

ICS homework3

张运吉 211300063

3/11/2022

1 P6

(1). $R[ebp] + 8, R[ebp] + 12, R[ebp] + 16$

(2).

```
1 void func(int *xptr, int *yptr, int *zptr) {  
2     int x = *xptr;  
3     int y = *yptr;  
4     int z = *zptr;  
5     *yptr = x;  
6     *zptr = y;  
7     *xptr = z;  
8  
9 }
```

2 P8

(1). 指令功能: $R[edx] \leftarrow R[edx] + M[R[eax]]$

即 $R[edx] \leftarrow 0xffffffff0 + 0x80$

所以 EDX 寄存器的值变为 $0x70$

$OF = 0, ZF = 0, SF = 0, CF = 1$

(2). 指令功能: $R[ecx] \leftarrow R[ecx] - M[R[eax] + R[ebx]]$

即 $R[ecx] \leftarrow 0x10 - 0x80000008 = 0x10 + 0x7fffffff8 = 0x80000008$

所以 ECX 寄存器的值变为 $0x80000008$

$OF = 1, ZF = 0, SF = 1, CF = 1$

(3). 指令功能: $R[bx] \leftarrow R[bx] + M[R[ecx] + 8 * R[ecx] + 4]$

即 $R[bx] \leftarrow 0x0100|0xf00 = 0xf00$

所以 BX 寄存器的值变为 $0xf00$

$OF = 0, ZF = 0, SF = 1, CF = 0$

(4). 指令功能: 根据运算结果改变条件寄存器中的条件标记

$\therefore 0x80 \& R[dl] = 0x80 \& 0x80 = 0x80$

$\therefore OF = 0, ZF = 0, SF = 1, CF = 0$

(5). 指令功能: $R[ecx] \leftarrow M[R[ecx] + R[edx]] * 32$

即 $R[ecx] \leftarrow 0x908f12a8 * 32 = 0x11e25500$

所以 ECX 寄存器的值变为 $0x11e25500$

$OF = 1, ZF = 0, SF = 1, CF = 1$

(6). 指令功能: $R[dx - ax] \leftarrow R[ax] * R[bx]$

即 $R[dx - ax] \leftarrow 0x9300 * 0x0100 = 0x00930000$

所以 DX 寄存器的值变为 $0x0093$, AX 寄存器的值变为 $0x0000$

$OF = CF = 1$

(7). 指令功能: $R[cx] \leftarrow R[cx] - 1$

即 $R[cx] \leftarrow 0x0010 - 1 = 0x0010 + 0xffff = 0x000f$

所以 CX 寄存器的值变为 $0x000f$

$OF = 0, ZF = 0, SF = 0$

3 P9

1.movl 12(%ebp),%ecx // $R[ecx] \leftarrow M[R[ebp] + 12]$, y 送 ECX

2.sall \$8,%ecx // $R[ecx] \leftarrow R[ecx] << 8$, $y * 256$ 送 ECX

3.movl 8(%ebp),%eax // $R[ecx] \leftarrow M[R[ebp] + 8]$, x 送 EAX

4.movl 20(%ebp),%edx // $R[edx] \leftarrow M[R[ebp] + 20]$, k 送 EDX

5.imull %edx,%eax // $R[ecx] \leftarrow R[ecx] + R[edx]$, kx 送 EAX

6.movl 16(%ebp),%edx // $R[edx] \leftarrow M[R[ebp] + 16]$, z 送 EDX

7.andl \$65520,%edx // $R[edx] \leftarrow R[edx] \& 65520$, z & $0xffff0$ 送 EDX

8.andl %ecx,%edx // $R[edx] \leftarrow R[edx] \& R[ecx]$

9.subl %edx,%eax // $R[ecx] \leftarrow R[ecx] - R[edx]$

\therefore 第 3 行: $int \quad v = k * x - (y * 256 + z \& 0xffff0)$

4 P11

(1) 目标地址: $0x804838c + 2 + 0x08 = 0x8048396$

执行 call 指令时, $(PC) = 0x804838e + 5 = 0x8048393$

$\therefore 0x80483b1 = 0x8048393 + 0x1e$

\therefore 可以得出 call 指令后 4 个字节是偏移量, 第一个字节是操作码

转移目标地址 = $(PC) +$ 偏移量

(4) 目标地址: $0x804829b + 0xfffffff00 = 0x804819b$

5 P14

(1)

1.movw 8(%ebp),%bx // $R[bx] \leftarrow M[R[ebp] + 8]$, x 送 BX

2.movw 12(%ebp),%si // $R[si] \leftarrow M[R[ebp] + 12]$, y 送 AX

3.movw 16(%ebp),%cx // $R[cx] \leftarrow M[R[ebp] + 16]$, k 送 CX

4. .L1 :

