CS370  Computer Architecture

- Instructor: Tao Xie
- Email address: txie@sdsu.edu
- Office: GMCS 535
- Office hours: MW 11 am – 12 pm (noon) or by appointment
- Course website: https://taoxie.sdsu.edu/cs370/index.html
- Prerequisite: CS237 or equivalent
Lab Software

- LogicWorks5 (Windows) software is needed for lab assignments and it has been installed on computers in GMCS 422 & 425; or you can buy it from Amazon.
- Lab1 is individual; Lab2&3 are group projects (each group no more than 2 students)
- You can work on lab assignments at home on your own computer
- Your grader will provide 3 sessions per week at GMCS 425 to help you on the 3 lab assignments.
Class Schedule in Online Now!

CS 370 Class Schedule for Spring 2020

January 22: Lecture 1 (Class Guidelines & Chapter 1 “Digital Systems and Information”)

January 27: Lecture 2 (Chapter 2 “Combinational Logic Circuits”, part 1 – Gate Circuits and Boolean Equations)

January 29: Lecture 3 (Chapter 2 “Combinational Logic Circuits”, part 1 – Standard Forms)

February 3: Lecture 4 (Chapter 2 “Combinational Logic Circuits”, part 2 – Circuit Optimization)

February 5: Lecture 5 (Chapter 2 “Combinational Logic Circuits”, part 2 – K-Map Manipulation)

February 10: Lecture 6 (Chapter 2 “Combinational Logic Circuits”, part 3 – Additional Gates and Circuits)

February 12: Lecture 7 (Chapter 3 “Combinational Logic Design”, part 1 – Implementation Technology and Logic Design)

February 17: Lecture 8 (Chapter 3 “Combinational Logic Design”, part 2 – Functions and Functional Blocks)

February 19: Lecture 9 (Preview for Midterm Exam One)

February 24: Midterm Exam One (75 minutes in class)
Graders and There Office Hours

Section 1: Mattew Rose
Email: MattRose370@gmail.com

Session 2: Samantha Quiroz
Email: samiq370@gmail.com

Office hours: TBD
A Hello Email to your Grader by Jan. 31

- Send a Hello email to your grader
- Email subject is “Hello, cs370!”
- Please tell him your Red ID, full name, and your working email address, and a screenshot of your transcript to show that you passed (D or above) the prerequisite of cs370 (i.e., CS 237 or equivalent).
- Periodically, the grader will multicast important messages to this group email list.
Evaluation

- Midterm exam one: 75-minute in class ...15%
- Midterm exam two: 75-minute in class ...15%
- Homework assignment1 ~ 3 ...15% (5% each)
- Lab1 (5%, individual); Lab2 (10%); Lab3 (15%).
- Final exam: 2 hours ...25%
- Weekly exercises* ...0%
- Please check our class web page regularly.
Class Guidelines

Please refer to https://taoxie.sdsu.edu/cs370/index.html
# Grading Policy

<table>
<thead>
<tr>
<th>Grade</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>100</td>
</tr>
<tr>
<td>A-</td>
<td>90</td>
</tr>
<tr>
<td>B+</td>
<td>86</td>
</tr>
<tr>
<td>B</td>
<td>82</td>
</tr>
<tr>
<td>B-</td>
<td>78</td>
</tr>
<tr>
<td>C+</td>
<td>74</td>
</tr>
<tr>
<td>C</td>
<td>70</td>
</tr>
<tr>
<td>C-</td>
<td>66</td>
</tr>
<tr>
<td>D+</td>
<td>62</td>
</tr>
<tr>
<td>D</td>
<td>58</td>
</tr>
<tr>
<td>D-</td>
<td>54</td>
</tr>
<tr>
<td>F</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
Tips for Learning CS370

- Before each class, spend 1 hour to preview lecture slides.
- After each class, spend another 1 hour to read the slides & textbook.
- Each week, spend 1 hour to do Weekly Exercises.
- On average, spend 4 to 5 hours per week on CS 370.
- Pay close attention on example questions on the slides and make sure you fully understand how to solve them.
Why We Need to Learn Hardware?

