_X86_64_PC32        global-0x4
   e:   5d                      pop    rbp
   f:   c3                      ret

Disassembly of section .data:

0000000000000000 <global>:
   0:   02 00                   add    al,BYTE PTR [rax]
   ...
```

```c
int global = 2;

int main()
{
    return global;
}
```

```
gcc -c main.c
objdump -Dr -Mintel main.o
```

```
Disassembly of section .text:

0000000000000000 <main>:
   0: f3 0f 1e fa           endbr64
   4: 55                    push   rbp
   5: 48 89 e5              mov    rbp,rsp
   8: 8b 05 00 00 00 00     mov    eax,DWORD PTR [rip+0x0] # e <main+0xe>
                    a: R_X86_64_PC32    global-0x4
   e: 5d                    pop    rbp
   f: c3                    ret

Disassembly of section .data:

0000000000000000 <global>:
   0: 02 00                 add    al,BYTE PTR [rax]
     ...
```

```c
int global = 2;

int main()
{
    return global;
}
```

```
gcc -c main.c
objdump -Dr -Mintel main.o
```

```
Disassembly of section .text:

0000000000000000 <main>:
   0: f3 0f 1e fa           endbr64
   4: 55                    push   rbp
   5: 48 89 e5              mov    rbp,rsp
   8: 8b 05 00 00 00 00     mov    eax,DWORD PTR [rip+0x0] # e <main+0xe>
                    a: R_X86_64_PC32    global-0x4
   e: 5d                    pop    rbp
   f: c3                    ret

Disassembly of section .data:

0000000000000000 <global>:
   0: 02 00                 add    al,BYTE PTR [rax]
     ...
```

```c
int global = 2;

int main()
{
    return global;
}
```

```
gcc -c main.c
objdump -Dr -Mintel main.o
```

```
Disassembly of section .text:

0000000000000000 <main>:
   0: f3 0f 1e fa           endbr64
   4: 55                    push   rbp
   5: 48 89 e5              mov    rbp,rsp
   8: 8b 05 00 00 00 00     mov    eax,DWORD PTR [rip+0x0] # e <main+0xe>
                    a: R_X86_64_PC32    global-0x4
   e: 5d                    pop    rbp
   f: c3                    ret

Disassembly of section .data:

0000000000000000 <global>:
   0: 02 00                 add    al,BYTE PTR [rax]
     ...
```

```c
int global = 2;

int main()
{
    return global;
}
```

```
gcc -o main main.o
objdump -Dr -Mintel main
```

```c
int global = 2;

int main()
{
    return global;
}
```

```
gcc -o main main.o
objdump -Dr -Mintel main
```

```
Disassembly of section .text:
(...)
0000000000001129 <main>:
  1129: f3 0f 1e fa           endbr64
  112d: 55                    push rbp
  112e: 48 89 e5              mov  rbp,rsp
  1131: 8b 05 d9 2e 00 00     mov  eax,DWORD PTR [rip+0x2ed9] # 4010 <global>
  1137: 5d                    pop  rbp
  1138: c3                    ret
  1139: 0f 1f 80 00 00 00 00  nop  DWORD PTR [rax+0x0]
(...)

Disassembly of section .data:
(...)
0000000000004010 <global>:
  4010: 02 00                 add    al,BYTE PTR [rax]
```

```c
int global = 2;

int main()
{
    return global;
}
```

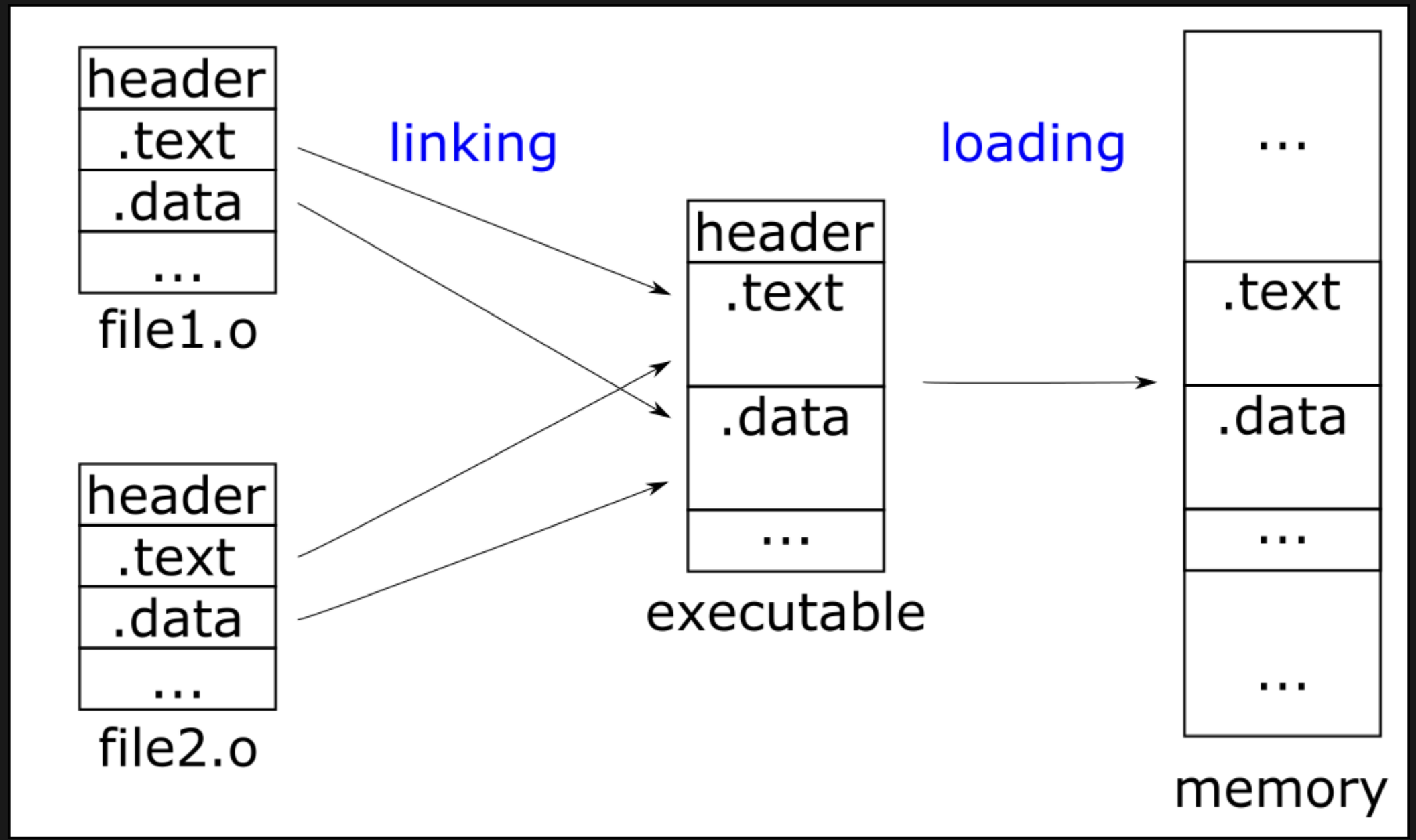
```
gcc -o main main.o
objdump -Dr -Mintel main
```

```
Disassembly of section .text:
(...)
0000000000001129 <main>:
  1129: f3 0f 1e fa           endbr64
  112d: 55                    push rbp
  112e: 48 89 e5              mov  rbp,rsp
  1131: 8b 05 d9 2e 00 00     mov  eax,DWORD PTR [rip+0x2ed9] # 4010 <global>
  1137: 5d                    pop  rbp
  1138: c3                    ret
  1139: 0f 1f 80 00 00 00 00  nop  DWORD PTR [rax+0x0]
(...)

Disassembly of section .data:
(...)
0000000000004010 <global>:
  4010: 02 00                 add    al,BYTE PTR [rax]
```

```c
int global = 2;

int main()
{
    return global;
}
```

```
gcc -o main main.o
objdump -Dr -Mintel main
```

```
Disassembly of section .text:
(...)
0000000000001129 <main>:
  1129: f3 0f 1e fa           endbr64
  112d: 55                    push rbp
  112e: 48 89 e5              mov  rbp,rsp
  1131: 8b 05 d9 2e 00 00     mov  eax,DWORD PTR [rip+0x2ed9] # 4010 <global>
  1137: 5d                    pop  rbp
  1138: c3                    ret
  1139: 0f 1f 80 00 00 00 00  nop  DWORD PTR [rax+0x0]
(...)

Disassembly of section .data:
(...)
0000000000004010 <global>:
  4010: 02 00                 add    al,BYTE PTR [rax]
```


header
.text
.data
...
file1.o
header
.text
.data
...
file2.o
linking
header
.text
.data
...
executable
loading
...
.text
.data
...
...
memory

```c
int global = 1;
static int internal = 2;

int unintialized;
int zero = 0;

const int const_ = 3;
const int const_uninitialized;

extern int extern_;
extern const int extern_const;

int main() {
  return global
    + internal
    + unintialized
    + zero
    + const_
    + const_uninitialized
    + extern_
    + extern_const
;}
```

```c
int global = 1;
static int internal = 2;

int unintialized;
int zero = 0;

const int const_ = 3;
const int const_uninitialized;

extern int extern_;
extern const int extern_const;

int main() {
  return global
    + internal
    + unintialized
    + zero
    + const_
    + const_uninitialized
    + extern_
    + extern_const
;}
```

```c
int global = 1;
static int internal = 2;

int unintialized;
int zero = 0;

const int const_ = 3;
const int const_uninitialized;

extern int extern_;
extern const int extern_const;

int main() {
  return global
    + internal
    + unintialized
    + zero
    + const_
    + const_uninitialized
    + extern_
    + extern_const
;}
```

```c
int global = 1;
static int internal = 2;

int unintialized;
int zero = 0;

const int const_ = 3;
const int const_uninitialized;

extern int extern_;
extern const int extern_const;

int main() {
  return global
    + internal
    + unintialized
    + zero
    + const_
    + const_uninitialized
    + extern_
    + extern_const
;}
```

```c
int global = 1;
static int internal = 2;

int unintialized;
int zero = 0;

const int const_ = 3;
const int const_uninitialized;

extern int extern_;
extern const int extern_const;

int main() {
  return global
    + internal
    + unintialized
    + zero
    + const_
    + const_uninitialized
    + extern_
    + extern_const
;}
```

```c
int global = 1;
static int internal = 2;

int unintialized;
int zero = 0;

const int const_ = 3;
const int const_uninitialized;

extern int extern_;
extern const int extern_const;

int main() {
  return global
    + internal
    + unintialized
    + zero
    + const_
    + const_uninitialized
    + extern_
    + extern_const
;}
```

```c
int global = 1;
static int internal = 2;

int unintialized;
int zero = 0;

const int const_ = 3;
const int const_uninitialized;

extern int extern_;
extern const int extern_const;

int main() {
  return global
    + internal
    + unintialized
    + zero
    + const_
    + const_uninitialized
    + extern_
    + extern_const
;}
```

```asm
endbr64
push   rbp
mov    rbp,rsp
mov    edx,DWORD PTR [rip+0x0]
mov    eax,DWORD PTR [rip+0x0]
add    edx,eax
mov    eax,DWORD PTR [rip+0x0]
add    edx,eax
mov    eax,DWORD PTR [rip+0x0]
add    eax,edx
mov    edx,0x3
add    eax,edx
mov    edx,0x0
add    edx,eax
mov    eax,DWORD PTR [rip+0x0]
add    edx,eax
mov    eax,DWORD PTR [rip+0x0]
add    eax,edx
pop    rbp
ret
```

```c
int global = 1;
static int internal = 2;

int unintialized;
int zero = 0;

const int const_ = 3;
const int const_uninitialized;

extern int extern_;
extern const int extern_const;

int main() {
  return global
    + internal
    + unintialized
    + zero
    + const_
    + const_uninitialized
    + extern_
    + extern_const
;}
```

```asm
endbr64
push   rbp
mov    rbp,rsp
mov    edx,DWORD PTR [rip+0x0]
mov    eax,DWORD PTR [rip+0x0]
add    edx,eax
mov    eax,DWORD PTR [rip+0x0]
add    edx,eax
mov    eax,DWORD PTR [rip+0x0]
add    eax,edx
mov    edx,0x3
add    eax,edx
mov    edx,0x0
add    edx,eax
mov    eax,DWORD PTR [rip+0x0]
add    edx,eax
mov    eax,DWORD PTR [rip+0x0]
add    eax,edx
pop    rbp
ret
```

```c
int global = 1;            //.data
static int internal = 2;   //.data

int unintialized;          //.bss
int zero = 0;              //.bss

const int const_ = 3;      //.rodata
const int const_uninitialized; //.rodat

extern int extern_;
extern const int extern_const;

int main() {
  return global
    + internal
    + unintialized
    + zero
    + const_
    + const_uninitialized
    + extern_
    + extern_const
;}
```

```
Disassembly of section .data:
0000000000000000 <global>:
   0: 01 00    add    DWORD PTR [rax]
    ...
0000000000000004 <internal>:
   4: 02 00    add    al,BYTE PTR [ra
    ...
Disassembly of section .bss:
0000000000000000 <unintialized>:
   0: 00 00    add    BYTE PTR [rax],
    ...
0000000000000004 <zero>:
   4: 00 00    add    BYTE PTR [rax],
    ...
Disassembly of section .rodata:
0000000000000000 <const_>:
   0: 03 00    add    eax,DWORD PTR [
    ...
0000000000000004 <const_uninitialized
   4: 00 00    add    BYTE PTR [rax],
    ...
```

