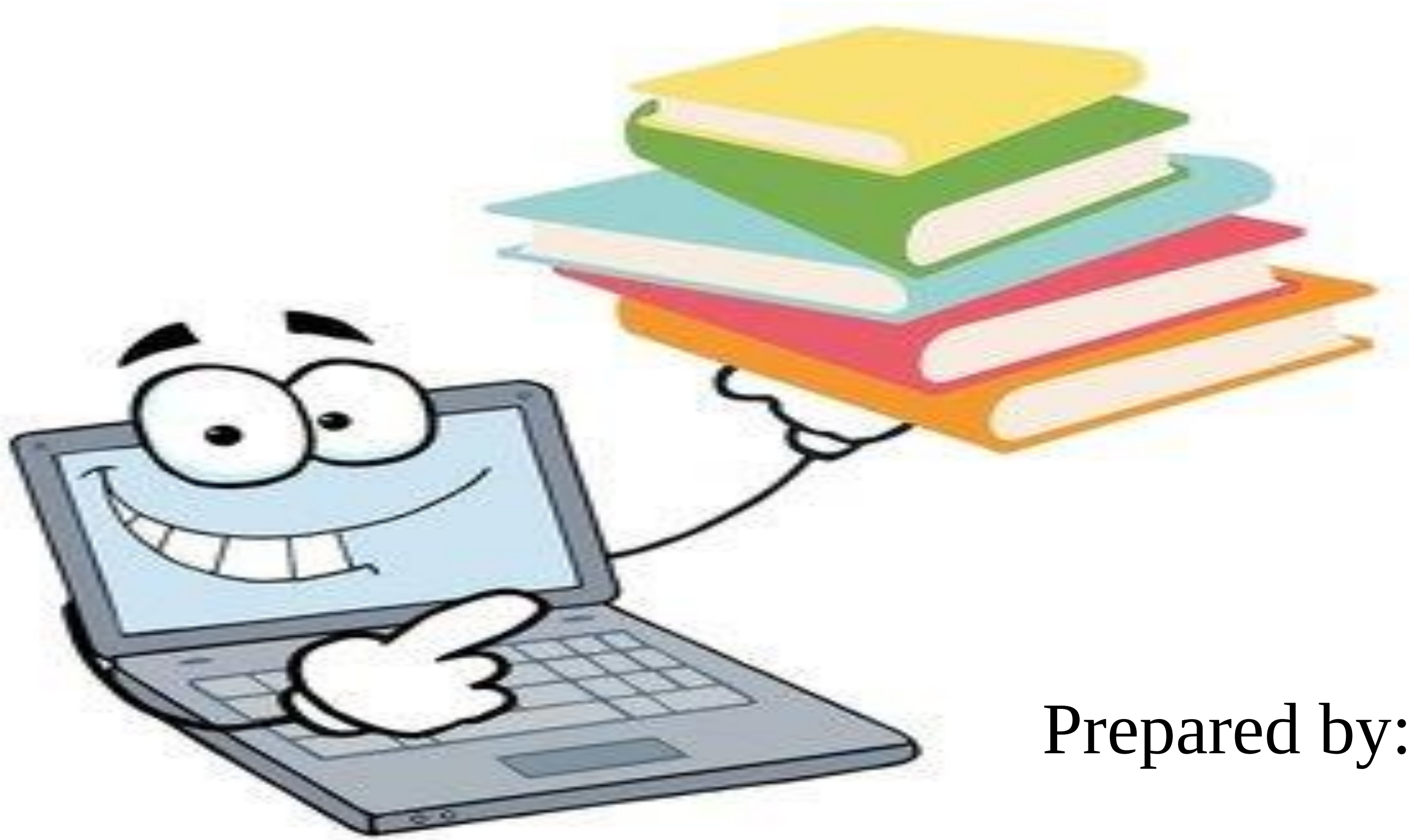
4CS015
Fundamentals of Computing



Prepared by: Uttam Acharya

A. Agenda

- Introduction to your Instructors
- Introduction to the Module
- Week 1 Lecture on Fundamentals of Computing

