3．

（1）后缀：w，源操作数：基址+比例变址+偏移， 目的操作数：寄存器

（2）后缀：b，源操作数：寄存器，        目的操作数：基址+偏移

（3）后缀：l， 源操作数：比例变址，     目的操作数：寄存器

（4）后缀：b，源操作数：基址，      目的操作数：寄存器

（5）后缀：l， 源操作数：立即数，     目的操作数：栈

（6）后缀：l， 源操作数：立即数，     目的操作数：寄存器

（7）后缀：w，源操作数：寄存器，     目的操作数：寄存器

（8）后缀：l， 源操作数：基址+变址+偏移，   目的操作数：寄存器

5.

movsbl %al, (%edx)

movb %al, (%edx)

movl %eax, (%edx)

movswl %ax, (%edx)

movzbl %al, (%edx)

movsbl %al, (%edx)

movl %eax, (%edx)

6.

（1）xptr、yptr和zptr对应实参所存放的存储单元地址分别为：R[ebp]+8、R[ebp]+12、

R[ebp]+16。

 （2）函数func的C语言代码如下：

void func(int \*xptr, int \*yptr, int \*zptr)   {

int tempx=\*xptr;

int tempy=\*yptr;

int tempz=\*zptr;

\*yptr=tempx;

\*zptr = tempy;

\*xptr = tempz;

 }

8.

（1）EDX中的内容改变为0x00000070。加法指令会影响OF、SF、ZF和CF标志。OF=0，ZF=0，SF=0，CF=1

（2）ECX中的内容改为0x80000008。减法指令会影响OF、SF、ZF和CF标志。OF=1，ZF=0，SF=1，CF=10=1。

（3）BX中的内容改为0xFF00。，OR指令执行后OF=CF=0；因为结果不为0，故ZF=0；因为最高位为1，故SF=1。

（4）test指令不改变寄存器。TEST指令执行后OF=CF=0；因为结果不为0，故ZF=0；因为最高位为1，故SF=1。

（5）单元0x8049380中的内容改变为0x11e25500。显然，这个结果是溢出的。乘法指令不影响标志位，也即并不会使OF=1。

（6）指令执行后CX中的内容从0x0010变为0x000F。DEC指令会影响OF、ZF、SF，根据上述运算结果，得到OF=0，ZF=0，SF=0。

11.

（1）因为je指令的操作码为01110100，所以机器代码7408H中的08H是偏移量，故转移目标地址为：0x804838c+2+0x8=0x8048396。

call指令中的转移目标地址0x80483b1=0x804838e+5+0x1e，由此，可以看出，call指令机器代码中后面的4个字节是偏移量，因IA-32采用小端方式，故偏移量为0000001EH。call指令机器代码共占5个字节，因此，下条指令的地址为当前指令地址0x804838e加5。

（2）jb指令中F6H是偏移量，故其转移目标地址为：0x8048390+2+0xf6=0x8048488。

movl指令的机器代码有10个字节，前两个字节是操作码等，后面8个字节为两个立即数，因为是小端方式，所以，第一个立即数为0804A800H，即汇编指令中的目的地址0x804a800，最后4个字节为立即数00000001H，即汇编指令中的常数0x1。

（3）jle指令中的7EH为操作码，16H为偏移量，其汇编形式中的0x80492e0是转移目的地址，因此，假定后面的mov指令的地址为x，则x满足以下公式：0x80492e0=x+0x16，故x=0x80492e0-0x16=0x80492ca。

（4）jmp指令中的E9H为操作码，后面4个字节为偏移量，因为是小端方式，故偏移量为FFFFFF00H，即-100H=-256。后面的sub指令的地址为0x804829b，故jmp指令的转移目标地址为0x804829b+0xffffff00=0x804829b-0x100=0x804819b。

14.

（1）每个入口参数都要按4字节边界对齐，因此，参数x、y和k入栈时都占4个字节。

1  movw  8(%ebp), %bx  //R[bx]←M[R[ebp]+8]，将x送BX

2  movw  12(%ebp), %si  //R[si]←M[R[ebp]+12]，将y送SI

3  movw 16(%ebp), %cx //R[cx]←M[R[ebp]+16]，将k送CX

4  .L1:

5  movw   %si, %dx  //R[dx]←R[si]，将y送DX

6   movw   %dx, %ax  //R[ax]←R[dx]，将y送AX

7  sarw  $15, %dx  //R[dx]←R[dx]>>15，将y的符号扩展16位送DX

8  idiv   %cx   //R[dx]←R[dx-ax]÷R[cx]的余数，将y%k送DX

//R[ax]←R[dx-ax]÷R[cx]的商，将y/k送AX

9    imulw   %dx, %bx  //R[bx]←R[bx]\*R[dx]，将x\*(y%k) 送BX

10   decw    %cx   //R[cx]←R[cx]-1，将k-1 送CX

11   testw  %cx, %cx  //R[cx] and R[cx]，得OF=CF=0，负数则SF=1,零则ZF=1

12  jle      .L2    //若k小于等于0，则转.L2

13  cmpw   %cx, %si  //R[si] - R[cx]，将y与k相减得到各标志

14   jg      .L1    //若y大于k，则转.L1

15  .L2:

16   movswl  %bx, %eax  // R[eax]←R[bx]，将x\*(y%k) 送AX

（2）被调用者保存寄存器有BX、SI，调用者保存寄存器有AX、CX和DX。

在该函数过程体前面的准备阶段，被调用者保存的寄存器EBX和ESI必须保存到栈中。

（3）因为执行第8行除法指令前必须先将被除数扩展为32位，而这里是带符号数除法，因此，采用算术右移以扩展16位符号，放在高16位的DX中，低16位在AX中。

18.

