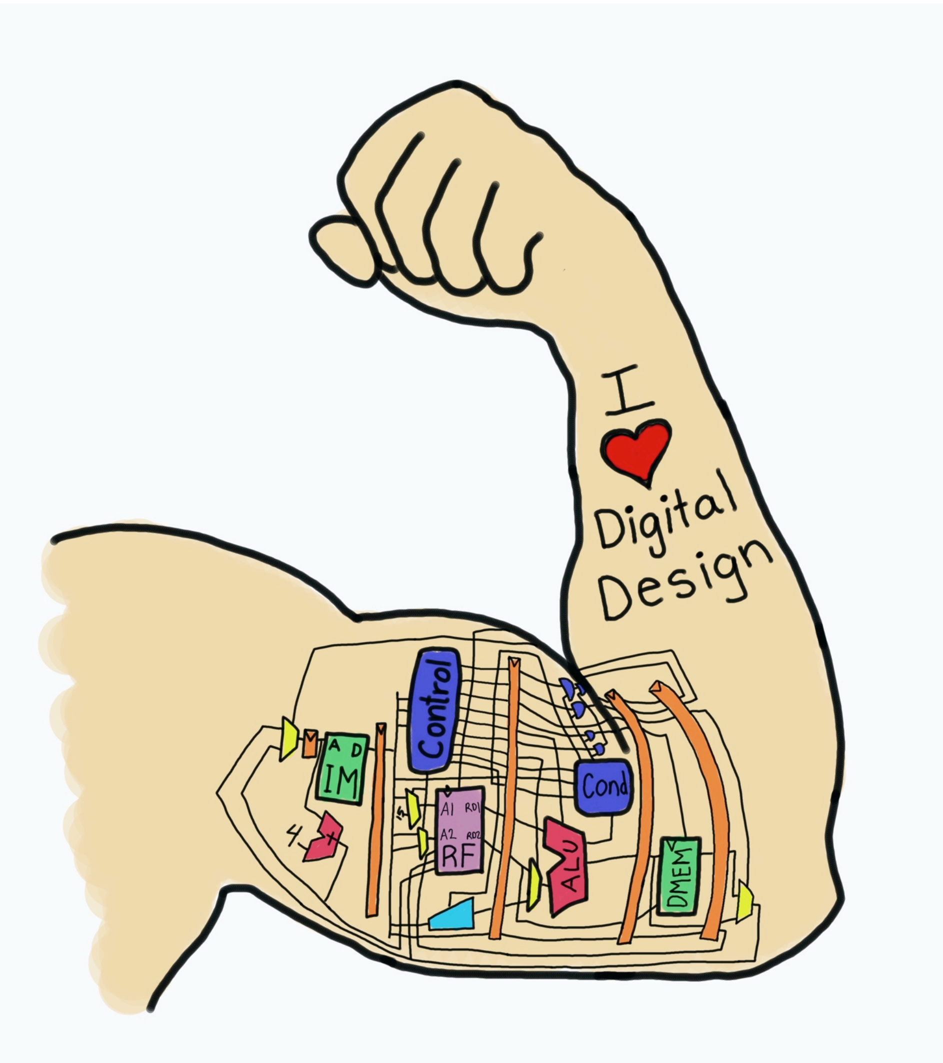
E85 Digital Design & Computer Engineering



Lecture 0:
Introduction

HARVEY MUDD COLLEGE

Lecture 0

- **Course Overview**
  - **Learning Objectives**
  - **Schedule**
  - **Assignments**
- **The Game Plan**
- **The Art of Managing Complexity**
- **The Digital Abstraction**
- **Number Systems**

Learning Objectives

- Build digital systems at all levels of abstraction from transistors through circuits, logic, microarchitecture, architecture, and C culminating with implementing and programming a microprocessor soft core on a field programable gate array.
- Manage complexity using the digital abstraction, data types, static and dynamic disciplines, and hierarchical design.
- Design and implement combinational and sequential digital circuits using schematics and hardware description languages.

