

计算机系统第三章作业

陈俊潼 10185101210

3.60

各个变量存储的寄存器如下：

x	n	result	mask
%rdi	%esi	%rax	%rdx

补全后的代码如下：

```

1  long loop(long x, int n)
2  {
3      long result = 0;
4      long mask;
5      for (mask = 1; mask != 0; mask = mask << 1) {
6          result |= x & mask;
7      }
8      return result;
9  }
```

2.63

查看并分析汇编代码：

rsi > 5

12 36

long switch_prob(long x, long n)
x in %rdi, n in %rsi

1	0000000000400590	<switch_prob>:			
2	400590:	48 83 ee 3c	sub	\$0x3c,%rsi	
3	400594:	48 83 fe 05	cmp	\$0x5,%rsi	
4	400598:	77 29	ja	4005c3 <switch_prob+0x33>	
5	40059a:	ff 24 f5 f8 06 40 00	jmpq	*0x4006f8(,%rsi,8)	$x = 48, 49, 50 \sim 53.$
6	4005a1:	48 8d 04 fd 00 00 00	lea	0x0(,%rdi,8),%rax	
7	4005a8:	00			
8	4005a9:	c3	retq		
9	4005aa:	48 89 f8	mov	%rdi,%rax	
10	4005ad:	48 c1 f8 03	sar	\$0x3,%rax	$x \gg 3$
11	4005b1:	c3	retq		
12	4005b2:	48 89 f8	mov	%rdi,%rax	$(x \ll 4 - x) * 2 + 75$
13	4005b5:	48 c1 e0 04	shl	\$0x4,%rax	
14	4005b9:	48 29 f8	sub	%rdi,%rax	$x * x$
15	4005bc:	48 89 c7	mov	%rax,%rdi	
16	4005bf:	48 01 af ff	imul	%rdi,%rdi	$48 +$
17	4005c3:	48 8d 47 4b	lea	0x4b(%rdi),%rax	$c1$
18	4005c7:	c3	retq		

这里是一个switch语句。第一行会把n减去48，后面的ja为无符号大于，所以减去后的数字也必须大于0。可以推测switch语句是从60开始的（上图截图草稿有误）。不同的跳转点对应的操作都比较简单。整理后补全后的switch语句如下：

```
long switch_prob(long x, long n) {
    long result = x;
    switch (n) {
        case 60:
        case 62:
            result = x * 8;
            break;
        case 63:
            result = x >> 3;
            break;
        case 64:
            x = x << 4 - x;
        case 65:
            x = x * x;
        default:
            result = x + 0x4B;
    }
    return result;
}
```

3.65

查看并分析汇编代码，得到答案如下：

```
1  .L6:
2  movq    (%rdx), %rcx
3  movq    (%rax), %rsi
4  movq    %rsi, (%rdx)
5  movq    %rcx, (%rax)
6  addq    $8, %rdx
7  addq    $120, %rax
8  cmpq    %rdi, %rax
9  jne     .L6
```

Handwritten annotations: $j++$ (pointing to line 6), $A[j+1][i], 120/8=15$ (pointing to line 7).

我们可以看到 GCC 把数组索引转换成了指针代码。

- 哪个寄存器保存着指向数组元素 $A[i][j]$ 的指针？ $\rightarrow \%rdx$
- 哪个寄存器保存着指向数组元素 $A[j][i]$ 的指针？ $\rightarrow \%rax$
- M 的值是多少？ $M=15$.

3.69

首句中 $0x120=288$ ，b_struct内的第一个int和第二个int相差了288个字节。

```

lea (%rdi,%rdi,4), %rax      # 这一句将i*5
lea (%rsi,%rax,8), %rax      # 这一句将bp + i*40赋值给了%rax, %rax为ap
mov 0x8(%rax), %rdx          # 再将%rax内的指针加上八个字节, 给%rdx, 推测%rdx就
                             # 是ap + 8

movsl %ecx, %rcx             # 同时可以得出b_struct以8字节对齐。
                             # 将整形n拓展为长整型存储在%rcx中, 推测a_struct中的
                             # x为long

```

又从后面的语句`mov %rcx, 0x10(%rax,%rdx,8)`中, `0x10=40`可以推断出每个`a_struct`的大小为40, `40*7+8=288`, 所以**CNT=7**。

`mov %rcx, 0x10(%rax,%rdx,8)`可以推断`a_struct`中有一个`long`类型的`x`数组和`long`类型的`idx`。结合每个`a_struct`的大小得到完整定义如下:

```

typedef struct a_struct{
    long idx,
    long x[4]
}

```

3.70

A. 下列字段的偏移量是多少(以字节为单位):

```

e1.p      0
e1.y      8
e2.x      0
e2.next   8

```

B. 这个结构总共需要多少个字节? **16**

C. 编译器为 `proc` 产生下面的汇编代码:

```
void proc (union ele *up)
```

```
up in %rdi
```

```
1 proc:
```

```
2 movq 8(%rdi), %rax
```

```
3 movq (%rax), %rdx
```

```
4 movq (%rdx), %rdx
```

```
5 subq 8(%rax), %rdx
```

```
6 movq (%rdx, (%rdi))
```

```
7 ret
```

\rightarrow `rax = *next`
 \rightarrow `rdx = next` (`*next` 中为 `e1`)
 \rightarrow `rdx = p`
 \rightarrow `rdx = p - 8`
 \rightarrow `e2.x`

$8(\%rax) = y$

分析汇编语句, `up`中应该为`e2`类型, 其中的`next`指向的是`e1`类型。

注意到前面对`%rdx`连续解了两次引用, 关键的`subq 8(%rax), %rdx`语句后面的`8(%rax)`存储的是`next`里的`y`, `%rdx`指向的是`next`里的`p`指向的`long`数据。

补全后的语句如下:

```
void proc (union ele * up){  
    up->e2.x = (*(up->e2.next).e1.p) - *(up->e2.next).e1.y;  
}
```

3.46

选做题好难，我好累Orz.