题目 1

%eax	0x10000000
%ecx	22
$0x10000004	268435460
0x10000012	0x50000
0xFFFFFF8	None
(%eax, %ecx, 8)	44

题目 2

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

题目 3

假设参数 x 和 y 分别存放在寄存器%edi 和%esi 中

```
movl  %edi, %eax      # 将 x 复制到 %eax
imull %esi, %eax      # %eax = x * y（计算第一个表达式）
movl  %edi, %ecx      # 将 x 复制到 %ecx
addl  %esi, %ecx      # %ecx = x + y
imull %esi, %ecx      # %ecx = (x + y) * y（计算第二个表达式）
cmpl  %esi, %edi      # 比较 x 和 y（设置条件码）
cmovge %ecx, %eax     # 如果 x >= y，则将 %ecx 的值移动到 %eax（否则 %eax 保持 x * y）
ret                   # 返回
```

编译器未使用 cmov 指令可能是由于在这个过程中，浪费的计算所需的性能大于由于分支预测错误所造成的性能处罚，特别是这个例子中，x*y 和 (x+y)*y 的计算均涉及乘法指令，计算开销较大。

题目 4

假设参数 x 和 result 分别存放在寄存器%rsi 和%rax 中

```
    .section .rodata
    .align 8
.L1:
    .quad .L2   #case 24
```

```
    .quad .L6   #case 25
    .quad .L4   #case 26
    .quad .L3   #case 27
    .quad .L3   #case 28
    .quad .L5   #case 29
    .quad .L5   #case 30
switch:
    subq $24, %rsi
    cmpq $6, %rsi
    ja .L6
    jmp *.L1(,%rsi,8)
.L2:                        #case 24
    leaq (,%rsi,2),%rax
    jmp .L7
.L3:                        #case 27,28
    leaq 10(%rsi),%rax
    jmp .L7
.L4:                        #case 26
    leaq (,%rsi,2),%rax
    jmp .L5
.L5:                        #case 29,30
    addq $5, %rax
    jmp .L7
.L6:                        #default
    movq $3, %rax
    jmp .L7
.L7:                        #exit
    ret
```