

시스템 프로그래밍

- 비트 연산 & Datalab -

2014. 09. 25.

박시형

sihyeong@cnu.ac.kr

Embedded System Lab. Computer Engineering Dept.
Chungnam National University

❖ 실습 명

- 비트 연산 & Datalab

❖ 목표

- 비트 연산의 이해와 사용

❖ 주제

- 비트 연산
 - Boolean Algebra
 - C언어의 비트 연산
 - 비트 논리 연산자
 - 비트 연산자
- Datalab

Boolean Algebra

❖ 불 대수(Boolean Algebra)

- 논리를 표현하기 위한 Algebra(대수학)이며, 컴퓨터 내부의 비트 정보 표시에 사용하는 변수들은 **0 또는 1**을 가지며, **2진수의 표시 및 연산**에 매우 유용하다.

❖ AND (&)

- ❖ $A \& B = 1$ when both $A = 1$ and $B = 1$

&	0	1
0	0	0
1	0	1

❖ OR (|)

- ❖ $A | B = 1$ when either $A = 1$ and $B = 1$

	0	1
0	0	1
1	1	1

❖ NOT (~)

- ❖ $\sim A = 1$ when $A = 0$

~	0
0	1
1	0

❖ Exclusive OR (XOR, ^)

- ❖ $A \wedge B = 1$ when either $A = 1$ or $B = 1$, but not both

^	0	1
0	0	1
1	1	0

비트 논리 연산자

❖ Bit 단위 논리연산에 사용하는 연산자

- $\&, |, \sim, ^$
- 일반 논리 연산자($\&\&, ||, !$) 보다 우선순위가 높음
- 아래의 정수형 변수에 사용 가능
 - int, short, long, char, unsigned

❖ 예제 (8-bit)

- $0x69 \& 0x55 = 0x41$
 - $0110\ 1001_{(2)} \& 0101\ 0101_{(2)} = 0100\ 0001_{(2)}$
- $0x69 | 0x55 = 0x7D$
 - $0110\ 1001_{(2)} | 0101\ 0101_{(2)} = 0111\ 1101_{(2)}$
- $\sim 0x41 = 0xBE$
 - $\sim 0100\ 0001_{(2)} = 1011\ 1110_{(2)}$
- $0x41 ^ 0x69 = 0x28$
 - $0100\ 0001_{(2)} ^ 0110\ 1001_{(2)} = 0010\ 1000_{(2)}$

비트 연산자

❖ Shift 연산자 (<< , >>)

- 어셈블리 언어나 기계어의 프로그램 작성에서 레지스터 또는 기억 장소 내에 비트 값들을 왼쪽이나 오른쪽으로 이동시키는 것.

❖ Left shift ($x \ll y$)

- Shift bit-vector x left y position
 - 좌측 끝의 MSB를 날려버림
 - 우측 끝의 LSB에 0을 채워줌
- Logical shift**와 **Arithmetic shift**의 동작이 동일함

❖ Right shift ($x \gg y$)

- Shift bit-vector x right y position
 - 우측 끝의 LSB를 날려버림
- Logical shift** (논리적 자리 이동)
 - 좌측 끝의 MSB에 0을 채워줌
- Arithmetic shift** (산술 자리 이동)
 - 좌측 끝의 MSB에 현재 부호를 유지함
 - Two's complement 정수 표현 시에 필요

Argument x	1010 0010
$x \ll 3$	0001 0000
Logical $x \gg 2$	0010 1000
Arithmetic $x \gg 2$	1110 1000

기본 비트 연산 - 따라 하기

❖ /home/syspro/operation_test.tar.gz를 자신의 홈(~)으로 복사

```
[b0000000000@eslab ~]$ cp /home/syspro/operation_test.tar.gz .
[b0000000000@eslab ~]$ ls
operation_test.tar.gz
[b0000000000@eslab ~]$
```

❖ 압축 해제 후 디렉터리 내 파일 확인

```
[b0000000000@eslab ~]$ tar xfvz operation_test.tar.gz
operation_test/
operation_test/LShiftFunc.c
operation_test/XorFunc.c
operation_test/NotFunc.c
operation_test/AndFunc.c
operation_test/InputAndPrint.c
operation_test/OrFunc.c
operation_test/Makefile
operation_test/header.h
operation_test/RShiftFunc.c
[b0000000000@eslab ~]$
```

기본 비트 연산 - 따라 하기

❖ 각 소스 내부의 함수를 작성

```
int AndFunc(int nA, int nB)
{
    int result = nA & nB;
    return result;
}
```

소스 코드	코드
AndFunc.c	nA & nB
NotFunc.c	~nA
OrFunc.c	nA nB
XorFunc.c	nA ^ nB
RShift.c	nA >> nB
LShift.c	nA << nB

