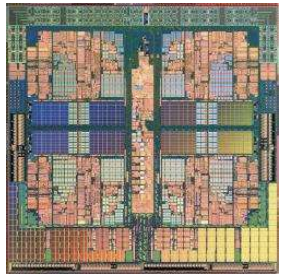


Computer Architecture Lab.

Lab 1
Datalab

Contents

- Introduction
- Instructions
- Administration
- Precautions
  - Warnings to ignore
  - The NOT DO's
  - Please 'Submit'

Introduction - Overview

- You will be solving 'puzzles'
  - Puzzles with a lot of constraints
- Total 15 puzzles to solve, puzzles are related to bitwise operation and floating point operations
- Must solve puzzle with a limited number of operations
- The purpose of this assignment is to become familiar with bit level representations

Introduction – The Puzzles(1)

Bit-level Manipulation

Name	Description	Rating	Max Ops
`bitAnd(x,y)`	`x & y` using only `|` and `~`	1	8
`getByte(x,n)`	Get byte `n` from `x`.	2	6
`logicalShift(x,n)`	Shift right logical.	3	20
`bitCount(x)`	Count the number of 1's in `x`.	4	40
`bang(x)`	Compute `!n` without using `!` operator.	4	12

Table 1: Bit-Level Manipulation Functions.

Two's Complement Arithmetic

Name	Description	Rating	Max Ops
`tmin()`	Most negative two's complement integer	1	4
`fitsBits(x,n)`	Does `x` fit in `n` bits?	2	15
`divpwr2(x,n)`	Compute $x/2^n$	2	15
`negate(x)`	`-x` without negation	2	5
`isPositive(x)`	`x > 0`?	3	8
`isLessOrEqual(x,y)`	`x <= y`?	3	24
`ilog2(x)`	Compute $\lfloor \log_2(x) \rfloor$	4	90

Table 2: Arithmetic Functions

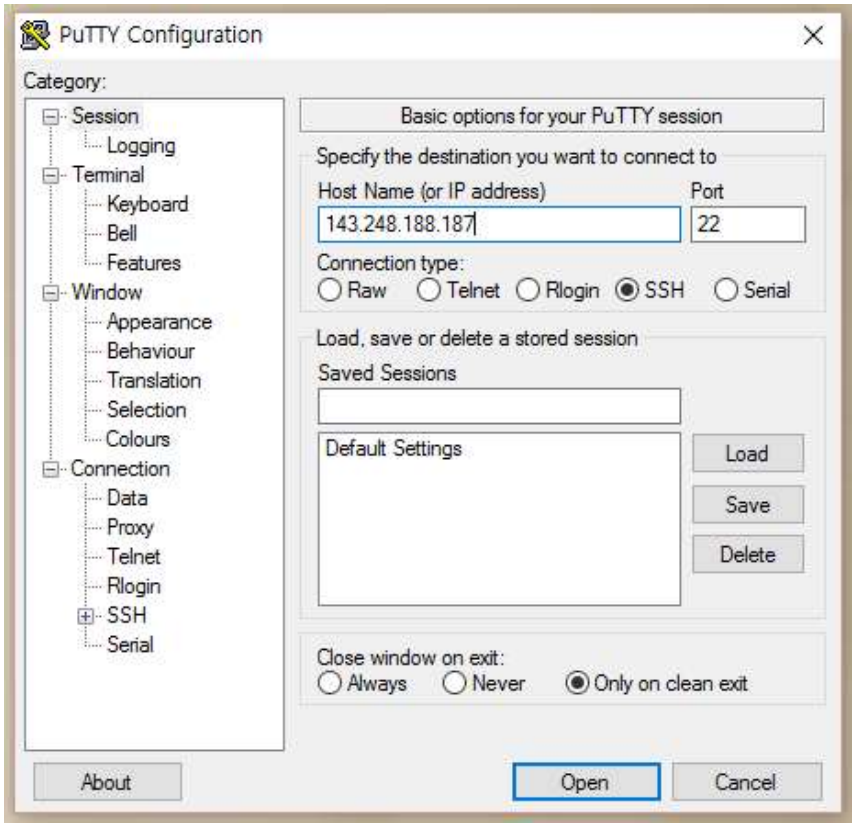
Introduction – The Puzzles(2)

