CS370 Computer Architecture

- Instructor: Tao Xie
- Email address: txie@mail.sdsu.edu
- Office: GMCS 535
- Office hours: MW 11 am – 12 pm (noon) or by appointment
- Course website:

http://taoxie.sdsu.edu/cs370/index.html

- Prerequisite: CS237 or equivalent (see Slide 4)

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 exactly 2 students)
- You can work on lab assignments at home on your own computer
- Your grader will provide 3 sessions per week at GMCS 422 or 425 to help you on the 3 lab assignments. Please refer to his office hours on our class web page: http://taoxie.sdsu.edu/cs370/index.html

Class Schedule in Online Now!

- **January 17: Lecture 1 (Class Guidelines & Chapter1 “Digital Systems and Information”)**
- **January 22: Lecture 2 (Chapter 2 “Combinational Logic Circuits”, part 1 – Gate Circuits and Boolean Equations)**
- **January 24: Lecture 3 (Chapter 2 “Combinational Logic Circuits”, part 1 – Standard Forms)**
- **January 29: Lecture 4 (Chapter 2 “Combinational Logic Circuits”, part 2 – Circuit Optimization)**
- **January 31: Lecture 5 (Chapter 2 “Combinational Logic Circuits”, part 2 – K-Map Manipulation)**
- **February 5: Lecture 6 (Chapter 2 “Combinational Logic Circuits”, part 3 – Additional Gates and Circuits)**
- **February 7: Lecture 7 (Chapter 3 “Combinational Logic Design”, part 1 – Implementation Technology and Logic Design)**
- **February 12: Lecture 8 (Chapter 3 “Combinational Logic Design”, part 2 – Functions and Functional Blocks)**
- **February 14: Lecture 9 (Preview for Midterm Exam One)**
- **February 19: Midterm Exam One (75 minutes in class)**

Grader and His Office Hours

Tony La

Email: tonyla858@gmail.com

Office: Monday & Wednesday GMCS 425; Thursday GMCS 422

Office hours (Temporary): Monday & Wednesday 12 PM ~ 1:45 PM; Thursday 5:30 PM ~ 7:00 PM.

A Hello Email to your Grader by Jan. 24

- Send a Hello email to your grader
- Email subject is “Hello, cs370!”
- In the email, please tell him your Red ID, full name, and your working email address if it is not the one in the Hello email, 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

- Test1 75-minute close book & in class ...20%
- Test2 75-minute close book & in class ...20%
- Homework assignment1 ~ 3 ...15% (5% each)
- Lab1 (5%, individual); Lab2 (7%); Lab3 (8%).
- Final exam ...25%
- Weekly exercises* ...0%
- Please check our class web page regularly.

Class Guidelines

- **Prerequisite**: The prerequisite for this course is CS237. This prerequisite will be strictly enforced.
- There will be **NO make-up** tests without a verified excuse.
- I will not sign **late drop** slips!
- The **final exam** is scheduled on **7 May 2018**, **13:00-15:00.** Failure to appear for the final exam will result in a grade of "F" in the course, unless you make prior arrangements with me for an Incomplete.
- **Incompletes**: To receive a grade of Incomplete ("I") in this class, you must meet all of the following criteria:….. (please refer to course web site)
- All homeworks are due at the start of class on the indicated due day. **No late homework will be accepted**.
- All the exams are **close-book, close-notes**. You can only bring one piece of paper (size A4, one side) for **each of the three exams** as "**cheating sheet**".
- Any questions about grading must be brought to the attention of the grader or the instructor within **one week** after the item in question is returned.
- **Cheating**: Any one caught cheating/collaborating on an exam or any individual assignment will receive zero for that assignment or exam. If a student is caught twice, he/she will receive an F in the course and the incident will be reported to the Office of Judicial Affairs for disciplinary proceedings. Note: If, for instance, you allow your assignment be copied by a classmate, you are considered as guilty as the copier.

