Data Representaton:
Bits, Data Types, Operations
(Chapter 2)

How do you represent data ?

- Our first requirement is to find a way to represent information (data) in a form that is mutually comprehensible by human and machine.
  - What kinds of data ?
    - Integers
    - Reals
    - Text
    - …what else
    - …

Data Type

- In a computer system, we need a representation of data and operations that can be performed on the data by the machine instructions or the computer language.
- This combination of *representation* + *operations* is known as a data type.
  - The type tells the compiler how the programmer intends to use it
- Prog. Languages have a set of data types defined in lang
  - In C: int, float, char, unsigned int, …

Type	Representation	Operations
Unsigned integers	binary	add, multiply, etc.
Signed integers	2's complement binary	add, multiply, etc.
Real numbers	IEEE floating-point	add, multiply, etc.
Text characters	ASCII	input, output, compare

Number systems

- A number is a mathematical concept
  - Natural numbers, Integers, Reals, Rationals,..
- Many ways to represent a number…..
  - Symbols used to create a representation
  - Example: Decimal representation uses the symbols (digits) 0,1,2…9
    - Binary uses the symbols 0,1
  - Roman numerals: I, II, V, X, etc.

Your first counting numbers experience ? How did you learn to count? How did you express a number ?

The Unary system is also used by Turing Machines …Why ?

5

5

In the CS world…..

- **There are 10 kinds of people in the world…**

Those who know binary, and those who don't

?

6

6

Computer is a Binary Digital System

- Digital = finite number of values (compared to ‘analog’= infinite values)
- Binary = only two values: 0 and 1
  - Unit of information = binary digit or “bit”

- Circuits (Chap 3) will pull voltage down towards zero or will pull up towards highest voltage
  - Grey areas represent noise margin – allowable deviation due to electrical properties (resistance, capacitance, interference,..)
  - More reliable than analog
- Alternative: can define multiple discrete values in voltage range
  - Problem: circuits would become much more complex

7

7

If we have more than two values…

- Basic unit of information = binary digit or *bit*
- Each “wire” in a logic circuit represents one bit = 0 or 1
- Values with more than 2 states require multiple wires (bits)
- With 2 bits → 4 possible values (states/strings): 00, 01, 10, 11
- 3 bits → 8 values: 000, 001, 010, 011, 100, 101, 110, 111
- **In general: with n bits can represent 2^n different values**

8

8

Bits – the universal data representation

- everything that is stored or manipulated on the computer is ultimately expressed as a group of bits.
  - Text – characters, strings,
  - Numbers – integer, fraction, real,…
  - Video, Audio, Images (using pixels…pixel can be 8 bits)
  - Logical – True (1) or False (0)
  - Instructions (program) are just 0's and 1's = programs are just another kind of data!
- We encode a value by assigning a bit pattern to represent that value
- We perform operations (transformations) on bits, and we interpret the results according to how the data is encoded

9

9

Hmmm……Machine Data Types

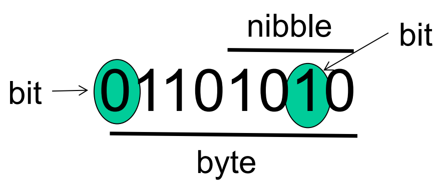
- devices that make up a computer are switches that can be on or off, i.e. at high or low voltage.
  - Thus they naturally provide us with two symbols to work with: we can call them *on* & *off*, or (more usefully) *0* and *1*.
- We don't want to keep referring to switches…
  - power of abstraction and problem transformation !

10

10

Terminology

- A single binary digit is referred to as a bit
- A collection of 8 bits is referred to as a byte
- A collection of 4 bits is referred to as a nibble
  - Also a *Hex digit*
- In a computer memory each storage location can only hold a finite number of bits



Data Representation

- We encode a value by assigning a bit pattern to represent that value
  - Encoding determines *how to interpret* the value of an n-bit binary ‘string’
- How to represent different types of data:
  - Start with Integers
    - Unsigned (non-negative)
    - Negative
  - Text …ASCII codes
  - Real numbers – floating point

(Unsigned) Integer Representation

- Non-positional notation (unary): 5 represented as 11111
- What are you used to ? Decimal representation (0..9) and…
- Decimal **Weighted positional representation**
  - *Position gives the weight of the location*
  - decimal number “329” (three hundred twenty nine)
  - “3” is worth 300, because of its position (most significant)
  - “9” is only worth 9 (least significant)

base-10 (decimal)

329

10^2 10^1 10^0

3x100 + 2x10 + 9x1 = 329

Value= Three hundreds,
Two tens, and
Nine ones.

13

13

Integer Representation

- Weighted positional representation in Binary

base-10 (decimal)

329

10^2 10^1 10^0

3x100 + 2x10 + 9x1 = 329

most significant 101 *least significant*

2^2 2^1 2^0

base-2 (binary)

1x4 + 0x2 + 1x1 = 5

Notations: the bit position *i* has weight of 2^i
n bit binary number $a_{n-1}a_{n-2},\ldots,a_1,a_0$

represents the decimal value/number

$$\sum_{i=0}^{i=n-1} a_i\, 2^i$$

14

14

Unsigned Integers

- An *n*-bit unsigned integer represents 2^n values
  - Values from 0 to 2^n-1

2^2	2^1	2^0	val
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

15

15

Question

- what number does the binary string 1011 represent

- What number does 00011 represent ?