Floating Point Operations

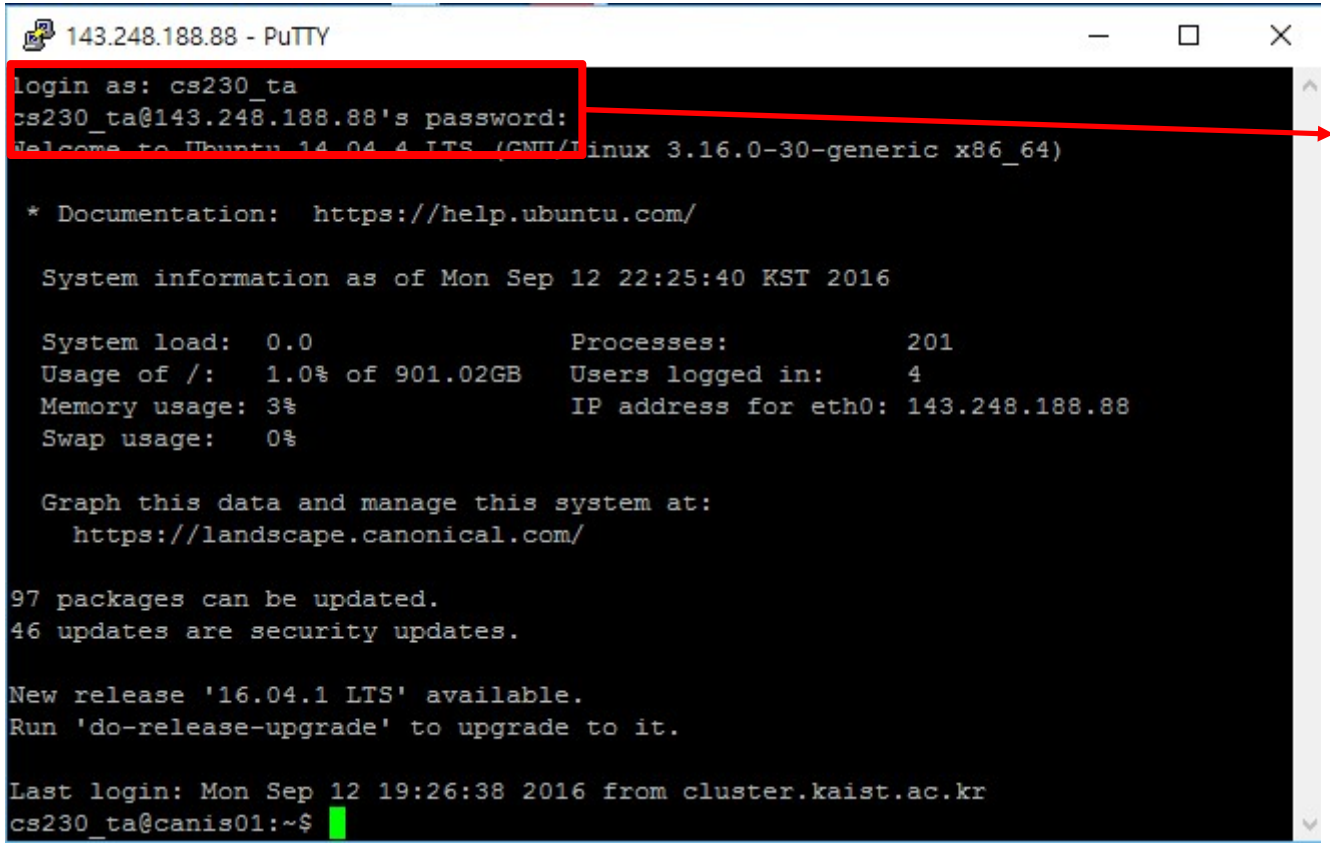
Name	Description	Rating	Max Ops
`float_neg(uf)`	Compute `-f`	2	10
`float_i2f(x)`	Compute `(float) x`	4	30
`float_twice(uf)`	Computer `2*f`	4	30