- Always Important
  Every hardware system is different
  Understanding hardware → more efficient algorithms, programs
  (Microsoft developers say so!)
  Understanding hardware → more effective use of OS
  Understanding hardware → more efficient database design

- Timely
  Multicore, hyper-threading, security, . . .

- Opens doors
  Yet another option!
  You won’t know what you need until you need it
  (and this will probably be after you graduate)
Software and Hardware

Application
Algorithm & Data Structure
Language
Machine Architecture, ISA (CS572)
Microarchitecture
Logic and IC (CS370)
Device

Software
Hardware

Software
Hardware

CPU

Device
Computer System Design

Diagram showing the relationships between Technology, Applications, Computer Architect, Interfaces, Machine Organization, and Measurement & Evaluation.
System, Component, Logic Circuit, IC, Transistor

- Single die
- Wafer

Going up to 12” (30cm)

- CPU chip, SOC chip
- Functional Block (ALU, FPU, ...)
- Logic Gate (NOT/NAND/NOR, ...)
- CMOS
- Transistor (MOSFET: PMOS/NMOS)
The Implementation of Computer System

- Personal Computer: Hardware & Software
- Circuit Board: ≈8 / system 1-16G devices
- Integrated Circuit: ≈8-16 / PCB 0.25M-1G devices
- Module: ≈8-16 / IC 100K devices

MOSFET

Scheme for representing information

Gate: ≈2-16 / Cell 8 devices

Cell: ≈1K-10K / Module 16-64 devices
Logic and Computer Design Fundamentals

Chapter 1 – Digital Computers and Information

Charles Kime & Thomas Kaminski

© 2004 Pearson Education, Inc.

Terms of Use
(Hyperlinks are active in View Show mode)
Overview

- Digital Systems and Computer Systems
- Information Representation
- Number Systems [binary, octal and hexadecimal]
- Arithmetic Operations
- Base Conversion
- Decimal Codes [BCD (binary coded decimal), parity]
Digital System

- Takes a set of discrete information inputs and discrete internal information (system state) and generates a set of discrete information outputs.
Types of Digital Systems

- No state present
  - Combinational Logic System
  - Output = Function (Input)

- State present
  - State updated at discrete times
    => Synchronous Sequential System
  - State updated at any time
    => Asynchronous Sequential System
  - State = Function (State, Input)
  - Output = Function (State) or Function (State, Input)
Digital System Example:

A Digital Counter (e. g., odometer):

Count Up  →  0 0 1 3 5 6 4
Reset  →  0 0 1 3 5 6 4

Inputs:  Count Up, Reset
Outputs:  Visual Display
State:  "Value" of stored digits
A Digital Computer Example

- **Inputs:** Keyboard, mouse, modem, microphone
- **Outputs:** CRT, LCD, modem, speakers
- **Synchronous or Asynchronous**

Diagram:
- Memory
- CPU
  - Control unit
  - Datapath
- Input/Output

Signal

- An information variable represented by physical quantity (e.g., voltages and currents).
- For digital systems, the variable takes on discrete values.
- Two level, or binary values are the most prevalent values in digital systems.
- Binary values are represented abstractly by:
  - digits 0 and 1
  - words (symbols) False (F) and True (T)
  - words (symbols) Low (L) and High (H)
  - and words On and Off.
- Binary values are represented by values or ranges of values of physical quantities (see Figure 1-1)
Signal Examples Over Time

**Analog**
- Continuous in value & time

**Digital**
- Discrete in value & continuous in time
- Discrete in value & time

**Asynchronous**
- Digital type

**Synchronous**
- Digital type

_Difference between Asy and Syn?_
Signal Example – Physical Quantity: Voltage

Volts

Threshold Region

HIGH

LOW

HIGH

LOW

OUTPUT

INPUT

Signal Example – Physical Quantity: Voltage

Volts

Threshold Region

HIGH

LOW

OUTPUT

INPUT
Binary Values: Other Physical Quantities

What are other physical quantities represent 0 and 1?

- CPU Voltage
- Disk Magnetic Field Direction
- CD Surface Pits/Light
- Dynamic RAM Electrical Charge
Number Systems – Representation

- Positive radix, positional number systems
- A number with *radix* \( r \) is represented by a string of digits:

\[
A_{n-1}A_{n-2} \ldots A_1A_0 \cdot A_{-1}A_{-2} \ldots A_{-m+1}A_{-m}
\]

in which \( 0 \leq A_i < r \) and \( . \) is the *radix point*.