Learning Objectives Cont.

- Program a commercial microcontroller in C and assembly language and use it in a physical system.
- Begin the practice of implementing and debugging digital systems with appropriate lab techniques including breadboarding, interpreting datasheets, and using field-programmable gate arrays and microcontroller boards, simulators, debuggers, and test-and-measurement equipment.

Big Picture

- Start from the fundamentals so you understand why, not just how.
  - What makes the system tick on the inside?
- Some of you will become computer engineers.
  - This material is the foundation of your career.
- Most of you will pursue other paths.
  - Digital systems are a tremendously valuable tool in your toolbox.
  - This course will get you to the point you can be dangerous!
  - Skills about managing complexity, designing nontrivial systems, debugging carry over to other fields.
  - Programmable microprocessors are one of humanities great ideas.
  - Computing has fundamentally changed the world we live in.
- If you like this course, take E155 next Fall.

ELSEVIER

Schedule

Lecture	Date	Topics	Readings	Assignment
0	1/22	Introduction: digital abstraction, numbers	1.1-1.5	
1	1/27	Logic gates, Static discipline, transistors	1.6-1.9, A1-A7	
10	1/29	Combinational logic design	2.1-2.8	PS 1 due
11	2/3	Timing, sequential circuits	2.9-2.10, 3.1-3.2	Lab 1 due Digital Circuits
100	2/5	Finite state machines	3.3-3.4	PS 2 due
101	2/10	Dynamic discipline, metastability	3.5-3.7	Lab 2 due Comb Logic
110	2/12	Hardware description languages: Verilog	4.1-4.3	PS 3 due
111	2/17	Verilog, Part II	4.4-4.10	Lab 3 due Structural FSM
1000	2/19	Arithmetic circuits	5.1-5.2	PS 4 due
1001	2/24	Fixed and floating-point number systems	5.3	Lab 4 due Behavioral FSM
1010	2/26	Sequential building blocks, arrays	5.4-5.7	PS 5 due
1011	3/2	Catchup / Midterm Review		Lab 5 due Building blocks
	3/4	Midterm		
1100	3/9	C Programming	C.1-C.7	
1101	3/11	C Programming	C.8-C.11	
	3/16	Spring Break!		
	3/18	Spring Break!		
1110	3/23	Microcontrollers: Memory-mapped I/O	9.1-9.3.3	Lab 6 due C Programming
1111	3/25	Parallel & serial interfacing, ADCs	9.3-9.4	PS 6 due
10000	3/30	I/O libraries and examples		Lab 7 due C I/O
10001	4/1	Assembly language	6.1-6.3.6	PS 7 due
10010	4/6	Function calls, machine language	6.3.7-6.9	Lab 8 due C Peripherals
10011	4/8	Single-cycle processor datapath	7.1-7.3.1	PS 8 due
10100	4/13	Single-cycle processor control, Verilog	7.3, 7.6	Lab 9 due Assembly
10101	4/15	Multicycle processor	7.4	
10110	4/20	Pipelining	7.5.1-2	PS 9 due
10111	4/22	Advanced architecture: a sampler	7.7	Lab 10 due Multicycle Control
11000	4/27	Case study: Processors	6.7, 8.7, 8.5	PS 10 due
11001	4/29	Class summary and review		Lab 11 due Multicycle CPU

Assignments

- Mondays: Labs (30%)
  - Must complete Lab 11 to pass the class
  - Digital Lab Tutoring Sat 12-2, Sun 12-6
- Wednesdays: Problem Sets (20%)
  - TBP TBD
- Midterm & Final (50%)

