CSED211 Homework 1

20190084 권민재

1. Exercise 2.61

- Any bit of x equals 1.
  - `!!x`
- Any bit of x equals 0.
  - `!x`
- Any bit in the least significant byte of x equals 1.
  - `!!(x & 0xff)`
- Any bit in the most significant byte of x equals 0.
  - `!(x>>24)`

2. Exercise 2.69

```
/*
 * Do rotating left shift. Assume 0 <= n < w
 * Examples when x = 0x12345678 and w = 32:
 *      n=4 -> 0x23456781, n=20 -> 0x67812345
 */
unsigned rotate_left(unsigned x, int n) {
    unsigned w = sizeof(unsigned) << 3;
    return (x << n) | (x >> (w-n-1) >> 1);
}
```

3. Exercise 2.77

- K = 17
  - `x + (x << 4)`
- K = -7
  - `x - (x << 3)`
- K = 60
  - `(x << 6) - (x << 2)`
- K = -112

- `(x << 4) - (x << 7)`

4. Exercise 2.83

A.

infinite string으로 나타낸 값을 x라고 하자. 이때, `x << k == x + Y`이므로, $x \cdot 2^k = x + Y$ 라고 할 수 있다. 그러므로, x는 아래와 같이 나타낼 수 있다.

$\frac{Y}{2^k - 1}$

B.

- (a) $\frac{5}{7}$
- (b) $\frac{2}{5}$
- (c) $\frac{19}{63}$

5. Exercise 2.88

- `0 10110 011`
  - Format A
    - 부호비트: 양수
    - 지수부: $10110_{(2)} - 15 = 7$
    - 가수부: $1.011_{(2)} = 1 + \frac{1}{4} + \frac{1}{8} = \frac{11}{8}$
    - $2^7 \cdot \frac{11}{8} = 176$
  - Format B
    - `0 1110 0110`, `176`
- `1 00111 010`
  - Format A
    - 부호비트: 음수
    - 지수부: $00111_{(2)} - 15 = -8$
    - 가수부: $1.010_{(2)} = 1 + \frac{1}{4} = \frac{5}{4}$
    - $-1 \cdot 2^{-8} \cdot \frac{5}{4} = -\frac{5}{1024}$
  - Format B
    - `1 0000 0101`, $-\frac{5}{1024}$
- `0 00000 111`
  - Format A
    - 부호비트: 양수
    - 지수부: $1 - 15 = -14$
    - 가수부: $0.111_{(2)} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} = \frac{7}{8}$

        - $2^{-14} \cdot \frac{7}{8} = \frac{7}{2^{17}}$
    - Format B
        - `0 0000 0000`, 0
- `1 11100 000`
    - Format A
        - 부호비트: 음수
        - 지수부: $11100_{(2)} - 15 = 13$
        - 가수부: $1.0_{(2)} = 1$
        - $-1 \cdot 2^{13} \cdot 1 = -8192$
    - Format B
        - `1 1111 0000`, $-\infty$
- `0 10111 100`
    - Format A
        - 부호비트: 양수
        - 지수부: $10111_{(2)} - 15 = 8$
        - 가수부: $1.100_{(2)} = 1 + \frac{1}{2} = \frac{3}{2}$
        - $2^8 \cdot \frac{3}{2} = 384$
    - Format B
        - `0 1111 1000`, $\mathbf{NaN}$

Format A		Format B	
Bits	**Value**	**Bits**	**Value**
1 01111 001	$\frac{-9}{8}$	1 0111 0010	$\frac{-9}{8}$
0 10110 011	176	0 1110 0110	176
1 00111 010	$-\frac{5}{1024}$	1 0000 0101	$-\frac{5}{1024}$
0 00000 111	$\frac{7}{2^{17}}$	0 0000 0000	0
1 11100 000	-8192	1 1111 0000	$-\infty$
0 10111 100	384	1 1111 1000	$\mathbf{NaN}$

6. Exercise 2.89

- A. Expression always yields 1.
    - `int`와 `double` 모두`float`로 캐스팅 될 때 손실 없이 rounds off 되기 때문이다.
- B. Expression can yield `0`.

    - `x = INT_MAX, y=INT_MIN`인 경우, 오버플로우가 발생하여 성립하지 않는다.
- C. Expression always yields 1.
    - 원래 `int`인 값 으로부터 만들어졌기 때문에 오버플로가 발생하지 않으므로, 성립한다.
- D. Expression can yield `0`.
    - 연산 순서와 rounding에 의해 결과 값이 달라질 수 있다.
    - ex. x: 0x574200aa, y: 0x2e39c5a7, z: 0x52df6ea6
- E. Expression can yield `0`.
    - 분모로 들어가는 값 (dz나 dx)이 0일 경우에 문제가 생길 수 있다.

7. Exercise 2.90

```
float fpwr2(int x) {
  /* Result exponent and fraction */
  unsigned exp, frac;
  unsigned u;

  if (x < -149) {
    /* too small. return 0.0 */
    exp = 0;
    frac = 0;
  } else if (x < -126) {
    /* Denormalized result */
    exp = 0;
    frac = 1 << (x + 149);
  } else if (x < 128) {
    /* Normalized result */
    exp = x + 127;
    frac = 0;
  } else {
    /* Too big, return +oo */
    exp = 0xFF;
    frac = 0;
  }

  /* pack exp and frac into 32 bits */
  u = exp << 23 | frac;
  /* Result as float */
  return u2f(u);
}
```

8. Exercise 2.95

```
float_bits float_half(float_bits f) {
    unsigned sign = f & 0x80000000;
    unsigned expo = f & 0x7f800000;
    unsigned frac = f & 0x007fffff;

    if(expo == 0x7f800000){
        return f;
    }

    if(expo > 0x00800000){
        return f - 0x00800000;
    }

    return sign | (((expo | frac) >> 1) + ((f & 3) == 3));

}
```

9. Describe the difference between RISC and CISC.

RISC는 Reduced Instruction Set Computer의 약자로, 고정된 길이의 적은 수의 명령어로 구성되어 있는 것을 말한다. 이에 반해, CISC는 (Complex Instruction Set Computer)의 약자로, 가변 길이의 다양한 명령어로 구성되어 있는 것을 말한다. RISC는 고정된 길이의 적은 명령어를 가지고 있기 때문에, CISC에 비해 하드웨어 구조가 단순하고 해석 속도가 빠르다. CISC는 RISC보다 하나의 명령어를 처리하는 과정이 복잡하며, RISC에서 여러 명령어를 사용해야 하는 것을 CISC에서는 더 적은 양으로 처리할 수 있다.

10. What is ISA (Instruction Set Architecture)?

ISA란, 마이크로프로세서가 실행할 수 있는 기계어 명령어들을 말하며, 프로세서가 실행할 수 있는 모든 명령어를 포함한다. ISA의 물리적 구현체를 우리는 microarchitecture라고 부른다.