作业 3

1．假设一个 C 语言程序有两个源文件：main.c 和 test.c，如下所示：

```
1    /* main.c */
2    int sum();
3
4    int a[4]={1，2，3，4};
5    extern int val;
6    int main( )
7    {
8      val = sum();
9      return val;
10   }
```

```
1    /* test.c*/
2    extern int a[];
3    int val=0;
4    int sum()
5    {
6      int i;
7      for (i = 0; i < 4; i++)
8      val += a[i];
9      return val;
10   }
```

对于编译生成的可重定位目标文件 test.o，填写表中各符号的情况，说明每个符号是否出现在 test.o 的符号表（.symtab 节）中，如果是，定义该符号的模块是 main.o 还是 test.o？该符号的类型是全局、外部，还是本地符号？该符号出现在相应定义模块的哪个节（.text、.data 或 .bss）？

题 1 表

符号	是否在 test.o 的符号表中	定义模块	符号类型	节
a				
val				
sum				
i				

2．一个 C 语言程序包含两个源文件 main.c 和 proc.c 如下所示，请分析回答：

（1）这两个文件中出现的符号哪些是强符号？哪些是弱符号？各变量的存储空间分配在 ELF 哪个节中？各占几个字节？

（2）程序执行后的打印结果是什么？请分别画出执行第 7 行 proc()函数，调用之前和调用之后，在地址&x 和&z 中存放的内容。

（3）若 main.c 的第 3 行改为“short y=1, z=2;”，结果又会怎样？

（4）修改文件 proc，使得 main.c 能输出正确的结果（即 x=257，z=2）。要求修改时不能改变任何变量的数据类型和名字。

/* main.c */

```
1    #include <stdio.h>
```

```
2    unsigned x = 257;
3    short y, z = 2;
4    void proc(void)
5    void main()
6    {
7      proc();
8      printf("x=%u,z=%d", x, z);
9      return 0;
10   }
/* proc.c */
1    double x;
2
3    void proc()
4    {
6      x = -1.5;
7    }
```

3．以下由两个目标模块 ml 和 m2 组成的程序，经编译、汇编、链接后在计算机上执行，结果发现即使 p1 函数中没有对数组变量 main 进行初始化，最终也能打印出字符串"0x5589\n"。为什么？要求解释原因。

```
1    /* m1.c */
2    void p1(void);
3
4    int main()
5    {
6      p1();
7      return 0;
8    }
```

```
1    /* m2.c */
2    #include <stdio.h>
3    char main[2];
4
5    void p1()
6    {
7      printf("0x%x%x\n", main[0], main[1]);
8    }
```

4．以下给出了用 OBJDUMP 显示的某个可执行目标文件的程序头表（段头表）的部分信息，其中，可读写数据段（Read/write data segment）的信息表明，该数据段对应虚拟存储空间中起始地址为 0x8049448、长度为 0x104 个字节的存储区，其数据来自可执行文件中偏移地址 0x448 开始的 0xe8 个字节。这里可执行目标文件中的数据长度和虚拟地址空间中的存储区大小之间相差了 28 字节。请解释可能的原因。

某可执行目标文件程序头表的部分内容

Read-only code segment

LOAD off 0x00000000 vaddr 0x08048000 paddr 0x08048000 align 2**12 filesz 0x00000448 memsz 0x00000448 flags r-x

Read/write data segment

LOAD off 0x00000448 vaddr 0x08049448 paddr 0x08049448 align 2**12 filesz 0x000000e8 memsz 0x00000104 flags rw-

5．a 和 b 是可重定位目标文件或静态库文件，a → b 表示 b 中定义了一个被 a 引用的符号，对于以下各种情形给出一个最短的命令行（含有最少可重定位目标文件或静态库文件参数），使链接器能够解析所有符号引用：

（1）p.o → libx.a → liby.a → p.o

（2）p.o → libx.a → liby.a 且 liby.a → libx.a

（3）p.o → libx.a → liby.a → libz.a 且 liby.a → libx.a → libz.a

6．一个 C 语言程序包含两个源文件 main.c 和 swap.c 如下所示

/* swap.c */

```
1    extern int buf[];
2
3    int *bufp0 = &buf[0];
4    static int *bufp1;
5
6    void swap()
7    {
8      int temp;
9      bufp1 = &buf[1];
10     temp = *bufp0;
11     *bufp0 = *bufp1;
12     *bufp1 = temp;
13   }
```

/* main.c */

```
1    extern void swap(void);
2
3    int buf[2] = {1, 2};
4
5    int main()
6    {
7      swap();
8      return 0;
9    }
```

main 函数源代码对应的 main.o 中的.text 节和.rel.text 节的内容如下所示，.text 节中有一处需要重定位，若 main 函数代码的起始地址是 0x8048386，紧跟在 main 函数后的是 swap 函数的代码，且首地址按 4 字节对齐。请问 main.o 的.text 节中需要重定位的符号名、相对于.text 节的起始位置的位移、所在指令的行号、重定位类型、重定位前的内容、重定位后的内容，

并给出重定位值的计算过程。

```
1     Disassembly of section .text:
2     00000000 <main>:
3     0:    55                          push    %ebp
4     1:    89 e5                       mov     %esp, %ebp
5     3:    83 e4 f0                    and     $0xfffffff0, %esp
6     6:    e8 fc ff ff ff              call    7 <main+0x7>
7                 7:    R_386_PC32 swap
8     b:    b8 00 00 00 00              mov     $0x0, %eax
9     10:   c9                          leave
10    11:   c3                          ret
```

提交作业

作业（word 或 pdf 格式）拷贝到一个文件夹中，命名为：

作业 n

其中，n=1…3 为第 n 次作业

课程结束时，将 3 次作业与 6 次实验拷贝到同一个文件夹中，命名为如下格式：

班号-学号-姓名

以班为单位一起提交。