Fundamental of Programming (C)

Lecturer: Omid Jafarinezhad

Lecture 2
Number Systems

Outline

- Numeral Systems
- Computer Data Storage Units
- Numeral Systems Conversion
- Calculations in Number Systems
- Signed Integer Representation
- Fractional and Real Numbers
- ASCII Codes

Numeral Systems

- Decimal number system (base 10)

- Binary number system (base 2)
  - Computers are built using digital circuits
  - Inputs and outputs can have only two values: 0 and 1
    - 1 or True (high voltage)
    - 0 or false (low voltage)
  - Writing out a binary number such as 1001001101 is tedious, and prone to errors

- Octal and hex are a convenient way to represent binary numbers, as used by computers
  - Octal number system (base 8)
  - Hexadecimal number system (base 16)

Numeral Systems

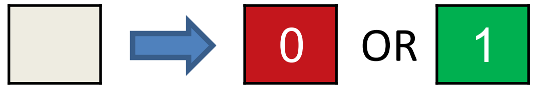
Base B : 0 ≤ digit ≤ B -1

Base 10 : 0 ≤ digit ≤ 9 (10 -1)
Base 2 : 0 ≤ digit ≤ 1 (2-1)
Base 8 : 0 ≤ digit ≤ 7 (8 -1)
Base 16 : 0 ≤ digit ≤ 15 (16 -1)

Decimal	Binary	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2		2	2
3		3	3
4		4	4
5		5	5
6		6	6
7		7	7
8			8
9			9
			A (decimal value of 10)
			B (decimal value of 11)
			C (decimal value of 12)
			D (decimal value of 13)
			E (decimal value of 14)
			F (decimal value of 15)

Computer Data Storage Units

- Bit



  - Each bit can only have a binary digit value: 1 or 0
  - basic capacity of information in computer
  - A single bit must represent one of two states: $2^1=2$

- How many state can encode by N bit?

Binary Encoding

	0
0	0
1	1

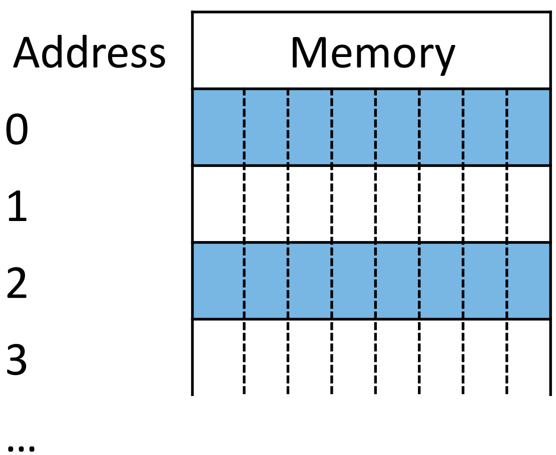
	1	0
0	0	0
1	0	1
2	1	0
3	1	1

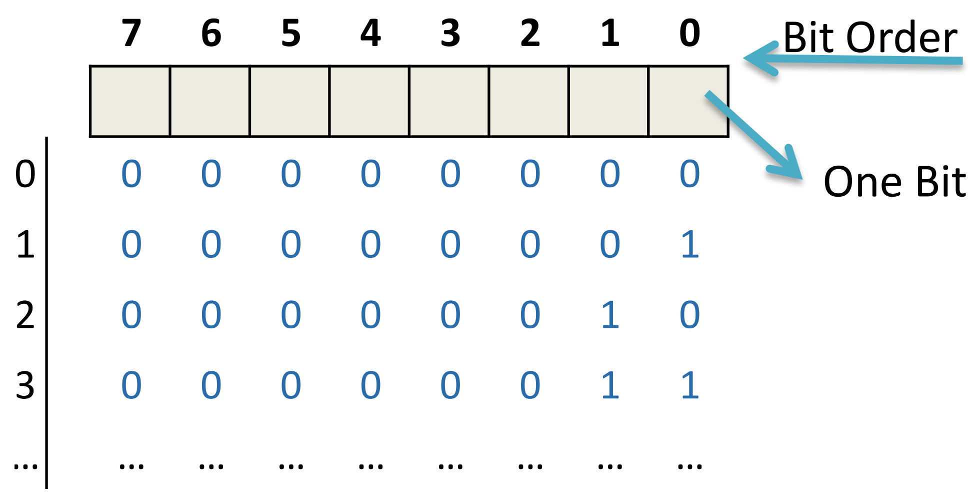
	2	1	0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