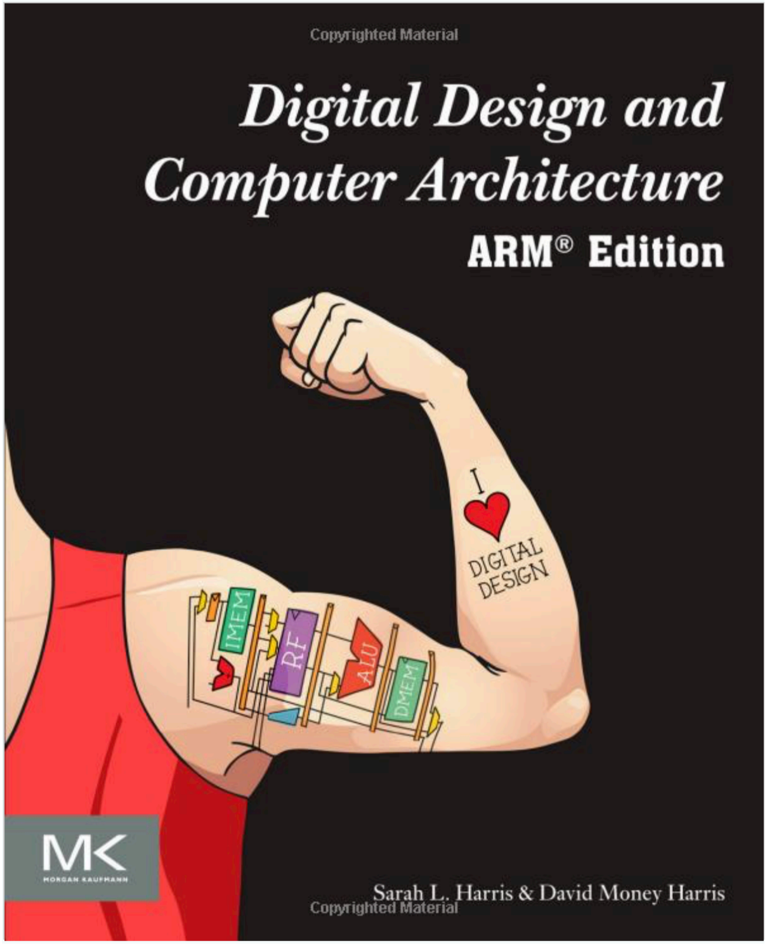
- You can have a 1-week extension on one assignment
  - Just turn it in with your assignment next week
- Your lowest lab and problem set score will be dropped

Collaboration Policy

- Speak with other students, instructor, tutors AFTER you have made an effort by yourself.
  - Ask about tool issues in the lab!
- Turn in your own work. Not identical to others.
  - Don't sit at adjacent computers and work in lockstep.
  - Pair programming prohibited.
- Credit classmates with whom you discussed ideas.
- Don't refer to old solutions!

Textbook

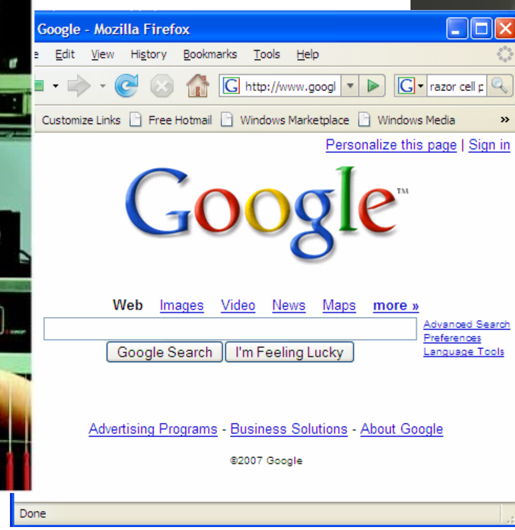
- Many students have found it enjoyable and useful
  - Suggest reading before class, come with questions
  - Reread key parts as you are doing the assignments
- Not everything in the assignments is covered in lecture.
- Copies available in the Eng. Lounge and Digital Lab.