每次执行pushl指令后，R[esp]=R[esp]-4，因此，第2行指令执行后R[esp]=0xbc00001c。

    （1）执行第3行指令后，R[ebp]=R[esp]=0xbc00001c。到第12条指令执行结束都没有改变EBP的内容，因而

执行第10行指令后，EBP的内容还是为0xbc00001c。执行第13行指令后，EBP的内容恢复为进入函数funct时的值0xbc000030。

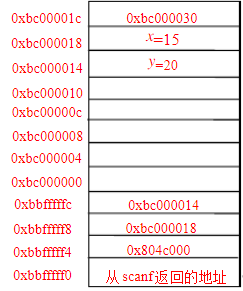
    （2）执行第3行指令后，R[esp]=0xbc00001c。执行第4行指令后R[esp]= R[esp]-40=0xbc00001c-0x28=0xbbfffff4。

因而执行第10行指令后，未跳转到scanf函数执行时，ESP中的内容为0xbbfffff4-4=0xbbfffff0；在从scanf函数返回后ESP中的内容为0xbbfffff4。执行第13行指令后，ESP的内容恢复为进入函数funct时的旧值，即R[esp]=0xbc000020。

    （3）第5、6两行指令将scanf的第三个参数&y入栈，入栈的内容为R[ebp]-8=0xbc000014；第7、8两行指令

将scanf的第二个参数&x入栈，入栈的内容为R[ebp]-4=0xbc000018。故x和y所在的地址分别为0xbc000018和0xbc000014。

    （4）执行第10行指令后，funct栈帧的地址范围及其内容如下：



17.

根据第2、3行指令可知，参数a是char型，参数p是指向short型变量的指针；根据第4、5行指令可知，参数

b和c都是unsigned short型，根据第6行指令可知，test的返回参数类型为unsigned int。因此，test的原型为：     unsigned int test(char a, unsigned short b, unsigned short c, short \*p);

19.

 x==0

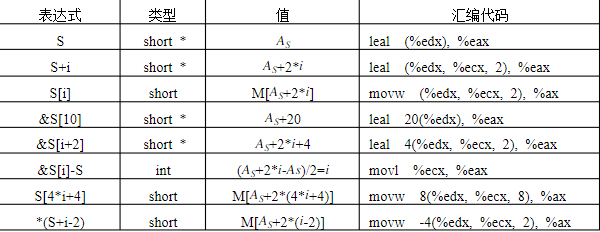
0

X》1

(x & 0x1) + rv

该函数的功能为计算x的各个数位中1的个数。

21.



22.

根据汇编指令功能可以推断最终在EAX中返回的值为：

M[a+28\*i+4\*j]+M[b+20\*j+4\*i]，因为数组a和b都是int型，每个数组元素占4B，因此，M=5, N=7。

23.

执行第11行指令后，a[i][j][k]的地址为a+4\*(63\*i+9\*j+k)，所以，可以推断出M=9，N=63/9=7。根据第12行指令，可知数组a的大小为4536字节，故L=4536/(4\*L\*M)=18。

28.

每个成员的偏移量如下：

c   d   i   s   p   l   g   v

0   8  16  20  24  28 32  40

结构总大小为48字节，因为其中的d和g必须是按8字节边界对齐，所以，必须在末尾再加上4个字节，即44+4=48字节。变量长度按照从大到小顺序排列，可以使得结构所占空间最小，因此调整顺序后的结构定义如下： struct {

double  d;

long long  g；

int   i;

char   \*p;

long   l;

void   \*v；

short  s;

char   c;

   } test；

d   g   i   p   l   v   s   c

0   8  16  20  24  28 32  34   结构总大小为34+6=40字节。

31.

（1）汇编指令注释如下：

1 movl 8(%ebp), %edx  //R[edx]←M[R[ebp]+8]，将x送EDX

2 movl 12(%ebp), %ecx  //R[ecx]←M[R[ebp]+12]，将k送ECX

3 movl $255, %esi   //R[esi]←255，将255送ESI

4 movl $-2147483648, %edi //R[edi]←-2147483648，将0x80000000送EDI

5  .L3:

6 movl %edi, %eax

 //R[eax]←R[edi]，将i送EAX

7 andl  %edx, %eax   //R[eax]←R[eax] and R[edx]，将i and x送EAX

8 xorl  %eax, %esi   //R[esi]←R[esi] xor R[eax]，将val xor (i and x)送ESI

9 movl %ecx, %ebx   //R[ebx]←R[ecx]，将k送ECX

10 shrl  %bl, %edi   //R[edi]←R[edi] >> R[bl]，将i逻辑右移k位送EDI

11 testl  %edi, %edi

12 jne  .L3   //若R[edi]≠0，则转.L3

13 movl %esi, %eax   //R[eax]←R[esi]

（2）x和k分别存放在EDX和ECX中。局部变量val和i分别存放在ESI和EDI中。

（3）局部变量val和i的初始值分别是255和-2147483648。

（4）循环终止条件为i等于0。循环控制变量i每次循环被逻辑右移k位。

（5）

255

 -2147483648

i != 0

(unsigned) i >> k

(i & x)