❖ make를 통해 컴파일 후 결과를 확인

```
[b0000000000@eslab operation_test]$ make
cc -o test.out -g -O2 AndFunc.c LShiftFunc.c NotFunc.c RShiftFunc.c InputAndPrint.c OrFunc.c XorFunc.c
[b0000000000@eslab operation_test]$ ls
AndFunc.c      LShiftFunc.c  NotFunc.c     RShiftFunc.c  header.h
InputAndPrint.c Makefile      OrFunc.c      XorFunc.c     test.out
[b0000000000@eslab operation_test]$
```

기본 비트 연산 - 따라 하기

❖ 실행 결과를 확인

```
[b0000000000@eslab operation_test]$ ./test.out
Input a hex pair values (ex : H1 FF) : 69 55
Your input values are 69(A) and 55(B)
```

```
NotFunc(~A)      : FFFFFFFF96
AndFunc(A&B)     : 41
OrFunc(A|C)      : 7D
XorFunc(A^B)     : 3C
LShiftFunc(A<<B) : D2000000
RShiftFunc(A>>B) : 0
[b0000000000@eslab operation_test]$
```

- main 함수는 InputAndPrint.c에 작성되어 있음

❖ 정수와 실수의 bit-level 표현에 대해 좀 더 친숙해 질 수 있도록 만들어진 랩

- 현 실습에서는 정수의 bit-level에 대한 표현만 진행
- 각 문제마다 사용 가능한 연산자들을 가지고 코드를 작성
 - 해당 연산자는 다음 슬라이드에서 확인

Datalab – 함수 설명

❖ Datalab은 아래와 같이 8개의 함수로 이루어져 있다.

함수	기능	사용 가능한 연산자
bitAnd(int x, int y)	~와 을 사용해서 x & y 연산	~
getBytes(int x, int n)	x의 n번째 (1 byte)를 추출	! ~ & ^ + << >>
logicalShift(int x, int n)	logicalShift를 이용해서 오른쪽으로 shift	! ~ & ^ + << >>
bitCount(int x)	x에서 1의 개수를 계산	! ~ & ^ + << >>
tmin(void)	2의 보수 중 최소값 (정수)을 계산	! ~ & ^ + << >>
fitsBits(int x, int n)	n-bit로 x의 2의 보수를 표현할 수 있으면 1을 반환	! ~ & ^ + << >>
isPositive(int x)	x > 0 이면	! ~ & ^ + << >>
rotateRight(int x, int n)	x를 n만큼 오른쪽으로 회전	~ & ^ + << >> !

Datalab - 구조

- ❖ /home/syspro/02_datalab_handout.tar.gz 파일을 자신의 계정에 복사 후 압축 해제 및 확인

```
[b0000000000@eslab 02]$ cp /home/syspro/02_datalab-handout.tar .
[b0000000000@eslab 02]$ tar xvf 02_datalab-handout.tar
02_datalab-handout/
02_datalab-handout/ishow.c
02_datalab-handout/Driverhdrs.pm
02_datalab-handout/decl.c
02_datalab-handout/bits.c
02_datalab-handout/README
02_datalab-handout/bits.h
02_datalab-handout/fshow.c
02_datalab-handout/dlc
02_datalab-handout/tests.c
02_datalab-handout/btest.c
02_datalab-handout/driver.pl
02_datalab-handout/Driverlib.pm
02_datalab-handout/btest.h
02_datalab-handout/Makefile
[b0000000000@eslab 02]$ █
```