```c
int global = 1;            //.data
static int internal = 2;   //.data

int unintialized;          //.bss
int zero = 0;              //.bss

const int const_ = 3;      //.rodata
const int const_uninitialized; //.rodat

extern int extern_;
extern const int extern_const;

int main() {
  return global
    + internal
    + unintialized
    + zero
    + const_
    + const_uninitialized
    + extern_
    + extern_const
;}
```

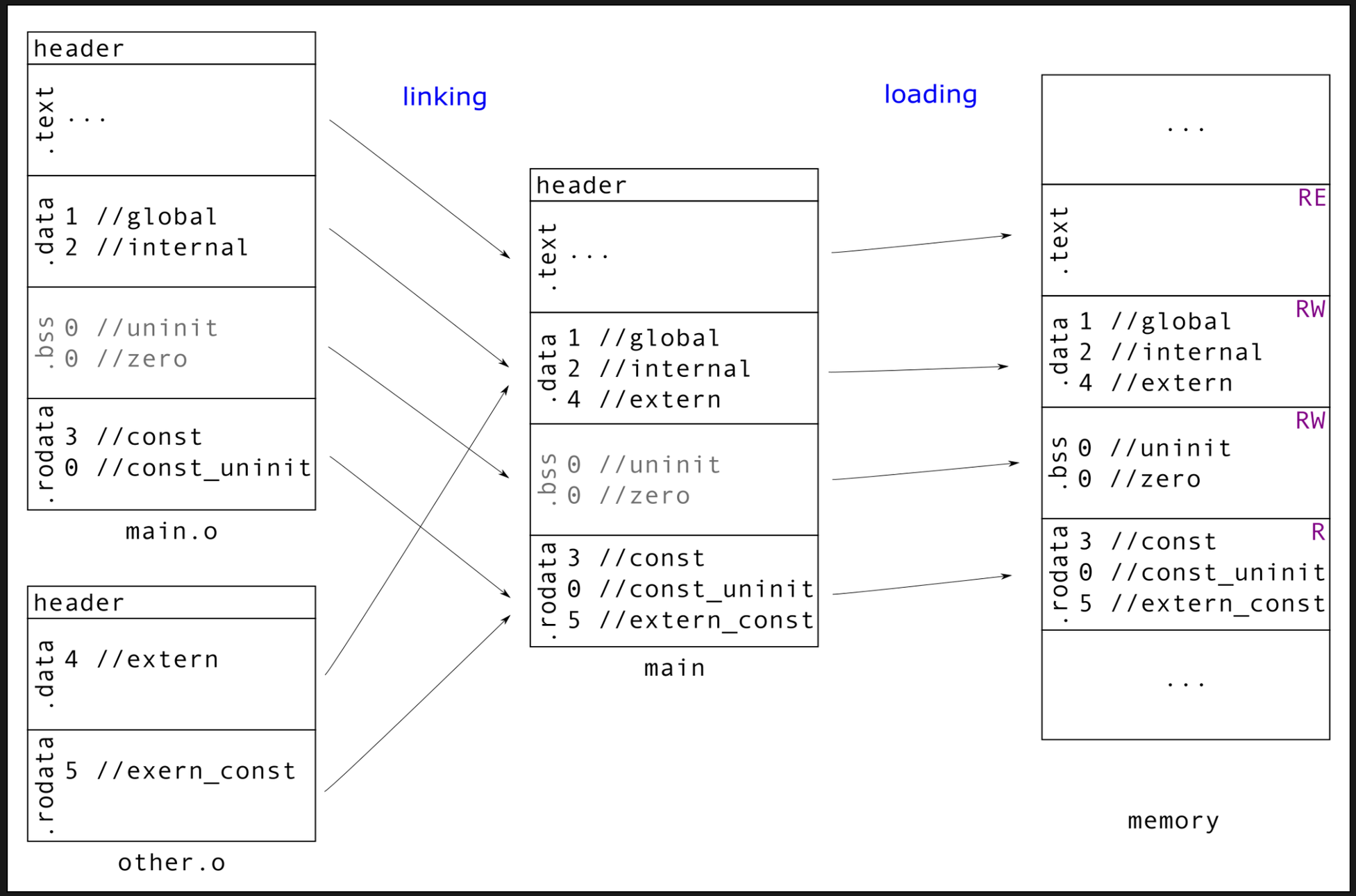
```
Disassembly of section .data:
0000000000000000 <global>:
   0: 01 00    add    DWORD PTR [rax]
    ...
0000000000000004 <internal>:
   4: 02 00    add    al,BYTE PTR [ra
    ...
Disassembly of section .bss:
0000000000000000 <unintialized>:
   0: 00 00    add    BYTE PTR [rax],
    ...
0000000000000004 <zero>:
   4: 00 00    add    BYTE PTR [rax],
    ...
Disassembly of section .rodata:
0000000000000000 <const_>:
   0: 03 00    add    eax,DWORD PTR [
    ...
0000000000000004 <const_uninitialized
   4: 00 00    add    BYTE PTR [rax],
    ...
```

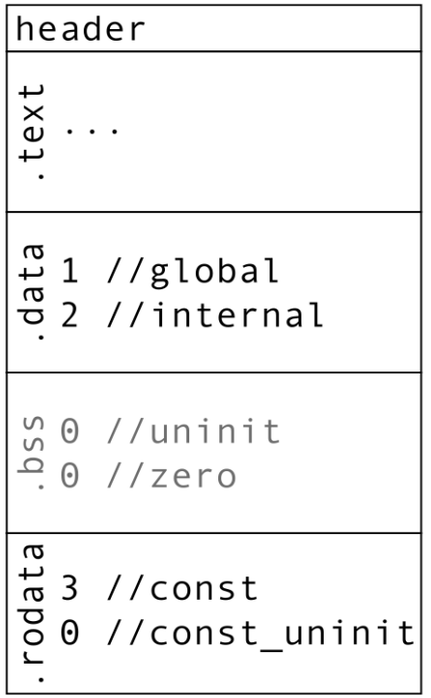
```c
int global = 1;            //.data
static int internal = 2;   //.data

int unintialized;          //.bss
int zero = 0;              //.bss

const int const_ = 3;      //.rodata
const int const_uninitialized; //.rodat

extern int extern_;
extern const int extern_const;

int main() {
  return global
    + internal
    + unintialized
    + zero
    + const_
    + const_uninitialized
    + extern_
    + extern_const
;}
```

```
Disassembly of section .data:
0000000000000000 <global>:
   0: 01 00    add    DWORD PTR [rax]
    ...
0000000000000004 <internal>:
   4: 02 00    add    al,BYTE PTR [ra
    ...
Disassembly of section .bss:
0000000000000000 <unintialized>:
   0: 00 00    add    BYTE PTR [rax],
    ...
0000000000000004 <zero>:
   4: 00 00    add    BYTE PTR [rax],
    ...
Disassembly of section .rodata:
0000000000000000 <const_>:
   0: 03 00    add    eax,DWORD PTR [
    ...
0000000000000004 <const_uninitialized
   4: 00 00    add    BYTE PTR [rax],
    ...
```

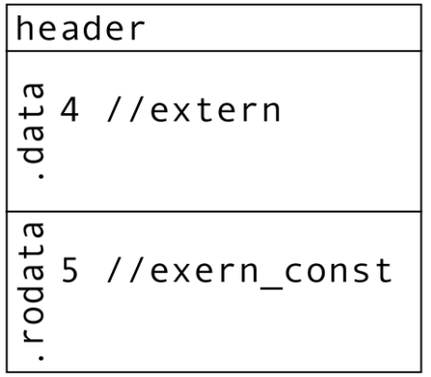
```c
int global = 1;            //.data
static int internal = 2;   //.data

int unintialized;          //.bss
int zero = 0;              //.bss

const int const_ = 3;      //.rodata
const int const_uninitialized; //.rodat

extern int extern_;
extern const int extern_const;

int main() {
  return global
    + internal
    + unintialized
    + zero
    + const_
    + const_uninitialized
    + extern_
    + extern_const
;}
```

```
Disassembly of section .data:
0000000000000000 <global>:
   0: 01 00    add    DWORD PTR [rax]
    ...
0000000000000004 <internal>:
   4: 02 00    add    al,BYTE PTR [ra
    ...
Disassembly of section .bss:
0000000000000000 <unintialized>:
   0: 00 00    add    BYTE PTR [rax],
    ...
0000000000000004 <zero>:
   4: 00 00    add    BYTE PTR [rax],
    ...
Disassembly of section .rodata:
0000000000000000 <const_>:
   0: 03 00    add    eax,DWORD PTR [
    ...
0000000000000004 <const_uninitialized
   4: 00 00    add    BYTE PTR [rax],
    ...
```

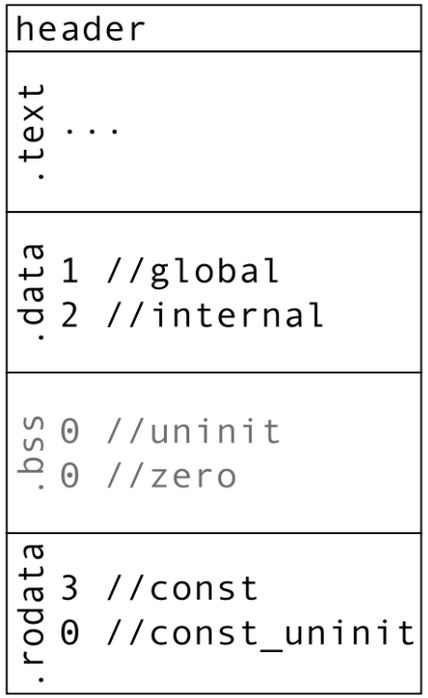

header
.text ...
.data 1 //global
2 //internal
.bss 0 //uninit
0 //zero
.rodata 3 //const
0 //const_uninit
main.o
header
.data 4 //extern
.rodata 5 //exern_const
other.o
linking
header
.text ...
.data 1 //global
2 //internal
4 //extern
.bss 0 //uninit
0 //zero
.rodata 3 //const
0 //const_uninit
5 //extern_const
main
loading
...
RE
.text
RW
.data 1 //global
2 //internal
4 //extern
RW
.bss 0 //uninit
0 //zero
R
.rodata 3 //const
0 //const_uninit
5 //extern_const
...
memory



main.o

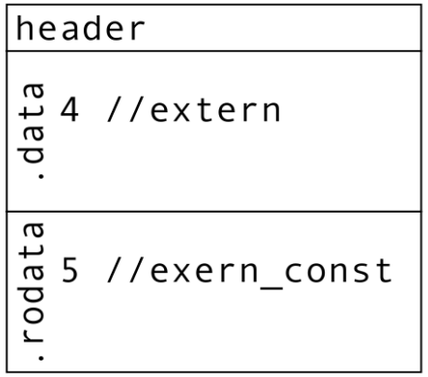


other.o



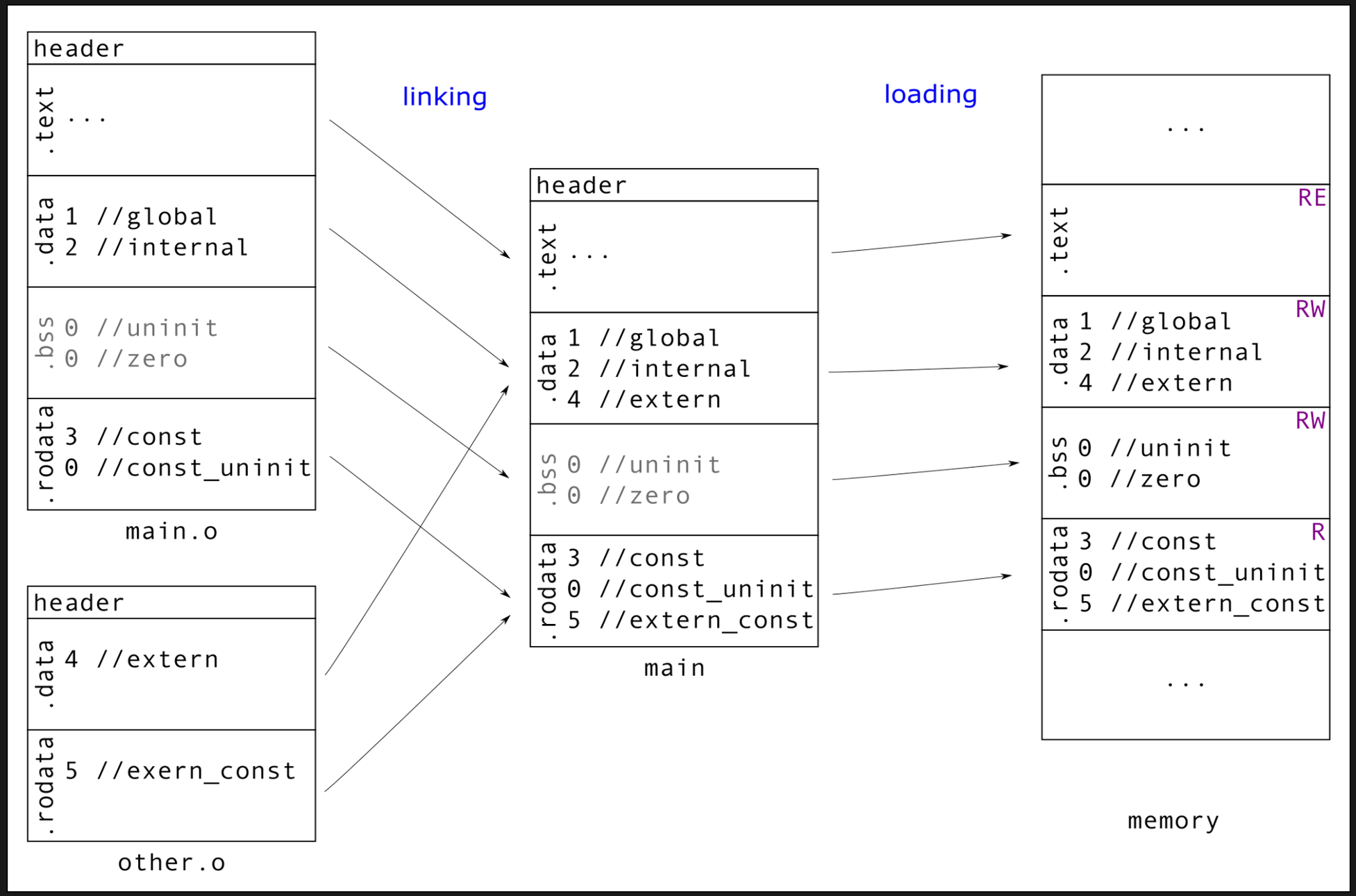
main.o