Grading Policy

100 90 86 82 78 74 70 66 62 58 54 50 0

| A | A- | B+ | B | B- | C+ | C | C- | D+ | D | D- | F |

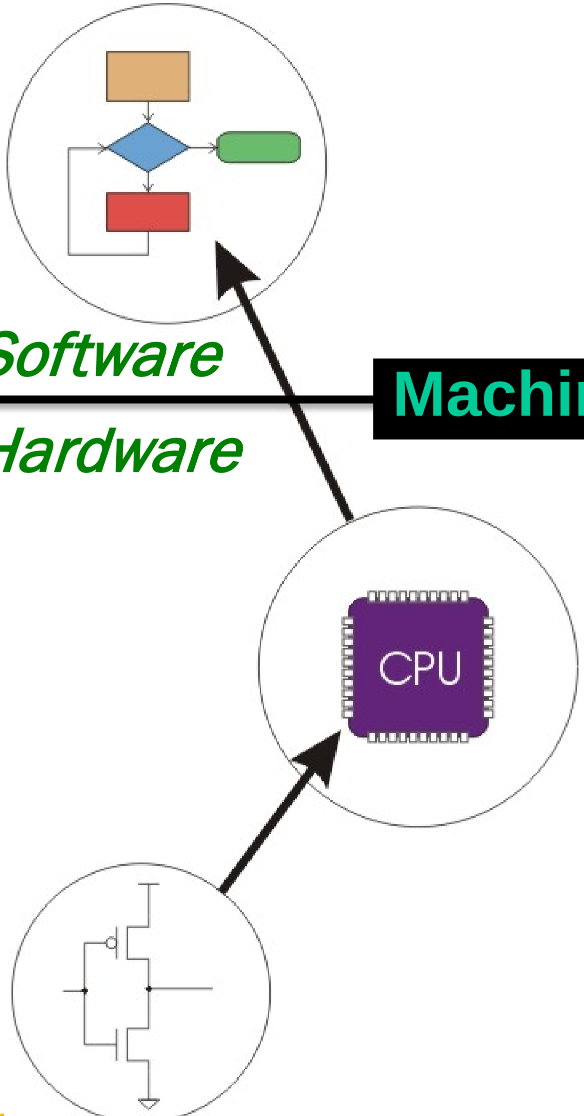
Tips for Learning CS370

- After each class, spend 1 hour to digest lecture slides.
- After each class, spend another 1 hour to read the 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

Software

Machine Architecture, ISA (CS572)

