CMPE12 Final Review
Lab 4 checkoffs as well
Friday 3-5 Office
Monday 3-5 Office

CMPE 13 this summer
21
10-week
Overview

- Final is June 6\textsuperscript{th} 8AM-11AM
- Same policy as before: no notes, books or calculators
- Extra Credit
  - If score is over 100\% will still help final grade
  - Two Problems
    - One repeat from the midterm exactly
    - One random one
- Show your work
  - No credit if we can't figure out how you did it
  - Partial credit
Partial List of Topics

- N-type, p-type transistors
- Realization of truth table from transistors and inverse
- Transistors to standard gates and inverse
- Truth table to gates and inverse
- Sum-of-Products and Product-of-Sums
- Boolean Algebra
- Common Logic elements
  - Mux etc including sequential

- Binary representations
  - Unsigned
    - Binary
    - Hex
    - Octal
  - Signed
    - Two’s complement
    - One’s complement
    - Sign magnitude
- Bias
- Binary Math
  - Overflow indications
Partial List of Topics Continued

- Floating Point
  - Conversions
  - Addition, Subtraction, Multiplication

- LC-3 Architecture

- LC-3 Assembly
  - Op-code translation
  - LC-3 coding and running
    - Subroutine methods
    - Basic data structures
IEEE 754 (SPFP)
Floating Point Format

Representation:

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>23</th>
<th>22</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>E</td>
<td></td>
<td></td>
<td>F</td>
</tr>
</tbody>
</table>
```

- **S** is one bit representing the sign of the number
- **E** is an 8 bit biased integer representing the exponent
- **F** is an 23-bit unsigned integer

The true value represented is: \((-1)^S \times f \times 2^e\)

- **S** = sign bit
- **e** = E − bias
- **\(f\)** = \(F/2^n + 1\)
- for single precision numbers \(n=23\), bias=127
Floating Point Conversion

What is the decimal value for this SP FP number 0x421A 0000?

128 + 4 = 132 - 127 = 5

\[ 1.001101 \times 2^5 \]

38.5
Floating Point Conversion

- What is the (SP FP) value of the decimal value -525.5?

\[ S = 1 \]

\[ 1.00000011011 \times 2^9 \]

\[ \frac{127}{136} \]

\[ 10001000 \]

hidden bit

\[ \text{0x} \text{C4036000} \]
Floating Point Math

0x45FFC000
+0x456B0000

1.01111010101

0.11010101010

1.00001100
+
0.00000001

1000 01100
11 \cdot 10^2
1100
1.1 \cdot 10^3
Floating Point Math part 2

0x45FFC000
+0x456B0000
LC-3 Assembly Coding

• Write LC-3 assembly code that will OR the values in R1 with R3 and store the result in R0.

```
R1 OR R3
R1 R3
```
LC-3 Sub-Routine Coding

• Write the Load Function from Lab 5
  - Assume R0,R1,R2 are used as arguments
  - Be sure to save off registers used
  - Label Base has first address of array and Label Size holds the column length

; save off

; restore at the end
Load R0, R1, R2

; save our registers

; calc RA + ( ... )

STR R0, R3

; restore

RET
LC-3 Data Structures
Array, Stacks and Queues

• Basic theory of each
  – Understanding of how to write basic routines

Push
  ↓
ADD R6, R6, #1 ; decrement stack ptr
STR R0, R6, #0 ; store data (R0)

Pop
  LDR R0, R6, #0 ; load data from TOS
  ADD R6, R6, #1 ; increment stack ptr
LC-3 Code Running

1. (10pts) LC-3 ISA

After the following LC-3 code executes what are the ending contents of the registers and memory? Assume some registers/memories have starting values as indicated. If blank, the content is unknown. Remember that both registers and memory locations are 16-bits wide. The memory portion starts at address 0x3200.

```
LEA    R1, label0
LDR    R2, R1, #0
STR    R0, R1, #4
LEA    R6, label12
ADD    R5, R0, R1
LEA    R0, label1
AND    R7, R2, R5
NOT    R3, R0
STR    R7, R6, #-2
STR    R2, R1, #1
```

<table>
<thead>
<tr>
<th>Memory</th>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0x3200</td>
<td>0xDEAD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x1234</td>
</tr>
<tr>
<td></td>
<td>0x4424</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Register</th>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>0x3234</td>
<td>0x3204</td>
</tr>
<tr>
<td>R1</td>
<td></td>
<td>0x3200</td>
</tr>
<tr>
<td>R2</td>
<td></td>
<td>0xDEAD</td>
</tr>
<tr>
<td>R3</td>
<td></td>
<td>0xC1DFB</td>
</tr>
<tr>
<td>R4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R5</td>
<td></td>
<td>0x4434</td>
</tr>
<tr>
<td>R6</td>
<td></td>
<td>0x3202</td>
</tr>
<tr>
<td>R7</td>
<td></td>
<td>0x4424</td>
</tr>
</tbody>
</table>
shift

100%
97%
17 - 3 = 14

0x4188 0x404
50  50
E4  E1
M 1.0001 M 1.1
4 - 1 = 3
M = 1.1000
\leftarrow
= 0.001100

SPFP Subtraction

0.0001
\times 0.0001
-----------------
0.000001

1.110
↑
E = 3

0x41600000
17 \times 3 = 51 \quad \text{SPFP multiplication}

\begin{align*}
\text{M} & : 1.0001 \quad \text{M} : 1.1 \\
\text{E} & : 4 \quad \text{E} : 1 \\
0 \times 4188 & \quad 0 \times 404 \\
50 & \quad 50 \\
5 + 127 & = 132
\end{align*}

\begin{align*}
1.0001 & \times 1.1 \\
\text{Result} & = 1.100010011 \\
\text{Q} & = 0
\end{align*}

0 \times 424 < 00000
Stack

Push q

Push b a

Pop

q
Stack

push 9

LC = 3

push 9

Pop

9
Queue

6 5 4 3 2 1
C 6 5 4 3 2
C 6 5 4 3

Drawing of a queue with elements 1, 2, and an entry point H.