MSI Review:
Only students who have been attending MSI can attend
Sat the 18th 3:00-4:30PM
ARC 202

CMPE12 Final Review
Overview

• Monday, March 20th, 4–7 p.m.
• Same policy as before: no notes, books or calculators
• Extra Credit
  – If score is over 100% will still help final grade
  – One Problem
• 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
- MIPS
  - Role of registers
  - Theory of Function Calls
- Embedded Uno32
  - I/O
  - Special Purpose Registers (TRIS, etc.)

\[-6.3 \cdot 10^4 \times 4.1 \cdot 10^{20}\]
Transistor and Gates
Truth Table to Gates

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Transistor and Gates

Truth Table to Transistors

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ AB + A\overline{B} \]
Transistors and Gates

Transistors to truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Logic Elements to Gates

• Draw the gate level diagram of a 2-4 decoder
## Base Conversion Table 8 bits

<table>
<thead>
<tr>
<th>Decimal</th>
<th>1’s Complement</th>
<th>2’s Complement</th>
<th>Signed Magnitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>-35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>0110 0001</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>1001 1101</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>10011101</td>
<td></td>
</tr>
</tbody>
</table>
## Base Conversion Table 8 bits

<table>
<thead>
<tr>
<th>Decimal</th>
<th>1’s Complement</th>
<th>2’s Complement</th>
<th>Signed Magnitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>-35</td>
<td>X</td>
<td>0110 0001</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>1001 1101</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>10011101</td>
<td></td>
</tr>
</tbody>
</table>
Bias One's

Binary Arithmetic

- Unsigned

- 2's Complement

- Signed Magnitude
Bias \[ -63 - 63 \]

\[ 4 + 63 = 67 \]
\[ -63 = 64 \]

\[ a + c = (a + b) + (c + b) = a + c + 2b \]

\[ a - c = (a + b) - (c + b) = a + b - c - b \]

\[ a - c \]
Fractional Representation

• 4.6 in Base 2?

100.1001

\[
\begin{align*}
2 \times 0.6 &= 1.2 \\
2 \times 0.2 &= 0.4 \\
2 \times 0.4 &= 0.8 \\
2 \times 0.8 &= 1.6 \\
2 \times 0.6 &= \\
\end{align*}
\]
Binary Division

• 101 | 101110111
Boolean Algebra

- \( X + X \)
- \( x' + x \cdot y \)

\[
XY + X(Y + Z) + Z(X + X') + (X + X')YZ + Y + Y'Z
\]
\[
\neg Y + x + XZ + xZ + yZ + y + \neg yZ
\]
\[
\neg Y + xZ + Z + y
\]
\[
Z(x + 1) + y(x + 1)
\]
\[
\neg Z + y + 1
\]
Floating Point Format

Representation:

![Diagram of floating point format]

- $S$ 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?

\[
\begin{align*}
0 & \quad 0 \quad 1 \quad 1 \quad \rightarrow \\
0 & \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad \rightarrow \\
64 + 32 + 4 & = 100 - 127 = -27 \\
1.001101 \times 2^{-27}
\end{align*}
\]
Floating Point Conversion

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

\[
\begin{align*}
1000001101.1 &= 1.10000011011_2 \\
127 + 9 &= 136 - 128 = 8 \\
10001000 &= 2^{7} + 2^{2} + 2^{1} + 2^{0} \\
\end{align*}
\]
\[ \times 40E \quad 00000 \quad 0 \quad 10000001\ 110 \rightarrow 0 \]
\[ \times 62 \quad 00000 \quad 1 \quad 10001100 \quad 010 \rightarrow 0 \]

\[ 10000001 = 128 + 1 = 129 - 127 = 2 \]

\[ 10001100 = 128 + 4 + 8 \rightarrow 127 = 140 - 127 = 13 \]

\[ 2 + 13 = [15] \]

\[ \begin{array}{c}
1.11 \\
+ 1 \\
\hline
1.01 \\
+ 111000 \\
\hline
1000000 \\
+ 1111000 \\
\hline
10000011
\end{array} \]
16
128 + 15
10001111
000110 \rightarrow 0
< 78C0000
7x - 10240
Floating Point Math

0x45FFC000
+0x456B0000
Floating Point Math part 2

0x45FFC000
+0x456B0000
Binary Arithmetic

• Unsigned

• 2’s Complement

• Signed Magnitude
Arbitrary Base Conversion

- $1210_3$ in base 10

$$27 \cdot 1 + 2 \cdot 9 + 3 \cdot 1 + 1 \cdot 0 = 27 + 18 + 3 + 0 = 48_{10} \implies 48/4 = 12 R 0$$
$$12/4 = 3 R 0$$
$$3/4 = 0 R 3$$
48_{10} \rightarrow ?_5 143_5

48 - 25 = 23
23 - 5 \cdot 4 = 3
3 = 3

\frac{1}{5} \frac{25}{125}
Arbitrary Base Conversion

- 1210₃ in base 10

\[
\begin{align*}
27 \cdot 1 + 2 \cdot 9 + 3 \cdot 1 + 1 \cdot 0 &= 81 + 18 + 3 + 0 = 48, \\
48/4 &= 12 \text{ R } 0, \\
12/4 &= 3 \text{ R } 0, \\
3/4 &= 0.75.
\end{align*}
\]
LC-3 Assembly Coding

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

- Write the Load Function from Lab 4
  - 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
LC-3 Data Structures
Array, Stacks and Queues

• Basic theory of each
  — Understanding of how to write basic routines
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>R0</th>
<th>0x1234</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td></td>
</tr>
<tr>
<td>R3</td>
<td></td>
</tr>
<tr>
<td>R4</td>
<td></td>
</tr>
<tr>
<td>R5</td>
<td></td>
</tr>
<tr>
<td>R6</td>
<td></td>
</tr>
<tr>
<td>R7</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label0</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Label1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>label2</td>
</tr>
</tbody>
</table>
PIC32 Architecture

– Word size, register count, similar to questions asked about the lc-3
– Standard registers available and their uses
PIC32 Ports

• What do TRIS LAT and PORT do?
• Pseudocode on how to use them
PIC32 Function Calls
\[ \begin{align*}
0 \times 4280 & \rightarrow 0 \times 3180 \\
0 & | 000010 \quad 1000 \rightarrow 1.000 \\
0 & | 011101 \quad 1000 \rightarrow 1.000 \\
133 - 127 & = 6 \\
123 - 127 & = -4 \\
133 - 1 & = 132
\end{align*} \]
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F</th>
<th>POS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>(A + B + C) (A + B + C)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>(A + B + C) (A + B + C)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>(A + B + C) (A + B + C)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(A + B + C) (A + B + C)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>(A + B + C) (A + B + C)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>(A + B + C) (A + B + C)</td>
</tr>
</tbody>
</table>