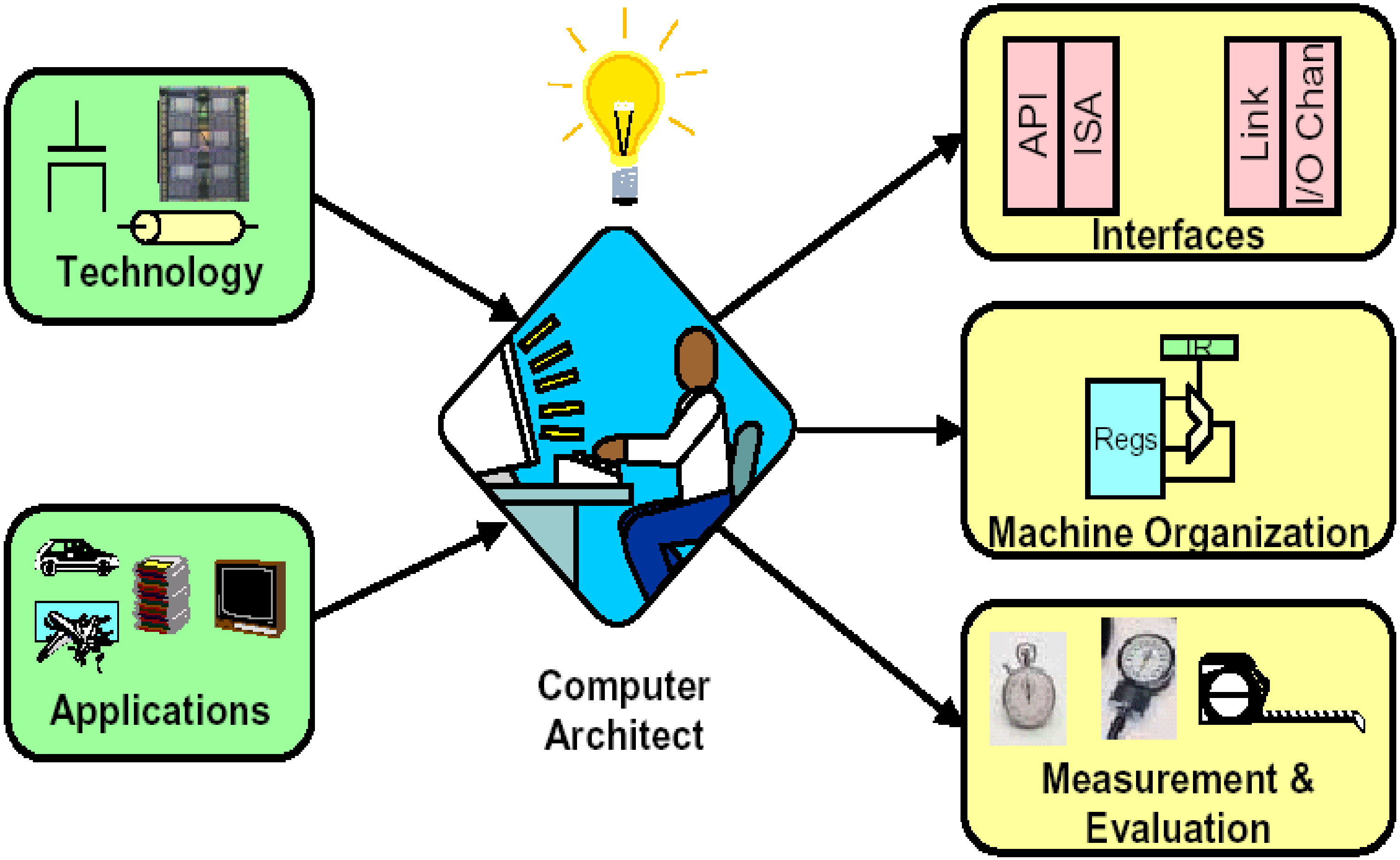
Hardware

Microarchiture

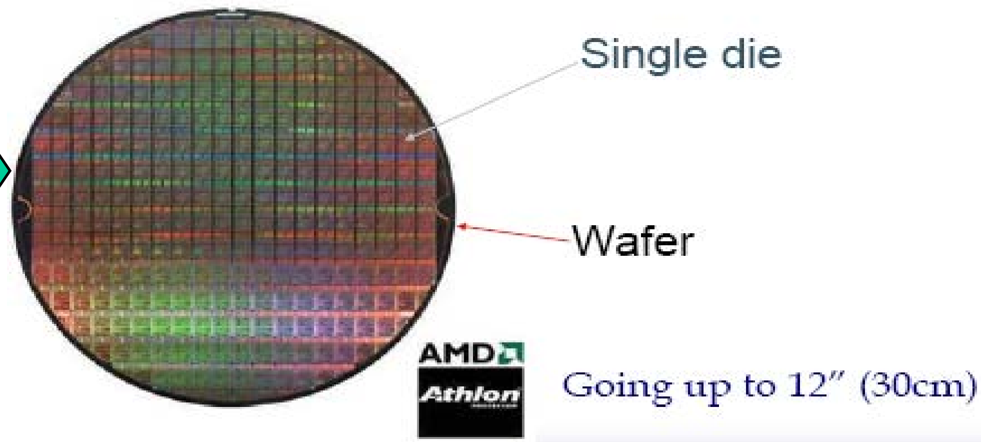
Logic and IC (CS370)

