

# 寄存器大小记忆

规律：

- b: 结尾为l
- w: 两个字符
- d: e开头
- r: r开头

%rax	%eax	%ax	%al
%rbx	%ebx	%bx	%bl
%rcx	%ecx	%cx	%cl
%rdx	%edx	%dx	%dl
%rsi	%esi	%si	%sil
%rdi	%edi	%di	%dil
%rbp	%ebp	%bp	%bpl
%rsp	%esp	%sp	%spl

# 易错点

- 复制double word
- movs与movz
- cltq

# T1详解

(1)从%ax复制16b: 0x0...0CDEF

(2)做有符号扩充, %bx最高位是C, 即1100, 于是扩1: 0xF...FCDEF

(3)注意复制32b的时候, 高位也会被清0: 0x00000000FFFCDEF

(4)Clcq = movslq %eax %rax: 0xF...F89ABCDEF

(5)~(8)查表填入对应大小即可。

(5)w

(6)wq

(7)l

# T2

- Long
- 一般机器（比如intel & ios）都是小端，题目多数都是在这个环境下的，不过提到网络传输一定要反应过来是大端
- 推导过程

# T2详解

转化后的C语言代码如右图

先定位出汇编代码中各jump的位置，然后将两份代码的代码块对应上

然后细化定位，还可以根据比较明显的立即数，和代码中出现较少的位运算辅助定位。

```
y=0;
if(x>1)
    goto L1;
return x;
m=(0x40000000<<32)+0;
goto test;

L2:
b=y+m;
y>>=1;
if(x<b)
    goto L3;
x-=b;
y+=m;
L3:
m>>=2;
test:
if(m!=0)
    goto L2;

return y;
```

# T2答案

(1)\$0x0

(2)4004e1

(3)40052f

(4)\$0x40000000

(5)400524

(6)(%rdx,%rax,1)

(7)40051f

(8)\$0x2

(9)4004f1

(10)%rax

# \*T3附加练习

8. 将下列汇编代码翻译成 C 代码

```
func:
    movl    $1, %eax
    jmp .L2
.L4:
    testb   $1, %sil
    je      .L3
    imulq   %rdi, %rax
.L3:
    sarq    %rsi
    imulq   %rdi, %rdi
.L2:
    testq   %rsi, %rsi
    jg      .L4
    rep ret
```

```
// a in %rdi, b in %rsi
long func(long a, long b) {
    long ans = _____;
    while (_____) {
        if (_____)
            ans = _____;
        b = _____;
        a = _____;
    }
    return ans;
}
```

## \*T3详解

由汇编第一句可得 $ans=1$ ，并且确定 $ans$ 为 $\%eax$

While结构，则L2为test判断，则为 $b>0$

由L2后的jg .L4推出L4为while正文，正文部分一一对应即可  
答案

```
long func(long a, long b) {  
    long ans = 1;  
    while (b > 0) {  
        if (b & 1)  
            ans = ans * a;  
        b = b >> 1;  
        a = a * a;  
    }  
    return ans;  
}
```