```c
//main.c
int global = 1;
static int internal = 2;

int unint;
int zero = 0;

const int const_ = 3;
const int const_uninit;

extern int extern_;
extern const int extern_cons
```



other.o

```c
//other.c
int extern_= 4;
const int extern_const = 5;
```


header
.text ...
.data 1 //global
2 //internal
.bss 0 //uninit
0 //zero
.rodata 3 //const
0 //const_uninit
main.o
header
.data 4 //extern
.rodata 5 //exern_const
other.o
linking
header
.text ...
.data 1 //global
2 //internal
4 //extern
.bss 0 //uninit
0 //zero
.rodata 3 //const
0 //const_uninit
5 //extern_const
main
loading
...
RE
.text
RW
.data 1 //global
2 //internal
4 //extern
RW
.bss 0 //uninit
0 //zero
R
.rodata 3 //const
0 //const_uninit
5 //extern_const
...
memory

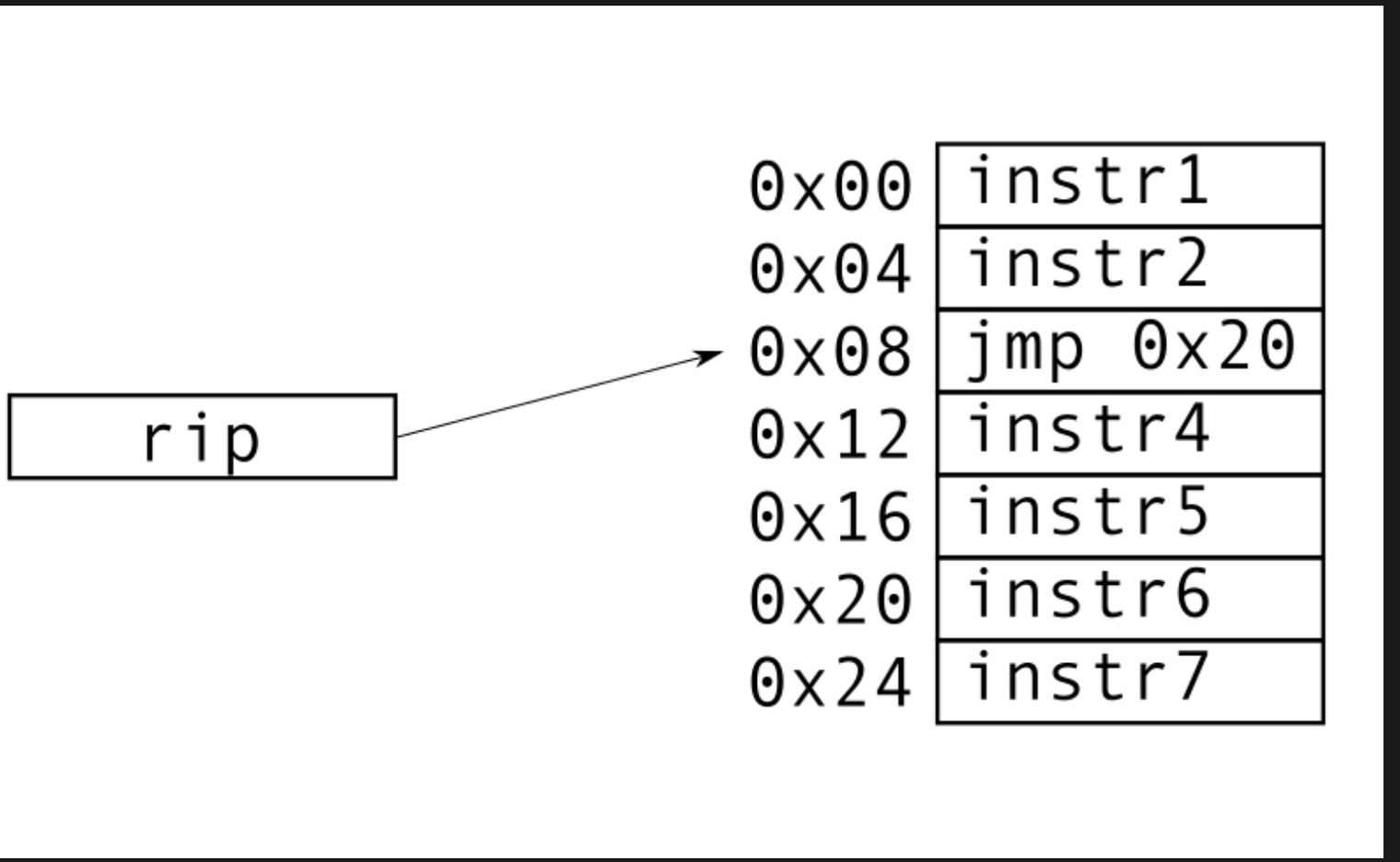
DYNAMIC LINKING

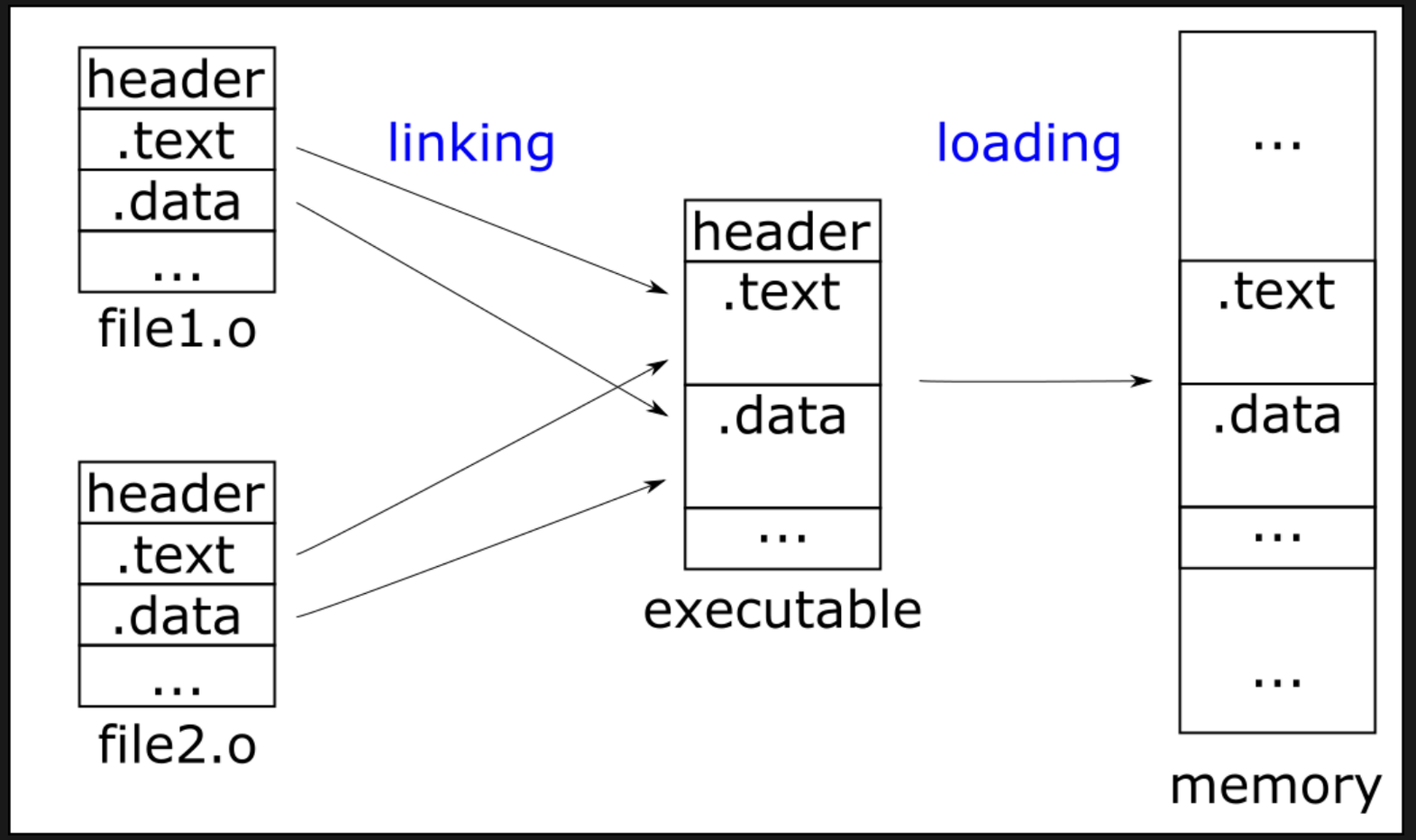
DYNAMIC LINKING

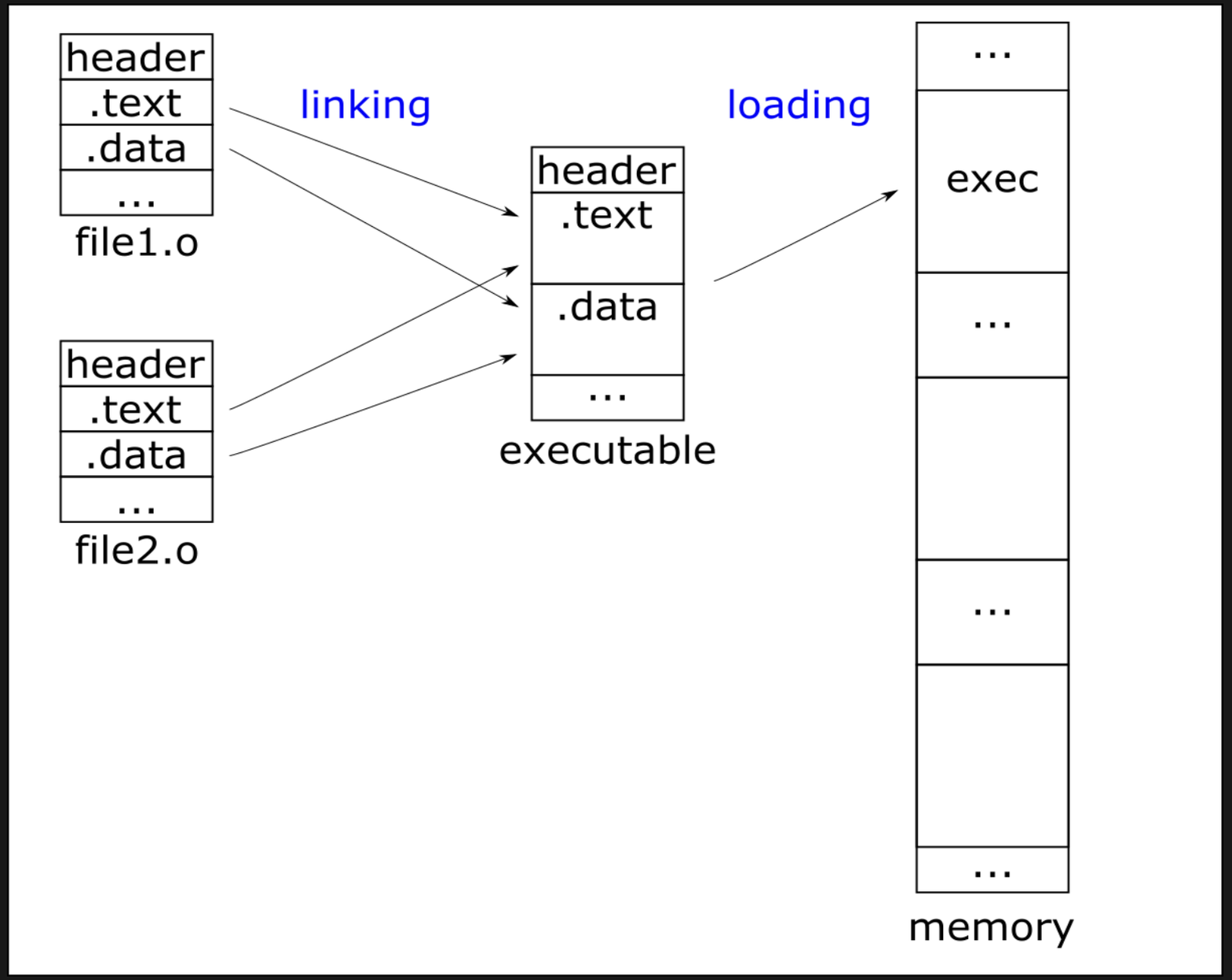
- Difference: What is known when (compiler->linker->loader)

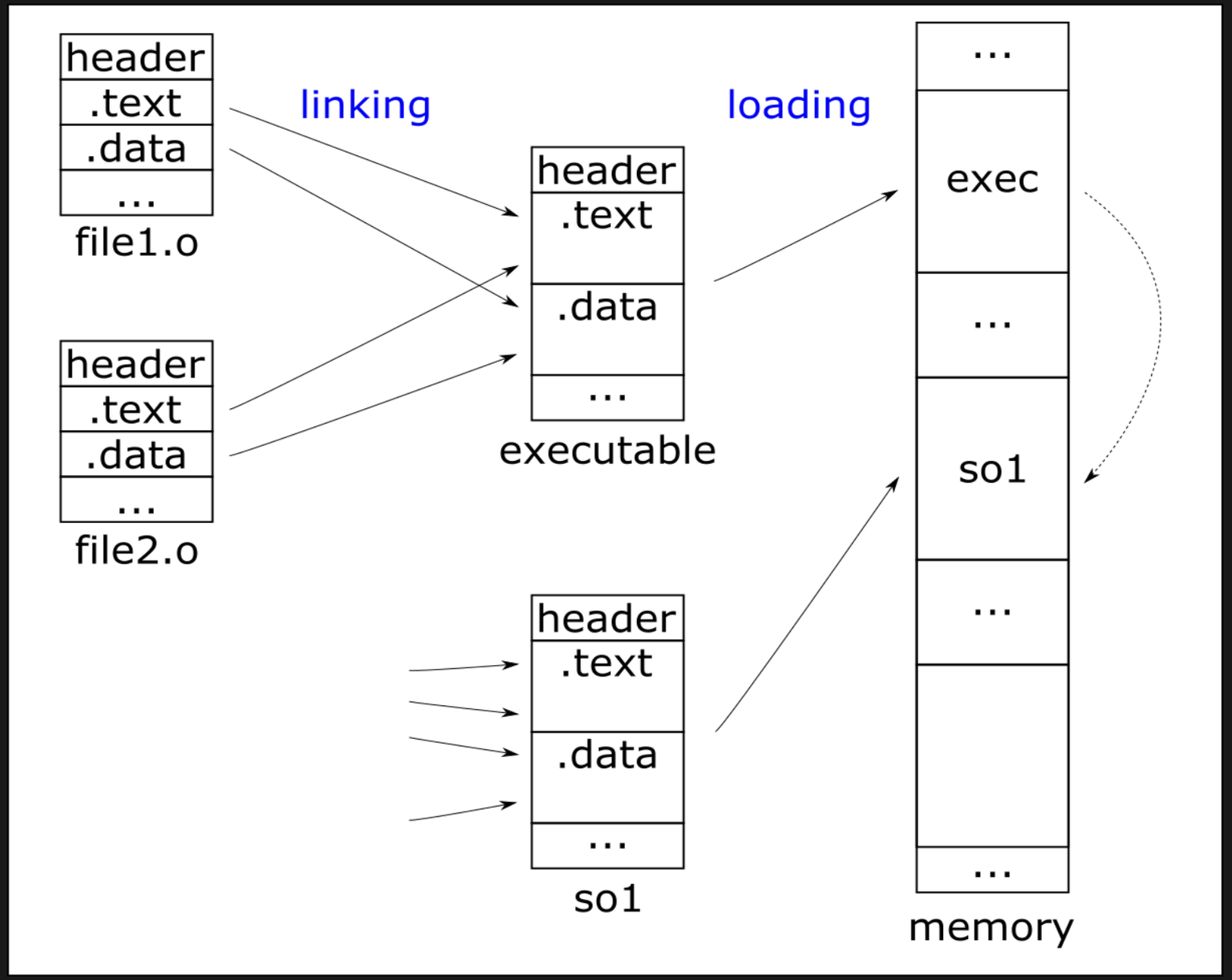
DYNAMIC LINKING

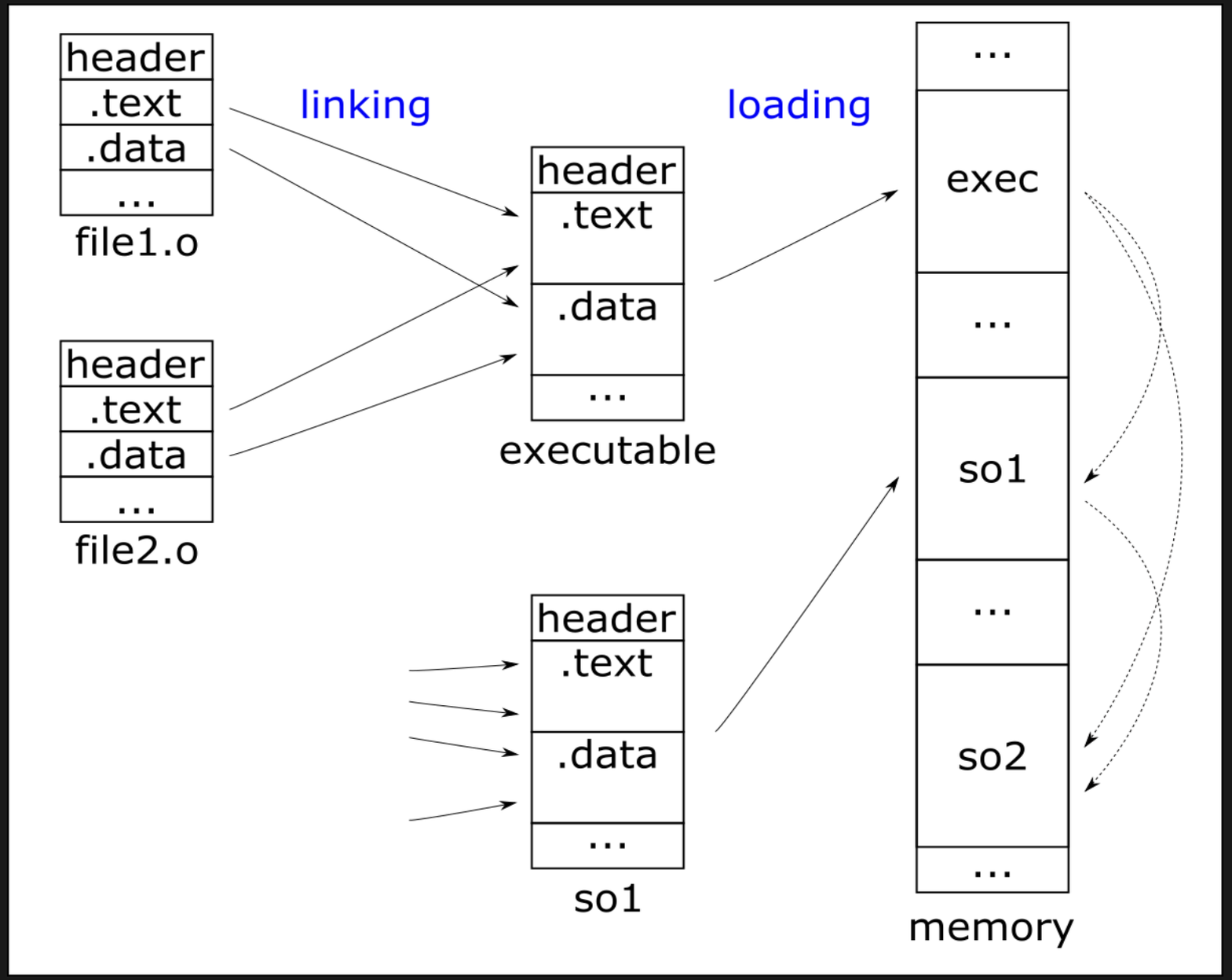
- Difference: What is known when (compiler->linker->loader)
- What can be done at those points


rip
0x00
instr1
0x04
instr2
0x08
jmp 0x20
0x12
instr4
0x16
instr5
0x20
instr6
0x24
instr7


header
.text
.data
...
