15-213-S09 Recitation #1

Jernej Barbic (S06)
Revised: Alex Gartrell (S09)
Little Endian to Big Endian

- 32-bit unsigned integer: 0x257950B2
  \[ n = 37 \times 2^{24} + 121 \times 2^{16} + 80 \times 2^8 + 178 \]

- Little Endian:
  \[
  \begin{array}{cccc}
  31 & 24 & 16 & 8 \\
  37 & 121 & 80 & 178 \\
  \end{array}
  \]

- Big Endian:
  \[
  \begin{array}{cccc}
  31 & 24 & 16 & 8 \\
  178 & 80 & 121 & 37 \\
  \end{array}
  \]
Little Endian to Big Endian

- Little Endian: 37 121 80 178
- Big Endian: 178 80 121 37

unsigned int LE2BE(unsigned int x)
{
    unsigned int result =
        ((x << 24) | ((x << 8) & (0xFF << 16)) | ((x >> 8) & (0xFF << 8)) | ((x >> 24) & 0xFF);

    return result;
}
If-then-else

```java
if (condition)
    expression1;
else
    expression2;
```
If-then-else

if (condition)
    expression1;
else
    expression2;

Rewrite as:
( condition & expression1) | (~condition & expression2)
Datalab Example - Max

Return the max of 2 ints \((x, y)\)

2 cases:
- Different signs – look at sign of \(x\)
- Same sign – look at sign of \(x-y\)

```c
int max (int x, int y) {
    int sub = x - y;
    int signcomp = x ^ y; // compare the signs
    // if signs are similar, take the sign of x-y, if not, take the sign of x.
    // This will be 1 if y is greater, 0 otherwise
    int mask = ((signcomp) & x) | (~signcomp & sub)) >> 31;
    return (mask & y) | (~mask & x);
}
```
Datalab Example - Least Bit Pos

Return a mask of the least bit position of n

Take a binary number x (1011...010..0)
Flip the bits (0100...101...1)
Add 1 to it (0100...110...0)
Only 1's occupying the same position between each the new number and original are in the lowest bit position

int leastBitPos(int x) {
    return x & (~x + 1);
}
2’s complement: some examples

- $0x0 = 0$
- $0x1 = 1$
- $0x7FFFFFFF = 2^{31}-1$  // largest 32-bit int
- $0xFFFFFFFF = -1$
- $0xFFFFFFFFFE = -2$
- $0x800000000 = -2^{31}$  // smallest 32-bit int
- $0x8000000001 = -2^{31} + 1$
Simple 8-bit Integer Example

Some simple, positive examples:

0b0001 0110
= 128(0) + 64(0) + 32(0) + 16(1) + 8(0) + 4(1) + 2(1) + 1(0)
= 16 + 4 + 2
= 22

53
= 32 + 16 + 4 + 1
= 128(0) + 64(0) + 32(1) + 16(1) + 8(0) + 4(1) + 2(0) + 1(1)
= 0011 0101
Signed 8-bit Integer Examples

Using $-x = (\sim x) + 1$

-14

$= -(14)$
$= -(8(1) + 4(1) + 2(1) + 1(0))$
$= -(0000 1110)$
$= \sim(0000 1110) + 1$
$= (1111 0001) + 1$
$= 1111 0010$

Using negative sign bit (for numbers close to tMin)

-120

$= -128(1) + 8(1)$
$= 1000 1000$
Other Signed 8-bit Integer Examples

12 =

63 =

72 =

-10 =

-100 =
Other Signed 8-bit Integer Examples

12 = 0000 1100

63 = 0011 1111

72 = 0100 1000

-10 = ~(0000 1010) + 1 = 1111 0101 + 1 = 1111 0110

-100 = -128 + 28 = 1001 1100
Floating point

Single precision:

k = 8, n = 23

Note: exponent boundary is NOT aligned with byte boundary
e.g. 0xFF7FFFFFFFFFF has lowest exponent bit zero (is normalized v.)

Double precision:
k = 11, n = 52
Floating point decision diagram

\[ x = (-1)^s \times 2^{1-Bias} \times 0.M \]

Is \( e = 0 \)?

- YES
- NO

Is \( e \) all 1's?

- YES
- NO

Is \( M = 0 \)?

- YES
- NO

Is \( s = 0 \)?

- YES
- NO

Bias = \( 2^{k-1} - 1 \)

\[ x = (-1)^s \times 2^{e-Bias} \times 1.M \]

YES: +\( \infty \)

NO: -\( \infty \)

NaN

Example

$1/4 = 1.0 \times 2^{-2}$

$s=0, \ e = -2 + \text{Bias} = 125, \ M = 0$

representation = $3E800000$
8-bit Floating Point Example

8-bit floating point (1 sign, 3 exponent, 4 fraction)
  bias = $2^{(3 - 1)} - 1 = 3$

encode 13/16

sign bit = 0 (13/16 $\geq$ 0)

$\frac{13}{16} = 13 \times 2^{-4}$
  $= 1101 \times 2^{-4}$
  $= 1.1010 \times 2^{-1}$ (remove the 1 because it's normalized)

exponent = $-1 +$ bias = 2 = 010

result = 0 010 1010
8-bit Floating Point Example

8-bit floating point (1 sign, 3 exponent, 4 fraction)
  bias = \(2^{(3 - 1)} - 1 = 3\)

encode -3/32

sign bit = 1 (-3/32 < 0)

\[\frac{3}{32} = 3 \times 2^{(-5)}\]
  \[= 0b11 \times 2^{(-5)}\]
  \[= 1.1 \times 2^{(-4)}\] Note: Lowest possible exponent is -2
  Must denormalize
  \[= .011 \times 2^{(-2)}\]
exponent = 0 (denormalized)

result = 1 000 0110
Other 8-bit Floating Point Examples

\[ \frac{10}{4} = \]

\[ -3 = \]

\[ -10 = \]

\[ \frac{1}{64} = \]
Other 8-bit Floating Point Examples

\[ 10/4 = 0\ 100\ 0100 \]

\[ -3 = 1\ 100\ 1000 \]

\[ -10 = 1\ 110\ 0100 \]

\[ 1/64 = 0\ 000\ 0001 \]
Good coding style

• Consistent indentation
• Avoid long sequences of commands without a comment
• Each source file should have an appropriate header
• Have a brief comment at the beginning of each function
Version Control - Do It

Version Control allows you to maintain old versions of your source code, potentially saving you if you were to do something like `rm -rf *`

Good SVN tutorial:

Note:
```bash
repoDir=`pwd` # of working dir
Path to repo is svn+ssh://unix.andrew.cmu.edu/${repoDir}
```

Example:
```
svn+ssh://unix.andrew.cmu.edu/afs/andrew.cmu.edu/usr17/agartrel/svn/15-410
```