- The string of digits represents the power series:

\[
(Number)_r = \left( \sum_{i=0}^{i=n-1} A_i \cdot r^i \right) + \left( \sum_{j=-m}^{j=-1} A_j \cdot r^j \right)
\]

(Integer Portion) + (Fraction Portion)
## Number Systems – Examples

<table>
<thead>
<tr>
<th>Radix (Base)</th>
<th>General</th>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>Digits</td>
<td>0 =&gt; r - 1</td>
<td>0 =&gt; 9</td>
<td>0 =&gt; 1</td>
</tr>
<tr>
<td>0</td>
<td>r^0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>r^1</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>r^2</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>r^3</td>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>r^4</td>
<td>10,000</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>r^5</td>
<td>100,000</td>
<td>32</td>
</tr>
<tr>
<td>-1</td>
<td>r^-1</td>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>-2</td>
<td>r^-2</td>
<td>0.01</td>
<td>0.25</td>
</tr>
<tr>
<td>-3</td>
<td>r^-3</td>
<td>0.001</td>
<td>0.125</td>
</tr>
<tr>
<td>-4</td>
<td>r^-4</td>
<td>0.0001</td>
<td>0.0625</td>
</tr>
<tr>
<td>-5</td>
<td>r^-5</td>
<td>0.00001</td>
<td>0.03125</td>
</tr>
</tbody>
</table>
Special Powers of 2

- $2^{10}$ (1024) is Kilo, denoted "K"
- $2^{20}$ (1,048,576) is Mega, denoted "M"
- $2^{30}$ (1,073,741,824) is Giga, denoted "G"
Positive Powers of 2

### Useful for Base Conversion

<table>
<thead>
<tr>
<th>Exponent</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
</tr>
<tr>
<td>10</td>
<td>1024</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exponent</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>2,048</td>
</tr>
<tr>
<td>12</td>
<td>4,096</td>
</tr>
<tr>
<td>13</td>
<td>8,192</td>
</tr>
<tr>
<td>14</td>
<td>16,384</td>
</tr>
<tr>
<td>15</td>
<td>32,768</td>
</tr>
<tr>
<td>16</td>
<td>65,536</td>
</tr>
<tr>
<td>17</td>
<td>131,072</td>
</tr>
<tr>
<td>18</td>
<td>262,144</td>
</tr>
<tr>
<td>19</td>
<td>524,288</td>
</tr>
<tr>
<td>20</td>
<td>1,048,576</td>
</tr>
<tr>
<td>21</td>
<td>2,097,152</td>
</tr>
</tbody>
</table>
Converting Binary to Decimal

- To convert to decimal, use decimal arithmetic to form $\Sigma$ (digit $\times$ respective power of 2).
- Example: Convert $11010_2$ to $N_{10}$:

  Powers of 2: $43210$

  $11010_2 \Rightarrow$
  
  $1 \times 2^4 = 16$
  
  $+ 1 \times 2^3 = 8$
  
  $+ 0 \times 2^2 = 0$
  
  $+ 1 \times 2^1 = 2$
  
  $+ 0 \times 2^0 = 0$

  $26_{10}$
Converting Decimal to Binary

- **Method 1**
  - Subtract the largest power of 2 (see slide 19) that gives a positive remainder and record the power.
  - Repeat, subtracting from the prior remainder and recording the power, until the remainder is zero.
  - Place 1’s in the positions in the binary result corresponding to the powers recorded; in all other positions place 0’s.