5.movw %si,%dx // $R[dx] \leftarrow R[si]$, y 送 DX

6.movw %dx,%ax // $R[ax] \leftarrow R[dx]$, y 送 AX

7.sarw \$15,%dx // $R[dx] \leftarrow R[dx] \gg 15$, DX 中存 y 的符号

8.idiv %cx // $R[ax] \leftarrow R[dx - ax] \div R[cx]$, $y \div k$ 的商送 AX

// $R[dx] \leftarrow R[dx - ax] \% R[cx]$, $y \div k$ 的余数送 CX

9.imulw %dx,%bx // $R[bx] \leftarrow R[bx] - R[dx]$, $x*(y\%k)$ 送 BX

10.decw %cx // $R[cx] \leftarrow R[cx] - 1$, k-1 送 CX

11.testw %cx,%cx // $R[cx] \& R[cx]$, OF=CF=0

12.jle .L2 // 若 k 小于等于 0, 则装.L2

13.cmpw %cx,%si // $R[si] - R[cx]$, $y - k$

14.jg .L1 // 若 $R[si] > R[cx] (y > k)$ 转 L2

15..L2

16.movswl %bx,%eax // $R[eax] \leftarrow R[bx]$, $x*(y\%k)$ 送 AX

(2) 被调用者保存寄存器: BX, SI

调用者保存寄存器: AX, CX, DX

EBX 和 ESI 必须保存到栈中

(3) 使 DX 中保存 y 的符号, 这样在第 8 行的除法操作中 $R[dx-ax]$ 保存的就是 y 符号拓展后的机器数。

6 P16

对应关系:

$$x + 3 = 0/x = -3 \quad .L7$$

$$x + 3 = 1/x = -2 \quad .L2$$

$$x + 3 = 2/x = -1 \quad .L2$$

$$x + 3 = 3/x = 0 \quad .L3$$

$$x + 3 = 4/x = 1 \quad .L4$$

$$x + 3 = 5/x = 2 \quad .L5$$

$$x + 3 = 6/x = 3 \quad .L6$$

$$x + 3 = 7/x = 4 \quad .L6$$

switch-case 语句原型:

```
1      switch(x) {
2          case -2:case -1:
3              ... // 标号.L2
4              break;
5          case 0:
6              ... // 标号.L3
7              break;
8          case 1:
9              ... // 标号.L4
10             break;
11         case 2:
12             ... // 标号.L5
13             break;
14         case 4:
15             ... // 标号.L6
16             break;
17         default:
18             ... // 标号.L7
19     }
```

由此可知, 当 x 不属于前 6 种 case 时会执行 default 分支, 标号为-2 和-1 会执行同一个 case 分支

7 P17

由第 2、3 行可知：a 是 char 型，因为 movsbw 是字符串处理指令，且是移动一个字节，p 是 short * 型。

由第 4、5 行可知：b、c 是 unsigned short 型，因为 movzwl 是从 2 字节零扩展到 4 字节并移动。

由第 6 行可知返回类型是 unsigned int。

综上，test 的原型为：

*unsigned int test(char a, unsigned short b, unsigned short c, short * b)*

8 P18

8.1 18.1

$$R[ebp] = 0xbc000020 - 0x4 = 0xbc00001c$$

$$R[ebp] = 0xbc00001c$$

$$R[ebp] = 0xbc000030$$

8.2 18.2

$$R[esp] = r[ebp] = 0xbc00001c$$

$$R[esp] = 0xbc00001c - 40 - 4 = 0xbbffffff0$$

$$R[esp] = 0xbc00001c + 4 = 0xbc000020$$

8.3 18.3

$$x \text{ 的地址: } 0xbc00001c - 4 = 0xbc000018$$

$$y \text{ 的地址: } 0xbc00001c - 8 = 0xbc000014$$

8.4 18.4

0xbc000030
$x = 15$
$y = 20$
...
...
...
0xbc000014
0xbc000018
0x804c0000
从 scanf 的返回地址
...
...
...

最顶层一格地址是 0xbc00001c, 往下每走一格减 4, 格子“从 scanf 的返回地址”地址是 0xbfffffff0

9 P23

执行第 11 行指令后, $a[i][j][k]$ 的地址是 $a + 4 * (63i + 9j + k)$

$$\therefore MN = 63, N = 9 \Rightarrow M = 7$$

由第 12 行知 $sizeof(a) = 4536$

$$\therefore LMN = \frac{4536}{64} = 1134$$

$$\therefore L = \frac{1134}{63} = 18$$

综上, $L = 18, M = 7, N = 9$

10 P26

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

11 P27

s	c	i	d
0	2	4	8

S1: 总共 12 字节，按 4 字节边界对齐。

i	s	c	d
0	4	6	7

S2: 总共 8 字节，按 4 字节边界对齐。

c	s	i	d
0	2	4	8

S3: 总共 12 字节，按 4 字节边界对齐。

s	c
0	6

S4: 总共 8 字节，按 2 字节边界对齐。

c	s	i	d	e
0	4	8	12	16

S5: 总共 24 字节，按 4 字节边界对齐。

c	s	d
0	36	40

S3: 总共 44 字节，按 4 字节边界对齐。

12 P30

```

1 void abc(int c, long *a, int *b);
2 void abc(unsigned c, long *a, int *b);
3 void abc(long c, long *a, int *b);
4 void abc(unsigned long c, long *a, int *b);

```

13 P32

```

1 typedef struct {
2     int idx;
3     unsigned a[6];
4 } line_struct;

```

由第 11、12 行指令可知，x 数组所占空间为 $0xc8 - 4 = 196B$

$$\therefore LEN = \frac{196}{28} = 7$$