CS F111: Computer Programming

(Second Semester 2021-22)

Lect 19: Number Systems

Nikumani Choudhury

Asst. Professor, Dept. of Computer Sc. & Information System

BITS Pilani

Hyderabad Campus

Number systems: Data Representation

- Our first requirement is to find a way to represent data in a form that is mutually understandable by human and machine.
  - Types: Numbers, Text, Images, Audio, Video, etc.
  - Uniform Representation: bit pattern (a string of 0’s and 1’s)
  - Specifically, the devices that make up a computer are switches that can be on or off, i.e. at high or low voltage. So, we have two symbols to work with: *on* & *off*, or (more usefully) *0* and *1*.

Decimal Numbers

- The <u>symbols</u> 0 through 9
- What is 546?
  - it is *five* hundreds plus *four* tens plus *six* ones.
- How about negative numbers?
  - we use two more symbols to distinguish positive and negative:
  - Will shortly cover these types

Binary Number System

$1101\ 0110_2 = 214_{10}$

Place	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Value	128	64	32	16	8	4	2	1
Evaluate	1 x 128	1 x 64	0 x 32	1 x16	0 x 8	1 x 4	1 x 2	0 x 1
Sum for Base 10	128	64	0	16	0	4	2	0

An example program for binary to decimal conversion

```c
#include <stdio.h>

void main()
{
    int  num, binary_val, decimal_val = 0, base = 1, rem;

    printf("Enter a binary number(1s and 0s) \n");
    scanf("%d", &num); /* maximum five digits */
    binary_val = num;
    while (num > 0)
    {
        rem = num % 10;
        decimal_val = decimal_val + rem * base;
        num = num / 10 ;
        base = base * 2;
    }
    printf("The Binary number is = %d \n", binary_val);
    printf("Its decimal equivalent is = %d \n", decimal_val);
}
```

```
Enter a binary number(1s and 0s)
110
The Binary number is = 110
Its decimal equivalent is = 6
```

Hexadecimal Representation

- Base 16 (hexadecimal)
  - More of a convenience to humans than a true data type
  - 0 to 9, A, B, C, D, E, F
  - $16 = 2^4$: i.e. every hexadecimal digit can be represented by a 4-bit binary (unsigned) and vice-versa.

104_{16} = ?

256's place 16's place 1's place

Place		16^2	16^1	16^0
Value		256	16	1
Evaluate		1 x 256	0 x 16	4 x 1
Sum for Base 10		**256**	**0**	**4**

Hexadecimal Representation continued…

- It is often convenient to write binary (base-2) numbers as hexadecimal (base-16) numbers instead.
- Why?
  - Fewer digits -- four bits per hex digit
  - Less error prone -- easy to corrupt long string of 1's and 0's

Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7

Binary	Hex	Decimal
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Octal Number System

- $8 = 2^3$: i.e. every octal digit can be represented by a 3-bit binary (unsigned) and vice-versa.

$524_8 = ?$

64's place 8's place 1's place

Place	8^2	8^1	8^0
Value	64	8	1
Evaluate	5x 64	2 x 8	4 x 1
Sum for Base 10	320	16	4