- **Example: Convert** $625_{10}$ to $N_2$

  $625 - 512 = 113 \Rightarrow 9$; $113 - 64 = 49 \Rightarrow 6$; $49 - 32 = 17 \Rightarrow 5$

  $17 - 16 = 1 \Rightarrow 4$; $1 - 1 = 0 \Rightarrow 0$. Placing 1’s in the result for the positions recorded and 0’s elsewhere,

9 8 7 6 5 4 3 2 1 0

1 0 0 1 1 1 0 0 0 1
### Commonly Occurring Bases

<table>
<thead>
<tr>
<th>Name</th>
<th>Radix</th>
<th>Digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>2</td>
<td>0,1</td>
</tr>
<tr>
<td>Octal</td>
<td>8</td>
<td>0,1,2,3,4,5,6,7</td>
</tr>
<tr>
<td>Decimal</td>
<td>10</td>
<td>0,1,2,3,4,5,6,7,8,9</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>16</td>
<td>0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F</td>
</tr>
</tbody>
</table>

- The six letters (in addition to the 10 integers) in hexadecimal represent: ?
# Numbers in Different Bases

- Good idea to memorize!

<table>
<thead>
<tr>
<th>Decimal (Base 10)</th>
<th>Binary (Base 2)</th>
<th>Octal (Base 8)</th>
<th>Hexadecimal (Base 16)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00000</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>00001</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>02</td>
<td>00010</td>
<td>02</td>
<td>02</td>
</tr>
<tr>
<td>03</td>
<td>00011</td>
<td>03</td>
<td>03</td>
</tr>
<tr>
<td>04</td>
<td>00100</td>
<td>04</td>
<td>04</td>
</tr>
<tr>
<td>05</td>
<td>00101</td>
<td>05</td>
<td>05</td>
</tr>
<tr>
<td>06</td>
<td>00110</td>
<td>06</td>
<td>06</td>
</tr>
<tr>
<td>07</td>
<td>00111</td>
<td>07</td>
<td>07</td>
</tr>
<tr>
<td>08</td>
<td>01000</td>
<td>10</td>
<td>08</td>
</tr>
<tr>
<td>09</td>
<td>01001</td>
<td>11</td>
<td>09</td>
</tr>
<tr>
<td>10</td>
<td>01010</td>
<td>12</td>
<td>0A</td>
</tr>
<tr>
<td>11</td>
<td>01011</td>
<td>13</td>
<td>0B</td>
</tr>
<tr>
<td>12</td>
<td>01100</td>
<td>14</td>
<td>0C</td>
</tr>
<tr>
<td>13</td>
<td>01101</td>
<td>15</td>
<td>0D</td>
</tr>
<tr>
<td>14</td>
<td>01110</td>
<td>16</td>
<td>0E</td>
</tr>
<tr>
<td>15</td>
<td>01111</td>
<td>17</td>
<td>0F</td>
</tr>
<tr>
<td>16</td>
<td>10000</td>
<td>20</td>
<td>10</td>
</tr>
</tbody>
</table>
Conversion Between Bases

- **Method 2**
  - To convert from one base to another:
    1) Convert the Integer Part
    2) Convert the Fraction Part
    3) Join the two results with a radix point
Conversion Details

- **To Convert the Integral Part:**
  Repeatedly divide the number by the new radix and save the remainders. The digits for the new radix are the remainders in *reverse order* of their computation. If the new radix is $> 10$, then convert all remainders $> 10$ to digits A, B, ...

- **To Convert the Fractional Part:**
  Repeatedly multiply the fraction by the new radix and save the integer digits that result. The digits for the new radix are the integer digits in *order* of their computation. If the new radix is $> 10$, then convert all integers $> 10$ to digits A, B, ...

Example: Convert $46.6875_{10}$ To Base 2

- Convert 46 to Base 2
- Convert 0.6875 to Base 2:
- Join the results together with the radix point:
Answer 1: Converting 46 as integral part

\[
\begin{align*}
46/2 &= 23 \text{ rem } = 0 \\
23/2 &= 11 \text{ rem } = 1 \\
11/2 &= 5 \text{ remainder } = 1 \\
5/2 &= 2 \text{ remainder } = 1 \\
5/2 &= 2 \text{ remainder } = 1 \\
2/2 &= 1 \text{ remainder } = 0 \\
1/2 &= 0 \text{ remainder } = 1
\end{align*}
\]

Reading off in the reverse direction: 101110
Answer 2: Converting 0.6875 as fractional part

\[
0.6875 \times 2 = 1.3750 \quad \text{int} = 1 \\
0.3750 \times 2 = 0.7500 \quad \text{int} = 0 \\
0.7500 \times 2 = 1.5000 \quad \text{int} = 1 \\
0.5000 \times 2 = 1.0000 \quad \text{int} = 1 \\
0.0000
\]

