ECE 2049: Lecture 1

- Intro to number representations

Administrivia

- HW1: online after class, due next Tues

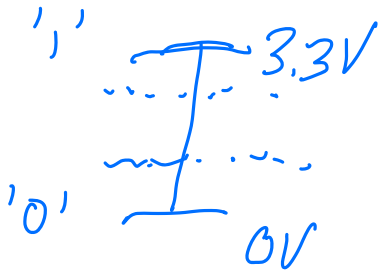
- Lab 0: starts Thursday – show up to get your boards!

- Please fill out the "background survey" on the website.

Module 1. Intro to Number Representations

Topics

- How do we store (or "encode") information in digital systems?
- Specifically: how do we store numbers?



First things first: Remembering Digital Logic

Before we can talk about how computing systems are built, we first need to talk about their basic building block: digital logic. In digital logic, information is represented in binary *bits*.

1 BIT = VALUE 0 or 1

Digital logic defines how we can process information using bits:

- LOGICAL (AND, OR, NOT, ...)
- ARITHMETIC (+, −, /, ...)

- STORAGE

⇒ COMBINE THESE ELEMENTS TO BUILD MORE COMPLEX COMPONENTS.

First things first: n bits differentiate among 2^n things.

N BITS = 2^N UNIQUE "CODES"

Ex. 2 BITS = $2^2 = 4$ "CODES"

00, 01, 10, 11

Terminology: 1 byte = 8 binary digits = 8 bits (e.g. 10010011)

½ byte = 1 nibble = 4 bits

1 word = 2 (or more) bytes --> MSP430 word = 2 bytes (16 BITS)

1 double word = 2 words (4 bytes on MSP430)

↳ (32 BITS)

In computers, ***information*** and ***memory space*** is organized in to multiples of bytes.
But what do the bytes mean?

The meaning of bits and bytes assigned by convention!

>> Under a given coding convention, a byte can represent up to $2^8 = 256$ things

For example, 1 byte (8 bits) could encode:

- A letter in an alphabet
- One or more decimal numbers
- The state of eight individual things (one per bit)
- An *instruction* that tells the CPU to do something:

We call these conventions **encoding formats**. They represent a kind of contract on how data will be stored and used. As programmers, it is up to us to assign meaning to those bits—which defines what operations we perform on them.

Conversion between Bases and Formats: Binary

Positional Number Systems

We write numbers in a positional system, which can be defined as:

RADIX (BASE)

$1734 \Rightarrow 1\times10^3 + 7\times10^2 + 3\times10^1 + 4\times10^0$

DECIMAL (BASE 10)

$$D = \sum_{i=-N}^{P-1} d_i \cdot r^i \quad \text{WHERE}$$

r = RADIX

d_i = i'TH DIGIT

N = DIGITS LEFT OF •

P = DIGITS RIGHT OF •

ddd.ddd

For binary numbers, we can write this definition as:

BASE 2: 0, 1

$$B = \sum_{j=-N}^{P-1} b_j (2^j) \quad \text{WHERE } b_j \in \{0,1\}$$

Unsigned integers = All bits used to convey magnitude (whole numbers ≥ 0)

Ex, 1001 0001b (MSB, LSB) $\Rightarrow 1\times2^7 + 0\times2^6 + 0\times2^5 + 1\times2^4 + 0\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0$

BINARY $\Rightarrow$ DECIMAL

$= 128 + 16 + 1 = 145d$

Decimal to Binary Conversion – Successive Division

↳ DIVIDE BY 2, LOOK AT REMAINDER

Ex. 44

44/2 = 22 R 0 ← LSB

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 ← MSB

5 4 3 2 1 0

101100b

$2^5 + 2^3 + 2^2$

$32 + 8 + 4 = 44$ ✓

Note: To differentiate numbers in different formats, we use notation to denote the radix used wo write it. For binary: 1010_2 or 1010b; decimal: 1010_{10} or 1010d (or just 1010)

Hexadecimal: A common way to write binary numbers

Since working in binary can be cumbersome, we often write numbers in *hexadecimal*, which is base 16.

Simple rule for conversion: ONE HEX "DIGIT" = 4 BITS $2^4 = 16$

--> 1 Hex character represents values from 0 to 15d using digits 0 – Fh

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

Ex. 158d =

158/16 = 9 R 14

9/16 = 0 R 9

9Eh

-OR- 0x9E

<u>If you memorize anything in this class, memorize these!</u>

Notation: Numbers in hex are written as 1010h or 0x1010

Conversion between hex and binary is piece of cake! Just convert each hex digit to a binary nibble...

