1、（1）R[ebp]+8、R[ebp]+12、R[ebp]+16

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

{

int x = \*xptr;

int y = \*yptr;

int z = \*zptr;

\*yptr = x;

\*zptr = y;

\*xptr = z;

}

2、（1）je指令的转移目标地址为0x804838c+0x8=0x8048396。

call指令中的转移目标地址0x80483b1=0x804838e+5+0x1e,其中5为call指令所占字节数，0x1e为偏移量。

（2）jb 指令的转移目标地址是0x8048390+2+0xf6=0x8048488。

movl指令的机器代码中，前两个字节为操作码，第一个立即数为0804A800H，第二个立即数为00000001H。

（3）mov指令的地址为0x80492e0-0x16=0x80492ca。

（4）jmp指令的转移目标地址为0x804829b+0xffffff00=0x804819b。

3、int f1(unsigned x)

{

int y = 0;

while(x!=0){

y ^= x;

x >>= 1;

}

return y&0x1;

}

函数f1的功能：当x中有奇数个1，则返回1，否则返回0。

4、（1）执行第3行指令后，R[ebp]=0xbc00001c;

执行第10行指令后，R[ebp]=0xbc00001c;

执行第13行指令后，R[ebp]=0xbc000030。

（2）执行第3行指令后，R[esp]=0xbc00001c;

执行第4行指令后，R[esp]=0xbc00001c-0x28=0xbbfffff4;

执行第10行指令后，R[esp]=0xbbfffff4;

执行第13行指令后，R[esp]=0xbc000020。

（3）x所在的地址为R[ebp]-4=0xbc000018,y所在的地址为R[ebp]-4=0xbc000014。

（4）

|  |  |  |
| --- | --- | --- |
| 0xbc00001c | 0xbc000030 | ebp |
| 0xbc000018 | x=15 |  |
| 0xbc000014 | y=20 |  |
| 0xbc000010 |  |  |
| 0xbc00000c |  |  |
| 0xbc000008 |  |  |
| 0xbc000004 |  |  |
| 0xbc000000 |  |  |
| 0xbbfffffc | 0xbc000014 |  |
| 0xbbfffff8 | 0xbc000018 |  |
| 0xbbfffff4 | 0x804c000 |  |
| 0xbbfffff0 | 从scanf返回的地址 | esp |

5、执行第8行指令后，R[eax]=M[a+28\*i+4\*j];

执行第9行指令后，R[eax]= M[a+28\*i+4\*j]+M[b+20\*j+4\*i],因此，M=5,N=7。

6、（1）M=76/4=19，存放在EDI中，变量j存放在ECX中。

（2）

1. void trans\_matrix(int a[M][M])
2. {
3. int i, j, t, \* p;
4. int c = (M << 2);
5. for (i = 0; i < M; i++)
6. {
7. p = &a[0][i];
8. for (j = 0; j < M; j++)
9. {
10. t = \*p;
11. \*p = a[i][j];
12. a[i][j] = t;
13. p += c;
14. }
15. }
16. }

7、

|  |  |  |
| --- | --- | --- |
| 表达式EXPR | TYPE类型 | 汇编指令序列 |
| uptr->s1.x | int | movl (%eax),%eax  movl %eax,(%edx) |
| uptr->s1.y | short | movw 4(%eax),%ax  movw %ax,(%edx) |
| &uptr->s1.z | short\* | leal 6(%eax),%eax  movw %eax,(%edx) |
| uptr->s2.a | short\* | movl %eax,(%edx) |
| uptr->s2.a[uptr->s2.b] | short | movl 4(%eax),%ecx  movl (%eax,%ecx,2),%eax  movl %eax,(%edx) |
| \*uptr->s2.p | char | movl 8(%eax),%eax  movb (%eax),%al  movb %al,(%edx) |

8、（1）

1 movl 8(%ebp), %edx //R[edx]<-M[R[ebp]+8]

2 movl 12(%ebp), %ecx //R[ecx]<-M[R[ebp]+12]

3 movl $255, %esi //R[esi]<-255

4 movl $-2147483648, %edi //R[edi]🡨2147483648

5 .L3:

6 movl %edi, %eax //R[eax]<-R[edi]

7 andl %edx, %eax //R[eax]<-R[eax] and R[edx]

8 xor1 %eax, %esi //R[esi]<-R[esi] xor R[eax]

9 movl %ecx, %ebx //R[ebx]<-R[ecx]

10 shrl %bl, %edi //R[edi]<-R[edi] >> R[bl]

11 testl %edi, %edi

12 jne .L3 //若R[edi]≠0,则转.L3

13 movl %esi, %eax //R[eax]<-R[esi]

（2）x放在EDX中，k放在ECX中，val放在ESI中，i放在EDI中。

（3）val的初值为255，i的初值为-2147483648。

（4）循环终止条件为i=0。循环控制变量i每次循环被逻辑右移k位。

（5）

1. int Iproc(int x, int k)
2. {
3. int val = 255;
4. int i;
5. for (i = -2147483648; i != 0; i = (unsigned)i >> k)
6. {
7. val ^= (i & x);
8. }
9. return val;
10. }

9、（1）n1.ptr、n1.data1、n2.data2、n2.next的偏移量分别是0、4、0、4。

（2）8字节。

（3）uptr->n2.next->n1.data1=\*(uptr->n2.next->n1.ptr)-uptr->n2.data2