Today’s Lecture

- Course organization
- Computing environment
- Overview of course topics
- Combinational circuits
Course Organization

- Course website
  
  http://www.cse.msu.edu/~cse320/

- Syllabus and calendar

- Enrollment

Computing Environment

- Hardware: array of ARM processors
  
  - pi1.cse.msu.edu
  
  - pi22.cse.msu.edu

- Operating system: Linux

- Accounts
Computer Project #1

- Due Wednesday, 5/17 (by 11:59 PM)
- Focuses on computing environment:
  - UNIX tutorial
  - Using the "vim" editor
  - Using the "handin" system

Architecture and Organization

- Computer architecture focuses on the functional behavior of a computing system as viewed by the programmer (such as the size of an integer data object in bytes).
- Computer organization focuses on the structural relationships that are not visible to the programmer (such as the clock frequency or the total size of RAM).
Levels of Abstraction

We can view a computing system at several levels, from the highest level (users running programs) to the lowest level (transistors operating according to the laws of physics).

The von Neumann Model

The von Neumann model consists of five major components:

1) input unit
2) output unit
3) arithmetic logic unit
4) memory unit
5) control unit
The System Bus Model

- Refine of the von Neumann model
- Communication between components handled by the system bus

The System Bus Model

- Fetch Phase:
  - RAM[PC] ==> IR

- Execute Phase:
  - decode IR
  - take appropriate action
  - update PC

The Fetch-Execute Cycle
Example

- Assume each instruction is 4 bytes long
- Assume PC: 00010700
- Fetch phase:
  - access RAM[00010700]
  - copy 4 bytes (E0827003) to IR
- IR now contains: E0827003

Example (continued)

- Assume IR: E0827003
- Execute phase:
  - decode IR
    - ADD instruction on ARM
  - take appropriate action
  - update PC
    - PC + 4 ==> PC
**RAM**

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td>00000001</td>
<td></td>
</tr>
<tr>
<td>00000002</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>00010700</td>
<td>E0</td>
</tr>
<tr>
<td>...</td>
<td>82</td>
</tr>
<tr>
<td>...</td>
<td>70</td>
</tr>
<tr>
<td>...</td>
<td>03</td>
</tr>
<tr>
<td>FFFFFFFD</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFE</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF</td>
<td></td>
</tr>
</tbody>
</table>

**CPU**

PC: 00010700
IR: E0827003

**ADD**

R[3] →
Number Systems

Must be able to convert between bases:

- Machines use base 2 (binary)
- Humans use base 10 (decimal)
- Humans abbreviate base 2 using base 16 (hexadecimal) or base 8 (octal)
### Powers of Two

- $2^0 = 1$
- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$
- $2^4 = 16$
- $2^5 = 32$
- $2^6 = 64$
- $2^7 = 128$
- $2^8 = 256$
- $2^9 = 512$
- $2^{10} = 1024$
- $2^{11} = 2048$
- $2^{12} = 4096$
- $2^{13} = 8192$
- $2^{14} = 16384$
- $2^{15} = 32768$

### Shorthand for Binary

Base 16 often used instead of base 2

Example:

0100101101111100 = 4b7c

Groups of four bits (from right):

0100 1011 0111 1100 = 4b7c
Shorthand for Binary

Base 8 sometimes used instead of base 2

Example:

010010110111110 = 22676

Groups of three bits (from right):

010 010 110 111 110 = 22676

Example: ASCII characters

A 1000001 100 0001 41
B 1000010 100 0010 42
C 1000011 100 0011 43
X 1011000 101 1000 58
Y 1011001 101 1001 59
Z 1011010 101 1010 5a
Example: UNIX file permissions

Permissions for each file:

```
  rwx rwx rwx (owner, group, world)
```

Make directory public:

```
chmod 755 my_directory
```

Make file private:

```
chmod 600 my_file
```

Convert Other Base to Decimal

Example: 2756 base 8 ==> base 10

```
2756 base 8  =  2 * 8^3 + 7 * 8^2 + 5 * 8^1 + 6 * 8^0
               =  1518 base 10
```

Nested form:

```
2756 base 8  = ((((((2) * 8) + 7) * 8) + 5) * 8) + 6)
               =  1518 base 10
```
Convert Other Base to Decimal

Algorithm:

answer = 0
iterate over digits in original number
    answer = answer * base + current digit

Convert Other Base to Decimal

2756 base 8 ==> base 10

answer = 0
    = (0 * 8) + 2 = 2
    = (2 * 8) + 7 = 23
    = (23 * 8) + 5 = 189
    = (189 * 8) + 6 = 1518