file1.o
header
.text
.data
...
file2.o
linking
header
.text
.data
...
executable
loading
...
.text
.data
...
...
memory


header
.text
.data
...
file1.o
header
.text
.data
...
file2.o
linking
header
.text
.data
...
executable
loading
...
exec
...
...
...
memory


header
.text
.data
...
file1.o
header
.text
.data
...
file2.o
linking
header
.text
.data
...
executable
loading
header
.text
.data
...
so1
...
exec
...
so1
...
...
memory


header
.text
.data
...
file1.o
header
.text
.data
...
file2.o
linking
header
.text
.data
...
executable
loading
header
.text
.data
...
so1
...
exec
...
so1
...
so2
...
memory

SHARED OBJECTS REALLY ARE VERY SHARED

SHARED OBJECTS REALLY ARE VERY SHARED

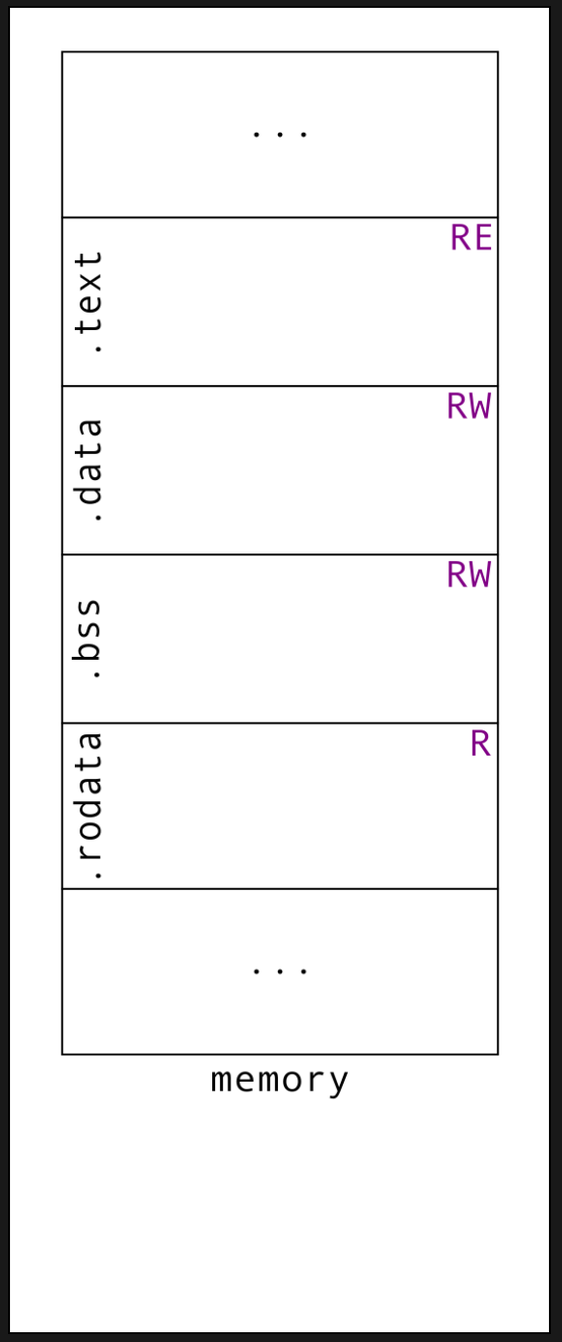
- `.text` mapped directly into memory

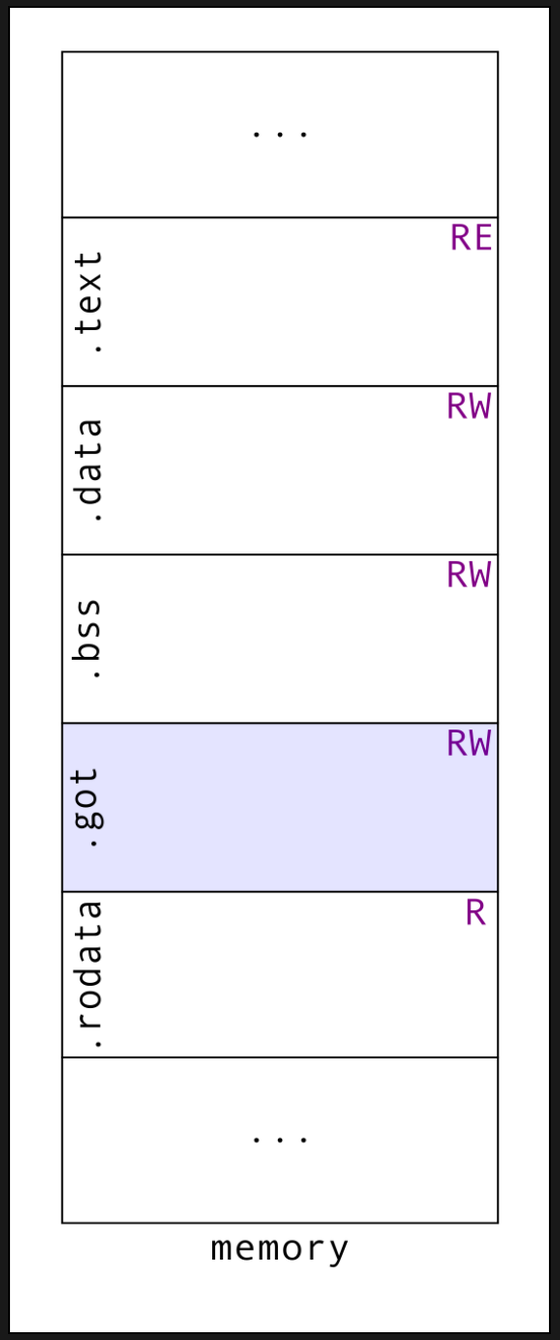
SHARED OBJECTS REALLY ARE VERY SHARED

- `.text` mapped directly into memory
- Many processes share the same `.text`

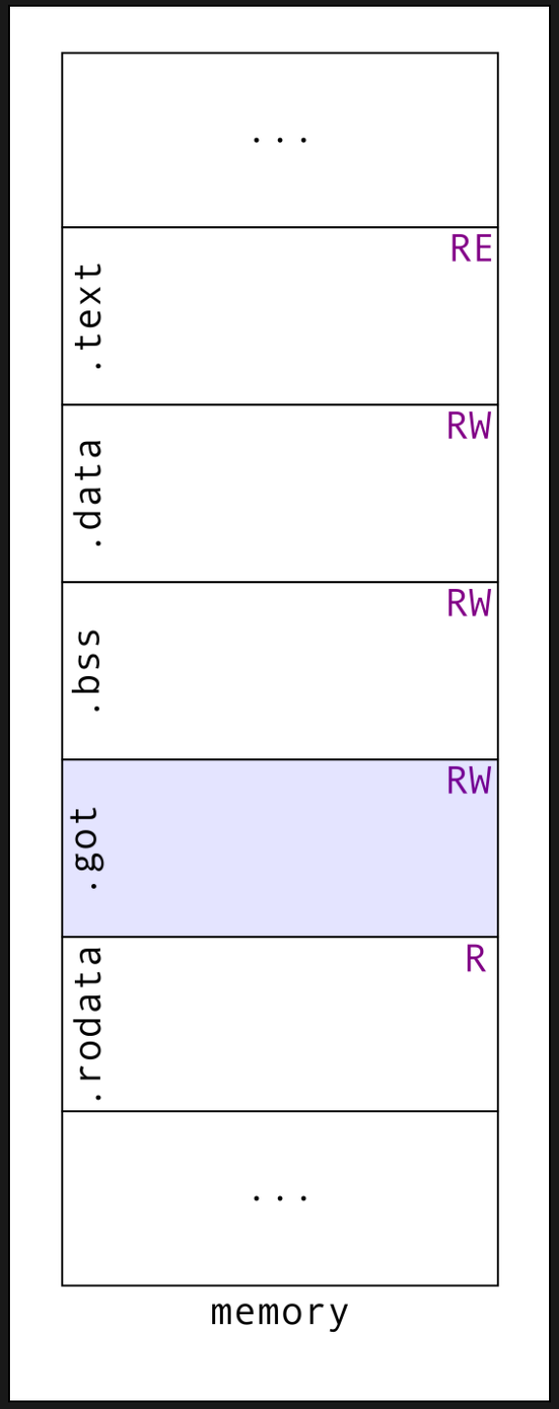
SHARED OBJECTS REALLY ARE VERY SHARED

- `.text` mapped directly into memory
- Many processes share the same `.text`
- Can't modify this!


...
.text
RE
.data
RW
.bss
RW
.rodata
R
...
memory

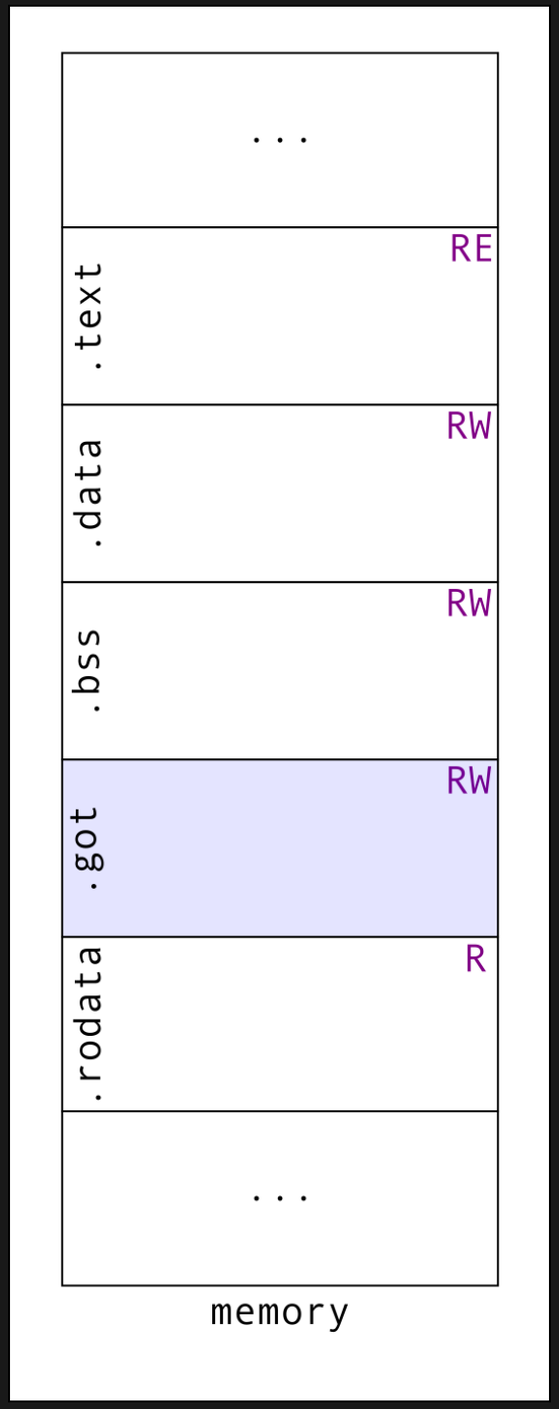

...
.text
RE
.data
RW
.bss
RW
.got
RW
.rodata
R
...
memory

