COMP1521 Tutorial 03

Signed and Unsigned Integers

- On a machine with 16-bit integers, the C expression (30000 + 30000) yields a negative result.
  - Why the negative result? How can you make it produce the correct result?

Twos Compliment

- Assume that of the following hexadecimal values are 16-bit twos-complement. Convert each to the corresponding decimal value.
  - 0x0013
  - 0x0444
  - 0x1234
  - 0xFFFF
  - 0x8000

Printf() formatting

- What does the following printf() statement display? ("man 7 ascii" will help with this)
  - printf("%c%c%c%c%c%c", 72, 101, 0x6c, 108, 111, 0x0a);

Unicode Encoding

- Another way of representing characters (Other way is ASCII)
- Uses up 4 bytes to represent characters

#bytes	#bits	Byte 1	Byte 2	Byte 3	Byte 4
1	7	0xxxxxxx	-	-	-
2	11	110xxxxx	10xxxxxx	-	-
3	16	1110xxxx	10xxxxxx	10xxxxxx	-
4	21	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Unicode Encoding

- Show the complete Unicode bit-string for each of the following Unicode characters (written in hexadecimal).
- If the character is ASCII, show its representation as a C char.

#bytes	#bits	Byte 1	Byte 2	Byte 3	Byte 4
1	7	0xxxxxxx	-	-	-
2	11	110xxxxx	10xxxxxx	-	-
3	16	1110xxxx	10xxxxxx	10xxxxxx	-
4	21	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

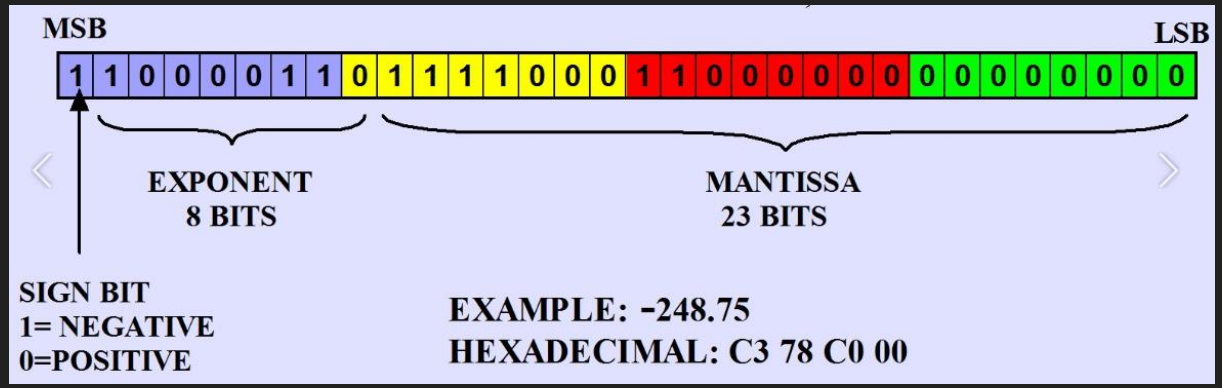
Symbol	Code
{	0x0007B
ë	0x000EB
ф	0x00444
≤	0x02264
𝔶	0x1D536

Floating Point

- Also used as doubles in C (Double precision floating point)
- Interprets bit-strings different to integers

- Sign Bit = Self-explanatory
- Exponent = 2^((integer of 8 bits) – 127)
- Mantissa = 1 + $(bit_{22} * 2^{-1} + bit21 * 2^{-2} + \cdots)$

- Result = Multiply everything together



Floating Point

- What decimal numbers do the following single-precision IEEE754-encoded bit-strings represent?
  - 0 00000000 00000000000000000000000
  - 1 00000000 00000000000000000000000
  - 0 01111111 10000000000000000000000
  - 0 01111110 00000000000000000000000
  - 0 01111110 11111111111111111111111
  - 0 01101110 10100000101000001010000

Floating Point

- Convert the following decimal numbers into IEEE754-encoded bit-strings
  - 2.5
  - 0.375
  - 27.0
  - 100.0

Pointer refresher (again… but useful this time [I promise])

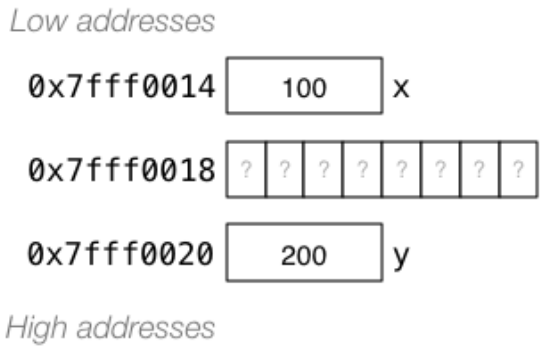
- Size of these variables
  - int
  - short int
  - char
  - double
  - struct xyz { int x; int y; int z; }
  - struct abc { char a; int b; float c; }

Buffer overflow (uh-oh)

Consider the following C program:

```
1. int main(void)
2. {
3.     int x = 100;
4.     char s[8];
5.     int y = 200;
6.     ...
7.     strcpy(s, "a long name");
8.     ...
9. }
```

If the memory looks like



Fun with unions

- Allows us to interpret a piece of memory as multiple explicit types!
- What will these printf()'s produce?
  - printf("%x\n", var.uval);
  - printf("%d\n", var.ival);
  - printf("%c\n", var.cval);
  - printf("%s\n", var.sval);
  - printf("%f\n", var.fval);
  - printf("%e\n", var.fval);

```
union _all {
    int   ival;
    char cval;
    char  sval[4];
    float fval;
    unsigned int uval;
};
```