B. Instructors

- **Uttam Acharya (Module Leader)**
  - MSc. Physics, TU-2017
    - *uttam.acharya@heraldcollege.edu.np*
- **Rukesh Shrestha**
  - BSc (First Class Hons) Computer Science
    - *rukesh.shrestha@heraldcollege.edu.np*
- **Yojana Ghimire**
  - BSc (First Class Hons) Computer Science
    - *yojana.ghimire@heraldcollege.edu.np*

C. Module Overview

- Number and data representation
- Boolean logic and logic gates
- Arithmetic Logic Units
- Sequential logic and Memory
- CPU Architecture and Peripheral devices
- Windows and Linux OS
- Batch file for Windows OS
- Database and SQL
- Networking
- Software upgrade and Security

C. Module Overview

This module has two distinct halves.

- Weeks 1 – 6

  The first half looks at the relationships between number systems, Boolean algebra, logic gates and digital electronic circuit design.

  By the end, you should have a good understanding of *how* a computer works and *why* it works that way.

- Weeks 7 – 12

  The second half examines how Computer Scientists get the most out of computers.

  You'll look at how we work with computers that don't have 'point and click' front ends, how we communicate with other computers and how to store and access data.

D. Teaching Strategy

- Taught over 1 Semester – 12 Weeks
  - Each Week Consists of:
    - 1 Lecture: 2 hrs => Discuss the conceptual idea of the topic
    - 1 Tutorial:2 hrs =>Discuss to clear confusions
    - 1 Workshop 2.5 hrs =>Practice the concept and ideas

D. Teaching Strategy
E. Recommended Textbooks
F. Module grading system in UK

E. Recommended Textbooks:

- Computer System Architecture
  - *M. Morris Mano (Pearson, 3rd Edition)*
- Logic and Computer Design Fundamentals
  - *M. Morris Mano and C.R. Cime (Pearson,2008)*

E. Recommended Textbooks
F. Module grading system in UK
G. Any Questions?

F. Module grading system in UK

Range of Marks	Grade	Remarks
70 - 100	A	Excellent: outstanding performance with only minor errors
60 - 69	B	Very Good: above the average standard but with some errors
50 - 59	C	Good: generally sound work with a number of notable errors
43 - 49	D	Satisfactory: fair but with significant shortcomings
40 - 42	E	Sufficient: performance meets the minimum criteria
0 - 39	F	Fail: performance does not meet the minimum criteria and considerable further work is required

H. Any Questions?

1. **Lecture- 1**
2. Overview

Lecture-1 Number and Data Representation

2. Overview

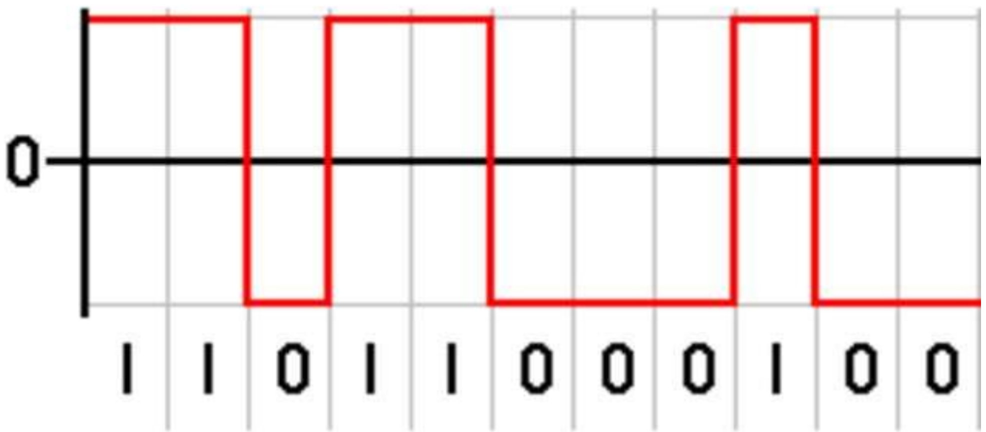
- Representation of Numbers
- Base/Radix:
  - Decimal
  - Binary
  - Octal
  - Hexadecimal
- Unsigned and Signed numbers
- Two's Complement Subtraction
- Non-integer numbers

2.1 Some Definitions

2.1.1 Analogue – Signals or information that are represented by a continuously variable physical quantity. For example, electricity, light, sound... In fact, most things we encounter in our everyday world.

