CS 356 Project 1

Data Lab (Part 1)

Overview

- Implement functions for a given task/puzzle
- Basic rules for 'integer' (i.e. non-floating point) puzzles
  - Use only allowed operators
    - Generally, allowed: `+ & | ^ ~ << >>`
      - `=` (assignment) is always allowed
    - Generally, disallowed: `- * / % < > ==`
  - Only integer variables
  - Only 8-bit constants (i.e. -128 to +127, 0x00-0xff)
- Utilize your knowledge of integer representation (2's comp. and unsigned) as well as integer operations

Hint: 2's complement Review

- What is the bit pattern of:
  - Max 2's comp. number (Tmax)
  - Min 2's comp. number (Tmin)

Hint: DeMorgan's Theorem

- DeMorgan's Theorem
  - $\neg(x \land y) = \neg x \lor \neg y$
  - $\neg(x \lor y) = \neg x \land \neg y$

Hint: bitMask

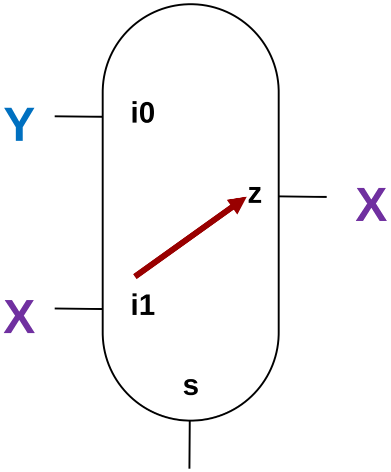
- If we could do binary subtraction, what would the following yield: 010000 – 000010?

- What about: 100000 – 000100?

Hint: Multiplexing

- Multiplexing refers to the process of choosing 1-of-n inputs and passing it to the output
  - Which input is chosen depends on the select
  - Analogy: Traffic cop
- Equivalent of an `if-else statement` (or ? : operator)

2-to-1 Mux



② **Thus, input 1 (i.e. X) is selected and passed to the output**

① **Select bit = 1**

```
if(cond)
   z = x
else
   z = y;

z = cond ? x : y;
```

Multiplexing and Logic

```
if(cond)
   z = x
else
   z = y;
```

- **We can replace the 'if' or '? :' control structure with &, |, and ~ operations**
- Use bitwise logic operations (ANDs and ORs to pass the appropriate value
  - $Z = (\neg S \land \text{In-0}) \lor (S \land \text{In-1})$
  - Analyze the above equation:

When S=0: $Z = (1 \land \text{In-0}) \lor 0 = \text{In-0}$

When S=1: $Z = 0 \lor (1 \land \text{In-1}) = \text{In-1}$

Identity	**0 OR Y = Y**	**1 AND Y = Y**
Null Ops	**1 OR Y = 1**	**0 AND Y = 0**

S (Cond.)	In-1	In-0	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Truth Table of a mux

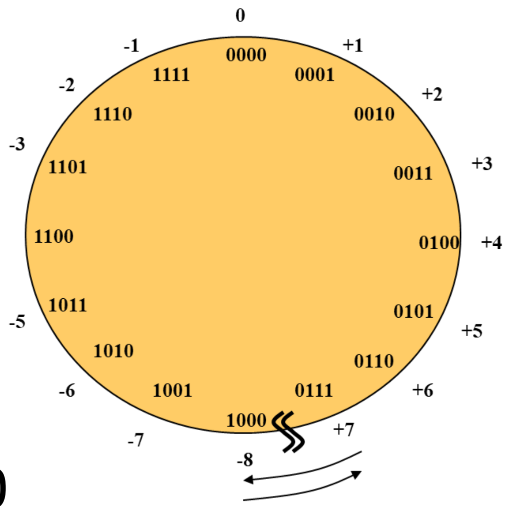
Let S = Cond, In-1 = X, In-0 = Y

Hint: Comparison Via Subtraction

- Suppose we want to compare two signed numbers: A & B
- Suppose we let DIFF = A-B…what could the result tell us
  - If DIFF < 0, then A < B
  - If DIFF = 0, then A=B
  - IF DIFF > 0, then A > B
- How would we know DIFF == 0?
  - If all bits of our answer are 0…
- How would we know DIFF < 0 (i.e. negative)?
  - Check MSB. **But what about overflow!!**

Computing A<B from "Negative" Result

- Recall overflow with signed numbers flips the sign to the **opposite** value of what it should be
- Perform A-B
- If ***there is no overflow***, simply check if MSB = 1 (it is trustworthy)
- So if ***there is overflow,*** check if MSB = 0 (i.e. positive) since that would mean the result truly should be negative
- Summary: A-B is "truly" negative if:
  - overflow & MSB=1 **OR**
  - no overflow & MSB=0



Overflow occurs when you cross this discontinuity

Hint: isTMax

- Consider how to solve the alternate problem: isTMin?
  - What is the set of binary numbers that when added to itself will yield 0?
- Consider the relationship between Tmax and Tmin