	3	2	1	0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Computer Data Storage Units

- Byte: A sequence of eight bits or binary digits
  - $2^8 = 256$ (0..255)
  - smallest addressable memory unit





Computer Data Storage Units

- Kilo byte: 2 10 = 1024
  - $2^{11} = 2$ K = 2048
  - $2^{16} = 64$ K = 65536
- Mega byte: $2^{20} = 1024 \times 1024$
  - $2^{21} = 2$ M
- Giga byte: 2^{30}

Numeral Systems Conversion

- Covert from Base-B to Base-10:
  - $(A)_B = (?)_{10}$
    - $(4173)_{10} = (4173)_{10}$
    - $(11001011.0101)_2 = (?)_{10}$
    - $(0756)_8 = (?)_{10}$
    - $(3b2)_{16} = (?)_{10}$

- Convert from Base-10 to Base-B:
  - $(N)_{10} = (?)_B$
    - $(4173)_{10} = (?)_2$
    - $(494)_{10} = (?)_8$
    - $(946)_{10} = (?)_{16}$

Covert from Base-B to Base-10

1. Define bit order

..	7	6	5	4	3	2	1	0		-1	-2	-3	...
									.				

- Example : Base-2 to Base-10

7	6	5	4	3	2	1	0		-1	-2	-3	-4
1	1	0	0	1	0	1	1	.	0	1	0	1

Covert from Base-B to Base-10

2. Calculate Position Weight

   – B bit order

Decimal Point

10^2	10^1	10^0		10^{-1}	10^{-2}
100s	10s	1s		1/10s	1/100s
9	8	7	.	5	6

- Example : Base-2 to Base-10

Position Weight

B = 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}
	7	6	5	4	3	2	1	0		-1	-2	-3	-4
	1	1	0	0	1	0	1	1	.	0	1	0	1

Covert from Base-B to Base-10

3. Multiplies the value of each digit by the value of its position weight

128	64	32	16	8	4	2	1		0.5	0.25	0.125	0.0625
7	6	5	4	3	2	1	0		-1	-2	-3	-4
1	1	0	0	1	0	1	1	.	0	1	0	1
127	64	32	16	8	4	2	1		0.5	0.25	0.125	0.0625
*	*	*	*	*	*	*	*		*	*	*	*
1	1	0	0	1	0	1	1		0	1	0	1
128	64	0	0	8	0	2	1	.	0	0.25	0	0.0625

Covert from Base-B to Base-10

4. Adds the results of each section

128	64	32	16	8	4	2	1		0.5	0.25	0.125	0.0625
7	6	5	4	3	2	1	0		-1	-2	-3	-4
1	1	0	0	1	0	1	1	.	0	1	0	1
128 * 1	64 * 1	32 * 0	16 * 0	8 * 1	4 * 0	2 * 1	1 * 1		0.5 * 0	0.25 * 1	0.125 * 0	0.0625 * 1
128	64	0	0	8	0	2	1	.	0	0.25	0	0.0625

203.3125 = **203** **+** **0.3125**

Covert from Base-B to Base-10

- Examples:

$(a_{n-1}\ a_{n-2}\ \dots\ a_0 . a_{-1} \dots a_{-m})_B = (N)_{10}$

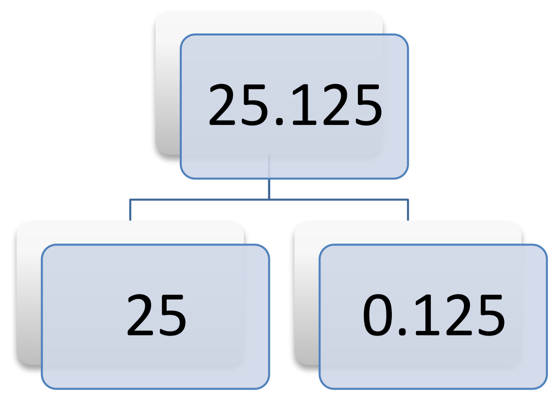
$N = (a_{n-1} * B^{n-1}) + (a_{n-2} * B^{n-2}) + \dots + (a_0 * B^0) + (a_{-1} * B^{-1}) + \dots + (a_{-m} * B^{-m})$