Pic Ref: bbc.co.uk

2.1.2 Digital – Discrete values typically represented by values of a physical quantity. For example, electricity



Pic Ref: answers.com

2.1.3 Digital Electronics

Digital electronics takes different values (or ranges of values) of analogue signals to represent different, discrete, values.

As an example, consider a light switch

- A light switch only has two positions.
- When it is ‘off’, we would measure 0 volts at the supply to the light.
- When it is ‘on’, we would measure 240 volts at the supply to the light.
- Thus, we can represent ‘on’ and ‘off’, or ‘0’ and ‘1’, with two arbitrary analogue values; ‘0’ and ‘240’ (In the USA, they use ‘0’ and ‘110’).

2.2 Representation

- All information when stored in a computer system, must use some form of internal representation

  - The program to be executed.
  - The numerical data to be processed.
  - Character strings (A....Z etc.).

2.3 The Decimal System

- What is a Decimal Value?
- What does 331 really mean?
- **The position of** each digit is assigned a weight.
- **Decimal Scheme right-most** digit has a weight of $10^0 = 1$.
- **Weighting increases** by a factor of 10 for each new symbol as the position moves to the left.

2.3.1 Decimal Example

Again taking the value 331:

10^2	10^1	10^0
3	3	1
3 * 100	3 * 10	1 * 1
300	30	1

Value = 300 + 30 +1.

2.4 Binary

- Binary number system follows similar rules to decimal system.

    Except base is **2** not **10**.

- By the **BASE** rules:

    Symbols: 0, 1 (all symbols are unique)

    Highest Symbol = **BASE - 1**.

    BASE = 2 this is BINARY.

- Example: 11_2

2.5 RADIX or BASE

- In any number scheme the base value defines the number of unique symbols. In the Decimal scheme we use the symbols: 0,1,2,3,4,5,6,7,8,9.

- The highest symbol = **BASE –1**.

- All Symbol values are **UNIQUE**.
- Example: 11_{10}

3. The Binary Representation Scheme

- Taking a simple value: **01011010**

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

- **Can you tell what this value is?**

3.1 Basic Conversion

- Converting this representation back to a more familiar DECIMAL form we can see that all the information necessary about the data is available.

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

$$0 + 64 + 0 + 16 + 8 + 0 + 2 + 0 = 90$$

3.2 Decimal To Binary Conversion

- Rules:
  - Divide the number by 2.
  - Look at the remainder ‘1’ or ‘0’.
  - Use the remainder as Bit value.
  - Loop till done.
- Convert 25_{10}

22

3.2 Decimal to Binary Conversion
3.2.1 Example

3.2.1 Example: Conversion of 25_{10}

Divisor	Quotient	Remainder
2	25	1 LSB
2	12	0
2	6	0
2	3	1
2	1	1 MSB
	0	

- Binary Value = 11001_2

3.3 Binary To Decimal Conversion

- Calculate the value for each Digit Position as **(Symbol * Base Weighting)** for the position. Then sum all these values.
  - **Binary** 1110_2

$= (1 * 2^3) + (1 * 2^2) + (1 * 2^1) + (0 * 2^0)$

$= (1 * 8) + (1 * 4) + (1 * 2) + (0 * 1)$

$= 14_{10}$

3.4 Most Significant Digit / Least Significant Digit

- The concepts of the significance of the Symbol positions is not unique to binary.
  - Units, Tens, Hundreds etc. In the decimal scheme.
- Most Significant Digit is the **Left - most** symbol of the Number.
- Least Significant Digit is the **Right – most** symbol of the Number.
- Example: 11001_2

4. 'Bits', 'Bytes' and 'Words'

- Within a computer system EVERYTHING is stored or used in binary format.
- Each **Bi**nary Digi**t** is commonly known as a **Bit.**
- Within any computer system information is stored or manipulated in groups of **‘n Bits’**.

