Computer Science 210

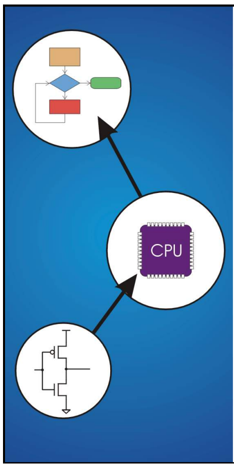
Computer Systems 1

Lecture Notes

Lecture 3

Introduction

Credits: Slides adapted from Gregory T. Byrd, North Carolina State University



Chapter 2

Bits, Data Types, and Operations

How do we represent data in a computer?

At the lowest level, a computer is an electronic machine.

- works by controlling the flow of electrons

Easy to recognize two conditions:

1. presence of a voltage – we'll call this state "1"
2. absence of a voltage – we'll call this state "0"

Could base state on *value* of voltage,
but control and detection circuits much more complex.

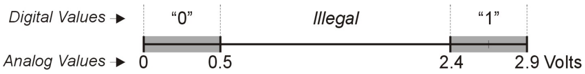
- compare turning on a light switch to measuring or regulating voltage

2-3

Computer is a Binary Digital System.

Digital system:
- *finite number of symbols*

Binary (base two) system:
- *has two states: 0 and 1*



Basic unit of information is the *binary digit*, or *bit*.

Values with more than two states require multiple bits.
- A collection of two bits has four possible states: 00, 01, 10, 11
- A collection of three bits has eight possible states: 000, 001, 010, 011, 100, 101, 110, 111
- *A collection of n bits has 2^n possible states.*

2-4

What kinds of data do we need to represent?

- Numbers – signed, unsigned, integers, floating point, complex, rational, irrational, …
- Text – characters, strings, …
- Images – pixels, colors, shapes, …
- Sound
- Logical – true, false
- Instructions
- …

Data type:
- *representation* and *operations* within the computer

We'll start with numbers…

2-5

Unsigned Integers

Non-positional notation
- could represent a number ("5") with a string of ones ("11111")
- problems?

Weighted positional notation
- like decimal numbers: "329"
- "3" is worth 300, because of its position, while "9" is only worth 9

329: 10^2 10^1 10^0

3x100 + 2x10 + 9x1 = 329

most significant → *101* ← *least significant*: 2^2 2^1 2^0

1x4 + 0x2 + 1x1 = 5

2-6

Unsigned Integers (cont.)

An n-bit unsigned integer represents any of 2^n (integer) values: from 0 to 2^n-1.

2^2	2^1	2^0	Value
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

How to convert decimal to binary video http://youtu.be/qWxiXU02ZQM

2-7

Unsigned Binary Arithmetic

Base-2 addition – just like base-10!

- add from right to left, propagating carry

```
  10010        10010       1111
+  1001      +  1011     +    1
  11011        11101      10000

               10111
             +   111
```

(carry)

Subtraction, multiplication, division, …

2-8

"There are 10 kinds of people in the world: those who understand binary, and those who don't".

"There are 10_2 kinds of people in the world: those who understand binary, and those who don't".

"There are 10_{10} kinds of people in the world: those who understand binary, and those who don't".

"There are 10_{two} kinds of people in the world: those who understand binary, and those who don't".

-- http://en.wikipedia.org/wiki/Mathematical_joke

1-9

Signed Integers

With n bits, we can distinguish 2^n unique values

- assign about half to positive integers (1 through 2^{n-1}) and about half to negative (-2^{n-1} through -1)
- that leaves two values: one for 0, and one extra

Positive integers

- just like unsigned, but zero in *most significant* (MS) bit
  00101 = 5

Negative integers

- Sign-Magnitude (or Signed-Magnitude) – set MS bit to show negative, other bits are the same as unsigned
  10101 = -5
- One's complement – flip every bit to represent negative
  11010 = -5
- In either case, MS bit indicates sign: 0=positive, 1=negative
- Video: http://youtu.be/9W67I2zzAfo

2-10

Two's Complement

Problems with sign-magnitude and 1's complement

- two representations of zero (+0 and –0)
- arithmetic circuits are complex
  - How to add two sign-magnitude numbers?
    - e.g., try 2 + (-3)
  - How to add two one's complement numbers?
    - e.g., try 4 + (-3)

Two's complement representation developed to make circuits easy for arithmetic.