Instructions – Getting Started(1)

- **Access your machine by ssh**
  - **Download and execute putty**
    - **Example) login to canis01(143.248.188.187)**



Instructions – Getting Started(2)



Enter password when prompted

Instructions – Getting Started(3)

- **Copy the compressed file required for this lab to your directory**

```
cs230_ta@canis01:~$ cp /home/lab/datalab-handout.tar ~
cs230_ta@canis01:~$ ls
datalab-handout.tar
cs230_ta@canis01:~$
```

*** If cp is done correctly there is <u>no</u> message**

*** Check if you copied correctly by typing 'ls'**

- **Decompress datalab-handout.tar**

```
cs230_ta@canis01:~$ tar xvf datalab-handout.tar
datalab-handout/
datalab-handout/ishow.c
datalab-handout/bits.c
datalab-handout/Driverhdrs.pm
datalab-handout/decl.c
datalab-handout/btest.c
datalab-handout/bits.h
datalab-handout/fshow.c
datalab-handout/tests.c
datalab-handout/dlc
datalab-handout/driver.pl
datalab-handout/Makefile
datalab-handout/btest.h
datalab-handout/README
datalab-handout/Driverlib.pm
cs230_ta@canis01:~$ ls
datalab-handout  datalab-handout.tar
cs230_ta@canis01:~$
```

*** If tar decompression is done correctly the list of files decompressed is displayed**

Insturctions – Getting Started(4)

- **Change current directory to datalab-handout/**

and Open bits.c with vim

```
cs230_ta@canis01:~$ cd datalab-handout/
cs230_ta@canis01:~/datalab-handout$ vi bits.c
```

'vim bits.c 'and 'vi bits.c' does the same thing!

- 'vi ' is linked to vim

- **How it looks like when you opened bits.c**

```
/*
 * CS:APP Data Lab
 *
 * <Please put your name and userid here>
 *
 * bits.c - Source file with your solutions to the Lab.
 *          This is the file you will hand in to your instructor.
 *
 * WARNING: Do not include the <stdio.h> header; it confuses the dlc
 * compiler. You can still use printf for debugging without including
 * <stdio.h>, although you might get a compiler warning. In general,
 * it's not good practice to ignore compiler warnings, but in this
 * case it's OK.
 */
```

Instructions – Coding(1)

- Access your machine by ssh and **change directory to datalab-handout/** and **open bits.c with vim**

- Press letter ‘a’, ‘i’, or ‘o’ to enter ‘insert’ mode

```
int ilog2(int x) {
  return 2;
}
/*
 * float_neg - Return bit-level equivalent of expression -f for
 *   floating point argument f.
 *   Both the argument and result are passed as unsigned int's, but
 *   they are to be interpreted as the bit-level representations of
 *   single-precision floating point values.
 *   When argument is NaN, return argument.
 *   Legal ops: Any integer/unsigned operations incl. ||, &&. also if, while
 *   Max ops: 10
 *   Rating: 2
 */
unsigned float_neg(unsigned uf) {

 return 2;
}
/*
 * float_i2f - Return bit-level equivalent of expression (float) x
 *   Result is returned as unsigned int, but
 *   it is to be interpreted as the bit-level representation of a
 *   single-precision floating point values.
 *   Legal ops: Any integer/unsigned operations incl. ||, &&. also if, while
 *   Max ops: 30
 *   Rating: 4
-- INSERT --
```

This white INSERT message will appear when vim is in insert mode

Instructions – Coding(2)

- **Save after you write something**
  - exit insert mode by pressing esc on your key board (the white INSERT line should disappear)
  - Press ':' on your keyboard and type 'wq' which means (w)rite and (q)uit
  - Write 'q!' instead of wq to (q)uit without writing

```
unsigned float_neg(unsigned uf) {
//bla bla bla bla bla bla bla
this is not the answer~!!

 return 2;
}
/*
 * float_i2f - Return bit-level equivalent of expression (float) x
 *   Result is returned as unsigned int, but
 *   it is to be interpreted as the bit-level representation of a
 *   single-precision floating point values.
 *   Legal ops: Any integer/unsigned operations incl. ||, &&. also if, while
 *   Max ops: 30
:wq
```

