Lecture 2

NUMBER SYSTEMS

FRACTIONS AND ARITHMETIC OPERATIONS

Outline

- ❑ Part A
  - ✓ Storage Unit
  - ✓ Number Systems
  - ✓ Converting from-to Number Systems
- ❑ Part B
  - ✓ Fractions
  - ✓ Converting from-to Number Systems

Storage Unit : Main Memory

- Main memory holds information such as computer programs, numeric data, or documents
- RAM consists of many capacitors and transistors. A capacitor and a transistor are paired together to make a memory cell.
- The capacitor represents one "bit" of data, the transistor is able to change the state of the capacitor to either a 0 or a 1. the Zero's and ones when read in a sequence represent the code which the computer understands.
- Memory is divided into cells, where each cell contains 8 bits (a 1 or a 0). Eight bits is called a byte.
- Each of these cells is uniquely numbered.
- The number associated with a cell is known as its address.
- Main memory is volatile storage. That is, if power is lost, the information in main memory is lost.

Main Memory

- All addresses in memory can be accessed in the same amount of time.
- We do not have to start at address 0 and read everything until we get to the address we really want (**sequential access**).
- We can go directly to the address we want and access the data (**direct**or **random access**).
- That is why we call main memory **RAM** (**Random Access Memory**).

Secondary Storage Media

- Provides permanent storage for information
- Retains information even when power is off

Examples of secondary storage:

- Hard Disks (sequential access)
- Tapes (sequential access)
- CD-ROMs (random access)
- DVDs (random access)

This type of storage is called persistent (permanent) storage because it is non-volatile.

Bits, Bytes, and Words

- A **bit**is a single **binary digit**(a 1 or 0).
- A **byte**is 8 bits
- A **word**is 32 bits or 4 bytes
- **Long word**= 8 bytes = 64 bits
- **Quad word**= 16 bytes = 128 bits
- Programming languages use these standard number of bits when organizing data storage and access.

Bits, Bytes, and Words

- The terms 32-bit and 64-bit refer to the way a computer's processor (also called a CPU), handles information.
- The 64-bit version of Windows handles large amounts of random access memory (RAM) more effectively than a 32-bit system.
- 64bit is much faster than 32bit. -Since 64-bit systems process more information and support greater RAM.
- Windows 7 is more responsive when you are running complex applications or many applications simultaneously.
- In the computer world, 32-bit and 64-bit refer to the type of central processing unit, operating system, driver, software program, etc.

Bits, Bytes

Unit	Symbol	Number of Bytes
kilobyte	KB	$2^{10} = 1024$
megabyte	MB	2^{20} (over 1 million)
gigabyte	GB	2^{30} (over 1 billion)
terabyte	TB	2^{40} (over 1 trillion)

Number Systems

There are two types of number systems:

- Non-positional number systems
- Positional number systems

Non-positional number systems

- **Characteristics**
  - Use symbols such as I for 1, II for 2, III for 3, IIII for 4, IIIII for 5, etc
  - Each symbol represents the same value regardless of its position in the number
  - The symbols are simply added to find out the value of a particular number

- **Difficulty**
  - It is difficult to perform arithmetic with such a number system

Positional number systems

Positional number systems

- Characteristics:
- Use only a few symbols called digits
- These symbols represents different values depending on the position they occupy in the number

Positional number systems

- The value of each digit is determined by:
  1. The digit itself
  2. The position of the digit in the number
  3. The base of the number system

(**base** = total number of digits in the number system)

- The maximum value of a single digit is always equal to one less than the value of the base

The Decimal Number System

The decimal number system is a positional number system.

- Example $(5621)_{10}$:

$$\begin{array}{cccc} 5 & 6 & 2 & 1 \\ 10^3 & 10^2 & 10^1 & 10^0 \end{array}$$

$$1 \times 10^0 = 1$$
$$2 \times 10^1 = 20$$
$$6 \times 10^2 = 600$$
$$5 \times 10^3 = 5000$$

- The decimal number system is also known as base 10.
- The values of the positions are calculated by taking 10 to some power.
- Why is the base 10 for decimal numbers?

Because we use 10 digits, the digits 0 through 9.(0,1,2,3,4,5,6,7,8,9)

The Binary Number System

The binary number system is also known as base 2. The values of the positions are calculated by taking 2 to some power.

- Why is the base 2 for binary numbers?

Because we use 2 digits, the digits 0 and 1.