- $(4173)_{10} = (4 * 10^3) + (1 * 10^2) + (7*10^1) + (3*10^0) = (4173)_{10}$
- $(0756)_8 = (0 * 8^3) + (7 * 8^2) + (5*8^1) + (6*8^0) = (494)_{10}$
- $(3b2)_{16} = (3 * 16^2) + (11*16^1) + (2*16^0) = (946)_{10}$
- $(2E6.A3)_{16} = (2 * 16^2) + (14*16^1) + (6*16^0) +$ $(10 * \underbrace{(1 / 16)}_{16^{-1}}) + (3 * \underbrace{(1 / (16 * 16))}_{16^{-2}}) = (?)_{10}$

Convert from Base-10 to Base-B

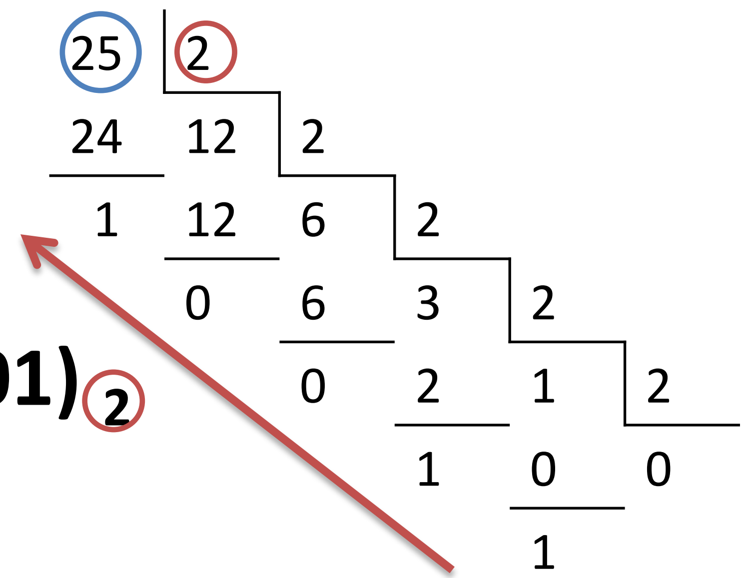
- $(N)_{10} = (\underbrace{a_{n-1}\ a_{n-2}\ \dots\ a_0}_{\text{Integer part}} \,.\, \underbrace{a_{-1}\ \dots\ a_{-m}}_{\text{Fraction part}})_B$

1. Convert integer part to Base-B
   – Consecutive divisions
2. Convert fraction part to Base-B
   – Consecutive multiplication



Convert Integer Part to Base-B

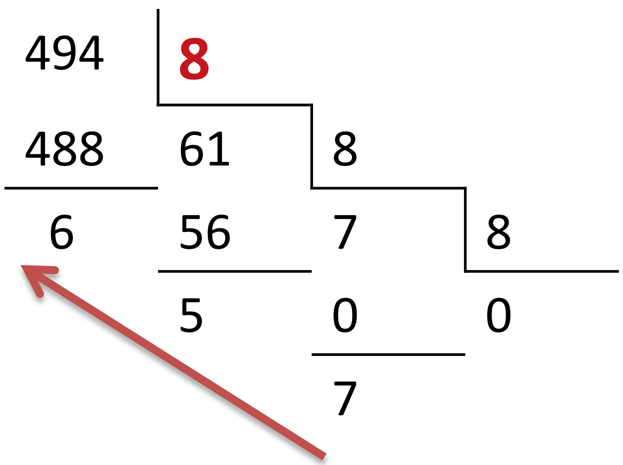
- Repeat until the quotient reaches 0
- Write the reminders in the reverse order
  - Last to first
- Examples:

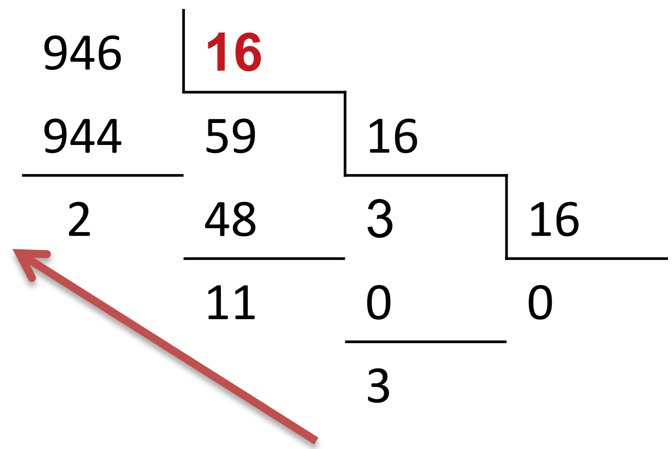


$$(25)_{10} = (11001)_2$$

Convert Integer Part to Base-B

- Examples:





$$(494)_{10} = (756)_{8}$$

$$(946)_{10} = (3B2)_{16}$$

Convert Fraction Part to Base-B

- Do
  - multiply fraction part by B (the result)
  - drop the integer part of the result (new fraction)
- While
  - (result = 0) OR (reach to specific precision)