```c
extern int data;

int using_data()
{
    return data;
}
```


...
.text
RE
.data
RW
.bss
RW
.got
RW
.rodata
R
...
memory

```c
extern int data;

int using_data()
{
    return data;
}
```

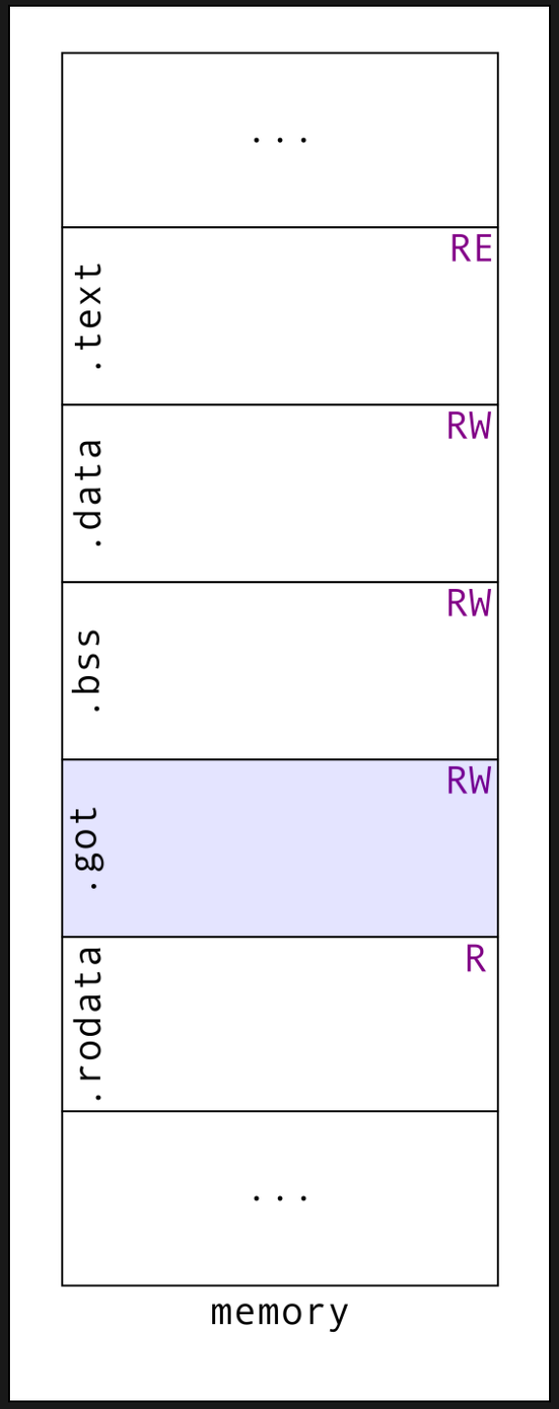
```
gcc -c file.c
```


...
.text
RE
.data
RW
.bss
RW
.got
RW
.rodata
R
...
memory

```c
extern int data;

int using_data()
{
    return data;
}
```

```
gcc -c file.c
```

```
8: 8b 05 00 00 00 00  mov eax,DWORD PTR [rip+0x0]
         a: R_X86_64_PC32 data-0x4
```

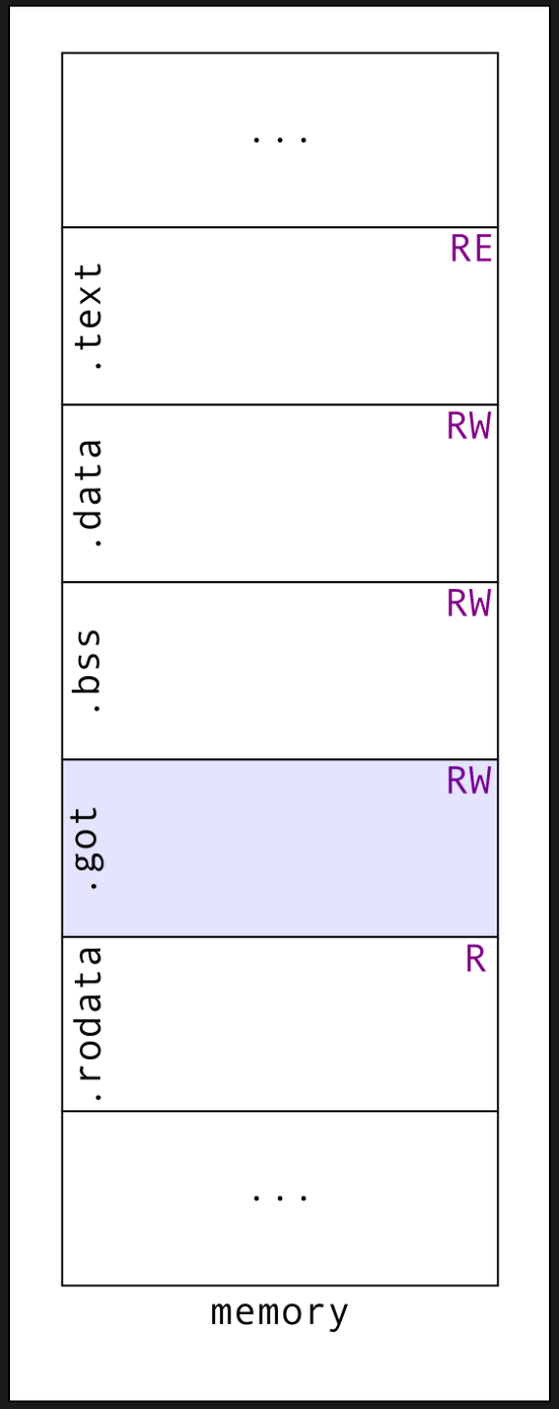

...
.text
RE
.data
RW
.bss
RW
.got
RW
.rodata
R
...
memory

```c
extern int data;

int using_data()
{
    return data;
}
```

```
gcc -c file.c
```

```
8: 8b 05 00 00 00 00  mov eax,DWORD PTR [rip+0x0]
         a: R_X86_64_PC32 data-0x4
```

```
gcc -fPIC -c file.c
```


...
.text
RE
.data
RW
.bss
RW
.got
RW
.rodata
R
...
memory

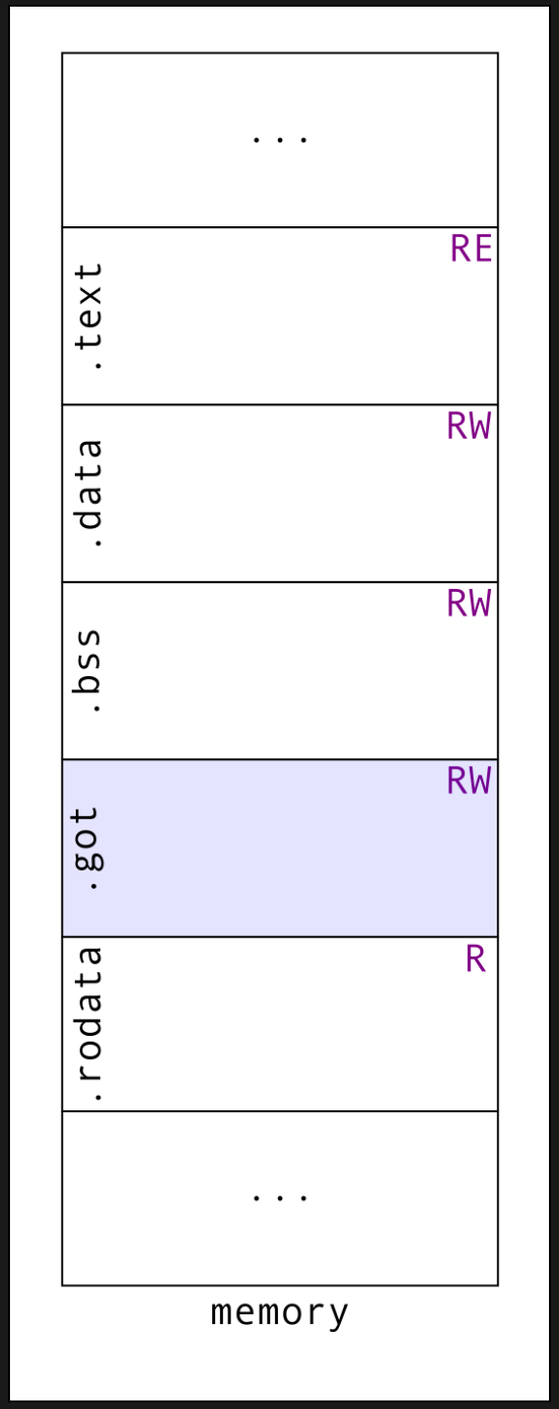
```c
extern int data;

int using_data()
{
    return data;
}
```

```
gcc -c file.c
```

```
8: 8b 05 00 00 00 00  mov eax,DWORD PTR [rip+0x0]
         a: R_X86_64_PC32 data-0x4
```

```
gcc -fPIC -c file.c
```

```
1 8: 48 8b 05 00 00 00 00   mov rax,QWORD PTR [rip+0x0]
2          b: R_X86_64_REX_GOTPCRELX  data-0x4
3 f: 8b 00                  mov eax,DWORD PTR [rax]
```


...
.text
RE
.data
RW
.bss
RW
.got
RW
.rodata
R
...
memory

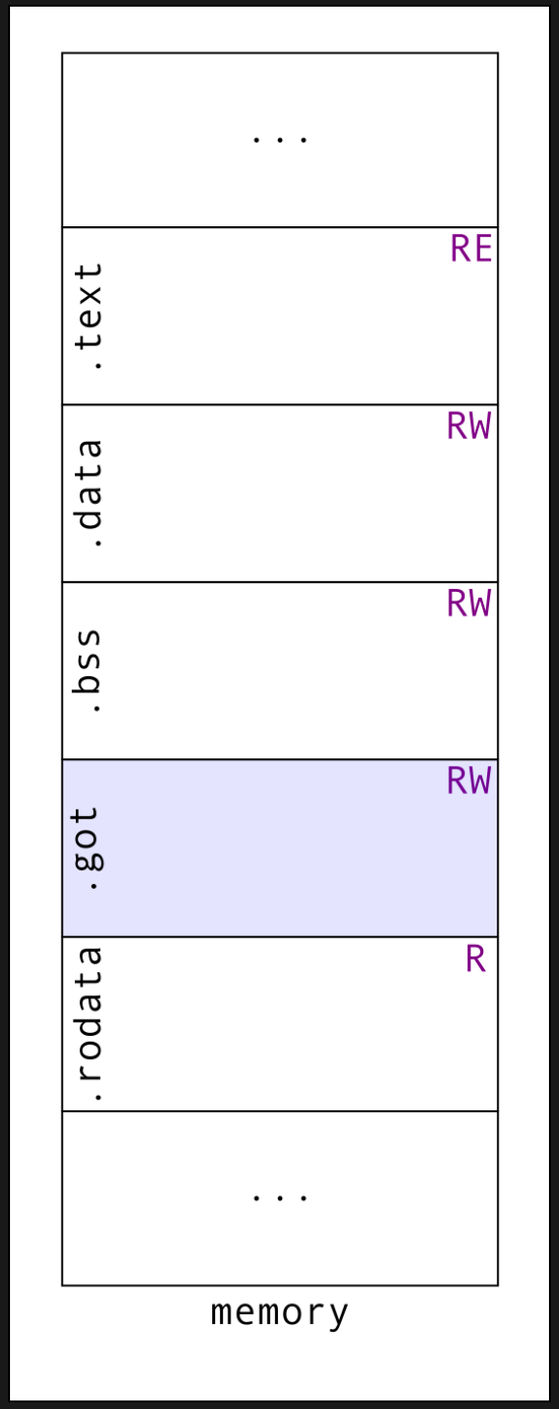
```c
extern int data;

int using_data()
{
    return data;
}
```

```
gcc -c file.c
```

```
8: 8b 05 00 00 00 00  mov eax,DWORD PTR [rip+0x0]
         a: R_X86_64_PC32 data-0x4
```

```
gcc -fPIC -c file.c
```

```
1 8: 48 8b 05 00 00 00 00   mov rax,QWORD PTR [rip+0x0]
2          b: R_X86_64_REX_GOTPCRELX  data-0x4
3 f: 8b 00                  mov eax,DWORD PTR [rax]
```


...
.text
RE
.data
RW
.bss
RW
.got
RW
.rodata
R
...
memory

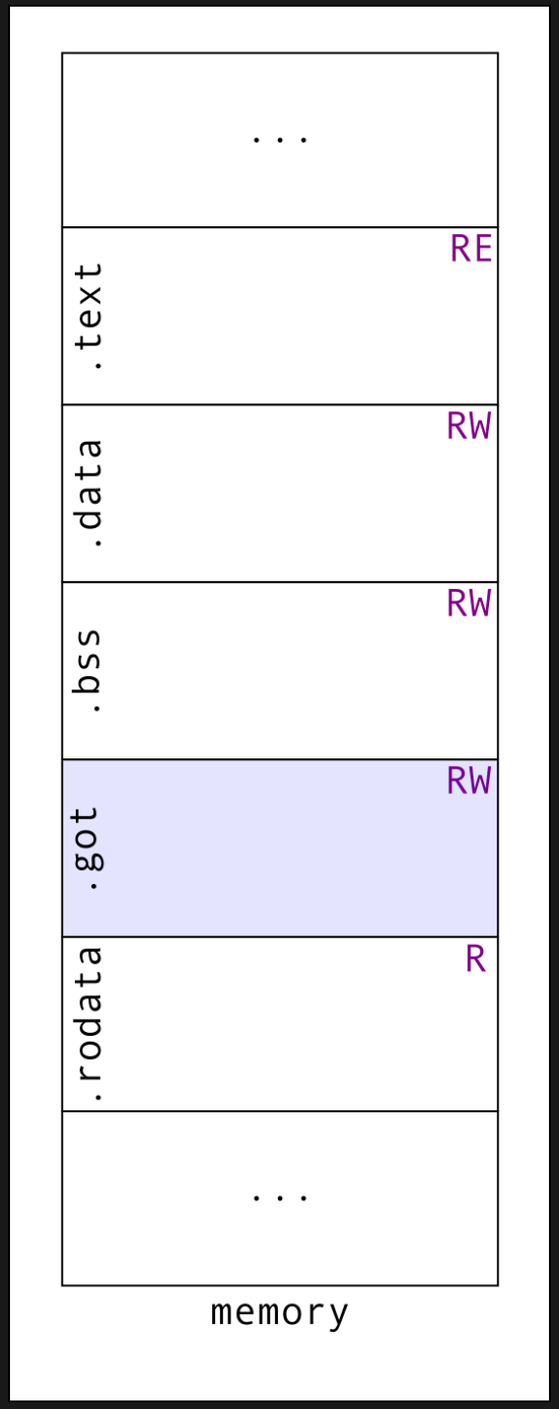
```c
extern int data;

int using_data()
{
    return data;
}
```

```
gcc -c file.c
```

```
8: 8b 05 00 00 00 00  mov eax,DWORD PTR [rip+0x0]
         a: R_X86_64_PC32 data-0x4
```

