

ICS EXE 10 Solution

May 13, 2020

1 Symbol

The following program consists of two modules: **main** and **foo**. Their corresponding source code files are shown below. (All the process of linking runs on an x86-64 machine.)

```
1  /* main.c */
2  #include <stdio.h>
3
4  extern char *names[];
5  static int id;
6  int foo(int n);
7  void main(void) {
8      id = 103;
9      char *str = names[foo(id)];
10     printf("%s %d\n", str, id);
11 }
```

```
1  /* foo.c */
2  char *names[] = {"Mario", "BenweiLu",
3                  "YYF", "Link"};
4  int id = 102;
5  int foo(int n) {
6      int res = 0;
7      switch(n) {
8          case 100:
9              res = 1; break;
10         case 103:
11             res = 2; break;
12         case 104:
13             res = 3; break;
14         default:
15             res = 0;
16     }
17     id = 233;
18     return res;
19 }
```

1. For symbols that are defined and referenced in **main.o** and **foo.o**, please complete the symbol tables. The format of them are the same as ones in **section 7.5** of your ICS book.

Module	Name	Type	Bind	Value(Hex)	Size	NDX
main.o	id	OBJECT	LOCAL	00000000	4	4
main.o	main	FUNC	GLOBAL	00000000	88	1
main.o	foo	NOTYPE	GLOBAL	00000000	0	UND
foo.o	id	OBJECT	GLOBAL	00000020	4	3

2. Please explain why the **Value** of **id** in **foo.o** is 0x00000020.
Value attribute of **id** is its offset in .data section. Before **id**, there must be names at offset 0, which occupies 32 bytes.
3. Please write down the output of **main.c**.
YYF 103

2 Relocation

The following program consists of two source files: **main.c** and **draw.c**, the relocatable object files are also listed. (The linking procedure runs on an x86_64 little-endian machine.)

```

1  /* main.c */
2  int y = 5;
3  static int x = 200;
4  int a[4];
5  int *ap = &a[1];
6  const int num = 8;
7  extern int draw(int n);
8
9  void main(){
10     int i = draw(x);
11     printf("Get %s using x = %d\n",
12           (char *)a[i], x);
13 }

```

```

1  /* main.o */
2  .text:
3  0000000000000000 <main>:
4  0: 55                push    %rbp
5  1: 48 89 e5           mov     %rsp,%rbp
6  4: 48 83 ec 10        sub     %0x10,%rsp
7  8: 8b 05 00 00 00 00  mov     0x0(%rip),%eax
8  e: 89 c7             mov     %eax,%edi
9  10: e8 00 00 00 00    callq   15 <main+0x15>

```

```

10 15: 89 45 fc          mov     %eax,-0x4(%rbp)
11 18: 8b 15 00 00 00 00  mov     0x0(%rip),%edx
12 1e: 8b 45 fc          mov     -0x4(%rbp),%eax
13 21: 48 98             cltq
14                // sign extend eax to rax
15 23: 8b 04 85 00 00 00 00 mov     0x0(,%rax,4),%eax
16 2a: 89 c6             mov     %eax,%esi
17 2c: bf 00 00 00 00     mov     $0x0,%edi
18 31: b8 00 00 00 00     mov     $0x0,%eax
19 36: e8 00 00 00 00     callq   3b //printf
20 ...
21 .data:
22 ...
23 0000000000000008 <ap>:
24 8: 00 00 00 00 00 00 00 00

```

```

1  /* draw.c */
2  char *a[] = {"BaiQi", "XuMo",
3              "LiZeyan", "ZhouQiluo"};
4  long y;
5  static long x = 20;
6  extern int num;
7
8  int draw(int n) {
9      static long x = 0;
10     x = 234;
11     const int num = 4;
12     y = x - n;
13     return y % num;
14 }

```

```

1  /* draw.o */
2  .text:
3  0000000000000000 <draw>:
4  0: 55             push    %rbp
5  1: 48 89 e5       mov     %rsp,%rbp
6  4: 89 7d ec       mov     %edi,-0x14(%rbp)
7  7: 48 c7 05 00 00 00 movq    $0xea,0x0(%rip)
8  d: 00
9  e: ea 00 00 00
10 12: c7 45 fc 04 00 00 movl    $0x4,-0x4(%rbp)
11 18: 00
12 19: 48 8b 15 00 00 00 mov     0x0(%rip),%rdx
13 1f: 00

```

```

14 20: 8b 45 ec          mov     -0x14(%rbp),%eax
15 23: 48 98              cltq
16 25: 48 29 c2          sub     %rax,%rdx
17 28: 48 89 d0          mov     %rdx,%rax
18 2b: 48 89 05 00 00 00  mov     %rax,0x0(%rip)
19 31: 00
20 ... // calculate y*num and return the value

```

- For symbols that are defined and referenced in **main.o** and **draw.o**, please complete the symbol tables. The format of them are the same as ones in **section 7.5** of your ICS book.

Module	Name	Value(Hex)	Size	Type	Bind	NDX
main.o	main	00000000	61	FUNCTION	GLOBAL	.text
main.o	num	00000000	4	OBJECT	GLOBAL	.rodata
main.o	x	00000004	4	OBJECT	LOCAL	.data
main.o	draw	00000000	0	NOTYPE	GLOBAL	UND
draw.o	a	00000000	0x20(32)	OBJECT	GLOBAL	.data
draw.o	y	00000008	8	OBJECT	GLOBAL	COMMON

- Please write down the output of **main.c**. NOTE: You don't need to consider .o files for this problem.
Get XuMo using $x = 0$
- Fill in the relocation entries of **main.o** and **draw.o**.

Relocation entries of **main.o**

Section	Offset(HEX)	Type	Symbol Name
.data	00000008	R_X86_64_64	a
.text	00000011	R_X86_64_PC32	draw
.text	00000026	R_X86_64_32S	a

Relocation entries of **draw.o**

Section	Offset(HEX)	Type	Symbol Name
.text	0000000a	R_X86_64_PC32	x
.text	0000002e	R_X86_64_PC32	y

- After relocation and the program is built, some changes will happen to the underlined instructions/data. Part of the symbol table and some comparison of relocations are given below. Fill in the blanks.

Name	Section	Type	Value(HEX)
num	.rodata	OBJECT	00400624
x	.bss	OBJECT	00600a20
a	.data	OBJECT	006009e0
y	.data	OBJECT	00600a08
draw	.text	FUNC	00400506
main	.text	FUNC	0040054f

Comparison of relocations of **main.o**

Section	Before relocation	After relocation
.text	8: 8b 05 <u>00 00 00 00</u>	af 04 20 00
.text	10: e8 <u>00 00 00 00</u>	a2 ff ff ff
.data	8: <u>00 00 00 00 00 00 00 00</u>	e4 09 60 00 00 00 00 00

Comparison of relocations of **draw.o**

Section	Before relocation	After relocation
.text	19: 48 8b 15 <u>00 00 00 00</u>	fa 04 20 00
.text	2b: 48 8b 05 <u>00 00 00 00</u>	d0 04 20 00