Reading off in the forward direction: 0.1011

Combining Integral and Fractional Parts:
101110.1011
Additional Issue - Fractional Part

- Note that in this conversion, the fractional part became 0 as a result of the repeated multiplications.
- In general, it may take many bits to get this to happen or it may never happen.
- Example: Convert $0.65_{10}$ to $N_2$
  - $0.65 = 0.1010011001001 \ldots$
  - The fractional part begins repeating every 4 steps yielding repeating 1001 forever!
- Solution: Specify number of bits to right of radix point and round or truncate to this number.
Checking the Conversion

- To convert back, sum the digits times their respective powers of $r$.

- From the prior conversion of $46.6875_{10}$

  $101110_2 = 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 0 \cdot 1$

  $= 32 + 8 + 4 + 2$

  $= 46$

  $0.1011_2 = 1/2 + 1/8 + 1/16$

  $= 0.5000 + 0.1250 + 0.0625$

  $= 0.6875$
Binary Numbers and Binary Coding

- **Flexibility of representation**
  - Within constraints below, can assign any binary combination (called a code word) to any data as long as data is uniquely encoded.

- **Information Types**
  - **Numeric**
    - Must represent range of data needed
    - Very desirable to represent data such that simple, straightforward computation for common arithmetic operations permitted
    - Tight relation to binary numbers
  - **Non-numeric**
    - Greater flexibility since arithmetic operations not applied.
    - Not tied to binary numbers
Non-numeric Binary Codes

- Given $n$ binary digits (called bits), a binary code is a mapping from a set of represented elements to a subset of the $2^n$ binary numbers.

- Example: A binary code for the seven colors of the rainbow

- Code 100 is not used

<table>
<thead>
<tr>
<th>Color</th>
<th>Binary Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>000</td>
</tr>
<tr>
<td>Orange</td>
<td>001</td>
</tr>
<tr>
<td>Yellow</td>
<td>010</td>
</tr>
<tr>
<td>Green</td>
<td>011</td>
</tr>
<tr>
<td>Blue</td>
<td>101</td>
</tr>
<tr>
<td>Indigo</td>
<td>110</td>
</tr>
<tr>
<td>Violet</td>
<td>111</td>
</tr>
</tbody>
</table>
Given $M$ elements to be represented by a binary code, the minimum number of bits, $n$, needed, satisfies the following relationships:

$$2^n > M > 2^{(n-1)}$$

$$n = \lceil \log_2 M \rceil$$

where $\lceil x \rceil$, called the ceiling function, is the integer greater than or equal to $x$.

Example: How many bits are required to represent decimal digits with a binary code?
Number of Elements Represented

- Given $n$ digits in radix $r$, there are $r^n$ distinct elements that can be represented.
- But, you can represent $m$ elements, $m < r^n$
- Examples:
  - You can represent 4 elements in radix $r = 2$ with $n = 2$ digits: (00, 01, 10, 11).
  - You can represent 4 elements in radix $r = 2$ with $n = 4$ digits: (0001, 0010, 0100, 1000).
  - This second code is called a "one hot" code.
Binary Coded Decimal (BCD)

- The BCD code is the 8,4,2,1 code.
- This code is the simplest, most intuitive binary code for decimal digits and uses the same powers of 2 as a binary number, but only encodes the first ten values from 0 to 9.
- Example: 1001 (9) = 1000 (8) + 0001 (1)
- How many “invalid” code words are there?
- What are the “invalid” code words?
Warning: Conversion or Coding?

- Do **NOT** mix up **conversion** of a decimal number to a binary number with **coding** a decimal number with a BINARY CODE.

