UNIT 8B

Computer Organization:
Levels of Abstraction
Announcements

• Midsemester grades will be assigned this weekend
• Ps7 is due Friday March 22 in class
• Pa7 will be assigned shortly (due after break)
Abstraction

• We can use layers of abstraction to hide details of the computer design.
• We can work in any layer, not needing to know how the lower layers work or how the current layer fits into the larger system.
  -> transistors
  -> gates
  -> circuits (adders, multiplexors, flip-flops)
  -> central processing units (ALU, registers, control)
  -> computer
Central Processing Unit (CPU)

• A CPU contains:
  – Arithmetic Logic Unit to perform computation
  – Registers to hold information
    • Instruction register (current instruction being executed)
    • Program counter (to hold location of next instruction in memory)
    • Accumulator (to hold computation result from ALU)
    • Data register(s) (to hold other important data for future use)
  – Control unit to regulate flow of information and operations that are performed at each instruction step
A sample CPU

http://cpuville.com/main.htm
Computer

Central Processing Unit (CPU)
- Control Unit
- ALU
- Registers

Main Memory
Secondary Memory
Storage

Bus

Input Devices
- Keyboard
- Mouse

Output Devices
- Display
- Printer

http://cse.iitkgp.ac.in/pds/notes/intro.html
To Gates

Properties (Similar to × and +)

- Commutative: \( a \land b = b \land a \) \( a \lor b = b \lor a \)
- Associative: \( a \land (b \land c) = (a \land b) \land c \)
  \( a \lor (b \lor c) = (a \lor b) \lor c \)
- Distributive: \( a \land (b \lor c) = (a \land b) \lor (a \land c) \)
  \( a \lor (b \land c) = (a \lor b) \land (a \lor c) \)
- Identity: \( a \land 1 = a \) \( a \lor 0 = a \)
- Dominance: \( a \land 0 = 0 \) \( a \lor 1 = 1 \)
- Idempotence: \( a \land a = a \) \( a \lor a = a \)
- Complementation: \( a \land \neg a = 0 \) \( a \lor \neg a = 1 \)
- Double Negation: \( \neg \neg a = a \)
### More gates

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>A nand B</th>
<th>A nor B</th>
<th>A xor B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **nand ("not and")**: \( A \text{ nand } B = \neg (A \land B) \)
- **nor ("not or")**: \( A \text{ nor } B = \neg (A \lor B) \)
- **xor ("exclusive or")**: 
  \[ A \text{ xor } B = (A \land \neg B) \lor (B \land \neg A) \]
All Gates

A ∧ B “AND”  

A ∨ B “OR”  

A → “NOT”  

¬(A ∧ B)  

¬(A ∨ B)  

A ⊕ B
Adding Binary Numbers
Adding Binary Numbers

Half Adder
A Full Adder

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>C_in</th>
<th>C_out</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
A Full Adder

\[
\begin{align*}
S &= A \oplus B \oplus C_{in} \\
C_{out} &= ((A \oplus B) \land C) \lor (A \land B)
\end{align*}
\]
Full Adder (FA)

\[
S = A \oplus B \oplus C_{in}
\]

\[
C_{out} = ((A \oplus B) \land C) \lor (A \land B)
\]
Another Full Adder (FA)

http://students.cs.tamu.edu/wanglei/csce350/handout/lab6.html
8-bit Full Adder

A\_7 \quad B\_7
\uparrow
\downarrow
\downarrow
1-bit Full Adder
\quad S\_7

\ldots

A\_2 \quad B\_2
\uparrow
\downarrow
\downarrow
1-bit Full Adder
\quad S\_2

A\_1 \quad B\_1
\uparrow
\downarrow
\downarrow
1-bit Full Adder
\quad S\_1

A\_0 \quad B\_0
\uparrow
\downarrow
\downarrow
1-bit Full Adder
\quad S\_0

A \quad B
\uparrow
\downarrow
\downarrow
8-bit FA
\quad S

C\_out \quad C\_in
\downarrow
\uparrow
\uparrow
8

15110 Principles of Computing,
Carnegie Mellon University - CORTINA
Exercise

- Convert expressions to circuit

\[
\begin{align*}
A \land B & \quad \text{“AND”} \\
A \lor B & \quad \text{“OR”} \\
\neg A & \quad \text{“NOT”}
\end{align*}
\]
Exercise

• Convert circuit to expressions
exercise

• Designing a control system
From Design to circuits

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