ELSEVIER

Background

- Microprocessors have revolutionized our world
  - Cell phones, Internet, rapid advances in medicine, etc.
- The semiconductor industry has grown from $21 billion in 1985 to $412 billion in 2017

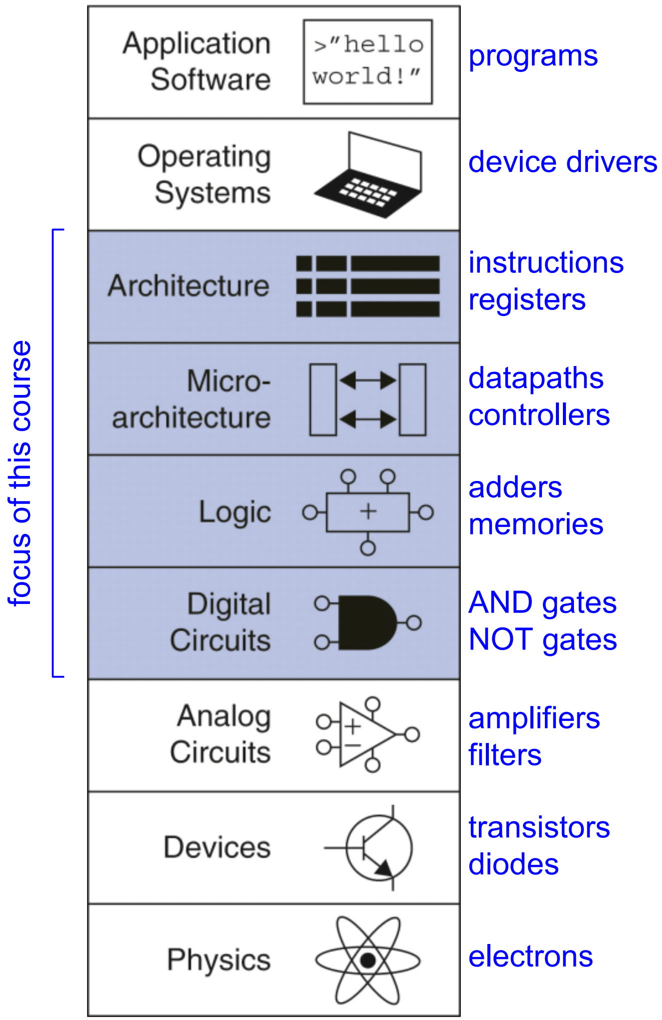


The Art of Managing Complexity

- Abstraction
- Discipline
- The Three –y's
  - Hierarchy
  - Modularity
  - Regularity

