15-213
Intro to Computer Systems
Recitation #1
By sseshadr
Today

• Introductions

• Datalab Tricks
  – Floating point questions? Go to Office Hours.

• Integer Puzzles

• Parity Example

• Style
Introductions
Datalab Tricks

• Basics
  - >>, <<
  - | vs. ||
  - & vs. &&
  - ! vs. ~

• What is x?
  - int x = (9 | 12) << 1;
  - x = 26
Datalab Tricks

- **Trick #1: Signed-ness**
  - The MOST significant bit
    - 0 -> positive or zero
    - 1 -> negative

- What is...
  - int x = (10 >> 31);
  - int y = (-10 >> 31);
    - It’s NOT 1 (what is arithmetic shifting?)
    - How do we fix that?

- Answers:
  - x = 0 and y = -1
Datalab Tricks

• Trick #2: Properties of Zero
  – Masking
    • 0 & (something) == 0
      [why?]
    • (0−1) & (something) == something
      [why?]
    • Why is this useful?

  – Positive zero vs. negative zero
    • int x = 0; int y = −x;
    • Neither x nor y is negative (MSB is 0 for both)
    • Why is this useful?
Datalab Tricks

• Trick #3: Negation
  – Review: take a 5-bit twos compliment system

\[
\begin{align*}
1 & \quad 0 & \quad 0 & \quad 1 & \quad 0 \\
\downarrow & \quad \downarrow & \quad \downarrow & \quad \downarrow & \quad \downarrow \\
-2^4 & \quad 2^3 & \quad 2^2 & \quad 2^1 & \quad 2^0 \\
\end{align*}
\]

\[-16 + 2 = -14\]
Datalab Tricks

- Trick #3: Negation
  - Review: take a 5-bit twos compliment system

\[ \begin{align*}
0 & \quad 1 & \quad 1 & \quad 1 & \quad 0 \\
-2^4 & + 2^3 & + 2^2 & + 2^1 & + 2^0 \\
8 & + 4 & + 2 & = & 14
\end{align*} \]
**Datalab Tricks**

- **Trick #3: Negation**
  
  - Example:

```
1 0 0 1 0
0 1 1 0 1
0 1 1 1 0
```

```
int x = -14; // -14
int y = ~x; // 13
int z = ~x+1; // 14
```
Datalab Tricks

• Trick #3: Negation
  – In general
    \[-x \equiv (\sim x + 1)\]
  – Does this always work?
    • Tmin?
      – No!
    • Tmax?
      – Yes!
    • Zero?
      – Yes!
    • Everything else?
      – Yes!
Integer Puzzles

Integer C Puzzles

Initialization

```c
int x = foo();
int y = bar();
unsigned ux = x;
unsigned uy = y;
```

- \( x < 0 \) \( \Rightarrow \) \((x \times 2) < 0\)
- \( ux >= 0 \)
- \( x & 7 == 7 \) \( \Rightarrow \) \((x << 30) < 0\)
- \( ux > -1 \)
- \( x > y \) \( \Rightarrow \) \(-x < -y\)
- \( x * x >= 0 \)
- \( x > 0 && y > 0 \) \( \Rightarrow \) \(x + y > 0\)
- \( x >= 0 \) \( \Rightarrow \) \(-x <= 0\)
- \( x <= 0 \) \( \Rightarrow \) \(-x >= 0\)
- \( (x-x)>>31 == -1 \)
- \( ux >> 3 == ux/8 \)
- \( x >> 3 == x/8 \)
- \( x & (x-1) != 0 \)
Integer Puzzles

• \((x < 0) \Rightarrow ((x*2) < 0)\)
  – Nope. Tmin?

• \((ux \geq 0)\)
  – Yup!

• \((x\&7 == 7) \Rightarrow ((x << 30) < 0)\)
  – Yup!
  – \((x\&7 == 7)\) means last 3 bits are 1
  – Examine the “negative bit” of \((x<<30)\)
Integer Puzzles

• \((ux > -1)\)
  – Nope. Unsigned comparison means -1 is Umax!

• \((x > y) \Rightarrow (-x < -y)\)
  – Nope. Boundary cases.
  – \(x = 0, y = \text{Tmin}\) (what is -Tmin?)

• \((x^2 >= 0)\)
  – Nope. Overflow into “negative bit”
  – int \(x = 65535; // 2^{16} - 1\)
Integer Puzzles

• \((x > 0 \land y > 0) \Rightarrow (x + y > 0)\)
  – Nope. Overflow into “negative bit”
  – \(x, y = \text{Tmax}\)

• \((x \geq 0) \Rightarrow (-x \leq 0)\)
  – Yup! Why doesn’t break for Tmax?

• \((x \leq 0) \Rightarrow (-x \geq 0)\)
  – Nope. What is \(-\text{Tmin}\)?
Integer Puzzles

• \((x | -x) >> 31 == -1\)
  – Nope. \(x = 0\)

• \((ux >> 3) == (ux / 8)\)
  – Yup!

• \((x >> 3) == (x / 8)\)
  – Nope. Careful on rounding!
  – \(int x = -19;\)
  – \(int y = x >> 3; \quad \text{// } y = -3\)
  – \(int z = x / 8; \quad \text{// } z = -2\)
Integer Puzzles

• $(x \ & \ (x-1)) \neq 0$
  
  — Nope. $x = 0$, $x = 1$
Parity Example

• Write a function which takes an integer and returns
  – 1 if there are an odd number of ‘1’ bits
  – 0 if there are an even number of ‘1’ bits

```c
int parity_check(int x){
    ...
}
```

• Any ideas?
Parity Example

• Inspiration:
  – If we could XOR all of the bits in the argument... we would get the answer!

\[
\begin{align*}
110110010110001111110010100101101 \\
110110010110001111100101001101101 \\
110110010110001111100101001101101 \\
\hline
\text{XOR} \\
110110010110001111100101001101101 \\
\hline
0011110001001110 \\
\end{align*}
\]
Parity Example

• Just keep going!

\[0011110001001110\]
\[00111100\]
\[01001110\]
\[00111100\]
\[01001110\]
\[00111100\]
\[01001110\]
\[00111100\]
\[01001110\]
\[00111100\]
\[01001110\]
\[00111100\]
\[01001110\]
\[00111100\]
\[01001110\]
\[01110010\]
\[\text{(down to 8 bits)}\]
Parity Example

- Just keep going!

01110010
01110010
01110010

XOR
0111
0010

0101 (down to 4 bits)
Parity Example

• You can take it from there.
  – Still confused on high-level algorithm? Can’t write the C code for the Parity Problem? Office Hours.
Style

• Here is what we grade on:

• It is in your best interest to read it ASAP!

• Autolab isn’t the whole grade. We read your code.
Style

- Documentation
- Whitespace
- Line length
- Variable names
- Magic Numbers
- Dead Code
- Modularity
- Error checking
- Consistency