16

16

Decimal to Binary Conversion:

1. What is the binary representation of decimal number 19
   - Express 19 as a sum of numbers each a power of 2
   - Algorithm to convert decimal (base 10) to binary (base 2)
     - Generalize to convert from base k to base m

k bit number: $b_{k-1},b_{k-2},\ldots,b_1,b_0$
Decimal integer N represented by this binary number is:

$$b_{k-1}\, 2^{k-1} + b_{k-2} 2^{k-2} + \ldots + b_1\, 2^1 + b_0\, 2^0$$

$19 = 1.16 + 0.8 + 0.4 + 1.2 + 1.1$
$= 1.2^4 + 0.2^3 + 0.2^2 + 1.2^1 + 1.2^0$
10011

17

17

Conversion from Decimal to Binary

```
//input is Decimal number N, output is list of bits b_i //
i=0;
while N > 0 do
   b_i = N % 2; // b_i = remainder; N mod 2
    N = N / 2; // N becomes quotient of division
    i++;
end while
```

- Replace 2 by *r* and you have an algorithm that computes the base *r* representation for N

18

18

Example: Conversion of 19 to Binary

```
//input is Decimal number N, output is list of bits b_i //
i=0;
while N > 0 do
   b_i = N % 2; // b_i = remainder; N mod 2
    N = N / 2; // N becomes quotient of division
    i++;
end while
```

- Iteration 1: b_0 = 19%2 =1 and N= 19/2= 9
- Iteration 2: b_1 = 9%2 = 1 and N=4
- Iteration 3: b_2 = 4%2 = 0 and N=2
- Iteration 4: b_3 = 2%2 =0 and N=1
- Iteration 5: b_4 = 1%2 =1 and N=0 so loop terminates
- Binary representation of 19 = 10011

19

19

Arithmetic Operations on Unsigned Integers

- Recall: Data type is representation and operations

20

20

Unsigned Binary Arithmetic

- Base-2 addition -- just like base-10
- Add from right to left, propagating carry.

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

                 10111
               +   111
                 -----
                 11110
```

- Can also do subtraction, multiplication, etc., using base-2.

21

21

Question:

- 1. Add two 4-bit binary numbers 0011 and 1010,
  - what is the 4-bit result ?
- 2. Add two 4 bit numbers: 0100 and 1100
  - What is the 4-bit result ?

22

22

Recap: Binary representation of integers

- We saw how Natural numbers can be represented in binary using weighted positional system
- Arithmetic operations work way as with decimal representation
- In general, base-K (radix-K) representation of numbers using weighted positional system
  - Decimal is base-10
  - Binary is base 2

23

23

Negative Integers, Operations (Arithmetic and Logical), Real Numbers

24

24

What About Negative Integers?

- Negative numbers have rights too
  - No negation without representation!!
- How do we represent negative integers in decimal:
  - sign followed by value
  - - 269
  - +169 is usually written as 169 (drop the + sign)
- Question: Is this a valid (as per math definition) base 10 (decimal) representation ?

25

25

Negative Integers in Binary?

- One option: sign-magnitude concept
  - What do we do with paper-and-pencil: put a '-' in front
  - No '–' in binary, just use a 1 in most significant bit to denote sign (0= positive, 1= negative)

    00101 = 5

    10101 = –5
- Another option: 1's Complement
  - Simply complement bits
  - 00101 = 5
  - 11010 = -5
- Note: in both these representations, we are using an extra bit to denote the sign

26

26

Examples

- 4 bit representation of -2 in
  - Signed magnitude binary
    - First represent 2 in binary: 0010
    - Since negative, the most significant bit (leftmost) should be=1
    - Therefore -2 in signed magnitude binary is: 1010
  - 1's complement binary – first represent 2 in binary= 0010
    - Complement all the bits to get 1101
- A and B are signed magnitude binary nos.
  - A=1010 (-2) and B= 0011 (+3)
  - A+B = ?

```
 1010
 0011
 ----
 1101 (-5)
```

- A and B are 1's complement binary nos.
  - A= 1010 and B=0011
  - A+B= ?

```
 1010
 0011
 ----
 1101 (-2)
```

27

27

What type of representation do we want ?

- We would like the same arithmetic 'algorithms' work for negative numbers
  - Keeps hardware circuits simple
- We want the same addition algorithm
  - Add starting with rightmost (least significant) bit and propagate the carry bit to the left
- Oops…Problem with signed magnitude and 1's Comp
  - Same addition algorithm does not work!!
- Furthermore two representations for zero
  - In signed magnitude both 1000 and 0000 represent 0
  - In 1's Complement both 1111 and 0000 represent 0
- Using Signed magnitude or 1C to represent negative integers is a bad idea!

28

28

Two's Complement Representation

- viewed as weighted position: but weight of most significant bit is (-2^{N-1})
- If number is positive or zero,
  - normal binary representation, zero in most significant bit
- If number is negative,
  - start with positive number
  - flip every bit (i.e., take the one's complement)
  - then add one

```
  00101  (5)
  11010  (1's comp)
+     1
  -----
  11011  (-5)
```

29

29

More 2C examples

- Find 2C of 9
- Find 2C of -6

```
  01001  (9)
  10110  (1's comp)
+     1
  -----
  10111  (-9)
```

```
  11010  (-6)
  00101  (1's comp)
+     1
  -----
  00110  (6)
```

30

30

Addition

- Two 2's Complement numbers
- A = 1010
  - = negative, therefore flip bits and add 1 to get 0101+1=0110
  - A = -6
- B = 0011
  - = positive, therefore B =3
- What is A+B

```
   1010 ( -6)
   0011 ( 3)
+______
   1101 ( -3)
```

31

31

2C Summary

- If you have the binary representation for a number, to find the negative in 2C representation, simply:
  - Flip all the bits and add 1
    - OR
  - Copy bits from right to left up to and including the first '1'
    - Flip remaining bits
  - Techniques work in reverse as well!

32

32