- $13_{10} = 1101_2$ (This is **conversion**)

- $13 \iff 0001|0011$ (This is **coding**)
Binary Arithmetic

- Single Bit Addition with Carry
- Multiple Bit Addition
- Single Bit Subtraction with Borrow
- Multiple Bit Subtraction
- Multiplication
Single Bit Binary Addition with Carry

Given two binary digits (X,Y), a carry in (Z) we get the following sum (S) and carry (C):

Carry in (Z) of 0:

<table>
<thead>
<tr>
<th>Z</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>+ Y</td>
<td>+ 0</td>
<td>+ 1</td>
<td>+ 0</td>
<td>+ 1</td>
<td></td>
</tr>
<tr>
<td>C S</td>
<td>0 0</td>
<td>0 1</td>
<td>0 1</td>
<td>1 0</td>
<td></td>
</tr>
</tbody>
</table>

Carry in (Z) of 1:

<table>
<thead>
<tr>
<th>Z</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>+ Y</td>
<td>+ 0</td>
<td>+ 1</td>
<td>+ 0</td>
<td>+ 1</td>
<td></td>
</tr>
<tr>
<td>C S</td>
<td>0 1</td>
<td>1 0</td>
<td>1 0</td>
<td>1 1</td>
<td></td>
</tr>
</tbody>
</table>
Multiple Bit Binary Addition

- Extending this to two multiple bit examples:

<table>
<thead>
<tr>
<th>Carries</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Augend</td>
<td>01100</td>
<td>10110</td>
</tr>
<tr>
<td>Addend</td>
<td>+10001</td>
<td>+10111</td>
</tr>
<tr>
<td>Sum</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Note: The 0 is the default Carry-In to the least significant bit.
Given two binary digits (X,Y), a borrow in (Z) we get the following difference (S) and borrow (B):

- **Borrow in (Z) of 0:**
  - Z: 0 0 0 0 0 0
  - X: 0 0 0 1 1 1
  - Y: -0 -1 -0 -1
  - BS: 0 0 1 1 0 1 0 0

- **Borrow in (Z) of 1:**
  - Z: 1 1 1 1 1 1
  - X: 0 0 0 1 1 1
  - Y: -0 -1 -0 -1
  - BS: 1 1 1 0 0 0 1 1
Multiple Bit Binary Subtraction

- Extending this to two multiple bit examples:

Borrows  \( \overline{0} \)  \( \overline{0} \)

Minuend  10110  10110

Subtrahend  -10010  -10011

Difference

- Notes: The \( \overline{0} \) is a Borrow-In to the least significant bit. If the Subtrahend > the Minuend, interchange and append a \( - \) to the result.
### Answer to Last Slide

<table>
<thead>
<tr>
<th>Borrows</th>
<th>00000</th>
<th>00110</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minuend</td>
<td>10110</td>
<td>10110</td>
</tr>
<tr>
<td>Subtrahend</td>
<td>10010</td>
<td>10011</td>
</tr>
<tr>
<td>Difference</td>
<td>00100</td>
<td>00011</td>
</tr>
</tbody>
</table>
Binary Multiplication

The binary multiplication table is simple:

\[
0 * 0 = 0 \quad | \quad 1 * 0 = 0 \quad | \quad 0 * 1 = 0 \quad | \quad 1 * 1 = 1
\]

Extending multiplication to multiple digits:

<table>
<thead>
<tr>
<th>Multiplicand</th>
<th>1011</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplier</td>
<td>x 101</td>
</tr>
<tr>
<td>Partial Products</td>
<td>1011</td>
</tr>
<tr>
<td></td>
<td>0000 -</td>
</tr>
<tr>
<td></td>
<td>1011 - -</td>
</tr>
<tr>
<td>Product</td>
<td>110111</td>
</tr>
</tbody>
</table>
Weekly Exercise 1-1

- Problems
- 1-1; 1-4; 1-7; 1-8; 1-10, 1-16 (a)
- Memorize the table on slide 31
Terms of Use

- © 2004 by Pearson Education, Inc. All rights reserved.
- The following terms of use apply in addition to the standard Pearson Education Legal Notice.
- Permission is given to incorporate these materials into classroom presentations and handouts only to instructors adopting Logic and Computer Design Fundamentals as the course text.
- Permission is granted to the instructors adopting the book to post these materials on a protected website or protected ftp site in original or modified form. All other website or ftp postings, including those offering the materials for a fee, are prohibited.
- You may not remove or in any way alter this Terms of Use notice or any trademark, copyright, or other proprietary notice, including the copyright watermark on each slide.

Return to Title Page