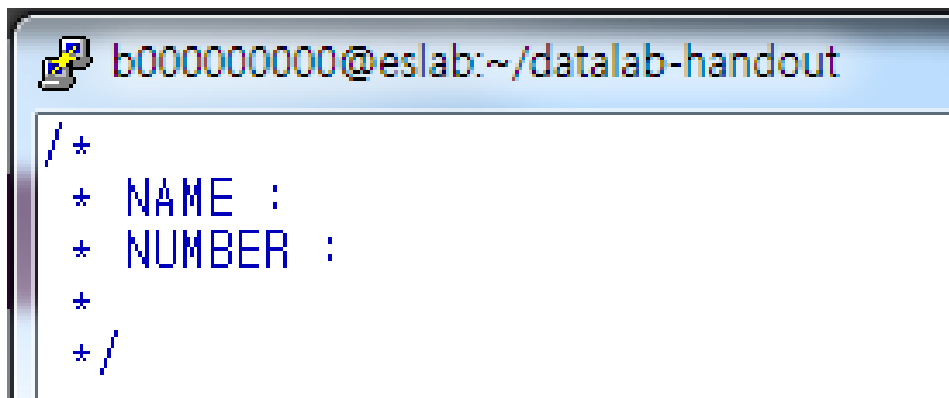
- ❖ 실습 소스코드 작성은 **bits.c** 파일에 한다

- ❖ datalab-handout 디렉터리에서 make 명령을 통해 컴파일하면 btest 라는 실행파일이 생성됨

```
[b0000000000@eslab datalab-handout]$ make
gcc -O -Wall -m32 -lm -o btest bits.c btest.c decl.c tests.c
      In function '':
      warning: variable 'i' set but not used [-Wunused-but-set-var
iable]
      int errors;
      ^
gcc -O -Wall -m32 -o fshow fshow.c
gcc -O -Wall -m32 -o ishow ishow.c
[b0000000000@eslab datalab-handout]$ ls
Driverhdrs.pm  README  btest  decl.c  fshow  ishow.c
Driverlib.pm   bits.c  btest.c  dlc     fshow.c  tests.c
Makefile       bits.h  btest.h  driver.pl  ishow
[b0000000000@eslab datalab-handout]$
```

Datalab – 진행 방법

- ❖ vi 편집기를 이용하여 bits.c 파일을 열어 맨 위에 자신의 이름과 학번을 기입한다.



```
b0000000000@eslab:~/datalab-handout
/*
 * NAME :
 * NUMBER :
 *
 */
```

- ❖ 함수를 문제에 알맞게 작성 후 저장한 후, make를 통해 다시 컴파일 해서 실행파일 생성

- ❖ 컴파일 후, 반드시 자신이 작성한 코드가 **허용된 연산자를 사용해서 작성**되었는지 확인
- ❖ **./dlc bits.c**
 - 해당 명령어를 사용하면 자신의 코드에서 허용되지 않은 연산자가 사용되었는지 확인 할 수 있음
 - 허용되지 않은 연산자를 사용하면, 해당 문제 점수 감점
 - for, while 과 같은 구문 등
- ❖ **bits.c외 다른 파일 수정 금지**

❖ 정답이 아닌 경우 다음과 같은 에러 발생

```
[b0000000000@eslab datalab-handout]$ ./btest
Score   Rating  Errors  Function
ERROR: Test bitAnd(-2147483648[0x80000000],-2147483648[0x80000000]) failed...
...Gives 2[0x2]. Should be -2147483648[0x80000000]
ERROR: Test getByte(-2147483648[0x80000000],0[0x0]) failed...
...Gives 2[0x2]. Should be 0[0x0]
ERROR: Test logicalShift(-2147483648[0x80000000],0[0x0]) failed...
...Gives 2[0x2]. Should be -2147483648[0x80000000]
ERROR: Test bitCount(-2147483648[0x80000000]) failed...
...Gives 2[0x2]. Should be 1[0x1]
ERROR: Test bang(-2147483648[0x80000000]) failed...
...Gives 2[0x2]. Should be 0[0x0]
ERROR: Test tmin() failed...
...Gives 2[0x2]. Should be -2147483648[0x80000000]
ERROR: Test fitsBits(-2147483648[0x80000000],1[0x1]) failed...
...Gives 2[0x2]. Should be 0[0x0]
ERROR: Test divpwr2(-2147483648[0x80000000],0[0x0]) failed...
...Gives 2[0x2]. Should be -2147483648[0x80000000]
ERROR: Test negate(-2147483648[0x80000000]) failed...
...Gives 2[0x2]. Should be -2147483648[0x80000000]
```

Datalab - 결과 확인

❖ 모든 함수를 작성한 후 ./btest를 통해 점수를 확인할 수 있음

- bits.c를 수정하였을 경우 make를 통해 다시 컴파일 해주어야만 자신의 결과가 반영된다.

```
[b0000000000@eslab datalab-handout]$ ./btest
Score  Rating  Errors  Function
1      1      0      bitAnd
2      2      0      getByte
3      3      0      logicalShift
4      4      0      bitCount
4      4      0      bang
1      1      0      tmin
2      2      0      fitsBits
2      2      0      divpwr2
2      2      0      negate

Total points: 41/41
```

❖ 특정 함수의 점수만 확인 하는 방법

- f 옵션을 사용하여 함수 별로 결과를 확인할 수 있음
- 사용법 : ./btest -f [함수명]

```
[b0000000000@eslab datalab-handout]$ ./btest -f bitAnd
Score  Rating  Errors  Function
1      1      0      bitAnd

Total points: 1/1
```


Datalab – 종합 결과 확인

- ❖ 종합적인 점수를 확인하기 위해서는 **./driver.pl** 파일 실행
 - 실행 전 **chmod 700 driver.pl**을 해서 사용자에게 해당 파일의 모든 권한을 준다.