Abstraction

Hiding details when they aren't important



Discipline

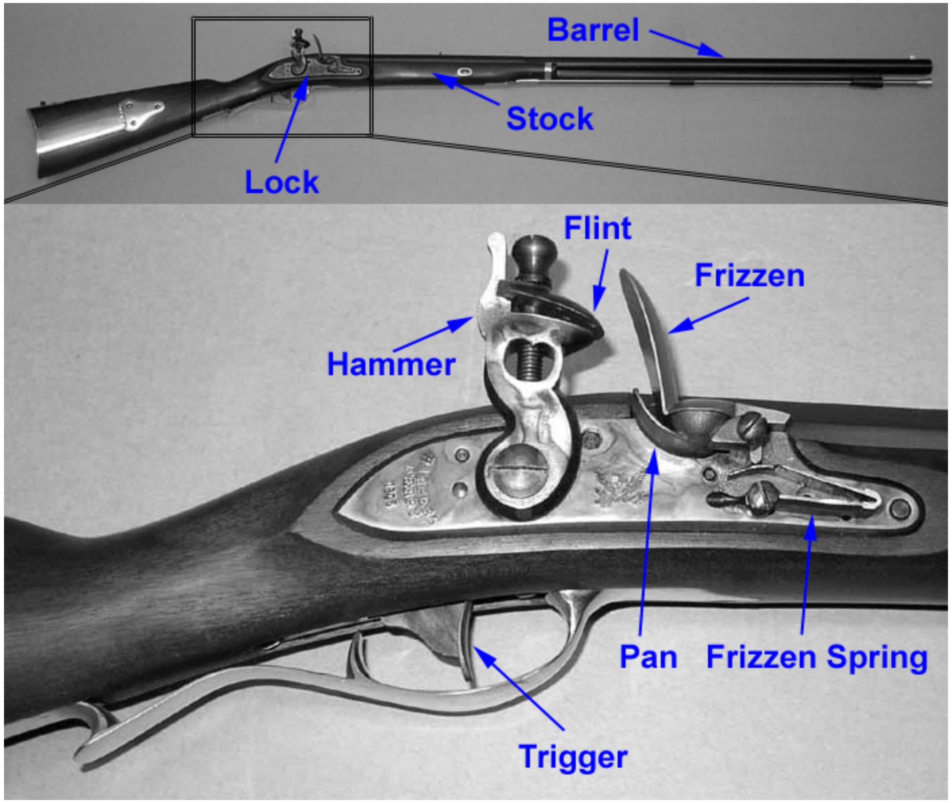
- Intentionally restrict design choices
- Example: Digital discipline
  - Discrete voltages instead of continuous
  - Simpler to design than analog circuits – can build more sophisticated systems
  - Digital systems replacing analog predecessors: i.e., digital cameras, digital television, cell phones, CDs

The Three -y's

- **Hierarchy**
  - –
- **Modularity**
  - –
- **Regularity**
  - –

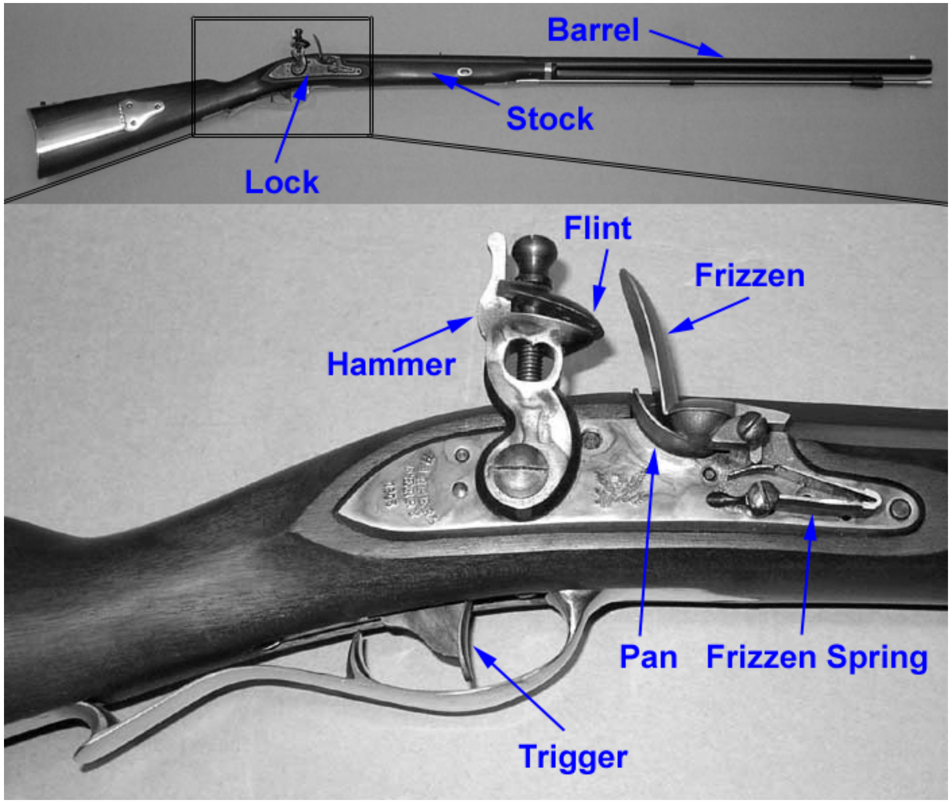
Example: The Flintlock Rifle

- **Hierarchy**
  - **Three main modules:** lock, stock, and barrel
  - **Submodules of lock:** hammer, flint, frizzen, etc.



