编译原理第一次作业

秦沁

中国科学技术大学
University of Science and Technology of China

第一题(a)

```
1  .file  “foo.c”  ; 原始文件名是“foo.c”
2      .text ; 代码段的开始
3  .globl fact ; 定义全局标签fact
4      .type fact,@function ; 定义fact的类型为function
5  fact:
6      pushl %ebp ; 基址指针入栈
7      movl %esp, %ebp ; 把栈顶指针的值赋给基址指针
8      subl $4, %esp ; 在栈上分配4个字节的空间
9      cmpl $0, 8(%ebp) ; ebp的值加8得到一个地址，这个地址的值和0比较
10     jg .L2 ; 如果大于0，跳转到.L2
11     movl $1, -4(%ebp) ; 把1赋给ebp的值减4得到的地址
12     jmp .L1 ; 无条件跳转到.L1
13 .L2:
14     subl $12, %esp ; 在栈上分配12个字节的空间
```

ebp：基址指针
esp：栈顶指针

第一题(a)

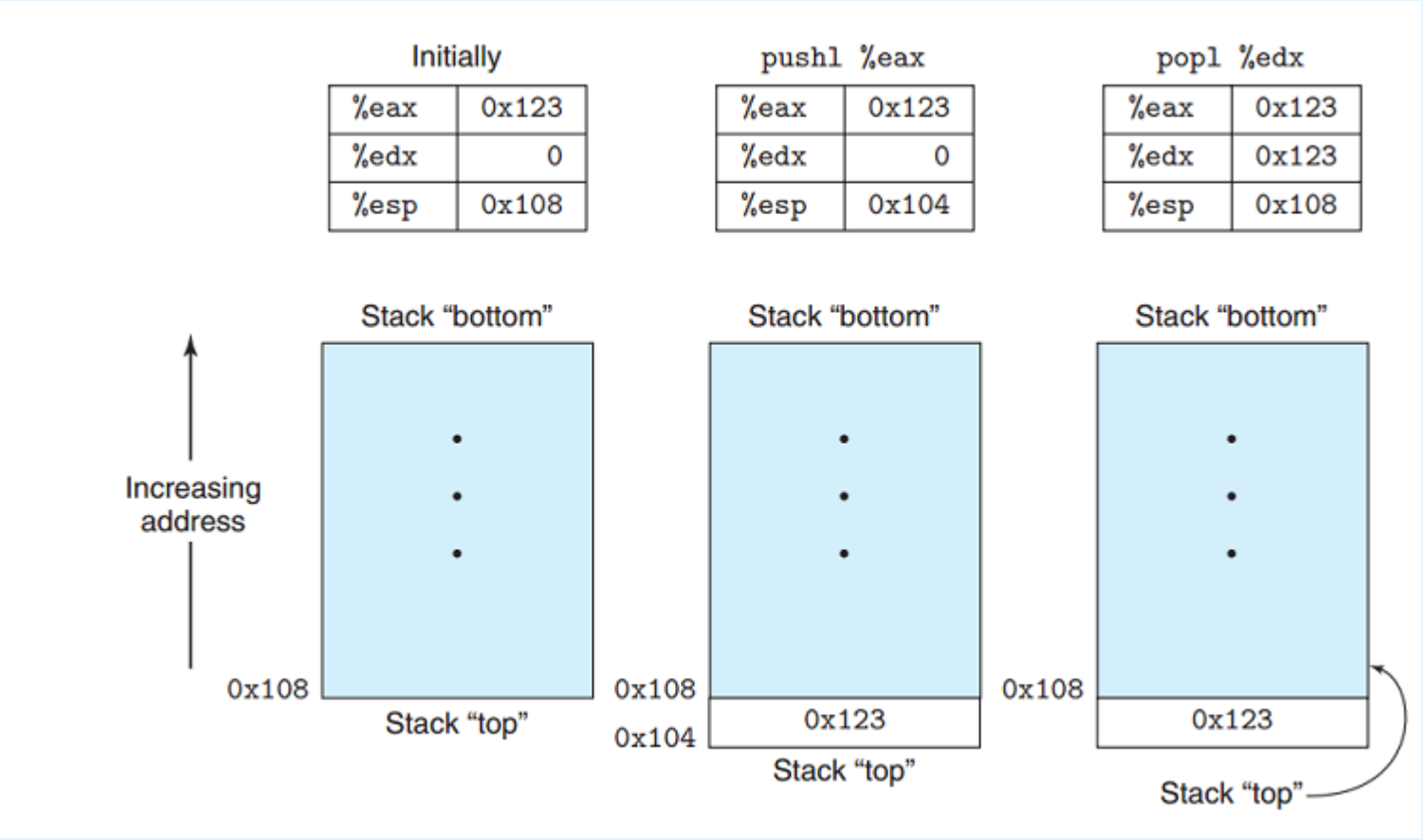
```
15    movl 8(%ebp), %eax ; 把n赋给eax寄存器
16    decl %eax ; eax的值减1
17    pushl %eax ; 把eax寄存器的值压入栈中
18    call fact ; 调用fact函数
19    addl $16, %esp ; 在栈上释放16个字节的空间
20    imull 8(%ebp), %eax ; 把n和eax的值相乘，结果赋给eax
21    movl %eax, -4(%ebp) ; 把eax的值赋给ebp的值减4得到的地址
22 .L1:
23    movl -4(%ebp), %eax ; 把函数返回值赋给eax寄存器
24    leave ; 等价于movl %ebp %esp; popl %ebp
25    ret ; 返回主程序
26 .Lfe1: ; fact函数结束
27    .size fact,.Lfe1-fact ; fact函数的大小
28    .ident "GCC: (GNU) 3.2.2 20030222 (Red Hat Linux 3.2.2-5)"
      ; 编译器版本
```

