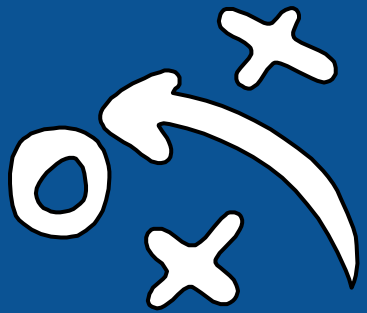
Архитектура ЭВМ и язык ассемблера

Семинар #38:

1. Вычисление значений ссылок при компоновке.

10.06.2025

Вычисление значений ссылок при компоновке



Пример кода с лекции

```c
extern void func();

char* buf = "Hello, world!\n";

int main(void)
{
    int ret_code = 0;
    func();
    return ret_code;
}
```

```c
#include <stdio.h>

extern char* buf;

void func()
{
    printf("%s", buf);
}
```

Таблицы символов объектных файлов

```
❯ nm build/hello1.o
00000000 D buf
         U func
00000000 T main
```

```
❯ nm build/hello2.o
         U buf
00000000 T func
         U printf
```

Типы символов:
B,b – символ из секции **.bss** (неинициализированные переменные)
D,d – символ из секции **.data** (инициализированные переменные)
T,t – символ из секции **.text** (функции)
R,r – символ из секции **.rodata** (константные значения)
C – **common** символ
U – неопределенный символ
V,v,W,w – слабый символ

Ссылка на внешнюю функцию

```
00000000 <main>:
   0:   f3 0f 1e fb           endbr32
   4:   55                    push   ebp
   5:   89 e5                 mov ebp,esp
   7:   83 e4 f0              and esp,0xfffffff0
   a:   83 ec 10              sub esp,0x10
   d:   c7 44 24 0c 00 00 00  mov DWORD PTR [esp+0xc],0x0
  14:   00
  15:   e8 fc ff ff ff        call   16 <main+0x16>
  1a:   8b 44 24 0c           mov eax,DWORD PTR [esp+0xc]
  1e:   c9                    leave
  1f:   c3                    ret
```

Ссылки на данные в .rodata/.data/.bss

```
00000000 <func>:
   0:   f3 0f 1e fb              endbr32
   4:   55                       push   ebp
   5:   89 e5                    mov ebp, esp
   7:   83 ec 08                 sub esp, 0x8
   a:   a1 00 00 00 00           mov eax, ds:0x0
   f:   83 ec 08                 sub esp, 0x8
  12:   50                       push   eax
  13:   68 00 00 00 00           push   0x0
  18:   e8 fc ff ff ff           call   19 <func+0x19>
  1d:   83 c4 10                add esp,0x10
  20:   90                      nop
  21:   c9                      leave
  22:   c3                      ret
```

Задание типов ссылок

```
❯ readelf --relocs build/hello1.o

Relocation section '.rel.text' at offset 0x1d0 contains 1 entry:
 Offset     Info    Type            Sym.Value  Sym. Name
00000016  00000c02 R_386_PC32      00000000   func

Relocation section '.rel.data' at offset 0x1d8 contains 1 entry:
 Offset     Info    Type            Sym.Value  Sym. Name
00000000  00000501 R_386_32        00000000   .rodata

Relocation section '.rel.eh_frame' at offset 0x1e0 contains 1 entry:
 Offset     Info    Type            Sym.Value  Sym. Name
00000020  00000202 R_386_PC32      00000000   .text
```

Задание типов ссылок

```
❯ readelf --relocs build/hello2.o

Relocation section '.rel.text' at offset 0x1c8 contains 3 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
0000000b  00000b01 R_386_32        00000000   buf
00000014  00000501 R_386_32        00000000   .rodata
00000019  00000c02 R_386_PC32      00000000   printf

Relocation section '.rel.eh_frame' at offset 0x1e0 contains 1 entry:
 Offset     Info    Type            Sym.Value  Sym. Name
00000020  00000202 R_386_PC32      00000000   .text
```

Вычисление значений ссылок

Схемы вычисления значений ссылок:

- **R_386_32**: S + A
- **R_386_PC32**: S + A - P

Слагаемые:

- **S** – абсолютный адрес памяти, которому символ соответствует после перемещения.
- **A** – дополнительное слагаемое (addend), хранимое непосредственно в байтах ссылки.
- **P** – абсолютный адрес ссылки.

Анализ результата компоновки

```
❯ objdump -M intel --disassemble=main build/relocs
000011cd <main>:
    11cd:   f3 0f 1e fb             endbr32
    11d1:   55                      push   ebp
    11d2:   89 e5                   mov ebp,esp
    11d4:   83 e4 f0                and esp,0xfffffff0
    11d7:   83 ec 10                sub esp,0x10
    11da:   c7 44 24 0c 00 00 00    mov DWORD PTR [esp+0xc],0x0
    11e1:   00
    11e2:   e8 06 00 00 00          call   11ed <func>
    11e7:   8b 44 24 0c             mov eax,DWORD PTR [esp+0xc]
    11eb:   c9                      leave
    11ec:   c3                      ret
```

Анализ результата компоновки

```
❯ objdump -M intel --disassemble=func build/relocs
000011ed <func>:
    11ed:   f3 0f 1e fb             endbr32
    11f1:   55                      push   ebp
    11f2:   89 e5                   mov ebp,esp
    11f4:   83 ec 08                sub esp,0x8
    11f7:   a1 08 40 00 00          mov eax,ds:0x4008
    11fc:   83 ec 08                sub esp,0x8
    11ff:   50                      push   eax
    1200:   68 17 20 00 00          push   0x2017
    1205:   e8 fc ff ff ff          call   1206 <func+0x19>
    120a:   83 c4 10                add esp,0x10
    120d:   90                      nop
    120e:   c9                      leave
    120f:   c3                      ret
```

Вычисление значений ссылок

```
❯ readelf -s build/relocs
Symbol table '.symtab' contains 72 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     0: 00000000     0 NOTYPE  LOCAL  DEFAULT  UND
    ...
    62: 000011ed    35 FUNC    GLOBAL DEFAULT   16 func
    ...
    67: 00004008     4 OBJECT  GLOBAL DEFAULT   25 buf

RSLT(buf)  = 00004008 + 00000000 = 00004008
RSLT(func) = 000011ed + fffffffc - 000011e2 = 00000006
```

Вопросы?

Красивые иконки взяты с сайта handdrawngoods.com