From Base 10 to Base 2

42_{10}

Remainder

Quotient

2)	**42**	(0
2)	**21**	(1
2)	**10**	(0
2)	**5**	(1
2)	**2**	(0
2)	1	

From Base 10 to Base 16

5735_{10}

Remainder

Quotient

16)	5,735	(7
16)	358	(6
16)	22	(6
16)	1	(1
16)	0	

From Base 10 to Base 8

427_{10}

Remainder

Quotient

8)	427	(3	LSB
8)	53	(5	
8)	6	(6	MSB
	0		

Binary to Hexadecimal and Binary to Octal using Substitution Codes

8 4 2 1

$0101011010101110_2 = ?_{16}$

0101 0110 1010 1110

5 6 A E

$\boxed{56AE_{16}}$

4 2 1

$010101101_2 = ?_8$

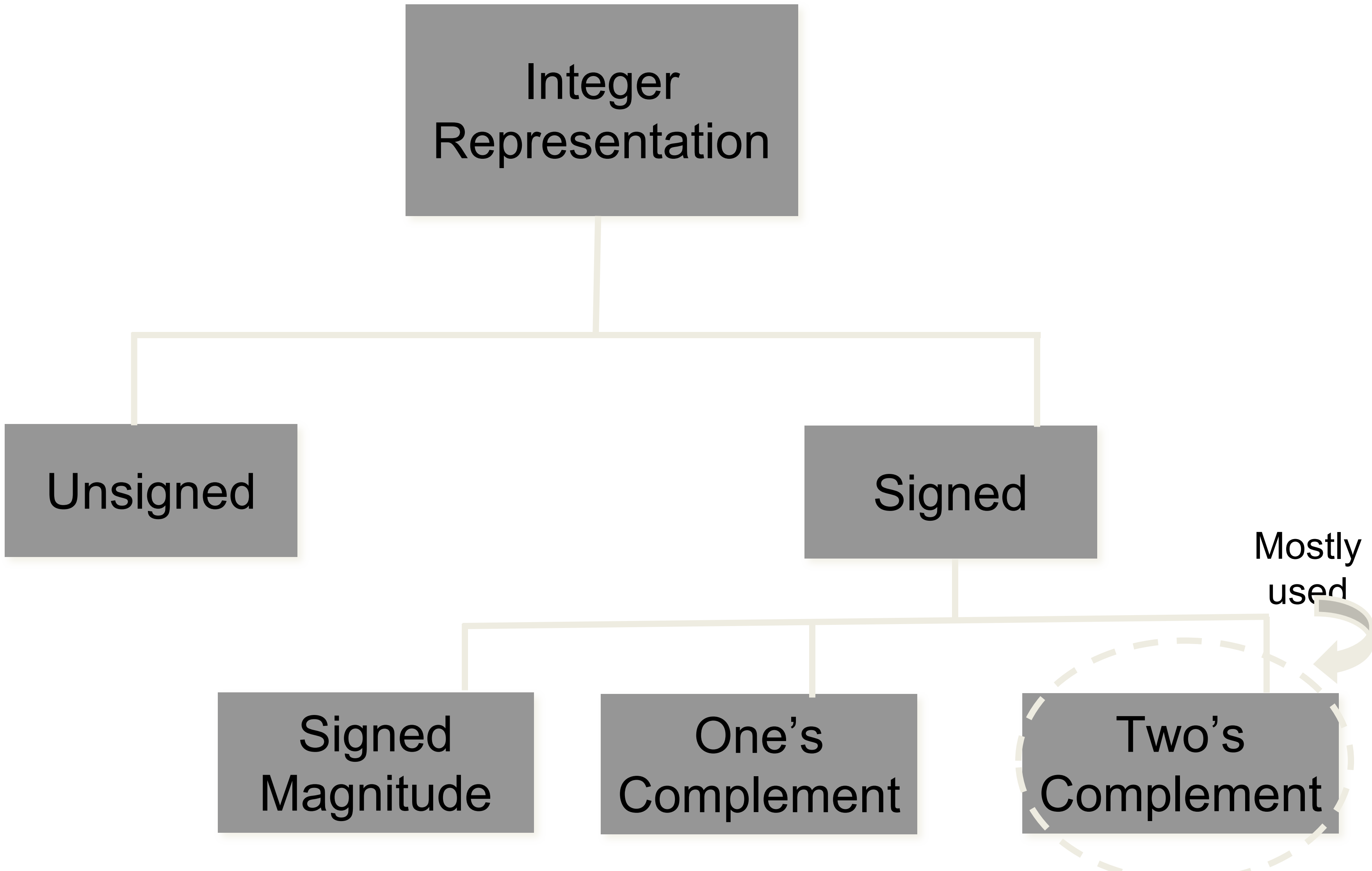
010 101 101

2 5 5

$\boxed{255_8}$

Try: $(1011)_2 = (-)_8$? and $(10111)_2 = (-)_{16}$?

Taxonomy of Integers



Unsigned Binary Integers

Applications: Counting, and Memory addressing

$(011)_2 = 0.2^2 + 1.2^1 + 1.2^0 = 2+1 = 3$

N = number of bits
Range is:
$0 \leq i <= (2^N - 1)$

	3-bits	5-bits	8-bits
0	000	00000	00000000
1	001	00001	00000001
2	010	00010	00000010
3	011	00011	00000011
4	100	00100	00000100
			…

<u>Problem:</u>

- How do we represent *negative* numbers?

Signed Magnitude

- How do we represent negative numbers?
- Leading bit is the sign bit

$Y = \text{“abc”} = (-1)^a\,(b.2^1 + c.2^0)$

where a, b, and c can each take on 0/1

N=no. of bits
Range is:
$-(2^{N-1} - 1)$ … $(2^{N-1} - 1)$

IBM 704, 709, etc…

…	
-4	10100
-3	10011
-2	10010
-1	10001
-0	10000
+0	00000
+1	00001
+2	00010
+3	00011
+4	00100
	…

<u>Problems:</u>

- There are two zeroes!
- Arithmetic is cumbersome (e.g. 4-3 using 4-bits)

Applications: When we do not need mathematical operations, to store analog and digital signals.