4. 'Bits', 'Bytes' and 'Words'

- **A Byte is** the smallest grouping and it is defined to contain **'n' Bits**. By common usage, **the Byte** usually refers to a grouping of **8** Bits.
- **A Word is** a larger grouping **of Bits**.
- Size reflected by a PC's architecture.
- Again by common usage, a **Word** is a grouping of **16 Bits** and therefore is **2 Bytes**.
- We also have **Nibbles**
  - **Which are?**

5.Unsigned Numbers

- Any number range is defined as:
  - **0** $\rightarrow$ **Max Number.**
- In the Decimal scheme, if we write a value as 27:
  - The **magnitude** of the value is 27.
  - By convention the value is Positive.
- On the other hand if we write the value as: -27
  - It is not the same since the **'-' SIGN** identifies it as another value.

5.1 Unsigned Binary Representation

- The Binary representation in an unsigned form has only a **MAGNITUDE.**
- All binary combinations of the **"n" Bits** will uniquely identify a magnitude
  - e.g. 0000 1111.
  - Range 0 to 2^n -1
- So program variable declarations can define the size and type of storage used!

5.2 Kilo - Mega - Giga

- Within **Binary** these terms have slightly different meaning to those in **Decimal.**
- Kilo = 10^3 = 1000 in Decimal.
- Kilo = 2^{10} = 1024 in Binary.
- Mega = Kilo * Kilo = (Kilo^2).
  - 1,048,576 Binary / 1,000,000 Decimal.
  - Decimal: Mega = 10^6 / Binary: Mega = 2^{20}.
- Giga = Kilo * Mega.

2^0	1
2^1	2
2^2	4
2^3	8
..	..
2^9	512
2^{10}	1K
2^{11}	2K
2^{20}	1M
2^{30}	1G
2^{40}	1T

5.3 Unsigned Numbers Table

Look at this table of 16 values we need **‘4’ Bits’** to represent all possible values.

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

Decimal	Binary
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

6. Signed Numbers

- The Decimal number system uses ‘-‘ *(minus)*
  - In Binary we only have two symbols 1 and 0.
- But we must be able to store signed numbers in computer memory!
- One possible solution - *signed magnitude*

00010111 = +23

sign — magnitude

10010111 = -23

sign — magnitude

6. Signed Numbers

- What effect does this have on the size of integers a computer can use?

- 8 bit *unsigned* binary number

  00000000 - 11111111

  0 - 255_{10}

- 8 bit *signed* binary number

  01111111 - 11111111

  $+127_{10}$ - -127_{10}

  Range $(2^{n-1} -1)$ to $2^{n-1} -1$

  but we have 00000000 and 10000000 Is this a problem?

6. Signed Numbers
7. Radix Complement

7. Radix complement

- 2's complement
  - Variation of sign magnitude but has some very useful characteristics.
- Subtraction becomes complementary addition.
- Convert -ve numbers to 2's complement and then add.
  - Get subtraction through adding a negative.

    30 + (-30) = 0

7.1 Subtraction and Two's Complement

- To create the 2's complement of a negative number.
  1. Invert all '1's to '0' and '0's to '1'
  2. Add 1.
- Each operand must have same number of bits.
- Example 25 - 7

  25 = 00011001

  7 = 00000111

7.1.1 Subtraction and 2’s Comp

- First convert -7 to 2's complement

- Step 1 : form 1's complement

  00000111
  11111000 1's complement

- Step 2 : add 1

  11111000
  +1
  ―――――――
  11111001 -7 in 2's complement representation

7.1.1 Subtraction and 2's Comp

- Now add

  25 = 00011001
  -7 = <u>11111001</u>
  Ans = 00010010 18_{10}

- Check answer.

7.1.2 2's complement answers.

- ONLY CONVERT NEGATIVE NUMBERS!
- When working out the real answer of a 2's complement sum.
  - 1. Look at the most significant bit.
  - 2. If it is a 1 then
    - The answer is negative
    - Change all ones to zero, all zeros to one and add one.
    - Convert this to decimal and write the sign.

Example

- Which of the following numbers are correctly converted to 8 bit 2’s complement?