5. Running './dlc -e' to get operator count of each function.

Correctness Results			Perf Results		
Points	Rating	Errors	Points	Ops	Puzzle
0	1	1	0	0	bitAnd
0	2	1	0	0	getByte
0	3	1	0	0	logicalShift
0	4	1	0	0	bitCount
0	1	1	0	0	tmin
0	2	1	0	0	fitsBits
0	3	1	0	0	isPositive
0	3	1	0	0	rotateRight

Score = 0/35 [0/19 Corr + 0/16 Perf] (0 total operators)

Datalab – 종합 결과 확인

❖ 웹 페이지를 통해 자신의 순위를 알 수 있다.

- <http://168.188.127.145:8080/>
- **./driver.pl -u 학번**
 - 해당 명령어를 입력하면 웹 서버로 자신의 점수가 등록된다.
 - **반드시 본인 학번 사용.**
 - 10초마다 갱신됨.

Scoreboard for the Data Lab – Sys02

Puzzle key: 1=bitAnd, 2=getByte, 3=logicalShift, 4=bitCount, 5=tmin, 6=fitsBits, 7=isPositive, 8=rotateRight

Last updated: Wed Sep 24 16:46:58 2014 (updated every 10 secs)

1	2	3	4	5	6	7	8	Winner?	Score	Nickname
999999	999999	999999	999999	999999	999999	999999	999999		0	eslab

점수 측정 기준

❖ 점수 평가

- 정확성 (Correctness) : 19점
- 성능 (Performance) : 16점
- **총 35점 만점**

- 정확성 평가표 (난이도에 따라 점수가 다름)

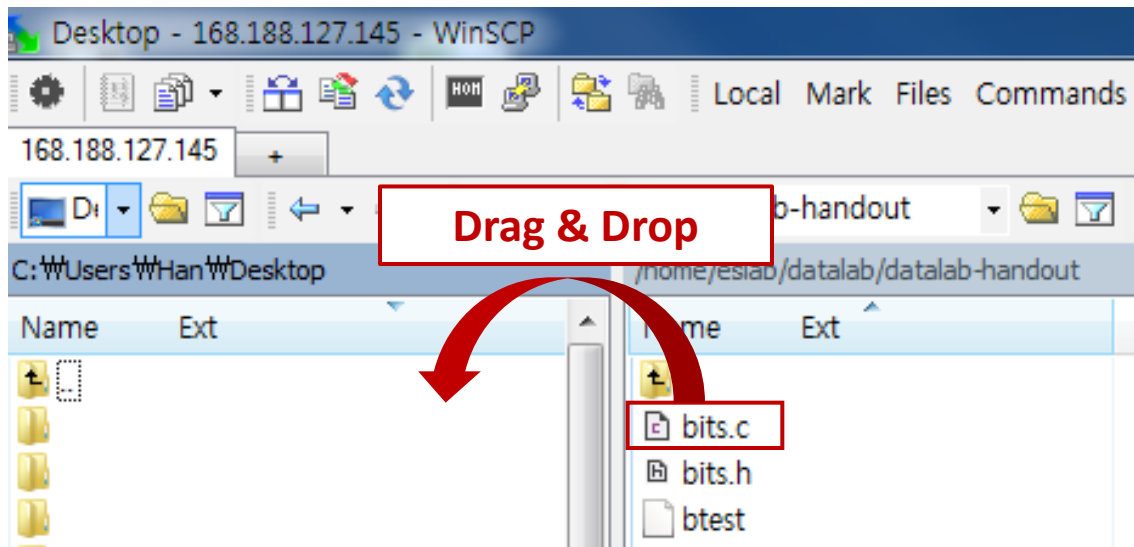
이름	점수
bitAnd	1
getBytes	2
logicalShift	3
bitCount	4
tmin	1
fitsBits	2
isPositive	3
rotateRight	3

제출 사항

- ❖ Datalab의 bits.c와 보고서를 하나의 파일로 압축
- ❖ 이메일과 서면으로 제출
 - sihyeong@cnu.ac.kr
 - 제목 양식 : [sys02]HW03_학번_이름
 - 압축 파일 제목 : [sys02]HW03_학번_이름
 - 반드시 메일 제목과 파일 양식을 지켜야 함. (위반 시 감점)
 - 보고서는 제공된 양식 사용
- ❖ COPY시 0점 처리
- ❖ 제출 일자
 - 이메일 : 2014년 10월 1일 23시 59분 59초
 - 서면 : 2014년 10월 2일 수업시간

참고 - 파일 옮기기

❖ WinSCP와 같은 프로그램 사용



❖ bits.c 파일을 옆의 탭으로 Drag & Drop