```
gcc -fPIC -c file.c
```

```
1 8: 48 8b 05 00 00 00 00   mov rax,QWORD PTR [rip+0x0]
2         b: R_X86_64_REX_GOTPCRELX  data-0x4
3 f: 8b 00                  mov eax,DWORD PTR [rax]
```


...
.text
RE
.data
RW
.bss
RW
.got
RW
.rodata
R
...
memory

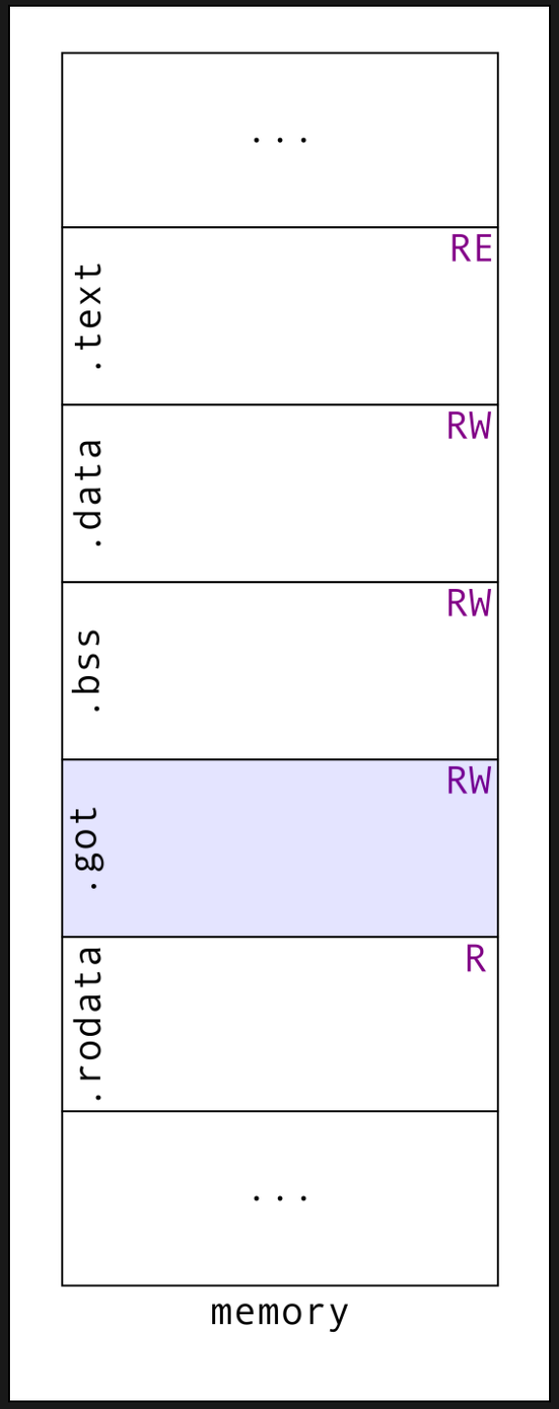
```c
extern int data;

int using_data()
{
    return data;
}
```

```
gcc -c file.c
```

```
8: 8b 05 00 00 00 00  mov eax,DWORD PTR [rip+0x0]
         a: R_X86_64_PC32 data-0x4
```

```
gcc -fPIC -c file.c
```

```
1 8: 48 8b 05 00 00 00 00   mov rax,QWORD PTR [rip+0x0]
2          b: R_X86_64_REX_GOTPCRELX  data-0x4
3 f: 8b 00                  mov eax,DWORD PTR [rax]
```


...
.text
RE
.data
RW
.bss
RW
.got
RW
.rodata
R
...
memory

```c
extern int data;

int using_data()
{
    return data;
}
```

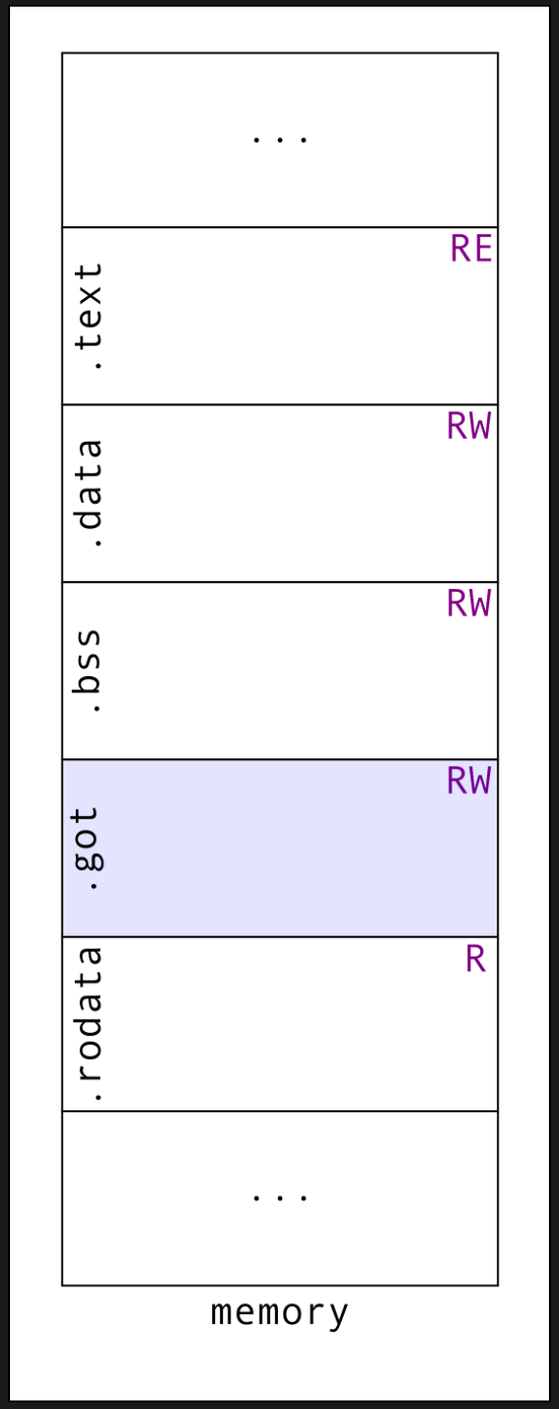
```
gcc -c file.c
```

```
8: 8b 05 00 00 00 00  mov eax,DWORD PTR [rip+0x0]
         a: R_X86_64_PC32 data-0x4
```

```
gcc -fPIC -c file.c
```

```
1 8: 48 8b 05 00 00 00 00   mov rax,QWORD PTR [rip+0x0]
2          b: R_X86_64_REX_GOTPCRELX  data-0x4
3 f: 8b 00                  mov eax,DWORD PTR [rax]
```

```
gcc -shared -o libshared.so file.o
```


...
.text
RE
.data
RW
.bss
RW
.got
RW
.rodata
R
...
memory

```c
extern int data;

int using_data()
{
    return data;
}
```

```
gcc -c file.c
```

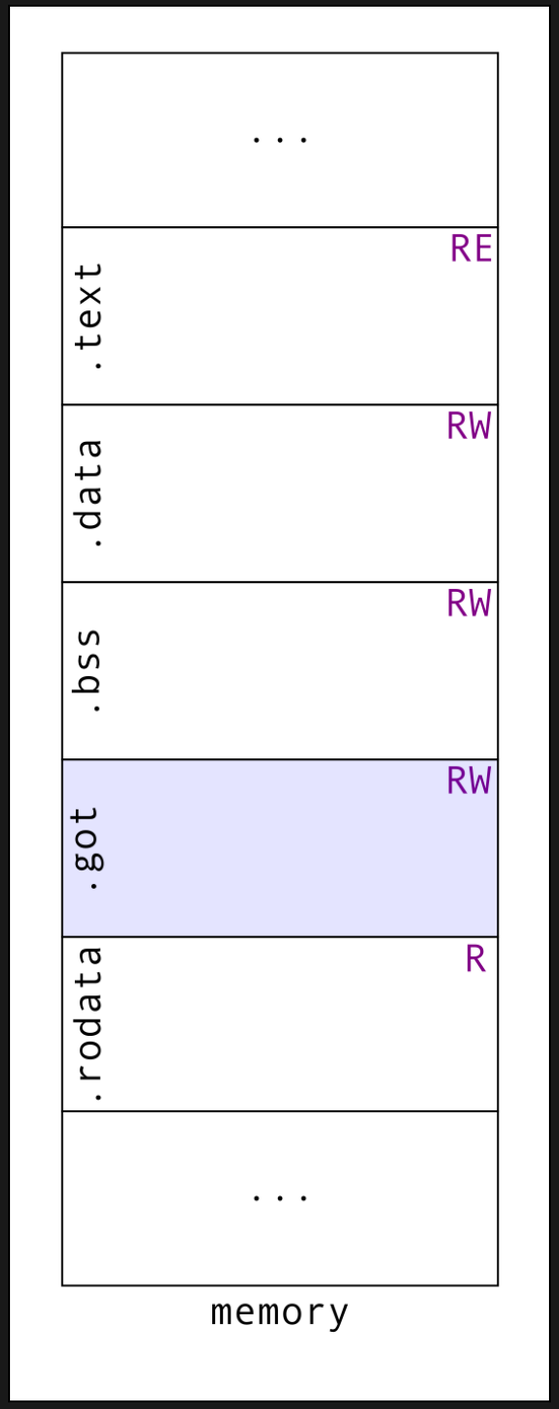
```
8: 8b 05 00 00 00 00  mov eax,DWORD PTR [rip+0x0]
         a: R_X86_64_PC32 data-0x4
```

```
gcc -fPIC -c file.c
```

```
8: 48 8b 05 00 00 00 00   mov rax,QWORD PTR [rip+0x0]
         b: R_X86_64_REX_GOTPCRELX  data-0x4
f: 8b 00                  mov eax,DWORD PTR [rax]
```

```
gcc -shared -o libshared.so file.o
```

```
1101: 48 8b 05 e8 2e 00 00  mov rax,QWORD PTR [rip+0x2ee8]
1108: 8b 00                 mov eax,DWORD PTR [rax]
```


...
.text
RE
.data
RW
.bss
RW
.got
RW
.rodata
R
...
memory

```c
extern int data;

int using_data()
{
    return data;
}
```

```
gcc -c file.c
```

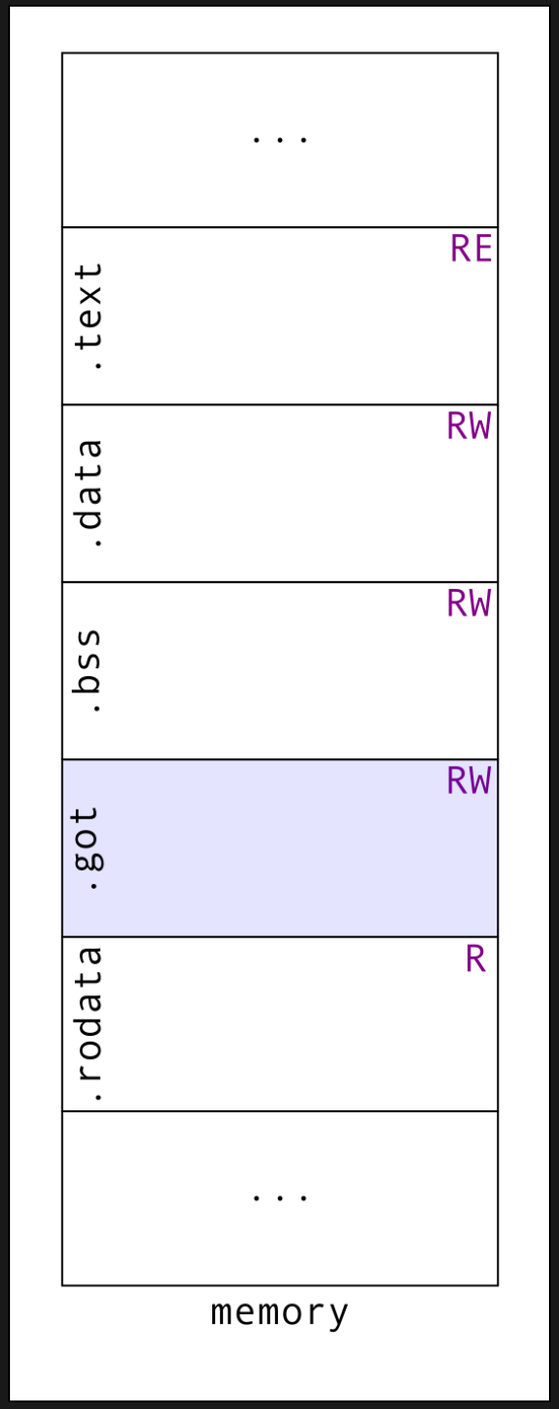
```
8: 8b 05 00 00 00 00  mov eax,DWORD PTR [rip+0x0]
         a: R_X86_64_PC32 data-0x4
```

```
gcc -fPIC -c file.c
```

```
1 8: 48 8b 05 00 00 00 00   mov rax,QWORD PTR [rip+0x0]
2          b: R_X86_64_REX_GOTPCRELX  data-0x4
3 f: 8b 00                  mov eax,DWORD PTR [rax]
```

```
gcc -shared -o libshared.so file.o
```

```
1 1101: 48 8b 05 e8 2e 00 00  mov rax,QWORD PTR [rip+0x2ee8]
2 1108: 8b 00                 mov eax,DWORD PTR [rax]
```


...
.text
RE
.data
RW
.bss
RW
.got
RW
.rodata
R
...
memory

```c
extern int data;

int using_data()
{
    return data;
}
```

```
gcc -c file.c
```

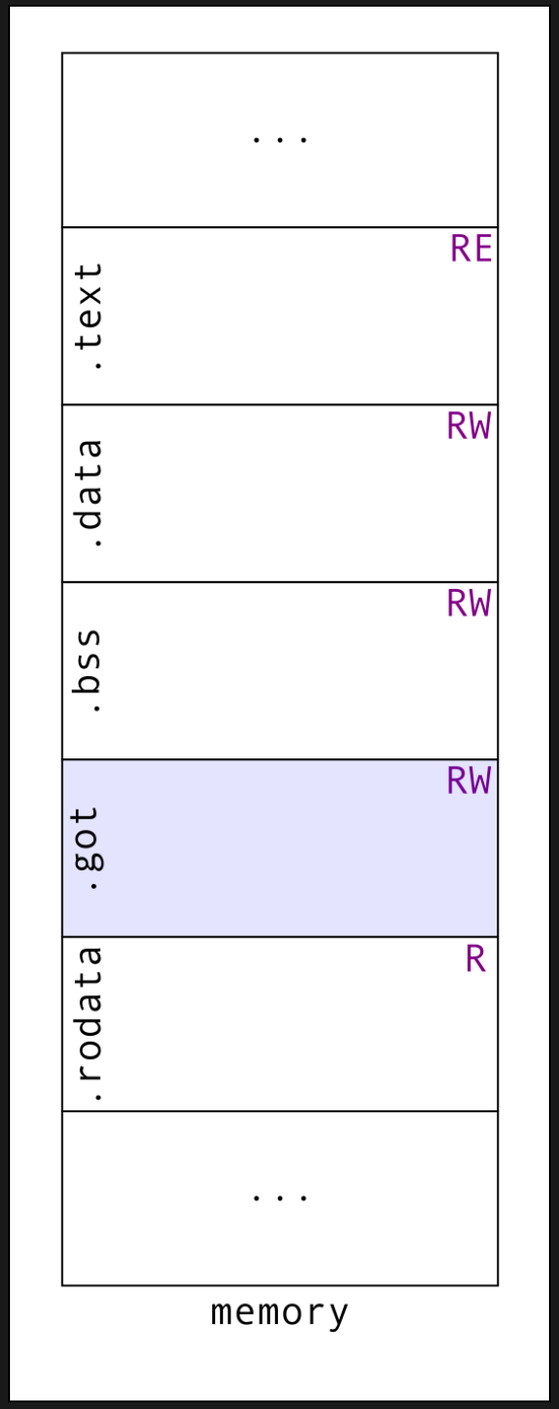
```
8: 8b 05 00 00 00 00  mov eax,DWORD PTR [rip+0x0]
         a: R_X86_64_PC32 data-0x4
```