- The binary number system is also a positional numbering system.
- Instead of using ten digits, 0 -9, the binary system uses only two digits, 0 and 1.
- Example of a binary number and the values of the positions $(1001101)_2$:

$$\begin{array}{ccccccc} \underline{1} & \underline{0} & \underline{0} & \underline{1} & \underline{1} & \underline{0} & \underline{1} \\ 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \end{array}$$

The Octal Number System

- The octal number system is also a positional numbering system.
- The octal numeral system, or oct for short, is thebase-8 number system, and uses the digits 0 to 7 (0,1,2,3,4,5,6,7).

Example $(112)_8$:

1	1	2
8^2	8^1	8^0

The Hexadecimal Number System

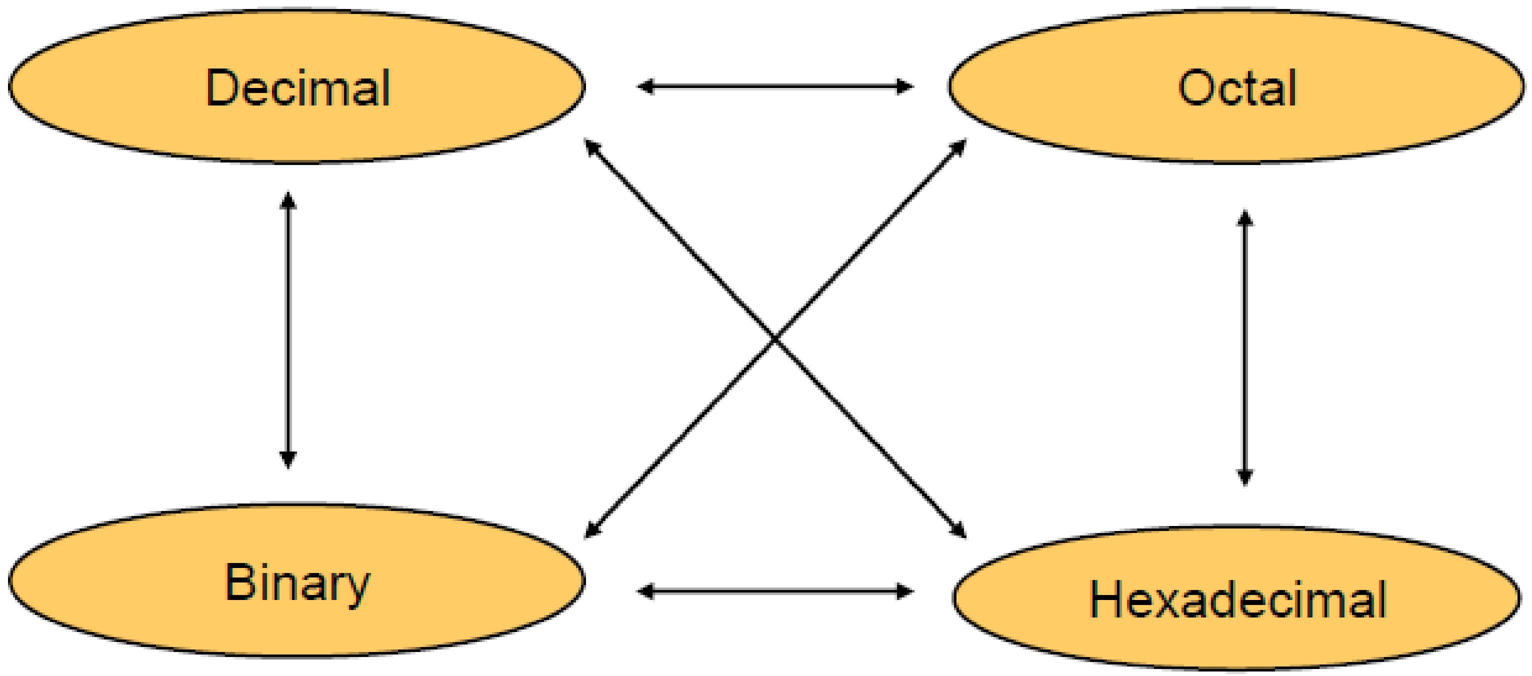
- The hexadecimal number system is also a positional numbering system.
- The hexadecimal(alsobase16, or hex) with aradix, or base, of 16.
- It uses sixteen distinct symbols, most often the symbols 0–9 to represent values zero to nine, and A, B, C, D, E, F (or alternatively a–f) to represent values ten to fifteen.