- the integral parts from top to bottom are arranged from left to right after the decimal point

Convert Fraction Part to Base-B

- Example:
  - 0.125 * 2 = 0.25
  - 0.25 * 2 = 0.50
  - 0.50 * 2 = 1.00

    ~~1~~.00

  - 0.00 * 2 = 0.00

$(0.125)_{10} = (0.001)_2$

Convert Fraction Part to Base-B

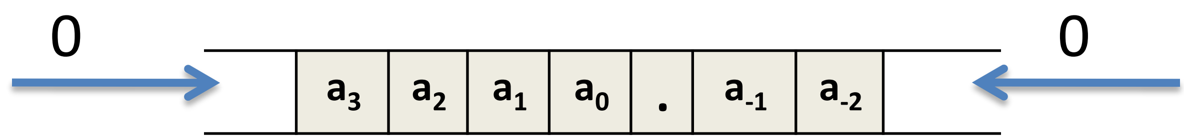
- Example:
  - 0.6 * 2 = 1.2
  - 0.2 * 2 = 0.4
  - 0.4 * 2 = 0.8
  - 0.8 * 2 = 1.6
  - 0.6 * 2 = …

$$(0.6)_{10} = (0.\overline{1001})_2$$

Conversion binary and octal

- Binary to Octal
  - $2^3 = 8$
  - Each digit in octal format: 3 digits in binary format
  - If the number of digits is not dividable by 3, add additional zeros:

Octal	0	1	2	3	4	5	6	7
Binary	000	001	010	011	100	101	110	111



- 43 = 043 = 000043 = 043.0 = 043.000
- $(10011.1101)_2 = (010011.110100)_2 = (23.64)_8$

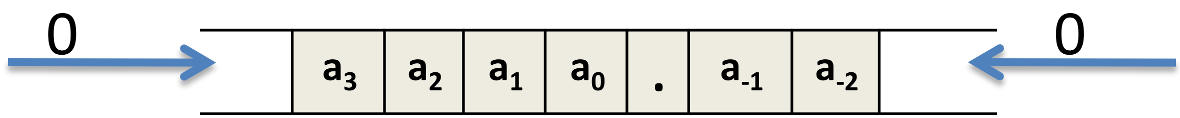
2 3 6 4

Conversion binary and octal

- Octal to Binary
  - Substitute each digit with 3 binary digits
    - $(5)_8 = (101)_2$
    - $(1)_8 = (001)_2$
    - $(51)_8 = (101\ 001)_2$
    - $(23.61)_8 = (010\ 011.110\ 001)_2$

Conversion binary and Hexadecimal

- Binary to Hexadecimal
  - $2^4 = 16$
  - Each digit in octal format: 4 digits in binary format
  - If the number of digits is not dividable by 4, add additional zeros:



  - $(1111101.0110)_{16} = (01111101.0110)_{16} = (7D.6)_2$

    7 D 6

Conversion binary and Hexadecimal

- Hexadecimal to Binary
  - Substitute each digit with 4 binary digits
    - $(F25.03)_{16} = (1111\ 0010\ 0101.0000\ 0011)_2$

1111 0010 0101 . 0000 0011

Conversion

- Octal ⟺ binary ⟺ Hexadecimal
  - $(345)_8 = (E5)_{16}$
  - $(345)_8 = (\underline{011}\,\underline{100}\,\underline{101})_2 = (\not{0}\underline{1110}\,\underline{0101})_2 = (E5)_{16}$

    3 4 5 E 5
  - $(3FA5)_{16} = (0011111110100101)_2 =$

    $(0011111110100101)_2 = (037645)_8$

    $= (37645)_8$

Calculations in Numeral Systems

- Addition
  - Binary

$(1 + 1)_{10} \rightarrow (2)_{10} = (10)_2 \Leftrightarrow$

$$\begin{array}{rr} 1 & \\ & 1 \\ + & 1 \\ \hline 1 & 0 \end{array} \qquad \begin{array}{r} 1 \\ +\;0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 0 \\ +\;1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 0 \\ +\;0 \\ \hline 0 \end{array}$$

$$\begin{array}{lcrrrrrr} & & & 1 & 1 & 1 & 1 & \leftarrow \text{carried digits} \\ & (13)_{10} & & & 1 & 1 & 0 & 1 \\ + & (23)_{10} & & 1 & 0 & 1 & 1 & 1 \\ \hline & (36)_{10} & 1 & 0 & 0 & 1 & 0 & 0 \end{array}$$