```
gcc -fPIC -c file.c
```

```
1 8: 48 8b 05 00 00 00 00   mov rax,QWORD PTR [rip+0x0]
2          b: R_X86_64_REX_GOTPCRELX  data-0x4
3 f: 8b 00                  mov eax,DWORD PTR [rax]
```

```
gcc -shared -o libshared.so file.o
```

```
1 1101: 48 8b 05 e8 2e 00 00  mov rax,QWORD PTR [rip+0x2ee8]
2 1108: 8b 00                 mov eax,DWORD PTR [rax]
```


...
.text
RE
.data
RW
.bss
RW
.got
RW
.rodata
R
...
memory

```c
extern int data;

int using_data()
{
    return data;
}
```

```
gcc -c file.c
```

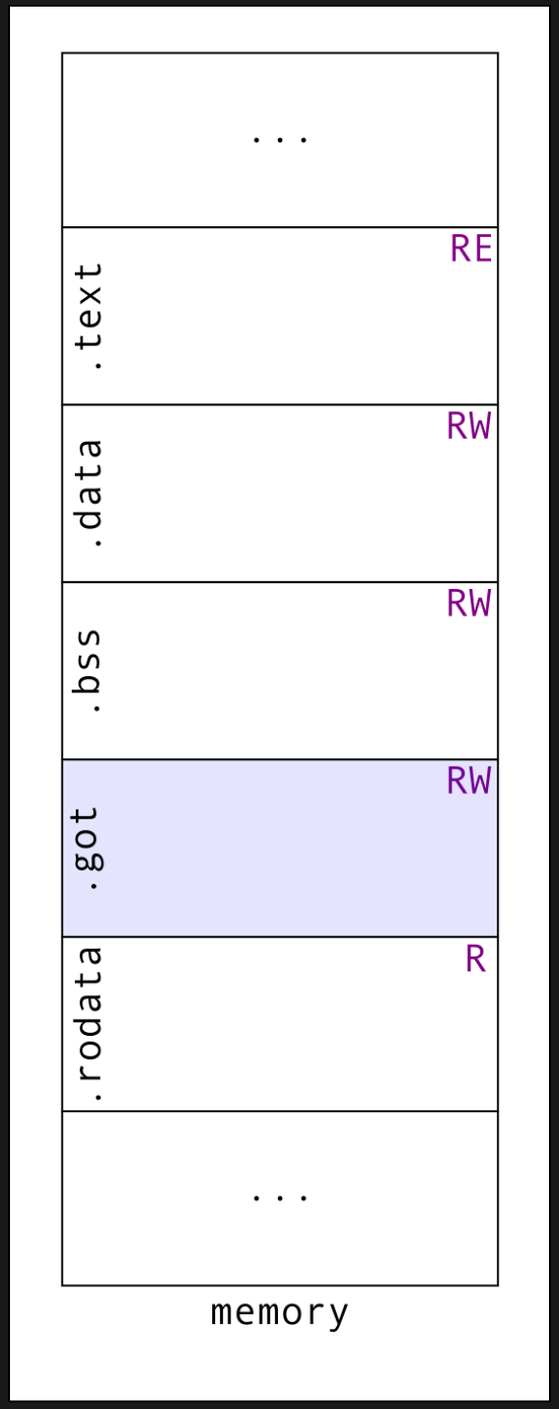
```
8: 8b 05 00 00 00 00  mov eax,DWORD PTR [rip+0x0]
         a: R_X86_64_PC32 data-0x4
```

```
gcc -fPIC -c file.c
```

```
1 8: 48 8b 05 00 00 00 00   mov rax,QWORD PTR [rip+0x0]
2          b: R_X86_64_REX_GOTPCRELX  data-0x4
3 f: 8b 00                  mov eax,DWORD PTR [rax]
```

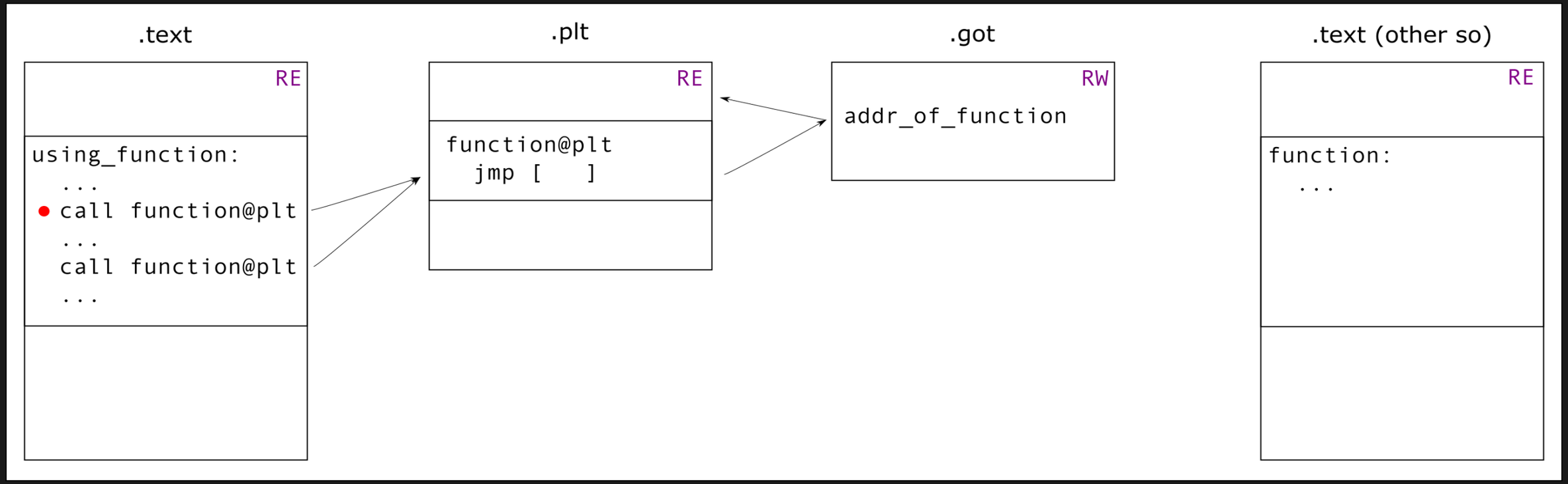
```
gcc -shared -o libshared.so file.o
```

```
1 1101: 48 8b 05 e8 2e 00 00  mov rax,QWORD PTR [rip+0x2ee8]
2 1108: 8b 00                 mov eax,DWORD PTR [rax]
```

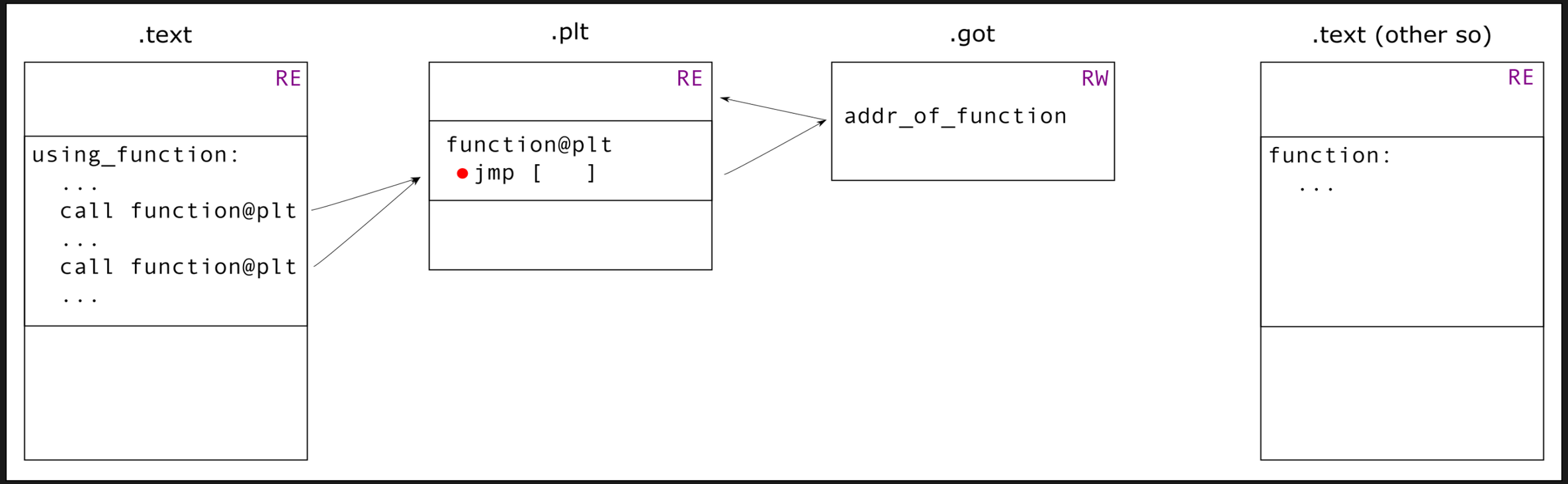

...
.text
RE
.data
RW
.bss
RW
.got
RW
.rodata
R
...
memory

```c
int function();

int using_function()
{
    int a = function();
    int b = function();
    return a+b;
}
```

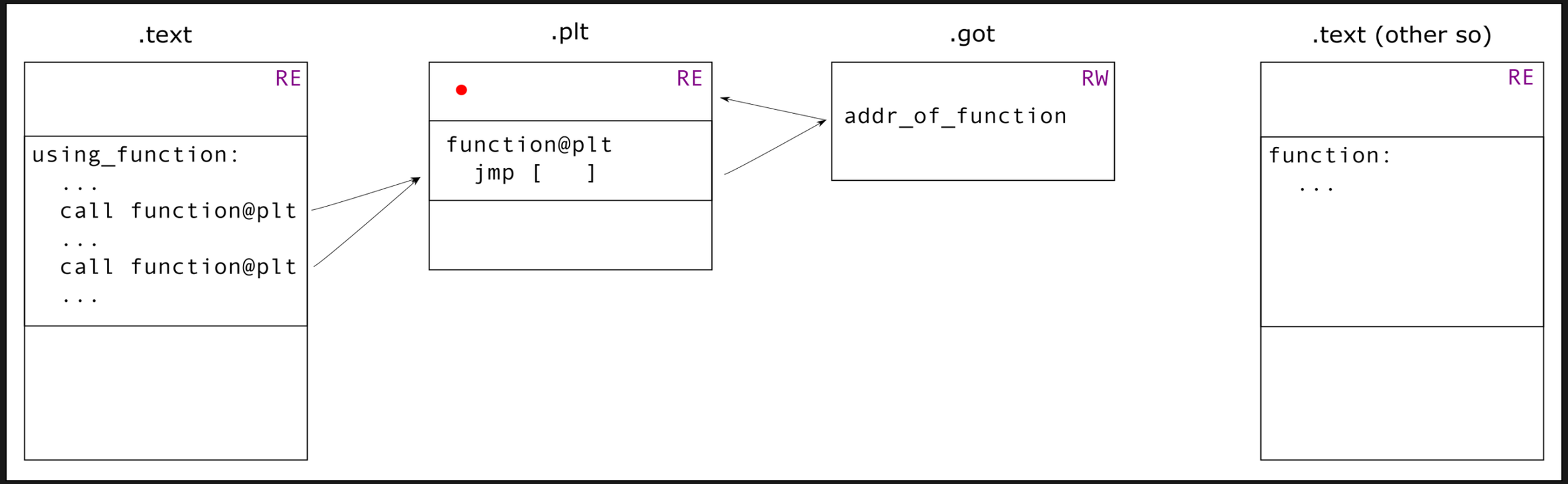
```c
int function();

int using_function()
{
    int a = function();
    int b = function();
    return a+b;
}
```



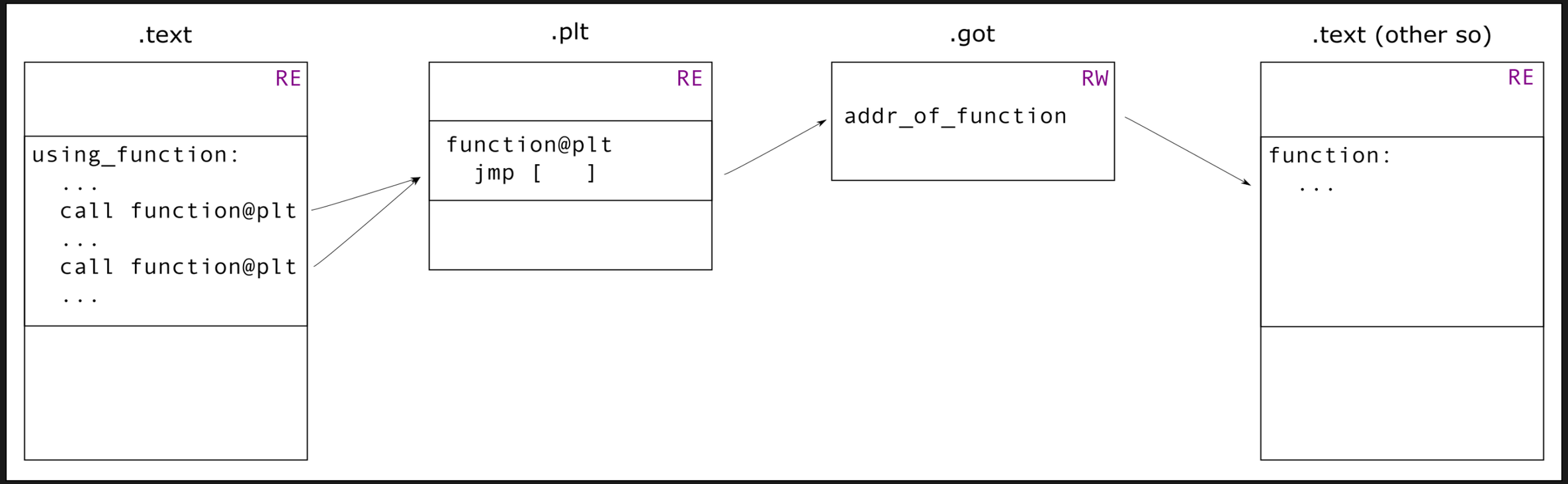
```c
int function();

