# 题目1

| %exa | 0x10000000 |
|---|---|
| %ecx | 22 |
| $0x10000004 | 0x10000004 |
| 0x10000012 | None |
| 0xFFFFFF8 | None |
| (%eax,%ecx,8) | 44 |

# 第二题

```
int dw_loop(int x, int y, int n) {
     do{
        x+=n;
        y*=n;
         n--;
      }while (n>0&&y<n);
      return x;
}
```

# 第三题

## 1

```
movl %rdi %rcx
movl %rsi %rdx
movl %rcx %rax
movl %rdx %rbx
imul %rdx %rax
addl %rcx %rbx
imul %rdx %rbx
cmpl %rcx %rdx
cmovge %rbx %rax
ret
```

## 2

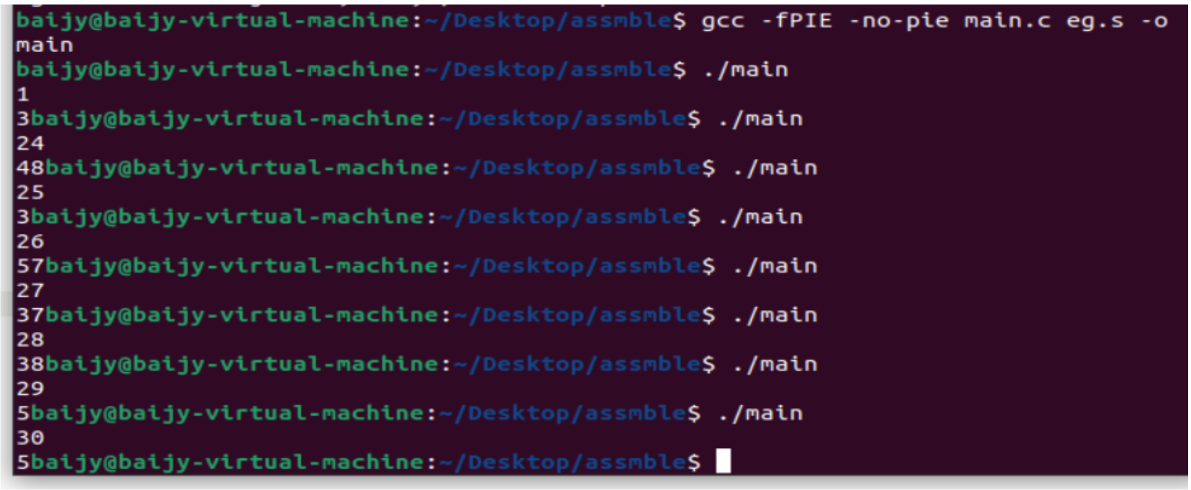
可能是因为对于这种情况下，一个要做两次运算，一个只做一次运算，不用cmove效率反而更高?

# 第四题

```
.section    .rodata
    .align 8
.jpt:
.quad   .to24
.quad   .default
.quad   .to26
.quad   .to27_28
.quad   .to27_28
.quad   .to29_30
.quad   .to29_30

.section .text
.global eg
eg:
push %rbp
mov %rsp,%rbp
mov %rdi,%rax
mov $0,%rdx
cmp $30,%rax
jg .default
cmp $24,%rax
jl .default
movq %rax,%rbx
subq $24,%rbx
lea .jpt(,%rbx,8),%rcx
jmp *(%rcx)
.to24:
lea (,%rax,2),%rdx
jmp .end
.to27_28:
lea 10(%rax),%rdx
jmp .end
.to26:
lea (,%rax,2),%rdx
.to29_30:
addq $5,%rdx
jmp .end
.default:
movq $3,%rdx
.end:
movq %rdx,%rax
pop %rbp
ret
```

运行结果：



```
baijy@baijy-virtual-machine:~/Desktop/assmble$ gcc -fPIE -no-pie main.c eg.s -o
main
baijy@baijy-virtual-machine:~/Desktop/assmble$ ./main
1
3baijy@baijy-virtual-machine:~/Desktop/assmble$ ./main
24
48baijy@baijy-virtual-machine:~/Desktop/assmble$ ./main
25
3baijy@baijy-virtual-machine:~/Desktop/assmble$ ./main
26
57baijy@baijy-virtual-machine:~/Desktop/assmble$ ./main
27
37baijy@baijy-virtual-machine:~/Desktop/assmble$ ./main
28
38baijy@baijy-virtual-machine:~/Desktop/assmble$ ./main
29
5baijy@baijy-virtual-machine:~/Desktop/assmble$ ./main
30
5baijy@baijy-virtual-machine:~/Desktop/assmble$ 
```

但注意到，编译时需要用以下命令才能成功：

`gcc -fPIE -no-pie main.c eg.s -o main`