第一题(a)

摘自CSDN：

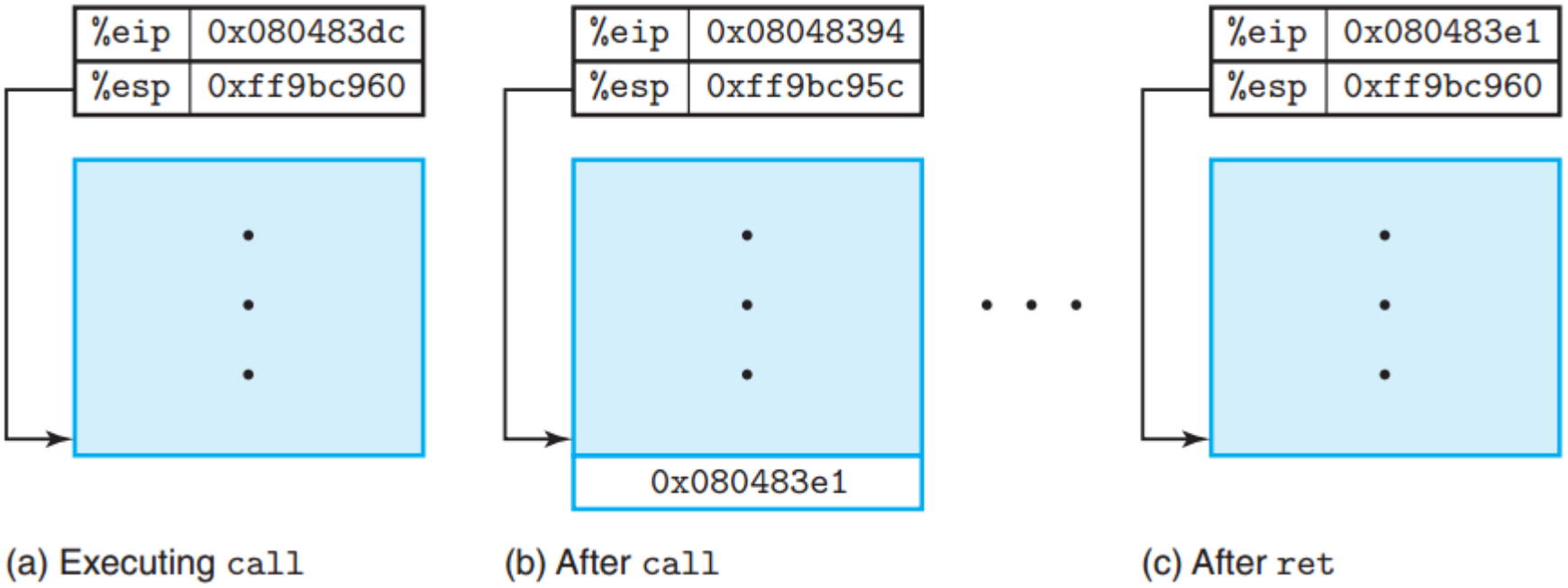
pushl %ebp等价于：
subl $4, %esp
movl %ebp (%esp)

popl %ebp等价于：
movl (%esp), %ebp
addl $4, %esp



第一题(a)