$1 + 1 \rightarrow 1\;0$

$1 + 1 + 1 \rightarrow 1\;1$

Calculations in Numeral Systems

- Addition
  - Hexadecimal

					1	1		
	4	5	6			7	8	4
+	7	8	4	+		B	D	A
	B	D	A		1	B	5	E

$(8 + D)_{16} \rightarrow (8 + 13)_{10} = (21)_{2} = (15)_{16}$

  - Octal

	1	1	1	1	1	
		7	7	7	1	4
+					7	6
1	0	0	0	1	2	

$(4 + 6)_{10} \rightarrow (10)_{10} = (12)_{8}$

Calculations in Numeral Systems

- Subtraction
  - Binary

Borrowed digit → $(2)_{10} = (10)_2$

1	1	0	0
1	0	1	0
0	1	1	0

		2			
	0	0	2		← Borrowed digits
$(13)_{10}$	~~1~~	~~1~~	~~0~~	1	
$(7)_{10}$		1	1	1	
$(6)_{10}$	0	1	1	0	

Calculations in Numeral Systems

- subtraction
  - Hexadecimal

	16 + 0 = 16	16 + 3 = 19	
2	~~0~~	~~3~~	5 + 16 =21
~~3~~	~~1~~	~~4~~	5
1	9	7	6
1	7	C	F

  - Octal

3	6+ 8 = 14
~~4~~	~~6~~
	7
3	7

Signed Integer Representation

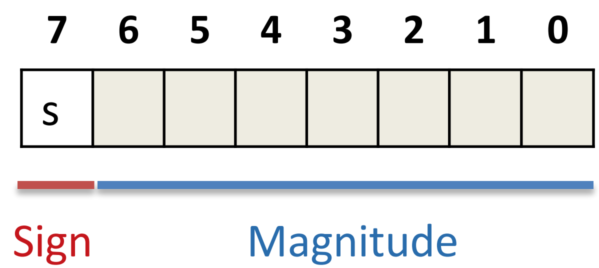
- Negative numbers must be encode in binary number systems

- Well-known methods
  - Sign-magnitude
  - One's Complement
  - Two's Complement

- Which one is better?
  - Calculation speed
  - Complexity of the computation circuit

Sign-magnitude

- One sign bit + magnitude bits
  - Positive: s = 0
  - Negative: s= 1
  - Range = {$(-127)_{10}$ .. $(+127)_{10}$}
  - Two ways to represent zero:
    - 00000000 (+0)
    - 10000000 (–0)
  - Examples:
    - $(+43)_{10}$ = 00101011
    - $(-43)_{10}$ = 10101011



- How many positive and negative integers can be represented using N bits?
  - Positive: $2^{N-1} - 1$
  - Negative: $2^{N-1} - 1$

Two's Complement

- Negative numbers:
  1. Invert all the bits through the number
     - ~(0) = 1　　　~(1) = 0
  2. Add one

- Example:
  - +1 = 00000001
  - - 1 = ?
    - ~(00000001) → 11111110
    - 11111110 + 1 → 11111111

- Only one zero (00000000)

- Range = {127 .. −128}

```
      -1    1 1 1 1 1 1 1 1
 +    -2    1 1 1 1 1 1 1 0
 ----------------------------
            1 1 1 1 1 1 0 1
```

Negative

two's complement (11111101)

~(11111101) + 1

-(00000011) = -3

ASCII Codes

- American Standard Code for Information Interchange
  - First decision: 7 bits for representation
  - Final decision: 8 bits for representation
    - 256 characters
    - ASCII (“P”) = $(50)_{16}$ ASCII (“=”) = $(3D)_{16}$

row number column number

Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	)	(	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	>	=	<	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	]	\	[	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	}	|	{	~	

Summary

- Numeral Systems
  – Decimal, Binary, Octal, Hexadecimal
- Computer Data Storage Units
  – Bit, Byte, Kilo byte, Giga byte,
- Numeral Systems Conversion
  – Convert between different bases
- Calculations in Number Systems
  – Addition and subtraction
- Signed Integer Representation
  – Sing-magnitude: one sign bit + magnitude bits
  – Two's complement : (-N) = ~(N) + 1
- Fractional and Real Numbers
- ASCII Codes