Example $(2AF3)_{16}$:

2	A	F	3
16^3	16^2	16^1	16^0

Conversion Among Bases

- The possibilities:



Decimal to Decimal

Weight

$$
\begin{aligned}
125_{10} \Rightarrow \quad & 5 \times 10^0 &= \quad & 5 \\
& 2 \times 10^1 &= \quad & 20 \\
& 1 \times 10^2 &= \quad & \underline{100} \\
& & & 125
\end{aligned}
$$

Base

Binary to Decimal

Technique

- Multiply each bit by 2^n, where n is the “weight” of the bit
- The weight is the position of the bit, starting from 0 on the right
- Add the results

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

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$

$2^4 = 16$
$2^5 = 32$
$2^6 = 64$

$1 \times 2^0 = 1$
$0 \times 2^1 = 0$
$1 \times 2^2 = 4$
$1 \times 2^3 = 8$
$0 \times 2^4 = 0$
$0 \times 2^5 = 0$
$1 \times 2^6 = \underline{64}$
77_{10}

Converting from Binary to Decimal

Converting from Binary to Decimal

$(101011)_2 = (????)_{10}$

Octal to Decimal

Technique

- Multiply each bit by 8^n, where n is the “weight” of the bit
- The weight is the position of the bit, starting from 0 on the right
- Add the results

$$724_8 \Rightarrow \begin{array}{r} 4 \times 8^0 = 4 \\ 2 \times 8^1 = 16 \\ 7 \times 8^2 = 448 \\ \hline 468_{10} \end{array}$$

Hexadecimal to Decimal

Technique

- Multiply each bit by 16^n, where n is the "weight" of the bit
- The weight is the position of the bit, starting from 0 on the right
- Add the results

$$
\begin{array}{lllllll}
ABC_{16} \Rightarrow & C \times 16^0 & = & 12 \times 1 & = & 12 \\
& B \times 16^1 & = & 11 \times 16 & = & 176 \\
& A \times 16^2 & = & 10 \times 256 & = & \underline{2560} \\
& & & & & 2748_{10}
\end{array}
$$

Decimal to Binary

Technique

- Divide by two, keep track of the remainder
- First remainder is bit 0 (LSB, least-significant bit)
- Second remainder is bit 1
- Etc.

$125_{10} = ?_2$

2	125	
2	62	1
2	31	0
2	15	1
2	7	1
2	3	1
2	1	1
	0	1

$125_{10} = 1111101_2$

Octal to Binary

Technique

- Convert each octal digit to a 3-bit equivalent binary representation

$$705_8 = ?_2$$

$$\begin{array}{ccc} 7 & 0 & 5 \\ \downarrow & \downarrow & \downarrow \\ 111 & 000 & 101 \end{array}$$

$$705_8 = 111000101_2$$

Hexadecimal to Binary

Technique

- Convert each hexadecimal digit to a 4-bit equivalent binary representation

$10AF_{16} = ?_2$

$$\begin{array}{cccc} 1 & 0 & A & F \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 0001 & 0000 & 1010 & 1111 \end{array}$$

$10AF_{16} = 0001000010101111_2$

Decimal to Octal

Technique

- Divide by 8
- Keep track of the remainder

$1234_{10} = ?_8$

8	1234
154	2
19	2
2	3
0	2

$1234_{10} = 2322_8$

Decimal to Hexadecimal

Technique

- Divide by 16
- Keep track of the remainder

$$1234_{10} = ?_{16}$$

$$\begin{array}{r|r l}
16 & 1234 & \\ \hline
16 & 77 & 2 \\ \hline
16 & 4 & 13 = D \\ \hline
 & 0 & 4
\end{array}$$

$$1234_{10} = 4D2_{16}$$

Binary to Octal

Technique

- Group bits in threes, starting on right
- Convert to octal digits

$1011010111_2 = ?_8$

$$\begin{array}{cccc} 1 & 011 & 010 & 111 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 1 & 3 & 2 & 7 \end{array}$$

$1011010111_2 = 1327_8$

Binary to Hexadecimal

Technique

- Group bits in fours, starting on right
- Convert to hexadecimal digits

$1010111011_2 = ?_{16}$

$$\begin{array}{ccc} 10 & 1011 & 1011 \\ \downarrow & \downarrow & \downarrow \\ 2 & B & B \end{array}$$