- -33_{10} = 00011111_2
- -30_8 = 10011000
- $\boxed{30_{10} = 00011110}$
- I really don’t know

7.2 2's Comp and Sign Magnitude

7.2 2's Comp and Sign Magnitude

Binary	S/M	2's Comp
00000000	0	0
00000001	1	1
00000010	2	2
…..	…..	…..
01111110	126	126
01111111	127	127
10000000	-0	-128
10000001	-1	-127
…..	…..	…..
11111110	-126	-2
11111111	-127	-1

7.2 2‘s Comp and Sign Magnitude

- Number range for 2's complement 8 and 16 bit integers?

  -128 to +127 8 bit

  -32768 to +32767 16 bit

  Range $-(2^{n-1})$ to $+ 2^{n-1} -1$

- Number range for the sign magnitude representation for 8 and 16 bit integers?

  -127 to +127 8 bit

  -32767 to +32767 16 bit

  Range $-(2^{n-1} -1)$ to $+ 2^{n-1} -1$

8. Hexadecimal (Hex)

- Hexadecimal (Hex) number system is base 16.
- 16 unique symbols.
- Symbols: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.
- **A** = 10_{10}, **B** = 11_{10}, **C** = 12_{10},
  D = 13_{10}, **E** = 14_{10}, **F** = 15_{10}.
- Example: $3E8_{16} = 1000_{10}$.

8. Hexadecimal (Hex)

- Weight is 16 raised to the power of the position.
- Example: $A1F_{16}$

$(A \times 16^2)$ +	(1×16^1) +	$(F \times 16^0)$	
$16^2 = 256$	16	1	
10 x 256 +	1 x 16	15 x 1	$=2591_{10}$

8.1 Conversion of Hexadecimal to / from Decimal

- **Hex $9AC_{16}$**

  $(9 * 16^2) + (A * 16^1) + (C * 16^0) = 2476$

  $2304 + 160 + 12 = 2476$

- **1371_{10}**

Divisor	Quotient	Remainder
16	1371	11 =B
16	85	5
16	5	5
	0	

- **Result = $55B_{16}$**

8.2 Binary to Hexadecimal Conversion

- Comparison of the binary and hexadecimal number systems.
- What do you notice?

Binary	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

8.2 Binary to Hexadecimal Conversion

- A single Hex digit can represent a 4-bit binary number.
- Example: $(110101111)_2$

0001	1010	1111
1	A	F

$$(110101111)_2 = (1AF)_{16}$$

Binary	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

8.3 Binary & Hex Conversion

- Consider the value:

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

- Split this into groupings of 4 Bits
- This gives the Hex – $5B_{16}$
  - **Hex is simply groupings of 4 Binary Bits.**

8.4 Making Binary Easier for Humans

- **Why was Hexadecimal chosen?**
  - Single hex digit = four bits.
  - 2 hex digits per **byte**.
  - Allows simple conversion to and from binary.
  - Simpler to convert between Base 2 and Base 10.
  - The meaning of a value can be communicated simply and effectively, both at the computer (i.e. Base 2) level and also so that humans (i.e. Base 10) can visualise the meaning.
    - Hex is easier for humans to understand and communicate.
    - Example: $A_{16} = 1010_2 = 10_{10}$.

9. OCTAL Radix 8

- The only other commonly used number base is 8 or Octal.
- Why use octal?
  - Direct conversion to binary
- How would you represent 23 decimal in octal?
  - Hint 23_{10} is 00010111_2 in 8 bit binary.
  - The answer is : 027

9.OCTAL Radix 8
10. Non Integer Numbers

10. Non Integer Numbers

- Can we always use **WHOLE** values?
- It would be nice if we could.
  - Bank balances might be interesting.
  - We need to look at what are called Floating Point numbers.
- Values such as **3.142**, etc.
  - Numbers with a decimal point!
  - Data types such as double (in Java and C).

10.1 Scientific Notation

- This is a way of expressing very large and very small numbers.
- **537,000,000** is written **5.37×10^8**
  - 5.37 is called the Mantissa.
  - 10^8 is called the Exponent.

- **0.000136** is written **1.36×10^{-4}**
  - 1.36 is called the Mantissa.
  - 10^{-4} is called the Exponent.

10.2 Scientific Notation: Floating Point Numbers

- Note that a **positive** exponent tells you how many places to **move** the decimal point to the **right**, making the **number. bigger**.
- A **negative** Exponent is how many places to **move** the decimal point to the **left**, making the **number smaller**.