1001 1110b = 158d

9 E

3 2 1 0

1 0 0 1 $= 2^3 + 2^0$

$2^3\ 2^2\ 2^1\ 2^0$

8 4 2 1

$= 8 + 1 = 9$

Or vice versa:

8AC4h = 8 A C 4

1000 1010 1100 0100 b

Note: A modern computer always stores information in binary form. Writing in hex is just a faster way for us to read and write these numbers—the machine's representation is still binary.

-6

How do we store negative numbers?

One way: Sign Magnitude integers = n-1 bits used to convey magnitude with "most significant bit" or MSB used for sign. Convention: 0 = +, 1 = -

FOR 8 BITS SMMM MMMM

"SIGN BIT" → "MAGNITUDE"

0|010 0001 = +33

1|010 0001 = -33

0|000 0000 = 0

1|000 0000 = -0 ???

AMBIGUOUS

INEFFICIENT

Note: This format has 2 representations of 0 = +0 and -0 !

Another way: Two's Complement integers = More common format for signed integers. For n bits, values range from $-2^{(n-1)}$ to $2^{(n-1)}-1$

How it works:

Positive numbers: Follow same format as unsigned numbers

1026 = 0000 0100 0000 0010b = 0402h

$2^{10} + 2^1 = 1024 + 2 = 1026$ ✓

ASSUME 4 BIT NUMBERS

Negative numbers: *Write* magnitude, *Complement* each bit, *Add* 1

-15 = ONLY IF NEGATIVE

0 + 0 = 0

0 + 1 = 1

1 + 1 = 10

0000 1111 MAGNITUDE

1111 0000 COMPLEMENT (FLIP ALL BITS)

\+ 1 ADD 1

1111 0001 b

SIGN BIT IS SET

Ex. -1

0000 0001

1111 1110

\+ 1

1111 1111 => -1

1 1

111 3

\+ 1 1

100

Range of Values for 2's Complement

FOR 16 BITS = (-2^{15}) TO $(2^{15}-1)$ = -32768 TO +32767

0111 1111 1111 1111b = +32767

0111 1111 1111 1110b +32766

. . .

0000 0000 0000 0001 = 1

0000 0000 0000 0000b = 0

1111 1111 1111 1111 = -1

. . .

1000 0000 0000 0001b

1000 0000 0000 0000b = -32768

Ex: Find the 8-bit two's complement representation of 104 and -80

Ex. 104 => POSITIVE, TREAT IT LIKE UNSIGNED

CAN CONVERT W/ SUCCESSIVE DIVISION...

104 /2 = 52 R0

52/2 = 26 R0

⋮

0110 1000b

$= 2^6 + 2^5 + 2^3 = 104$

Ex. -80 => NEGATIVE, SO NEED 2'S COMP PROCEDURE.

↳ FOR SUCCESSIVE DIVISION EXAMPLE, SEE NEXT PAGE...

80 => 0101 0000 MAG

1010 1111 COMP

+0000 0001

1011 0000 = -80 ✓

Ex: What are the decimal equivalent values of these 2's complement values

0010 0011b

POSITIVE

$2^5 + 2^1 + 2^0 = 32 + 2 + 1 = 35d$

1000 0011b

NEGATIVE, SO NEED 2'S COMP PROCEDURE

1000 0011 NUMBER

0111 1100 COMP

+ 1 ADD 1

0111 1101 = 125 => -125

FOUND MAG.

CONVERTING $80 TO BINARY...

80/2 = 40 R 0
40/2 = 20 R 0
20/2 = 10 R 0
10/2 = 5 R 0
5/2 = 2 R 1
2/2 = 1 R 0
1/2 = 0 R 1

0 1 0 1 0 0 0 0

Ex: What decimal value does 8008h represent as an…
(a) unsigned integer (b) 2's comp integer (c) sign-magnitude integer

$2^3 + 2^2 + 2^1$

UNSIGNED INTEGER

A. 8008h = 15 1000 0000 0000 3 2 1 0 1000

$= 2^{15} + 2^3 = \boxed{32776}$

B. 2's COMP.
NEGATIVE, SO NEED 2's COMP PROCEDURE

1000 0000 0000 1000
0111 1111 1111 0111 COMP
\+ 0000 0000 0000 0001
0111 1111 1111 1000

$= 2^{14} + 2^{13} + \ldots + 2^3 = \boxed{-32760}$

C. SIGN MAGNITUDE 8008h.
SIGN => NEGATIVE
MAGNITUDE = $2^3 = 8$ => $\boxed{-8}$

15 1000 0000 0000 3 2 1 0 1000

NEGATIVE ↑ MAGNITUDE ↓ $2^3 = 8$ => -8