2756 base 8 = 1518 base 10
Convert Decimal to Other Base

Example: 44 base 10 ==> base 2

\[
\begin{align*}
44 / 2 &= 22 \ R \ 0 \\
22 / 2 &= 11 \ R \ 0 \\
11 / 2 &= 5 \ R \ 1 \\
5 / 2 &= 2 \ R \ 1 \\
2 / 2 &= 1 \ R \ 0 \\
1 / 2 &= 0 \ R \ 1
\end{align*}
\]

44 base 10 = 101100 base 2

Convert Decimal to Other Base

Example: 44 base 10 ==> base 8

\[
\begin{align*}
44 / 8 &= 5 \ R \ 4 \\
5 / 8 &= 0 \ R \ 5
\end{align*}
\]

44 base 10 = 54 base 8
Convert Decimal to Other Base

Algorithm:

\[
\text{value} = \text{original number} \\
\text{loop until value == 0} \\
\quad \text{current digit} = \text{value} \mod \text{base} \\
\quad \text{value} = \text{value} / \text{base}
\]

Summary: base conversions

Base 2 ==> Base 16: group digits
\[
\text{ex: } 1100011 \text{ base } 2 \Rightarrow 63 \text{ base } 16
\]

Base 16 ==> Base 2: decompose digits
\[
\text{ex: } 5C \text{ base } 16 \Rightarrow 1011100 \text{ base } 2
\]

Other Base ==> Base 10: multiply and add

Base 10 ==> Other Base: repeated division
Combinational Circuits

- Circuit design based on Boolean algebra
- Three equivalent representations
  - algebraic expressions
  - truth tables
  - circuit diagram

Example: Exclusive OR

- Expression in Boolean algebra:
  \[ F(A,B) = A'B + AB' \]
- Truth table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F(A,B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
- Circuit diagram:

![Circuit Diagram]

**Boolean algebra**

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Dual</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A + B = B + A$</td>
<td>$A + B = B + A$</td>
<td>Commutative</td>
</tr>
<tr>
<td>$A(B + C) = A(B + C)$</td>
<td>$A + B(C + A) = A + B$</td>
<td>Associative</td>
</tr>
<tr>
<td>$1A = A$</td>
<td>$0 + A = A$</td>
<td>Identity</td>
</tr>
<tr>
<td>$A + 0 = A$</td>
<td>$A + A = A$</td>
<td>Complement</td>
</tr>
<tr>
<td>$0A = 0$</td>
<td>$1A = 1$</td>
<td>Zero and one theorems</td>
</tr>
<tr>
<td>$AA = A$</td>
<td>$A + A = A$</td>
<td>Idempotence</td>
</tr>
<tr>
<td>$A(B + C) = (A + B)C$</td>
<td>$A(B + C) = (A + B)C$</td>
<td>Involution</td>
</tr>
<tr>
<td>$A + B = A + B$</td>
<td>$A + B = A + B$</td>
<td>DeMorgan’s Theorem</td>
</tr>
<tr>
<td>$AB + AC + BC = A(B + C)$</td>
<td>$AB + AC + BC = (A + B)(A + C)$</td>
<td>Consensus Theorem</td>
</tr>
<tr>
<td>$A(A + B) = A$</td>
<td>$A + AB = A$</td>
<td>Absorption Theorem</td>
</tr>
</tbody>
</table>
All possible functions on two inputs

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>( B )</td>
</tr>
<tr>
<td>0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0 1</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1 1</td>
<td>0 1 0 0</td>
</tr>
</tbody>
</table>

Basic Logic Gates

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>( B )</td>
</tr>
<tr>
<td>0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0 1</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1 1</td>
<td>0 1 0 0</td>
</tr>
</tbody>
</table>

33

34
Other Logic Gates

- Any circuit can be defined using only: 
  \{ NOT, AND, OR \}

- Other complete gate sets:
  \{ NAND \}
  \{ NOR \}
More Than Two Inputs

- Useful to define gates which have more than 2 inputs.
  
  AND: output is 1 if all inputs are 1
  
  OR: output is 1 if any input is 1

- Not meaningful for NOT

More Than Two Inputs

- Can be implemented using cascading:

- Can also be implemented directly (more efficient)
Standard Forms

- Canonical Sum-of-Products Form:
  \[ F(A,B) = A'B + AB' \]

- The expression is the sum of a series of products, where each product is a minterm

- A minterm is a product where each variable is present (complemented or uncomplemented)

Minterms

- For a function with two inputs, there are four possible minterms:
  
m0: A'B'  
m1: A'B  
m2: AB'  
m3: AB  

- Canonical SOP form has a subset of all possible minterms