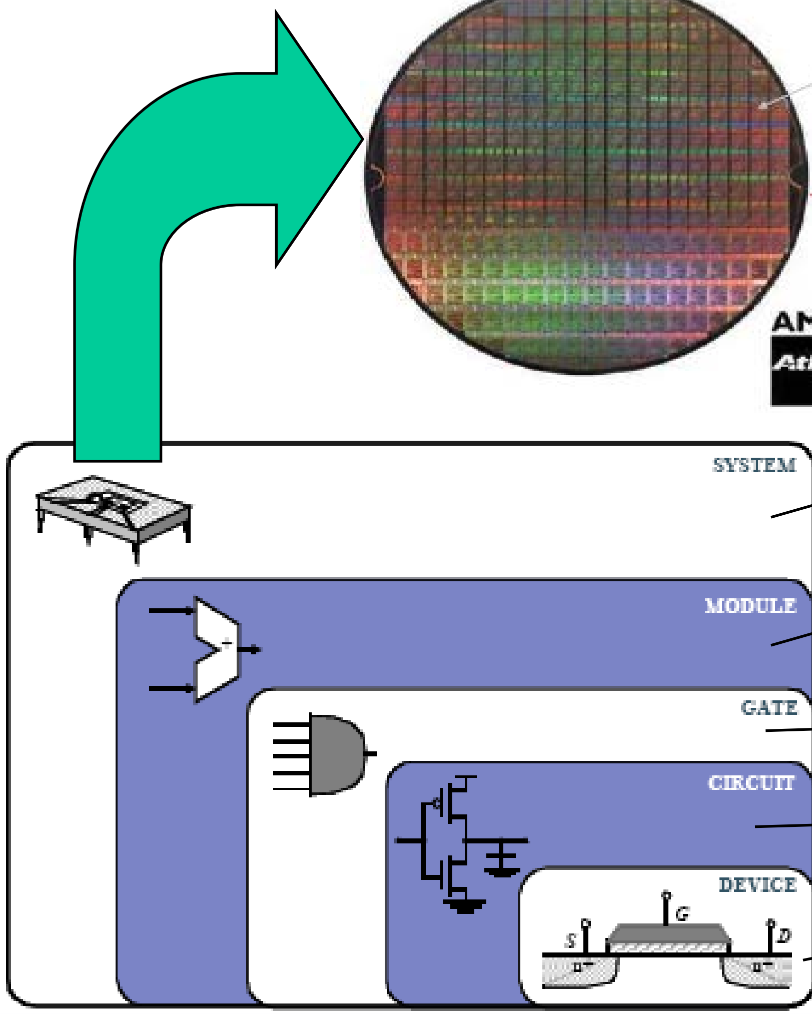
Device

Computer System Design



System, Component, Logic Circuit, IC, Transistor





CPU chip，SOC chip

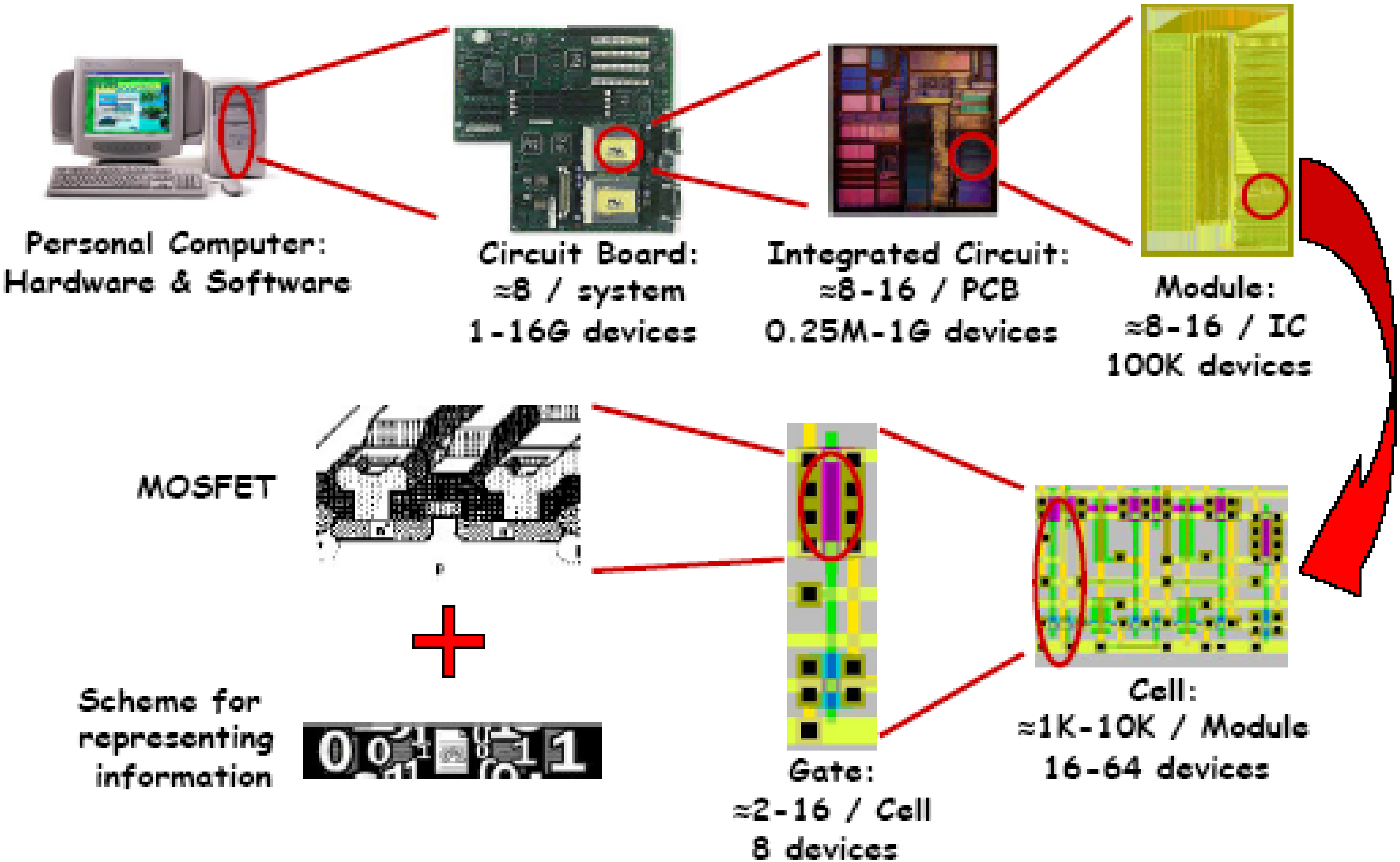
Functional Block（ALU，FPU，...）

Logic Gate（NOT/NAND/NOR。。。）

CMOS

Transistor（MOSFET：PMOS/NMOS）

The Implementation of Computer System



Logic and Computer Design Fundamentals

Chapter 1 – Digital Computers and Information

Charles Kime & Thomas Kaminski

Terms of Use

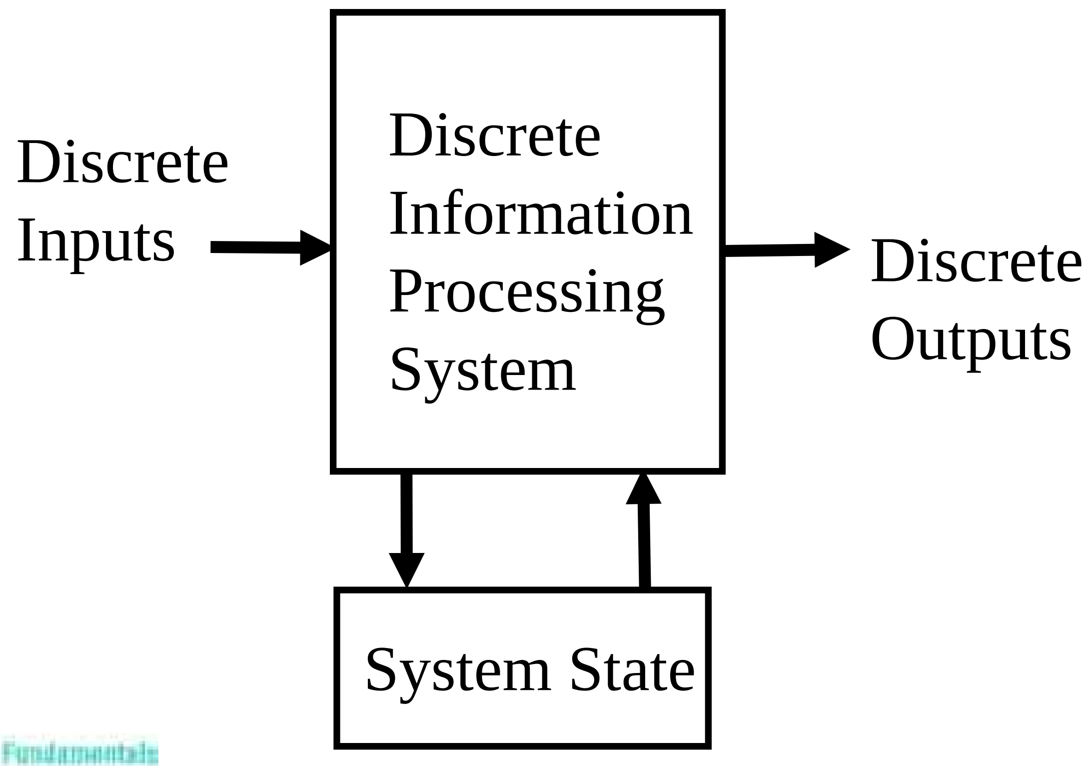
(Hyperlinks are active in View Show mode)

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

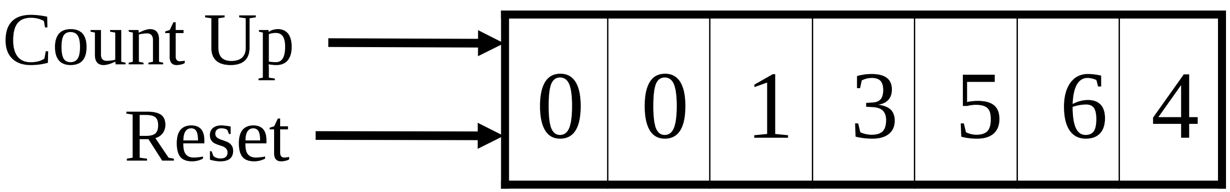
- No state present
  - Combinational Logic System
  - Output = Function (Input)
- State present
  - State updated at discrete times

    => Synchronous Sequential System
  - State updated at any time

    =>Asynchronous Sequential System
  - State = Function (State, Input)
  - Output = Function (State)
    or Function (State, Input)

