Homework 2 & Homework 3

郭天魁
信息科学技术学院
1300012790

November 4, 2014

1 Homework 2

1.1 2.85

	Extended precision	
Description	Value	Decimal
Smallest positive denormalized	2^{-16445}	$3.645200 \times 10^{-4951}$
Smallest positive normalized	2^{-16382}	$3.362103 \times 10^{-4932}$
Largest normalized	$(2 - 2^{-63}) \times 2^{16382}$	$5.948657 \times 10^{4931}$

1.2 2.93

```
float_bits float_half(float_bits uf) {
        unsigned sign = uf & 0x80000000;
        unsigned exp = uf & 0x7F800000;
        unsigned frac = uf & 0x7FFFFF;
        unsigned round = (frac & 3) == 3;
        unsigned frac_half = (frac >> 1) + round;
        if (exp == 0x7F800000)
                return uf;
        if (exp)
                frac = (exp -= 0x800000) ? frac : 0x400000 + frac_half;
        else
                frac = frac_half;
        return sign | exp | frac;
}
```

在f为NaN时，直接返回f与直接进行浮点运算的结果不符，具体规则待考。

1.3 2.95

```c
float_bits float_i2f(int i) {
        unsigned sign = i & 0x80000000u;
        unsigned f = i;
        if (!i)
                return 0;
        if (sign)
                f = (~f) + 1;
        int exp = 0x4F000000;
        while (~f & 0x80000000) {
                f <<= 1;
                exp -= 0x800000;
        }
        f ^= 0x80000000;
        f += 0x7f;
        if ((f & 0x1ff) == 0x1ff)
                f++;
        f >>= 8;
        return sign | (exp + f);
}
```

通过全部测试。

2 Homework 3

2.1 3.55

注意到对于64-bit的数x，其作为有符号形式所表示的数与无符号形式所表示的数对于2^{64}是同余的，所以如果x是64-bit的，那么只需要执行x与y的无符号乘法并使其自然溢出即可。此时令x_1, x_2, y_1, y_2分别表示x, y的高位与低位，A_1, A_2分别表示答案xy的高位与低位，则有

$$A_1 = x_1y_2 + x_2y_1, A_2 = x_2y_2,$$

其中A_2可能向A_1溢出，A_1可能自然溢出。

而此时x为32-bit的数，则$x_2 = x, x_1 = x >> 31 = \begin{cases} 0, & x \geq 0, \\ -1, & x < 0. \end{cases}$

以下为算法过程:

```asm
movl    16(%ebp), %esi          # get y2
movl    12(%ebp), %eax          # get x2
movl    %eax, %edx
sarl    $31, %edx               # x1 = x >> 31
movl    20(%ebp), %ecx          # get y1
imull   %eax, %ecx              # calculate x2y1
movl    %edx, %ebx
imull   %esi, %ebx              # calculate x1y2
addl    %ebx, %ecx              # A1 = x1y2 + x2y1
```

```
mull    %esi                 # A2 = [edx:eax] = x2y2
leal    (%ecx,%edx), %edx    # add A1 to edx
movl    8(%ebp), %ecx        # get dest
movl    %eax, (%ecx)         # save xy to *dest
movl    %edx, 4(%ecx)
```

2.2 3.56

A. %esi - x, %ebx - n, %edi - result, %edx - mask.

B. -1(0xffffffff) - result, 1 - mask.

C. mask != 0.

D. mask = mask << n. 在执行过程中，实际只考虑n的后8位。

E. result ^= x & mask.

F.

```c
int loop(int x, int n)
{
        int result = -1;
        int mask;
        for (mask = 1; mask != 0; mask = mask << n) {
                result ^= x & mask;
        }
        return result;
}
```

2.3 3.58

```c
int switch3(int *p1, int *p2, mode_t action)
{
        int result = 0;
        switch(action) {
                case MODE_A:
                        result = *p1;
                        *p1 = *p2;
                        break;
                case MODE_B:
                        result = *p1 + *p2;
                        *p2 = result;
                        break;
                case MODE_C:
                        *p2 = 15;
```

```
                result = *p1;
                break;
            case MODE_D:
                *p2 = *p1;
                result = 17;
                break;
            case MODE_E:
                result = 17;
                break;
            default:
                result = -1;
        }
        return result;
```

2.4 3.60

A. &A[i][j][k]= $x_A + L(i \cdot S \cdot T + j \cdot T + k)$，其中$x_A$为A的起始地址，$L$为类型长度，此题中为4。

B. $T = 1 + 2 \times (4 + 1) = 11, S = \dfrac{99}{T} = 9, R = \dfrac{1980}{99 \times 4} = 5.$

2.5 3.63

```
#define E1(n) (2 * (n) + 1)
#define E2(n) (3 * (n))
```