- for each positive number (X), assign value to its negative (-X), such that X + (-X) = 0 with "normal" addition, ignoring carry out

```
   00101  (5)        01001  (9)
+  11011  (-5)   +  ______  (-9)
   00000  (0)        00000  (0)
```

2-11

Two's Complement

To get a negative number first "flip the bits" of its positive binary representation

```
00000101  (+5)
11111010  (1's complement)
```

Then add 1

```
  11111010
+        1
  11111011  (-5)
```

2-12

Two's Complement Signed Integers

MS bit is sign bit

Range of an n-bit number: -2^{n-1} through $2^{n-1} - 1$

- The most negative number (-2^{n-1}) has no positive counterpart.

-2^3	2^2	2^1	2^0		-2^3	2^2	2^1	2^0	
0	0	0	0	0	1	0	0	0	-8
0	0	0	1	1	1	0	0	1	-7
0	0	1	0	2	1	0	1	0	-6
0	0	1	1	3	1	0	1	1	-5
0	1	0	0	4	1	1	0	0	-4
0	1	0	1	5	1	1	0	1	-3
0	1	1	0	6	1	1	1	0	-2
0	1	1	1	7	1	1	1	1	-1

2-13

"Biased" Representation of Signed Integers

All integers (positive & negative) are represented as an unsigned integer supplemented with a "bias" to be subtracted out.

Range of an n-bit number: (0 - bias) through (2^n-1 - bias).

Bias 7:

2^3	2^2	2^1	2^0	Bias-7	2^3	2^2	2^1	2^0	Bias-7
0	0	0	0	-7	1	0	0	0	1
0	0	0	1	-6	1	0	0	1	2
0	0	1	0	-5	1	0	1	0	3
0	0	1	1	-4	1	0	1	1	4
0	1	0	0	-3	1	1	0	0	5
0	1	0	1	-2	1	1	0	1	6
0	1	1	0	-1	1	1	1	0	7
0	1	1	1	0	1	1	1	1	8

2-14

Converting Binary (2's C) to Decimal

1. If leading bit is zero (a positive number) just convert as normal

```
X  =  01101000
   =  2^6+2^5+2^3
   =  64+32+8
X  =  104
```

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

Assuming 8-bit 2's complement numbers.

2-15

Converting Binary (2's C) to Decimal

Same as before **EXCEPT** the MS bit is negative

```
X= 11100110
 = -2^7+2^6+2^5+2^2+2^1
 = -128+64+32+4+2
X=-26
```

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	-128

Assuming 8-bit 2's complement numbers.

2-16

Converting Decimal to Binary (2's C)

First Method: *Division*

1. Find magnitude of decimal number. (Always positive.)
2. Divide by two – remainder is least significant bit.
3. Keep dividing by two until answer is zero, writing remainders from right to left.
4. Append a zero as the MS bit; if original number was negative, take two's complement.

$X = 104_{ten}$

104/2 =	52 r0	bit 0
52/2 =	26 r0	bit 1
26/2 =	13 r0	bit 2
13/2 =	6 r1	bit 3
6/2 =	3 r0	bit 4
3/2 =	1 r1	bit 5
1/2 =	0 r1	bit 6

$X = 01101000_{two}$

2-17

Converting Decimal to Binary (2's C)

Second Method: *Subtract Powers of Two*

1. Find magnitude of decimal number.
2. Subtract largest power of two less than or equal to number.
3. Put a one in the corresponding bit position.
4. Keep subtracting until result is zero.
5. Append a zero as MS bit; if original was negative, take two's complement.

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

$X = 104_{ten}$

104 - 64 =	40	bit 6
40 - 32 =	8	bit 5
8 - 8 =	0	bit 3

$X = 01101000_{two}$

2-18