汇编语言第九次作业

石曜铭

2025 年 7 月 5 日

3.58

```
long decode2(long x, long y, long z) {
        y -= z;
        x *= y;
        long res = y;
        res >>= 63;
        res <<= 63;
        res ^= x;
        return res;
}
```

3.59

解释：首先将 x, y 分别符号扩展至 128 位，然后将 x, y 分别表示成 $x = x_h \cdot 2^{64} + x_l, y = y_h \cdot 2^{64} + y_l$ 。

这样

$$x \cdot y = x_h \cdot y_h \cdot 2^{128} + (x_h \cdot y_l + x_l \cdot y_h) \cdot 2^{64} + x_l \cdot y_l.$$

高于 128 位的被忽略。实际上因为 x_h, y_h 是符号位扩展的结果，所以结果只会是 $-1, 1$，而这实际上就是结果的符号位。

将结果 $p = x \cdot y$ 表示为 $p = p_h \cdot 2^{64} + p_l$ ，对比系数可得

$$\begin{cases} p_h &= x_h \cdot y_l + x_l \cdot y_h, \\ p_l &= x_l \cdot y_l. \end{cases}$$

代码首先取出 x 的最高位，作位扩展和 y 相乘，然后再求出 x, y 低位相乘的结果。最后将乘法的高位和低位结果存储在 `rdi` 所指向的内存地址。

3.60

(A) x 在 rdi 寄存器中，n 在 esi 寄存器中，result 在 rax 寄存器中，mask 在 rdx 寄存器中。

(B) result 初值为 0，mask 初值为 1 。

(C) 不等于 0 。

(D) mask $\leftarrow$ mask $>> n$ ，其中 n 只取低八位。

(E) result $\leftarrow$ result|mask&x 。

(F) $0, 1, != 0$, mask $>> n$, x&mask 。

3.61

```
long cread_alt(long *xp) {
        long test = xp == 0;
        long result = 0;
        if (!test) result = *xp;
        return result;
}
```

3.62

```
typedef enum {
        MODE_A, MODE_B, MODE_C, MODE_D, MODE_E
} mode_t;

long switch3(long *p1, long *p2, mode_t action) {
  long result = 0;
  switch(action) {
    case MODE_A:
      result = *p2;
      *p2 = *p1;
      break;
    case MODE_B:
      *p1 = *p1 + *p2;
      result = *p1;
      break;
    case MODE_C:
      *p1 = 59;
      result = *p2;
      break;
    case MODE_D:
      *p1 = *p2;
      result = 27;
      break;
    case MODE_E:
      result = 27;
      break;
    default:
      result = 12;
      break;
  }
  return result;
}
```

3.63

```
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;
}
```

3.64

(A) 设第一、二、三维大小分别为 R, S, T，设首地址为 A，每个元素的 *size* 为 L，那么对元素 (i, j, k)，其地址为

$$\texttt{dest} = A + L \times (i \times S \times T + j \times T + k).$$

(B) 由代码易知，首先计算了 $j \times 13$ ，然后计算了 $i \times 65$，所以

$$\begin{cases} S \times T & = 65, \\ T & = 13, \\ 8 \times R \times S \times T & = 3640. \end{cases}$$

所以 $R = 7, S = 5, T = 13$ 。

3.65

(A) `rdx` 。

(B) `rax` 。

(C) $M = 120/8 = 15$.

3.66

由代码可知，每次 `rcx` 加的偏移量是 `r8`，而 $\texttt{r8} = 8 \times (4 \times n + 1)$ ，所以 $NC(n) = 4 \times n + 1$ 。又，跳出循环的条件是计数器 `rdi` == `rdx`，而 $\texttt{rdx} = 3 \times n$ ，所以 $NR(n) = 3 \times n$ 。

3.67

(A) $0 \sim 7 : x; 8 \sim 15 : y; 16 \sim 23 : \&z; 24 \sim 31 : z$.

$\texttt{rsp} = 0, \texttt{rdi} = 64$.

(B) 传递了 s.a[0], s.a[1], s.p，以及一个新的地址 $\texttt{rdi} = \texttt{rsp} + 64$ 。

(C) 通过 `rsp` 加偏移量。

(D) 以 `rdi` 为起始地址。

(E) $0 \sim 7 : x; 8 \sim 15 : y; 16 \sim 23 : \&z; 24 \sim 31 : z; 64 \sim 71 : y, 72 \sim 80 : x, 81 \sim 87 : z$.

(F) 由调用者开辟内存并把起始地址传给被调用者，然后被调用者在指定内存把结构体的内容保存，然后返回首地址。

3.68

由 `movslq 8(%rsi),%rax`，且 t 是 int 类型，4 字节对齐，有 $4 < B \leq 8$ 。

由 `addq 32(%rsi),%rax`，且 u 是 $long$ 类型，8 字节对齐，有 $24 < 12 + A \times 2 \leq 32$ 。

由 `movq %rax,184(%rdi)`，且 y 是 $long$ 类型，8 字节对齐，有 $176 < A \times B \times 4 \leq 184$ 。

综上，只有 $A = 9, B = 5$ 满足要求。