Homework6

姓名 邢添程
学号 2024202862

Q1

一个C函数fun具有如下代码体：(参数从右向左入栈)

```
*p = d;
return x-c;
```

执行这个函数体的IA32代码如下：

```
Movsbl    12(%ebp), %edx // 较小的byte->dword, s表示符号填充，z表示0填充
Movl      16(%ebp), %eax
Movl      %edx, (%eax)
Movswl    8(%ebp), %eax
Movl      20(%ebp), %edx
Subl      %eax, %edx
Movl      %edx, %eax
```

写出函数fun的原型，给出参数p, d, x, c的类型和顺序。写出求解过程。

Answer1

- 函数原型：

```
int fun(short c, char d, int *p, int x);
```

分析：

结合代码分析，可知第一阶段 `d` 存储在 `%edx` 中，指针 `p` 存储在 `%eax` 中。因此推断出， `d` 为 `12(%ebp)` ， `p` 为 `16(%ebp)` 。第二阶段 `x` 存储在 `%edx` 中， `c` 存储在 `%eax` 中。因此推断出， `x` 为 `20(%ebp)` ， `c` 为 `8(%ebp)` 。由于栈是从高地址向低地址增长，再结合参数从右向左入栈可知参数顺序为： `c d p x` 。

再结合指令分析， d 对应的指令是 movsbl ，因此 d 的类型长度为 1 字节，为 char 类型。同理可知， p 为 int * 类型， c 为 short 类型， x 为 int 类型。

综上，参数从右到左入栈的顺序是 x， p， d， c 。函数原型为： int fun(short c, char d, int *p, int x); 。

Q2

- Suppose the initial value of %esp is 0x7FFFFFC4, initial value of %ebp is 0x7FFFFFF4.
- The value stored in address 0x7FFFFFC0 is 0x120, value stored in address 0x7FFFFFC4 is 0x200, the value stored in address 0x7FFFFFF4 is 0x2710.
- We have following x86 assembly code executed sequentially:

```
pushl %ebp (instruction 1)
movl %esp,%ebp (instruction 2)
popl %ebp (instruction 3)
```

Question: After each instruction executed, what is the value of %esp and %ebp

Answer2

(1) Instruction 1:

- %esp = 0x7FFFFFC4 - 4 = 0x7FFFFFC0
- %ebp = 0x7FFFFFF4

(2) Instruction 2:

- %esp = 0x7FFFFFC0
- %ebp = 0x7FFFFFC0

(3) Instruction 3:

- %esp = 0x7FFFFFC4
- %ebp = (%esp) = 0x7FFFFFF4

Q3(1)

右边是C语言源代码文件func.c对应的汇编代码，请写出对应的C语言代码；

```
.LC0:
    .string "%d %d"
.LC1:
    .string "%d %d %d\n"

main:
    subq    $24, %rsp
    leaq    8(%rsp), %rdx
    leaq    12(%rsp), %rsi
    movl    $.LC0, %edi
    movl    $0, %eax
    call    __isoc99_scanf
    movl    12(%rsp), %ecx
    movl    8(%rsp), %edx
    movl    %edx, %esi
    xorl    %ecx, %esi
    movl    $.LC1, %edi
    movl    $0, %eax
    call    printf
    movl    $0, %eax
    addq    $24, %rsp
    ret
```

Answer3(1)

对应的 c 代码为：

```c
int main() {
    int x, y;
    scanf("%d %d", &x, &y);
    int temp = x ^ y;
    printf("%d %d %d\n", temp, y, x);
    return 0;
}
```

Q3(2)

- 画出Line 24执行前栈的状态，以及此时寄存器%edi, %esi, %edx, %ecx, %rsp的值；
- 假设进入main函数前%rsp的值为0x8000420（代码中出现的局部变量，要标记在栈图中；图中标记内存地址）

Answer3(2)

结合一开始的 `subq $24, %rsp` 指令可知，在栈中 `main` 函数返回地址下开辟了 `24` 字节的空间，其中 `a` 在新 `%rsp` 高 `8` 字节处， `b` 在新 `rsp` 高 `12` 字节处，于是可以画出示意图：

```
高地址
|
| 返回地址                <- 0x8000420
|
├──────────────────────
| 未使用                  <- 0x8000418
|
|
├──────────────────────
| int b  (scanf 第二个)   <- 0x8000414
├──────────────────────
| int a  (scanf 第一个)   <- 0x8000410
├──────────────────────
| 未使用                  <- 0x8000408
|
└── rsp
低地址
```