Instructions – Compiling

- **Before proceeding please type 'make' inside datalab-handout directory!!**
  - **Just typing 'make' makes all binary(btest, ishow and fshow)**
- **Use dlc to check compilation**
  - **Type './dlc bits.c'**
- **To make a testing binary for bits.c you must 'make' the binary**
  - **Type 'make btest' to compile testing binary**
  - **Type 'make clean' for erasing all compiled binary**
- **Execute the binary by typing './btest'**
  - **Shows expected score to receive per puzzle**
  - **Please refer to datalab.pdf uploaded at KLMS for more information of btest**

Instructions – Helpful Tools

- **Must type 'make' before using the tools provided**
- **Use ishow to see the hexadecimal, signed, unsigned representation of a number**

```
cs230_ta@canis01:~/datalab-handout$ ./ishow
Usage: ./ishow val1 val2 ...
Values may be given in hex or decimal
cs230_ta@canis01:~/datalab-handout$ ./ishow 0xffffffff
Hex = 0xffffffff,       Signed = -1,    Unsigned = 4294967295
cs230_ta@canis01:~/datalab-handout$ ./ishow 50
Hex = 0x00000032,       Signed = 50,    Unsigned = 50
cs230_ta@canis01:~/datalab-handout$ ./ishow -1
Hex = 0xffffffff,       Signed = -1,    Unsigned = 4294967295
```

- **Use fshow to see the interpretation of binary as a floating point number**

```
cs230_ta@canis01:~/datalab-handout$ ./fshow 0x7bf80000

Floating point value 2.575379242e+36
Bit Representation 0x7bf80000, sign = 0, exponent = 0xf7, fraction = 0x780000
Normalized.  +1.9375000000 X 2^(120)
```

Instructions - Grading

- **You can know how much points you will get for your solution, type '. /driver.pl '**

```
Correctness Results     Perf Results
Points  Rating  Errors  Points  Ops     Puzzle
1       1       0       2       4       bitAnd
2       2       0       2       3       getByte
3       3       0       2       10      logicalShift
4       4       0       2       25      bitCount
0       4       1       0       3       bang
1       1       0       2       1       tmin
0       2       1       0       11      fitsBits
0       2       1       0       4       divpwr2
0       2       1       0       2       negate
3       3       0       2       6       isPositive
0       3       1       0       0       isLessOrEqual
4       4       0       2       37      ilog2
0       2       1       0       2       float_neg
0       4       1       0       0       float_i2f
0       4       1       0       0       float_twice

Score = 32/71 [18/41 Corr + 14/30 Perf] (108 total operators
```

Ex) 18 for correctness
14 for performance

*** You will receive maximum 41 points for correctness and 30 points for performance, 5 points for style**

= total 76 points

Instructions - Submit

- After finishing the assignment you **must submit your code to the server**

1) Go to directory ~/datalab-handout
   - cd ~/datalab-handout
2) Type 'submit Lab1 bits.c'
3) You should see the following messages

```
cs230_ta@canis01:~/datalab-handout$ ls
Driverhdrs.pm  Driverlib.pm  Makefile  README  bits.c  bits.h  btest.c  btest.h  decl.c  dlc  driver.pl  fshow.c  ishow.c  tests.c
cs230_ta@canis01:~/datalab-handout$ submit Lab1 bits.c

[CS230] System Programming Submit tool
=====================================================
 * Your ID :  cs230_ta
 * Lab Number :  Lab1
 * Making first submission for Lab1
 * Upload Status : Success
 * Upload Time :  2016-09-10 22:31:03
 * Upload Count (submission version) : 1

If there seems to be a problem with your submission
please email cs230_ta@calab.kaist.ac.kr.
=====================================================

cs230_ta@canis01:~/datalab-handout$
```

