Lab02: Bits, Bytes and Integers

CSE4009: System Programming

Bit-Level Rep. & Operation in C

- Bit-level Representation

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

- Bit-level Operations

```
  01101001      01101001
& 01010101    | 01010101
  --------    --------
  01000001      01111101

  01101001
^ 01010101    ~ 01010101
  --------    --------
  00111100      10101010
```

Exercise 1

- Write your own C program
  - Takes two numbers and a character: **`%d %c %d`**
  - For unary operations, use 0 as the first number

```
#include <stdio.h>
#include <stdlib.h>

void printb(char ch) {
  // TODO: implement here....
  // 41 -> 00101001
}

int main(int argc, char** argv) {
  char op;
  int num1, num2;

  // TODO: implement here ...
  // Supported operations: & | ^ ~
  return 0;
}
```

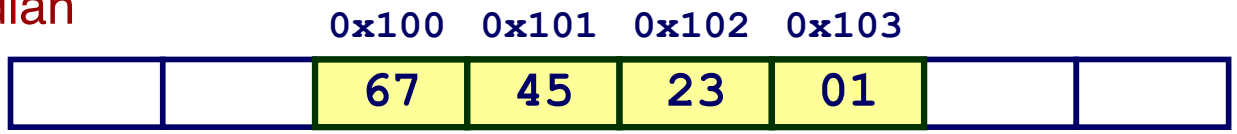
```
0 ~ 41
~ 00101001
----------
  11010110


55 & 59
  00110111
& 00111011
----------
  00110011
```

Byte Ordering

- Our lab environment might be on x86-64 (i.e., Little endian)
  - If we have 4-byte value of 0x01234567 at 0x100

Little Endian



Exercise 2

- Write a sample code and run GDB to examine memory
  - r: run
  - b: break on a certain point (i.e., a function name or line number)
  - p: print a value
  - x: print memory

```
#include <stdio.h>

void show_bytes(char* start, size_t len) {
  size_t i;
  for (i = 0; i < len; i++)
    printf("%p\t%.2x\n", start + i, start[i]);
  printf("\n");
}

int main(void) {
  int val = 0x01234567;
  show_bytes((char*) &val, sizeof(val));
  return 0;
}
```