Example: The Flintlock Rifle

- **Modularity**
  - **Function of stock:** mount barrel and lock
  - **Interface of stock:** length and location of mounting pins
- **Regularity**
  - Interchangeable parts



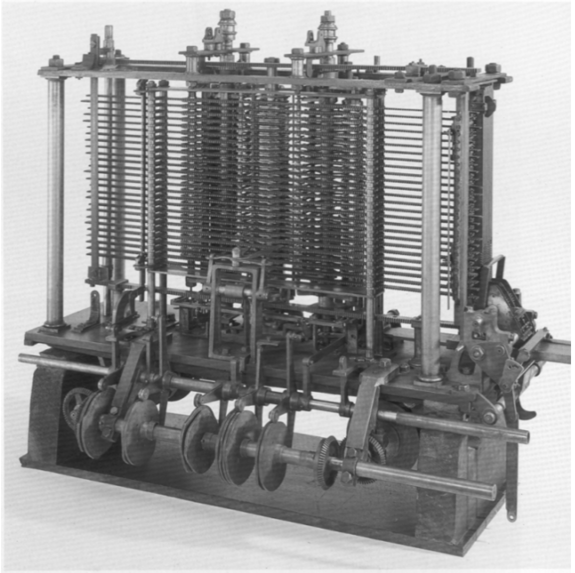
ELSEVIER

The Digital Abstraction

- Most physical variables are **continuous**
  - Voltage on a wire
  - Frequency of an oscillation
  - Position of a mass
- Digital abstraction considers **discrete subset** of values

The Analytical Engine

- Designed by Charles Babbage from 1834 – 1871
- Considered to be the first digital computer
- Built from mechanical gears, where each gear represented a discrete value (0-9)
- Babbage died before it was finished



Digital Discipline: Binary Values

- **Two discrete values:**
  - 1's and 0's
  - 1, TRUE, HIGH
  - 0, FALSE, LOW
- **1 and 0:** voltage levels, rotating gears, fluid levels, etc.
- Digital circuits use **voltage** levels to represent 1 and 0
- ***Bit***: *B*inary dig*it*

George Boole, 1815-1864

- Born to working class parents
- Taught himself mathematics and joined the faculty of Queen's College in Ireland
- Wrote *An Investigation of the Laws of Thought* (1854)
- Introduced binary variables
- Introduced the three fundamental logic operations: AND, OR, and NOT

Scanned at the American Institute of Physics

Number Systems

- Decimal numbers

1000's column
100's column
10's column
1's column

$$5374_{10} = 5 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$$

five thousands | three hundreds | seven tens | four ones

- Binary numbers

8's column
4's column
2's column
1's column

$$1101_2 =$$

Powers of Two

- $2^0 =$
- $2^1 =$
- $2^2 =$
- $2^3 =$
- $2^4 =$
- $2^5 =$
- $2^6 =$
- $2^7 =$

- $2^8 =$
- $2^9 =$
- $2^{10} =$
- $2^{11} =$
- $2^{12} =$
- $2^{13} =$
- $2^{14} =$
- $2^{15} =$

Handy to memorize

Number Conversion

- Binary to decimal conversion:
  - Convert 10011_2 to decimal
  -

- Decimal to binary conversion:
  - Convert 47_{10} to binary
  -

Decimal to Binary Conversion

- Two methods:
  - **Method 1:** Find the largest power of 2 that fits, subtract and repeat
  - **Method 2:** Repeatedly divide by 2, remainder goes in next most significant bit