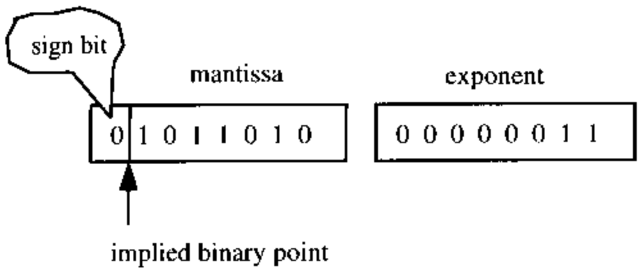
$1.005 \times 10^{2} = 100.5$ $1.005 \times 10^{-2} = 0.01005$

$1.005 \times 10^{0} = 1.005$ $1.005 \times 10^{4} = 10,050$

- These are called Floating Point numbers.

10.3 Binary: Floating Point Format

- Expressing a number as a Mantissa and an Exponent allows us to store Floating Point numbers in a format that only requires whole numbers.
- Sign = 1 bit.
- Mantissa = 7 bits.
- Exponent = 8 bit signed magnitude.
- Total = 16 bits.



5.625_{10}
$+101.1010_2$
1.011010×2^2

10.4 Floating Point in Decimal

What does 3.142_{10} actually mean?

10^0	Point	10^{-1}	10^{-2}	10^{-3}
1		1/10	1/100	1/1000
3	**.**	**1**	**4**	**2**

3.142 = (3 * 1) + (1 * 1/10) + (1 * 4/100) + (1 * 2/1000)

= 3 + 0.1 + 0.04 + 0.002

10.5 Floating Point in Binary

What is 1.101_2?

2^0	Point	2^{-1}	2^{-2}	2^{-3}
1		1/2	1/4	1/8
1	.	1	0	1

1.101_2 = (1 * 1) + (1 * 1/2) + 0 + (1 * 1/8)

1 + 0.5 + 0 + 0.125

= 1.625_{10}

10.6 Limitation in Binary

- The Columns as we go RIGHT after the POINT have the place values.

1/2	**1/4**	**1/8**	**1/16**	**1/32**
0.5	**0.25**	**0.125**	**0.0625**	**0.03125**

- When converting Floating Point decimal numbers to binary using a fixed number of places.
- **NOT ALL VALUES ARE POSSIBLE!**

10.7 Converting Decimal to Binary

- Use continuous **multiplication** by 2.
- Each 1 or 0 to the **left** of the point of the result contributes to the binary number.
- Continue until the fractional part is 0.
- Or until you obtain the answer to the accuracy you need.

10.7.1 Example: Decimal to Binary

Take the Decimal Fraction: 0.35

0.35	**x**	**2**	**0.70**	**0**	MSB
0.70	**x**	**2**	**1.4**	**1**	
0.4	**x**	**2**	**0.8**	**0**	
0.8	**x**	**2**	**1.6**	**1**	
0.6	**x**	**2**	**1.2**	**1**	LSB
0.2	**x**	**2**	**0.4**	**0**	

Answer: 0.35 = 0.010110 …and so on

and so on

10.7.2 How close is this answer to 0.35?

- 0.010110_2

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

$0*2^0$	0
$1* 2^{-2} = 1* 0.25$	0.25
$1* 2^{-4} = 1* 0.0625$	0.0625
$1* 2^{-5} = 1* 0.0625$	0.03125
	0.34375