int using_function()
{
    int a = function();
    int b = function();
    return a+b;
}
```



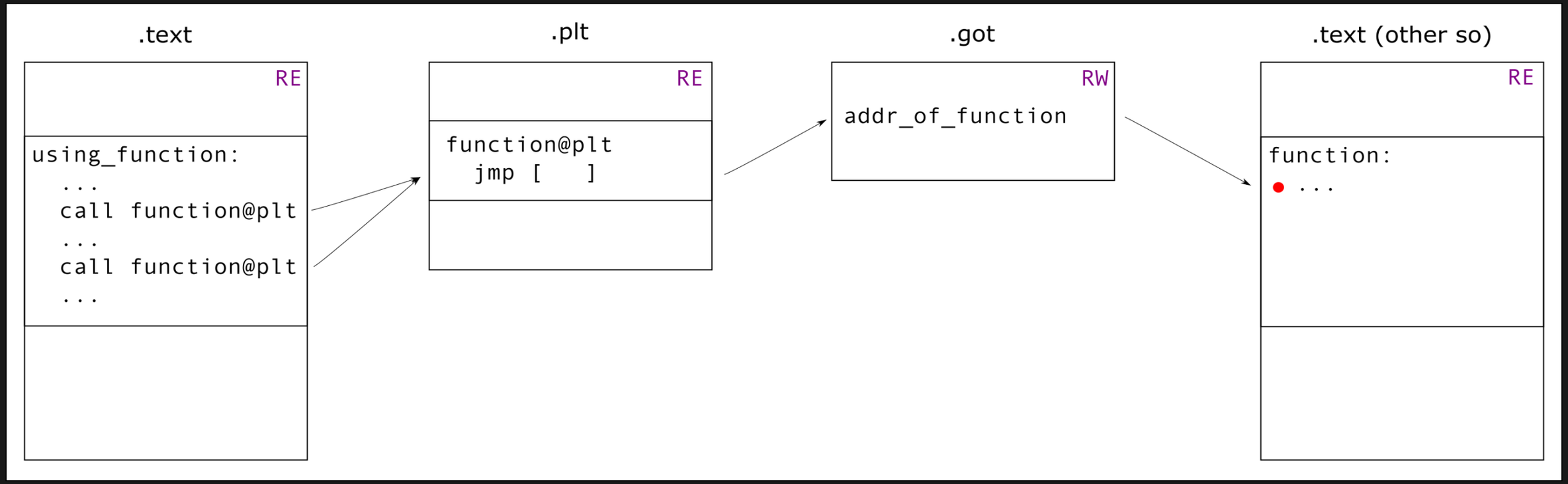
```c
int function();

int using_function()
{
    int a = function();
    int b = function();
    return a+b;
}
```



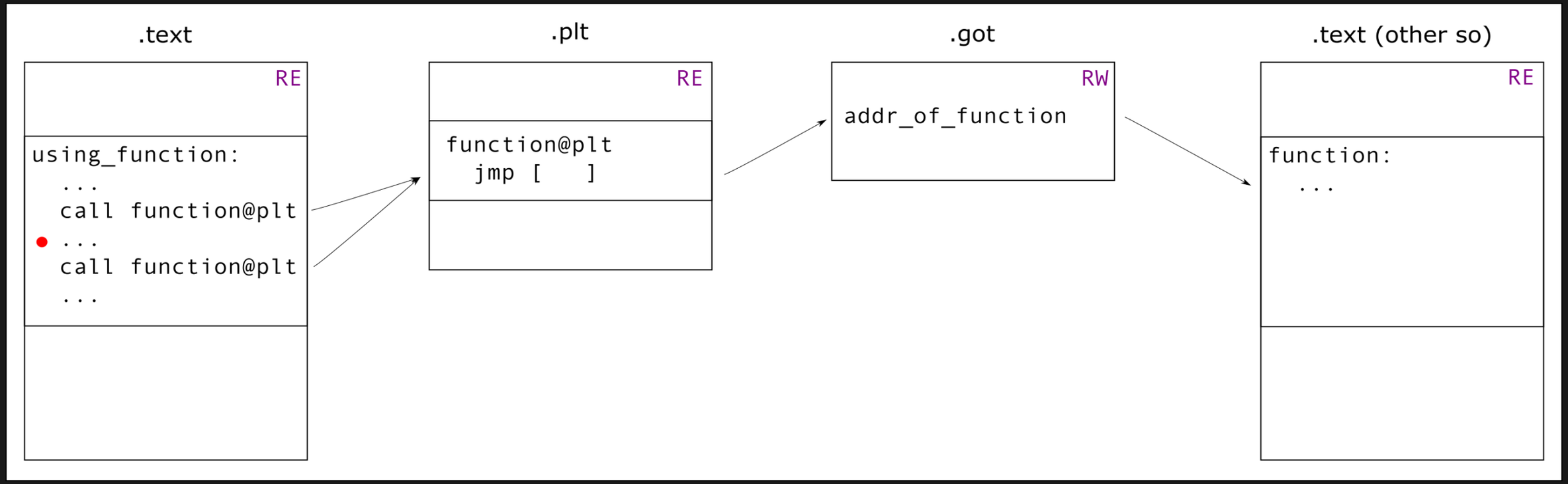
```c
int function();

int using_function()
{
    int a = function();
    int b = function();
    return a+b;
}
```



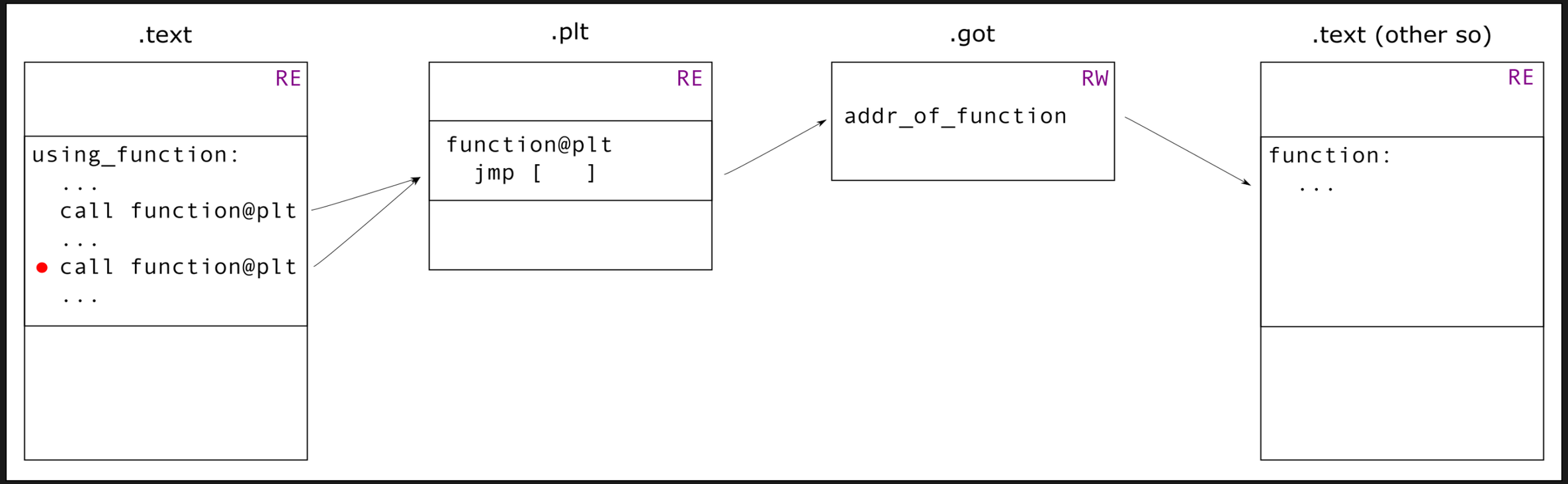
```c
int function();

int using_function()
{
    int a = function();
    int b = function();
    return a+b;
}
```



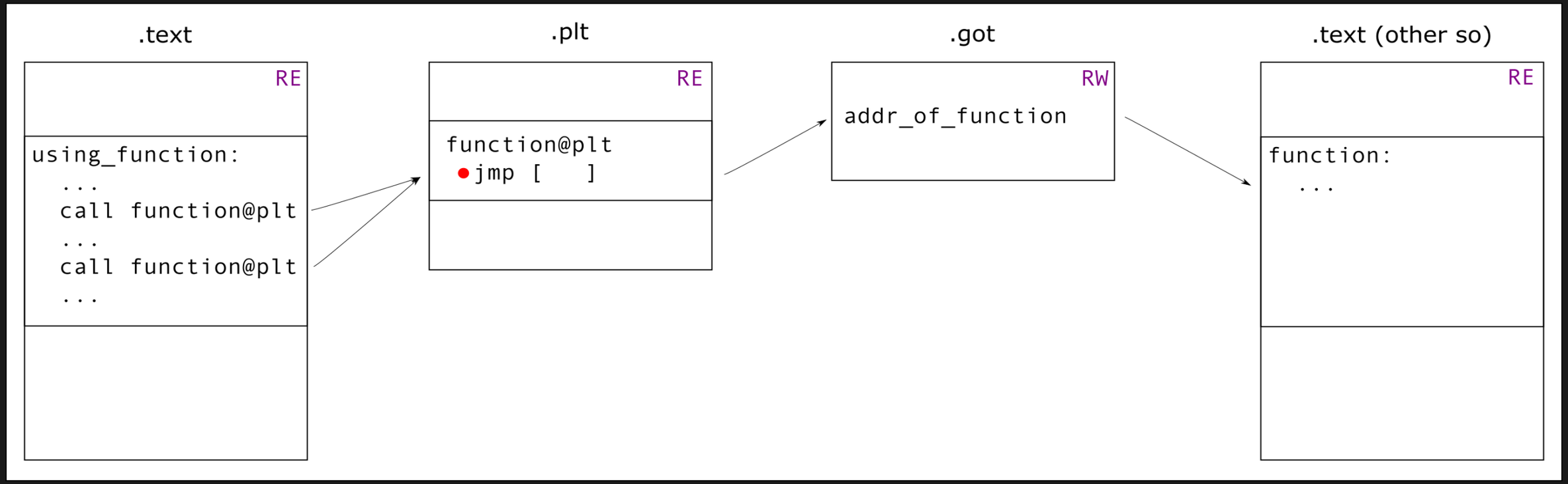
```c
int function();

int using_function()
{
    int a = function();
    int b = function();
    return a+b;
}
```



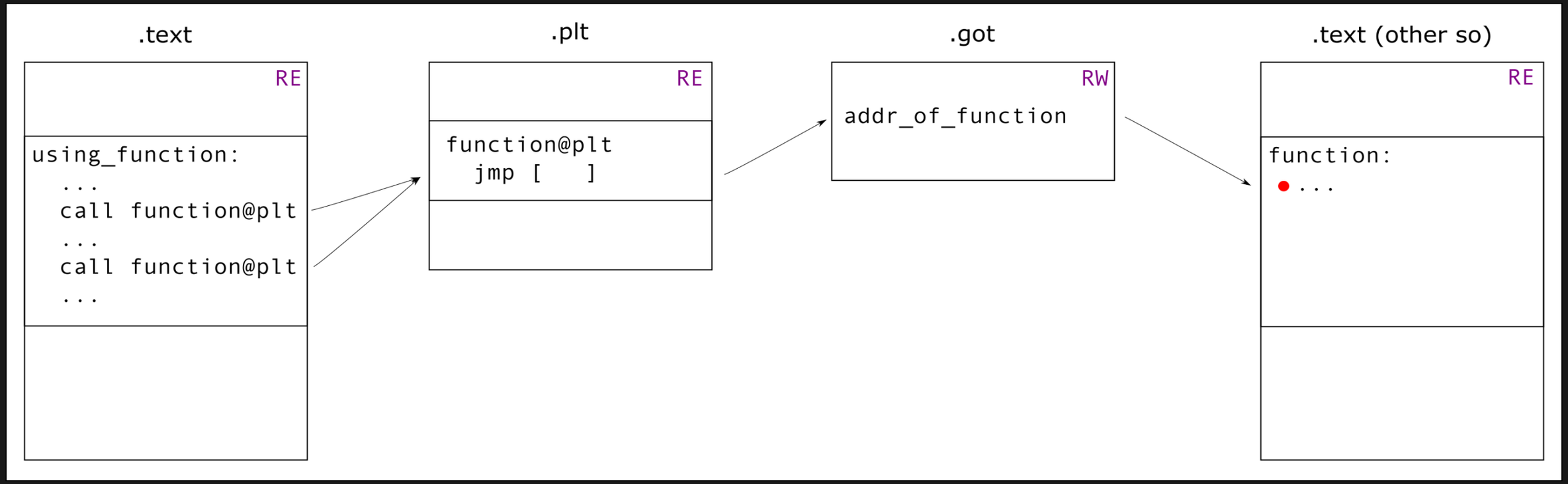
```c
int function();

int using_function()
{
    int a = function();
    int b = function();
    return a+b;
}
```



```c
int function();

int using_function()
{
    int a = function();
    int b = function();
    return a+b;
}
```



```c
int function();

int using_function()
{
    int a = function();
    int b = function();
    return a+b;
}
```



SUMMARY

SUMMARY

- Locals, parameters and returns: registers or stack

SUMMARY

- Locals, parameters and returns: registers or stack
- Static linking:

SUMMARY

- Locals, parameters and returns: registers or stack
- Static linking:
  - Global data: Relocations resolved by linker

SUMMARY

- Locals, parameters and returns: registers or stack
- Static linking:
  - Global data: Relocations resolved by linker
  - Functions: Relocations resolved by linker

SUMMARY

- Locals, parameters and returns: registers or stack
- Static linking:
  - Global data: Relocations resolved by linker
  - Functions: Relocations resolved by linker
- Dynamic linking:

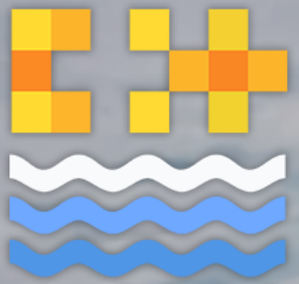
SUMMARY

- Locals, parameters and returns: registers or stack
- Static linking:
  - Global data: Relocations resolved by linker
  - Functions: Relocations resolved by linker
- Dynamic linking:
  - Global data: Global Offset Table (GOT)

SUMMARY

- Locals, parameters and returns: registers or stack
- Static linking:
  - Global data: Relocations resolved by linker
  - Functions: Relocations resolved by linker
- Dynamic linking:
  - Global data: Global Offset Table (GOT)
  - Functions: Procedure Linkage Table (PLT) + GOT

What Happens After the Compiler



Anders Schau Knatten