Decimal to Binary Conversion

53_{10}

Method 1: Find the largest power of 2 that fits, subtract and repeat

Method 2: Repeatedly divide by 2, remainder goes in next most significant bit

Decimal to Binary Conversion

Another example: Convert 75_{10} to binary.

or

Binary Values and Range

- ***N*-digit decimal number**
  - How many values?
  - Range?
  - Example: 3-digit decimal number:

- ***N*-bit binary number**
  - How many values?
  - Range:
  - Example: 3-digit binary number:

Hexadecimal Numbers

Hex Digit	Decimal Equivalent	Binary Equivalent
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Hexadecimal Numbers

- Base 16
- Shorthand for binary

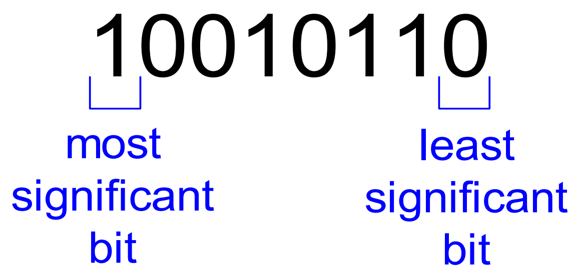
Hexadecimal to Binary Conversion

- Hexadecimal to binary conversion:
  - Convert $4AF_{16}$ (also written 0x4AF) to binary
  -

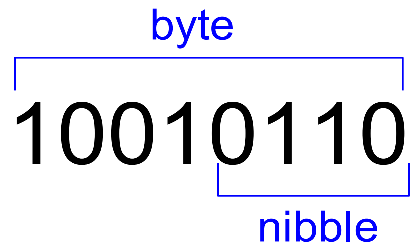
- Hexadecimal to decimal conversion:
  - Convert $4AF_{16}$ to decimal
  -

Bits, Bytes, Nibbles…

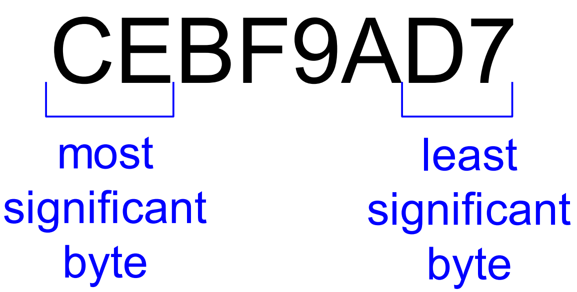
- Bits



- Bytes & Nibbles



- Bytes



Large Powers of Two

- 2^{10} = 1 kilo ≈ 1000 (1024)
- 2^{20} = 1 mega ≈ 1 million (1,048,576)
- 2^{30} = 1 giga ≈ 1 billion (1,073,741,824)
- 2^{40} = 1 tera ≈ 1 trillion
- 2^{50} = 1 peta ≈ 1 quadrillion
- 2^{60} = 1 exa ≈ 1 quintillion

Estimating Powers of Two

- What is the value of 2^{24}?

- How many values can a 32-bit variable represent?

Addition

- Decimal

11 ← carries
3734
+ 5168
8902

- Binary

11 ← carries
1011
+ 0011
1110

Binary Addition Examples

- Add the following 4-bit binary numbers

$$\begin{array}{r} 1001 \\ +\ 0101 \\ \hline \end{array}$$

- Add the following 4-bit binary numbers

$$\begin{array}{r} 1011 \\ +\ 0110 \\ \hline \end{array}$$

Overflow

- Digital systems operate on a **fixed number of bits**
- Overflow: when result is too big to fit in the available number of bits
- See previous example of 11 + 6

