컴퓨터 SW시스템개론 Lab1 보고서

20210207 이지현

Ⅰ. 소개

Bits.c의 5가지 문제에 대하여 함수를 작성하는 lab이다. bitNor, isZero, addOk, logicalShift, absVal 5개의 문제가 있고, 각 함수의 예시와 조건을 참고하여 문제를 해결할 수 있다.

Name	Description	Rating
bitNor (x, y)	~(x | y) using only ~ and &	1
isZero (x)	return 0 if x is non-zero, else 1	1
addOk(x,y)	Determine if can compute x+y without overflow	3
absVal(x)	absoulte value of x	4
logicalShift(x, n)	Shift right logical.	3

Table 1: Bit-Level Manipulation Functions.

Ⅱ. 코드 설명

(1) int bitNor (int x, int y)

| operator에도 ~이 적용하여 (~x) (~y) 와 &로 코드를 작성한다.

(2) int isZero (int x)

0이면 true가 되도록 ! operator를 이용해 코들르 작성한다.

(3) int addOK (int x, int y)

x, y의 부호비트만 남기고 x+y의 부호비트를 sum_msb로 남긴다.

x,y가 다른 부호이면 오버플로우가 일어나지 않는다.

만약 같은 부호이면 x와 sum이 같은 부호인지, y와 sum의 부호가 같은지 확인한다.

만약 같다면 오버플로우가 일어나지 않고, 다르면 오버플로우가 일어남을 이용해서 코드를 작성했다.

(4) int absVal (int x)

arithmetic shift를 이용하면 mask는 양수라면 000..000이고, 음수라면 111..111이다. 즉, x와 mask를 XOR과 -를 통해 양수라면 그대로, 음수라면 ~x + 1을 의미할 수 있다.

(5) int logicalShift (int x, int n)

예를 들어 n=3 이면 0001111..111인 비트를 만들면 &연산을 통해 앞부분을 0으로 만들

수 있다. 이를 이용해 코드를 작성하였다.

Ⅲ. 실행 사진 첨부

(1) Problem 1-5

```c
int bitNor(int x, int y) {

    return (~x) & (~y);

    return 0;
}
```

```c
int isZero(int x) {

    return !x;
}
```

```c
int addOK(int x, int y) {

    int sum_msb = (x + y) >> 31;
    x >>= 31;
    y >>= 31;

    return !!(x^y) | (!(x^sum_msb)&!(y^sum_msb));
}
```

```c
int absVal(int x) {

    int mask = x >> 31;

    x=x^mask;
    x=x+(~mask+1);

    return x;
}
```

```c
*/
int logicalShift(int x, int n) {

    x >>= n;
    int mask = ~(((1<<31)>>n)<<1);

    return mask & x;

}
```

(2) ./btest 실행

```
[io0818@programming2 ~]$ ls
Driverhdrs.pm  README  bits.h.gch  btest.h      dlc        fshow.c  tests.c
Driverlib.pm   bits.c  btest       btest.h.gch  driver.pl  ishow
Makefile       bits.h  btest.c     decl.c       fshow      ishow.c
[io0818@programming2 ~]$ ./btest
Score   Rating  Errors  Function
 1      1       0       bitNor
 1      1       0       isZero
 3      3       0       addOK
 4      4       0       absVal
 3      3       0       logicalShift
Total points: 12/12
[io0818@programming2 ~]$
```

Ⅳ. 결론 및 고찰

-bit단위의 operator들만을 이용해 코드를 구성하는 기초를 배울 수 있었다.

-operator들의 사용 개수의 최대값이 조건에 있어 가장 효율적이고 간단한 코드를 구현하는 것에 가장 많은 시간이 소요되었다.

-리눅스 xshell을 이용해 서버를 이용하는 방법을 익힐 수 있어 유익했다.

-int의 bit단위를 쪼개서 생각할 수 있는 함수들이 매우 많았고, 오늘 했던 5개의 함수 이외에도 여러 함수들의 코드를 직접 짜볼 계획이다.