$1010111011_2 = 2BB_{16}$

Octal to Hexadecimal

Technique

- Use binary as an intermediary

$1076_8 = ?_{16}$

1	0	7	6
001	000	111	110

0010 | 0011 | 1110

2 | 3 | E

$1076_8 = 23E_{16}$

Hexadecimal to Octal

Technique

- Use binary as an intermediary

$1F0C_{16} = ?_8$

1	F	0	C
0001	1111	0000	1100

0 | 001 | 111 | 1000 | 0 1 | 100

1 7 4 1 4

$1F0C_{16} = 17414_8$

Lecture 3

FRACTIONS AND ARITHMETIC OPERATIONS

Part B

FRACTIONS

Fractions

Decimal fractions to decimal fractions

$$
\begin{aligned}
3.14 \Rightarrow \quad & 4 \times 10^{-2} = 0.04 \\
& 1 \times 10^{-1} = 0.1 \\
& 3 \times 10^{0} = \underline{3} \\
& \phantom{3 \times 10^{0} = } 3.14
\end{aligned}
$$

Fractions

The conversions of binary fractions to the decimal fractions is similar to conversion of binary numbers to decimal numbers. Here, instead of a decimal point we have a binary point. The exponential expressions (or weight of the bits) of each fractional placeholder is 2^{-1}, 2^{-2}

Fractions

Binary fractions to decimal fractions

$$
10.1011_2 \Rightarrow \quad
\begin{aligned}
1 \times 2^{-4} &= 0.0625 \\
1 \times 2^{-3} &= 0.125 \\
0 \times 2^{-2} &= 0.0 \\
1 \times 2^{-1} &= 0.5 \\
0 \times 2^{0} &= 0.0 \\
1 \times 2^{1} &= \underline{2.0\quad\quad} \\
&\ 2.6875_{10}
\end{aligned}
$$

Fractions

Octal fractions to decimal fractions

The weight of the bit of the fraction placeholder is 8^{-1}, 8^{-2},…

55.6_8=>

$$
\begin{array}{r}
6 \times 8^{-1} = 0.75 \\
5 \times 8^{0} = 5 \\
5 \times 8^{1} = \underline{40} \\
45.75_{10}
\end{array}
$$

Fractions

Hexadecimal fractions to decimal fractions

The weight of the bit of the fraction placeholder is 16^{-1}, 16^{-2},…

$2D.C_{16}$=>

$$
\begin{aligned}
12 \times 16^{-1} &= 0.75 \\
13 \times 16^{0} &= 13 \\
2 \times 16^{1} &= 32 \\
\hline
&\ \ 45.75_{10}
\end{aligned}
$$

Fractions: **Decimal fractions to binary fractions**

- **Technique:**
  A. Multiply the decimal fraction by 2.
  B. If a non-zero integer is generated, record the non-zero integer otherwise record 0.
  C. Remove the non-zero integer and repeat the above steps till the fraction value becomes 0.
  D. Write down the number according to the occurrence.

Fractions: **Decimal fractions to binary fractions**

Example:

0.75_{10}=>

$$
\begin{array}{r}
0.75 \times 2 = 1.50 \\
0.50 \times 2 = \underline{1.00\quad} \\
0.11_2
\end{array}
$$

Fractions: **Decimal fractions to binary fractions**

- **Remark-** If the conversion is not ended and still continuing; we write the approximation in 16 bits.
- Example: 0.9_{10}

Thus $(0.9)_{10} = (0.1110011001100110 01)_2$

$0.9 \times 2 = 1.8$
$0.8 \times 2 = 1.6$
$0.6 \times 2 = 1.2$
$0.2 \times 2 = 0.4$
$0.4 \times 2 = 0.8$
$0.8 \times 2 = 1.6$
$0.6 \times 2 = 1.2$
$0.2 \times 2 = 0.4$
$0.4 \times 2 = 0.8$
$0.8 \times 2 = 1.6$
$0.6 \times 2 = 1.2$
$0.2 \times 2 = 0.4$
$0.4 \times 2 = 0.8$
$0.8 \times 2 = 1.6$
$0.6 \times 2 = 1.2$
$0.2 \times 2 = 0.4$
$0.4 \times 2 = 0.8$
$0.8 \times 2 = 1.6$

Fractions: **Decimal fractions to octal fractions**

Example:

0.75_{10}

$0.75 \times 8 = 6.00$

$0.75_{10} = 0.6_8$

$(45.75)_{10} = (55.6)_8$.