Signed Binary Numbers

- Sign/Magnitude Numbers
- Two’s Complement Numbers

Sign/Magnitude Numbers

- 1 sign bit, *N*-1 magnitude bits
- Sign bit is the most significant (left-most) bit
  - Positive number: sign bit = 0
  - Negative number: sign bit = 1

$$A: \{a_{N-1}, a_{N-2}, \ldots a_2, a_1, a_0\}$$

$$A = (-1)^{a_{N-1}} \sum_{i=0}^{N-2} a_i \, 2^i$$

- Example, 4-bit sign/mag representations of ± 6:

  +6 =

  - 6 =

- Range of an *N*-bit sign/magnitude number:

Sign/Magnitude Numbers

Problems:

- Addition doesn't work, for example -6 + 6:

```
   1110
 + 0110
 ------
  10100 (wrong!)
```

- Two representations of 0 (± 0):

```
   1000
   0000
```

Two's Complement Numbers

- Don't have same problems as sign/magnitude numbers:
  - **Addition works**
  - **Single representation for 0**

Two's Complement Numbers

- msb has value of -2^{N-1}

$$A = a_{N-1}(-2^{N-1}) + \sum_{i=0}^{N-2} a_i \, 2^i$$

- Most positive 4-bit number:
- Most negative 4-bit number:
- The most significant bit still indicates the sign (1 = negative, 0 = positive)
- Range of an *N*-bit two's complement number:

"Taking the Two's Complement"

- "Taking the Two's complement" **flips the sign** of a two's complement number
- **Method:**
  1. Invert the bits
  2. Add 1
- **Example:** Flip the sign of $3_{10} = 0011_2$
  1.
  2.

Two's Complement Examples

- Take the two's complement of $6_{10} = 0110_2$
  1.
  2.

- What is the decimal value of the two's complement number 1001_2?
  1.
  2.

Two's Complement Addition

- Add 6 + (-6) using two's complement numbers

```
    0110
+   1010
--------
```

- Add -2 + 3 using two's complement numbers

```
    1110
+   0011
--------
```

Two's Complement Addition

- Add 6 + (-6) using two's complement numbers

$$\begin{array}{r} 0110 \\ +\ \ 1010 \\ \hline \end{array}$$

- Add -2 + 3 using two's complement numbers

$$\begin{array}{r} 1110 \\ +\ \ 0011 \\ \hline \end{array}$$

Increasing Bit Width

Extend number from *N* to *M* bits (*M* > *N*) :

- Sign-extension
- Zero-extension

Sign-Extension

- Sign bit copied to msb's
- Number value is same
- **Example 1:**
  - 4-bit representation of 3 = **0**011
  - 8-bit sign-extended value: **0000**0011
- **Example 2:**
  - 4-bit representation of -5 = **1**011
  - 8-bit sign-extended value: **1111**1011

Zero-Extension

- Zeros copied to msb's
- Value changes for negative numbers
- **Example 1:**
  - 4-bit value = 0011 = 3_{10}
  - 8-bit zero-extended value: **0000**0011 = 3_{10}
- **Example 2:**
  - 4-bit value = 1011 = -5_{10}
  - 8-bit zero-extended value: **0000**1011 = 11_{10}

Number System Comparison

Number System	**Range**
Unsigned	$[0, 2^N-1]$
Sign/Magnitude	$[-(2^{N-1}-1), 2^{N-1}-1]$
Two's Complement	$[-2^{N-1}, 2^{N-1}-1]$

For example, 4-bit representation:

	Unsigned	Two's Complement	Sign/Magnitude
-8		1000	
-7		1001	1111
-6		1010	1110
-5		1011	1101
-4		1100	1100
-3		1101	1011
-2		1110	1010
-1		1111	1001
0	0000	0000	0000 1000
1	0001	0001	0001
2	0010	0010	0010
3	0011	0011	0011
4	0100	0100	0100
5	0101	0101	0101
6	0110	0110	0110
7	0111	0111	0111
8	1000		
9	1001		
10	1010		
11	1011		
12	1100		
13	1101		
14	1110		
15	1111		