Instructions – Check Submission

- **You can check your submission status**

1)Type 'submit check Lab1'

2)You should see something similar to the following messages

```
cs230_ta@canis01:~/datalab-handout$ submit check Lab1

[CS230] System Programming Submit tool
=====================================================
 * Your ID : cs230_ta
 * Lab Number : Lab1
 * Upload Time(Latest) : 09-10_22:31:03

If there seems to be a problem with your submission
please email cs230_ta@calab.kaist.ac.kr.
=====================================================
```

Precautions(1)

- **Warnings to ignore**
  - **The following warnings can be ignored**
- **1) when testing compilation with dlc**

```
cs230_ta@canis01:~/datalab-handout$ ./dlc bits.c
/usr/include/stdc-predef.h:1: Warning: Non-includable file <command-line> included from includable file /usr/include/stdc-predef.h.

Compilation Successful (1 warning)
cs230_ta@canis01:~/datalab-handout$
```

- **2) when compiling btest**

```
cs230_ta@canis01:~/datalab-handout$ make btest
gcc -O -Wall -m32 -lm -o btest bits.c btest.c decl.c tests.c
btest.c: In function 'main':
btest.c:528:9: warning: variable 'errors' set but not used [-Wunused-but-set-variable]
     int errors;
         ^
cs230_ta@canis01:~/datalab-handout$
```

Precautions(2)

- **Write straightline Code : Code with no control constructs such as if, do, while, for, switch**
  - **Except for 3 puzzles for floating points**
  - **float_neg, float_i2f, float_twice**
- **Example) code for negating bits**

```c
//acceptable code
int negateBits(int x)
{
  int tx=x;
  int result=~tx;
  return result;
}
```

```c
//unacceptable code
int negateBits(int x)
{
  int num=31;
  int mask=0;
  int i; //index
  for(i=31; i>=0; i--)
  {
    if( !((x>>i) & 1))
    {
     mask>>i;
     mask |=1;
    mask<<i;
    }
  }
  re
}
```

Precautions(3)

- **C Code Style: C code should always follow this style**

```
int Funct(arg1, arg2, ...) {
 /* brief description of how your implementation works */
 int var1 = Expr1;
 ...
int varM = ExprM;
varJ = ExprJ;
 ...
varN = ExprN;
return ExprR;
 }
```

Example

```
int negateBits(int x)
{
  int tx;
  int result;
  tx=x;
  result = ~x;
  return result;
}
```

Precautions(4)

- **Constraints on constants**
  - **You are not allowed to use big constants such as 0xffffffff**
  - **Use integer constants from 0 to 255 (0xff)**

- **Don't #include<stdio.h>**
  - **Results in non-intuitive error messages**
  - **You can still use printf without stdio.h for this assignment**

- **Use the dlc to check that your solutions conform to the coding rules**
  - **Avoid unwanted grading surprises**

Contest!! Beat the Instructor

- The person who has solved all 15 puzzles and uses the fewest number of operations can participate in the contest

- The top 3 students who beats the instructor(TA) will receive a special prize from the TAs

- How to check the score board:
  - Open up a browser(Internet explorer, Chrome)
  - Type 'http://143.248.188.15:18080' as address

- Let's have a look at the score board~ now!

Administration(1)

- **Due Date : ~ 2016/9/22, 23:59**

- **Only Electronic handins (no handwritten reports)**

- **Results are graded on submitted code**
  - **If you do not submit, no grade!**
  - **Will be graded on the most recently submitted code**

Administration(2)

- **Late Policy:**
  - **Accept submits until 3 days over due**
  - **Receive 15% penalty per day over due work**

- **Definition of 'late':**
  - **The most recent submission is over due**
    - **Even though you have a submission before due date, the latest submission will be used for grading.**
  - **Be cautious when submitting! TAs will not accept complains such as "I accidently submitted", "My friend submitted for me accidently"**

- **TA's E-mail Address :**
  - **Seungbeom Choi : sbchoi@calab.kaist.ac.kr**
  - **Seungheun Jeon : shjeon@calab.kaist.ac.kr**

Thank You!!