8	45	**Remainder**
8	5	5
8	0	5

Fractions: **Decimal fractions to hexadecimal fractions**

Technique

we multiply the fraction by 16 instead of 2 or 8. If the non-zero integer is in between 10 to 16, then the number is represented by A to F respectively.

Example: 0.75_{10}

$0.75 \times 16 = 12.00$

$0.75_{10} = 0.C_{16}$

$(45.75)_{10} = (2D.C)_{16}$

		Remainder
16	45	
16	2	D
16	0	2

Fractions: **Octal fractions to binary fractions**

Example :

55.6_8

6->110

5->101

5->101

101101.110_2

Fractions: **Hexadecimal fractions to binary fractions**

Example:

$2D.C_{16}$

C->1100

D->1101

2->0010

00101101.1100_2

Fractions: **Binary fractions to octal fractions**

Technique:

A. Break the fraction into 3-bit sections starting from MSB to LSB.
B. In order to get a complete grouping of 3 bits, we add trailing zeros in LSB.
C. Write the 3-bit binary number to its octal equivalent.

Note- In every number system-

(a) The first bit from the right is referred as LSB (Least Significant Bit)
(b) The first bit from the left is referred as MSB (Most Significant Bit)

Fractions: **Binary fractions to octal fractions**

Example:

101101.11_2

101 101 . 110

101-5

101->5

110->6

55.6_8

Fractions: **Hexadecimal fractions to octal fractions**

Example:

$2D.C_{16}$

```
C->1100
D->1101
2->0010
```

$$00\ \underline{101}\ \underline{101}.\underline{110}\ 0$$

$$= 55.6_8$$

Fractions: **Binary fractions to hexadecimal fractions**

101101.11_2

0010 1101 . 1100

1101->13->D

0010->2

1100->12->C

$2D.C_{16}$

Fractions: **Octal fractions to hexadecimal fractions**

Example:

55.6_8

5->101

5->101

6->110

0010 1101.1100

$= 2D.C_{16}$

Arithmetic Operation using Binary

Binary Addition

Two 1-bit values

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	10

“two”

Binary Addition

Two *n*-bit values

- Add individual bits
- Propagate carries
- E.g.,

```
 1     1
  10101        21
+ 11001      + 25
 101110        46
```

Multiplication

Decimal

```
    35
 x 105
------
   175
  000
  35
------
  3675
```

Multiplication

Binary, two 1-bit values

A	B	$A \times B$
0	0	0
0	1	0
1	0	0
1	1	1

Multiplication

Binary, two *n*-bit values

- As with decimal values
- E.g.,

```
      1110
    x 1011
  --------
      1110
     1110
    0000
   1110
  --------
  10011010
```

Binary Subtraction

weight	Difference	Borrow
0 - 0	0	0
1- 1	0	0
1 - 0	1	0
0 - 1	1	1

Binary Subtraction

Examples:

$$\begin{array}{r} 10110 \\ -\ 10010 \\ \hline 00100 \end{array}$$

$$\begin{array}{rr} 2 & \\ 0\cancel{1}01 & 5 \\ -0011 & -3 \\ \hline 0010 & \end{array}$$

$$\begin{array}{rr} 111 & \\ 0\cancel{1}\cancel{1}\cancel{1}22 & 16 \\ \cancel{1}0000 & \\ \rightarrow -00011 & -3 \\ \hline 01101 & = 13 \end{array}$$

Fraction Operation: Binary addition

Examples

10.00

00.11 +

10.11

11.10 +

1.10

101.00

Fraction Operation: Binary Subtraction

Examples

$$
\begin{array}{r}
0.1010 \\
0.1000 \quad - \\
\hline
0.0010
\end{array}
$$

Fraction Operation: Binary

Binary Multiplication

```
      10.01
  x    1.01
      1001
     0000
    1001
  10.1101
```

References

- Computer Fundamental –Pradeep K. Sinha & Priti Sinha
- Introduction to Information Technologies - ITEC 1011 - YORK University

ANNOUNCEMENTS

❖ The **Lecture 2 & Lecture 3** were posted Online Facebook Group last week.

Please read them carefully.

❖**Sheet # 2** were posted online this week.

❖**Submissions of Sheet #2** is during next week's Labs

❖**Quiz # 2** will be held in the week after

Thank you