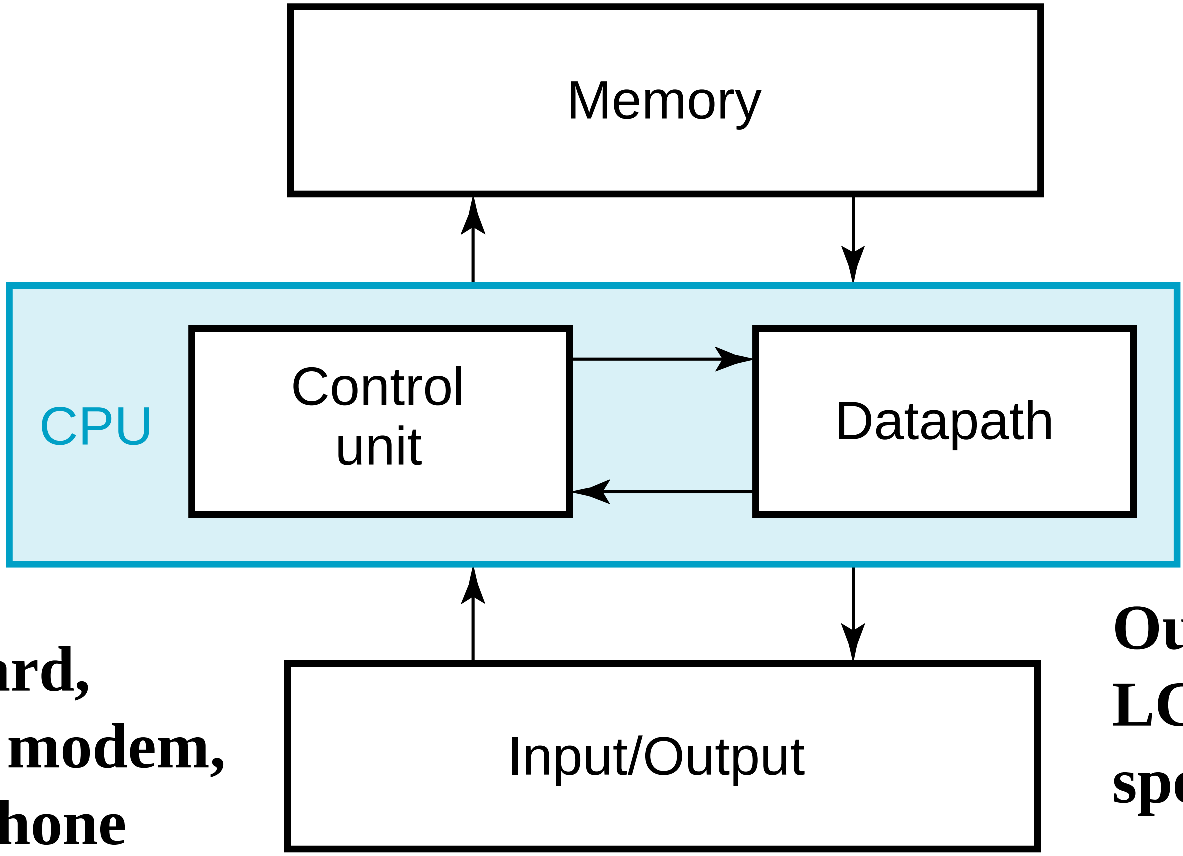
Digital System Example:

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



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

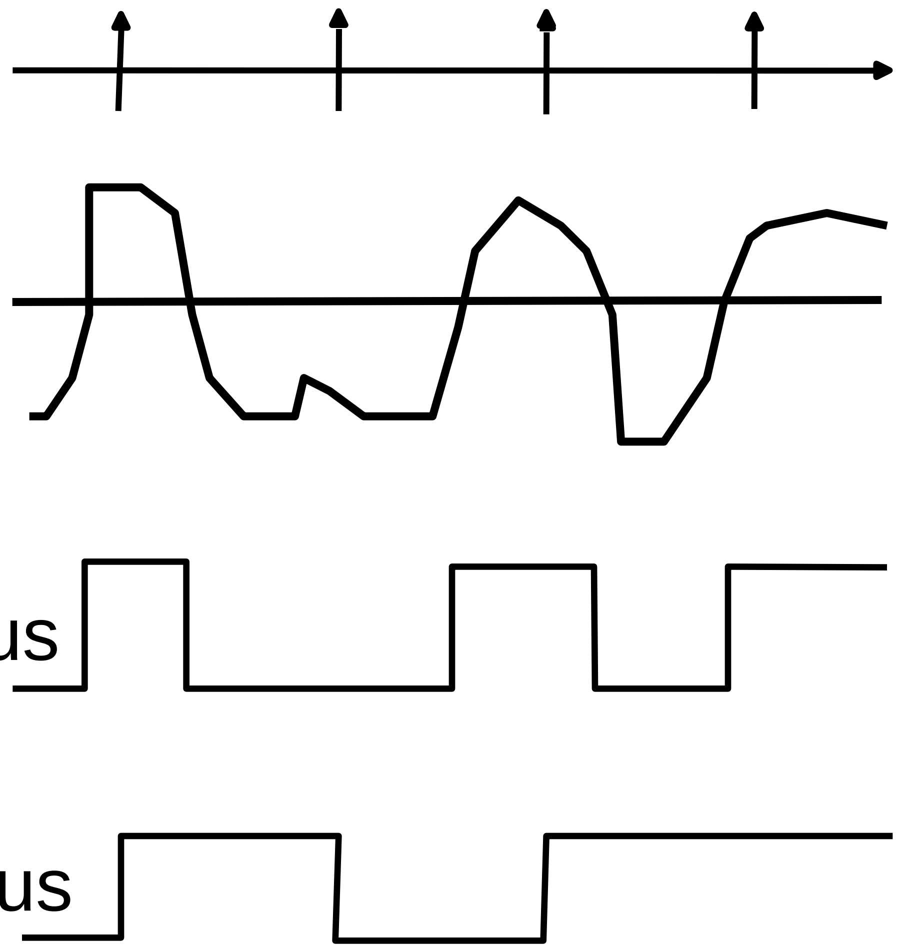
Synchronous or Asynchronous

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

Time



Analog

Continuous in value & time

Digital

Asynchronous

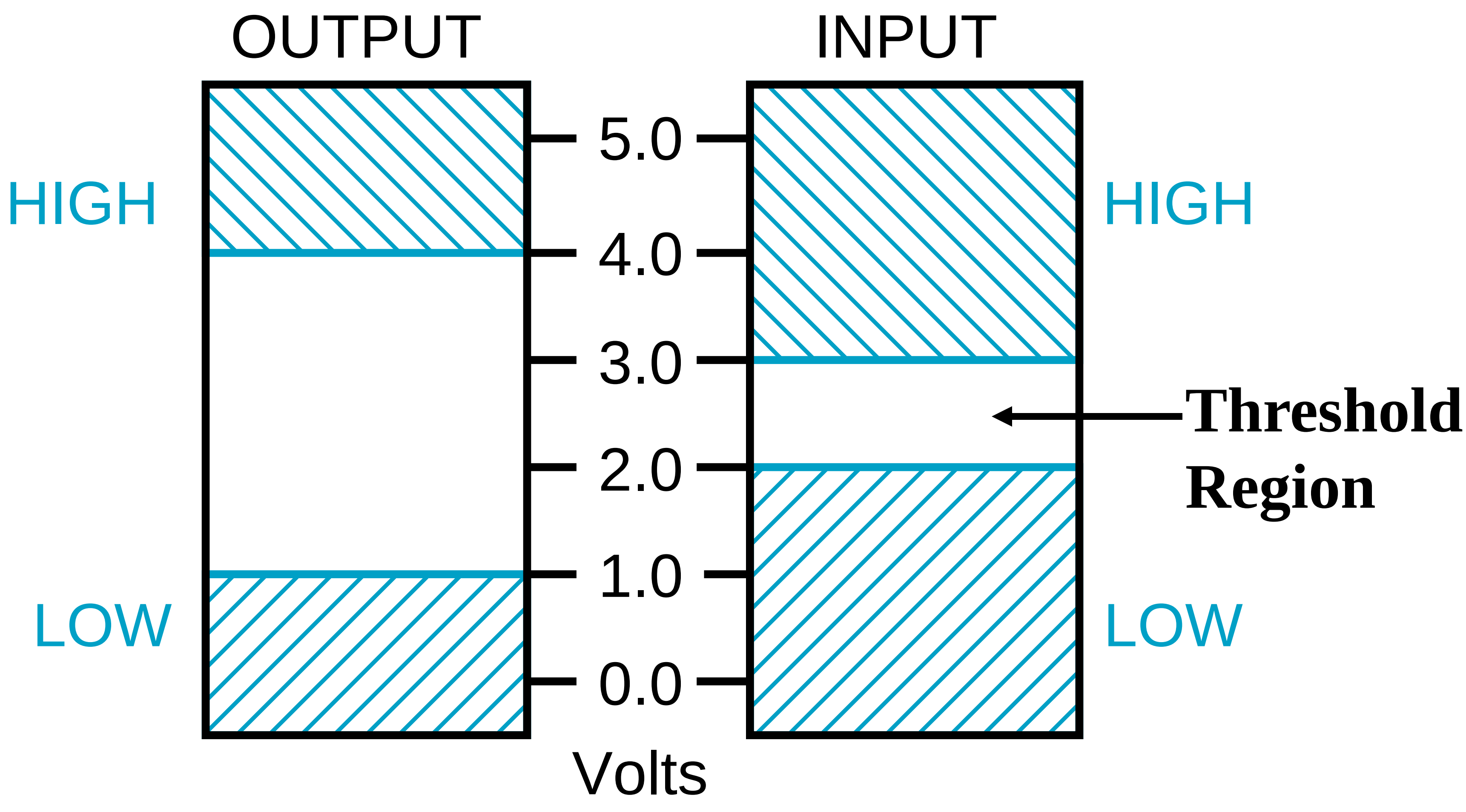
Discrete in value & continuous in time

Synchronous

Discrete in value & time

Difference between Asy and Syn?