LEAVE 等价于：
movl %ebp %esp
popl %ebp



(a) Executing call (b) After call (c) After ret

CALL：首先将返回地址（也就是call指令要执行时EIP的值）压入栈顶，然后将程序跳转到当前调用的方法的起始地址。执行push和jump指令。

RET：将栈顶的返回地址弹出到EIP，然后按照EIP此时指示的指令地址继续执行程序。

第一题(b)

- n在汇编中是如何表示的：8(%ebp)，即在栈上基址指针偏移8的位置。
- 函数返回值放于何处：存放在寄存器eax中
- if语句对应哪几条汇编代码：

```
9       cmpl $0, 8(%ebp) ；判断n是否大于0
10      jg .L2 ; n>0的情况
11      movl $1, -4(%ebp) ; else的情况
12      jmp .L1
```

第一题(c.a)

```
1   int fact( int n )
2       {
3           if (n<= 0) return 1;
4           else return ( n*fact(n-1));
5       }
```

```
1   fact(int):
2           push    rbp
3           mov     rbp, rsp
4           sub     rsp, 16
5           mov     DWORD PTR [rbp-4], edi
6           cmp     DWORD PTR [rbp-4], 0
7           jg      .L2
8           mov     eax, 1
9           jmp     .L3
10  .L2:
11          mov     eax, DWORD PTR [rbp-4]
12          sub     eax, 1
13          mov     edi, eax
14          call    fact(int)
15          imul    eax, DWORD PTR [rbp-4]
16  .L3:
17          leave
18          ret
```

第一题(c.a)

```
1  fact(int):
2          push    rbp ; 基址指针入栈
3          mov     rbp, rsp ; 把栈顶指针的值赋给基址指针
4          sub     rsp, 16 ; 在栈上分配16个字节的空间
5          mov     DWORD PTR [rbp-4], edi ; 把n赋给ebp的值减4得到的地址
6          cmp     DWORD PTR [rbp-4], 0 ; 判断n是否大于0
7          jg      .L2 ; 如果大于0，跳转到.L2
8          mov     eax, 1 ; 把1赋给eax寄存器
9          jmp     .L3 ; 无条件跳转到.L3
```

第一题(c.a)

```
10 .L2:
11      mov     eax, DWORD PTR [rbp-4] ; 把n赋给eax寄存器
12      sub     eax, 1 ; eax的值减1
13      mov     edi, eax ; 把eax的值赋给edi寄存器
14      call    fact(int) ; 调用fact函数
15      imul    eax, DWORD PTR [rbp-4]
        ; 把n和eax的值相乘，结果赋给eax
16 .L3:
17      leave ; 等价于mov rsp, rbp; pop rbp
18      ret ; 返回主程序
```

第一题(c.b)

- n在汇编中是如何表示的：存放在寄存器edi中
- 函数返回值放于何处：存放在寄存器eax中
- if语句对应哪几条汇编代码：

```
6       cmp     DWORD PTR [rbp-4], 0 ; 判断n是否大于0
7       jg      .L2 ; n>0的情况
8       mov     eax, 1 ; else的情况
9       jmp     .L3
```

第二题

1. int (**ap[20])[30];

ap是一个有20个元素的数组，数组的每个元素是一个指向指针的指针，这些指针指向一个含30个元素的数组，数组的每个元素是整型变量。

2. int (*fpa(int i,int *j))[20];

fpa是一个函数。它有两个参数，一个参数是整型，另一个参数是指向整型的指针；它的返回值是一个指针，指向一个含20个元素的数组，数组的每个元素是整型变量。

第二题

3. int * fa(int i)[20];
错误：函数的返回值不能是数组。

4. int af[20](int k);
错误：不能使用函数数组。

第二题

5. void (*(*paa)[10])(int a);

paa是一个指针，它指向一个含10个元素的数组，数组的每一个元素是指向函数的指针，函数有一个整型参数，没有返回值。

6. void (*afp[10]) (int b);

afp是一个有10个元素的数组，每个元素是一个指向函数的指针，函数有一个整型参数，没有返回值。

第二题

解题思路：

- 优先级：

  () > [] > * > int

  符号依次与标识符结合（注意标识符的定义）。

- 助记：
  - int *p[10];　VS　int (*p)[10];
  - int *a(int b);　VS　int (*a)(int b);

参考资料

- https://blog.csdn.net/striver1205/article/details/25695437
- https://blog.csdn.net/weixin_56462041/article/details/128069491
- https://blog.csdn.net/weixin_56462041/article/details/127721426

谢谢！

祝开心~