10.7.1 Example: Decimal to Binary
10.7.2 How close is this answer to 0.35?
10.8 Accuracy of Floating Point Data

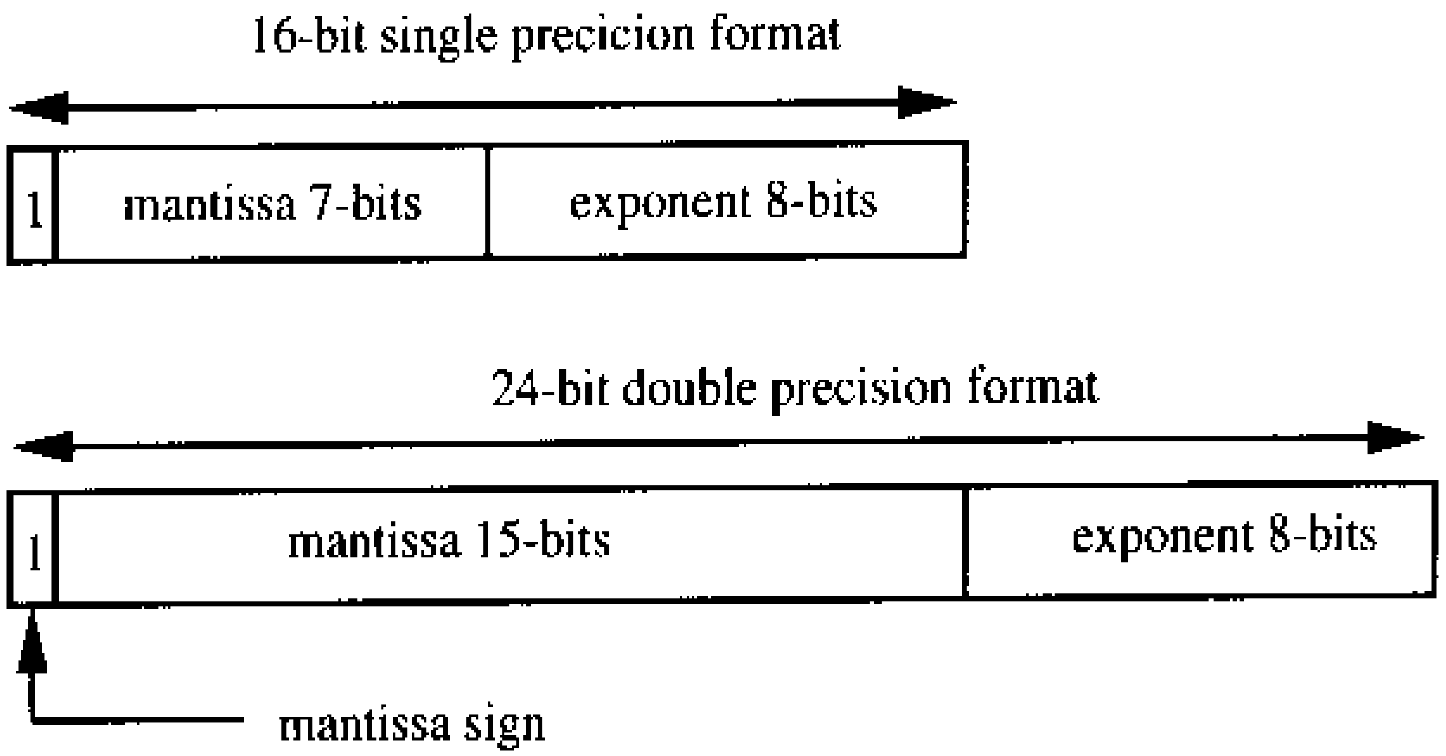
10.8 Accuracy of Floating Point Data

- Accuracy defined as **closeness to true value**.
- Data is inputted to a program in Decimal floating point format (e.g. via the keyboard).
- It is converted to a binary floating point format of fixed size.
- This may involve some loss of accuracy.
- Floating Point arithmetic is only approximate.
- Integer arithmetic is exact.

11.1 Precision

- Determined by the number of bits used to store the **Mantissa**.
- A high precision format will result in more accurate floating point arithmetic.
- Most languages provide a choice:
  - **Single precision.**
  - **Double precision.**

11.3 Formats That have been used



11.4 IEEE Format now Used

- IEEE 754 floating point standard.
- Single Precision (32 bits):
  - Mantissa:
    - 1 **Bit** for Sign / 23 **Bits** for Value.
  - Exponent:
    - 8 **Bits.**
- Double Precision (64 bits):
  - Mantissa:
    - 1 **Bit** for Sign / 52 **Bits** for Value.
  - Exponent:
    - 11 **Bits.**

12. Summary

- Representation of Numbers
- Base/Radix: Hex and Octal
- Non-integer numbers and Floating Point numbers.


Thank you!!