Signal Example – Physical Quantity: Voltage



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:

$$\boldsymbol{A_{n-1}A_{n-2}\ \ldots\ A_1A_0\ .\ A_{-1}\ A_{-2}\ \ldots\ A_{-m+1}\ A_{-m}}$$

in which $\mathbf{0} \leq \boldsymbol{A_i} < \boldsymbol{r}$ and **.** is the *radix point*.

- The string of digits represents the power series:

$$\textbf{(Number)}_\mathbf{r} = \left(\sum_{i=0}^{i=n-1} A_i \cdot r^i\right) + \left(\sum_{j=-m}^{j=-1} A_j \cdot r^j\right)$$

(Integer Portion) + (Fraction Portion)

Number Systems – Examples

		General	Decimal	Binary
Radix (Base)		r	10	2
Digits		0 => r - 1	0 => 9	0 => 1
	0	r^0	1	1
	1	r^1	10	2
	2	r^2	100	4
	3	r^3	1000	8
Powers of	4	r^4	10,000	16
Radix	5	r^5	100,000	32
	-1	r^{-1}	0.1	0.5
	-2	r^{-2}	0.01	0.25
	-3	r^{-3}	0.001	0.125
	-4	r^{-4}	0.0001	0.0625
	-5	r^{-5}	0.00001	0.03125

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

Exponent	Value
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

Exponent	Value
11	2,048
12	4,096
13	8,192
14	16,384
15	32,768
16	65,536
17	131,072
18	262,144
19	524,288
20	1,048,576
21	2,097,152

Converting Binary to Decimal

- To convert to decimal, use decimal arithmetic to form Σ (digit × respective power of 2).
- Example: Convert 11010_2 to N_{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**

Commonly Occurring Bases

Name	Radix	Digits
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Decimal	10	0,1,2,3,4,5,6,7,8,9
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

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

Numbers in Different Bases

- **Good idea to memorize!**

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

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:**

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 …**
  - **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

Color	Binary Number
Red	000
Orange	001
Yellow	010
Green	011
Blue	101
Indigo	110
Violet	111

Number of Bits Required

- **Given M elements to be represented by a binary code, the minimum number of bits, *n*, needed, satisfies the following relationships:**

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

  $n = \lceil \log_2 M \rceil$ 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 <u>decimal digits</u> 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 "<u>one hot</u>" 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 $\Leftrightarrow$ 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:

Z	0	0	0	0
X	0	0	1	1
+ Y	+ 0	+ 1	+ 0	+ 1
C S	0 0	0 1	0 1	1 0

Carry in (Z) of 1:

Z	1	1	1	1
X	0	0	1	1
+ Y	+ 0	+ 1	+ 0	+ 1
C S	0 1	1 0	1 0	1 1

Multiple Bit Binary Addition

- **Extending this to two multiple bit examples:**

Carries	**0**	**0**
Augend	**01100**	**10110**
Addend	**+10001**	**+10111**
Sum		

- **Note: The 0 is the default Carry-In to the least significant bit.**

Single Bit Binary Subtraction with Borrow

- 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
X	0	0	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
X	0	0	1	1
- Y	-0	-1	-0	-1
BS	11	1 0	0 0	1 1

Multiple Bit Binary Subtraction

- Extending this to two multiple bit examples:

Borrows	<u>0</u>	<u>0</u>
Minuend	10110	10110
Subtrahend	<u>- 10010</u>	<u>- 10011</u>
Difference		

- Notes: The <u>0</u> is a Borrow-In to the least significant bit. If the Subtrahend > the Minuend, interchange and append a – to the result.

Binary Multiplication

The binary multiplication table is simple:

$$0 * 0 = 0 \mid 1 * 0 = 0 \mid 0 * 1 = 0 \mid 1 * 1 = 1$$

Extending multiplication to <u>multiple digits</u>:

Multiplicand	**1011**
Multiplier	**<u>x 101</u>**
Partial Products	**1011**
	0000 -
	<u>1011 - -</u>
Product	**110111**

Weekly Exercise 1-1

- Problems (P33 on text book)
- 1-1; 1-4; 1-7; 1-8; 1-9, 1-12 (a); 1-16 (a)
- Memorize the table on slide 31

Terms of Use

-
- 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