Lecture 1

Introduction to Computers

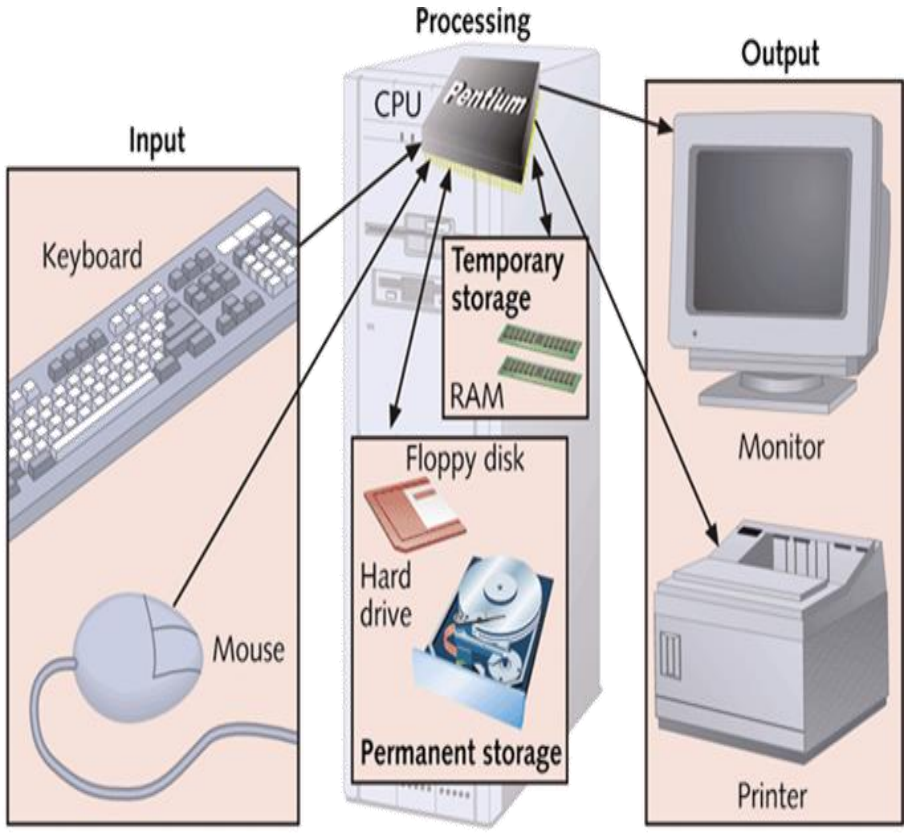
Communication Engineering Dept.
College of Electronics Engieneering
Ninevah University
2020/21

What Is A Computer?

A computer is an electronic device,

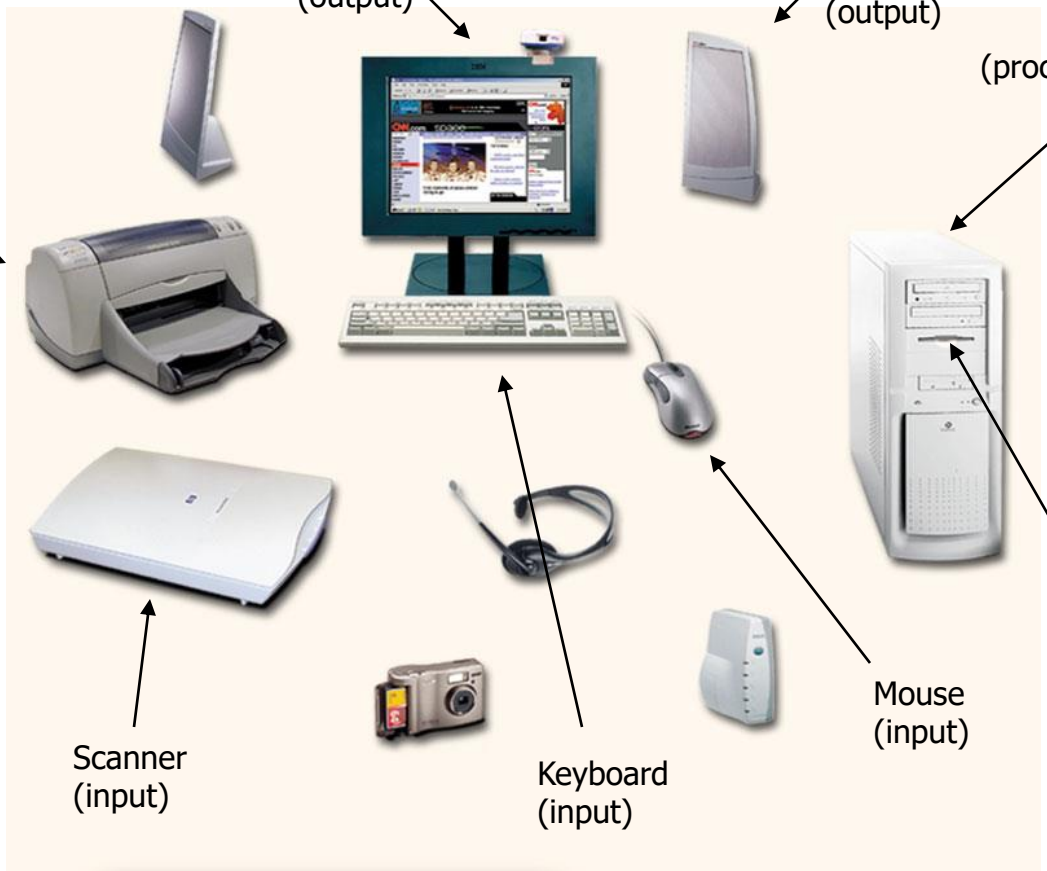
operating under the control of instructions (software) stored in its own memory unit, that can accept data (input), manipulate data (process), and produce information (output) from the processing.

Computer System: collection of devices that operate together.



Computer activity consists of input, processing, storage, and output

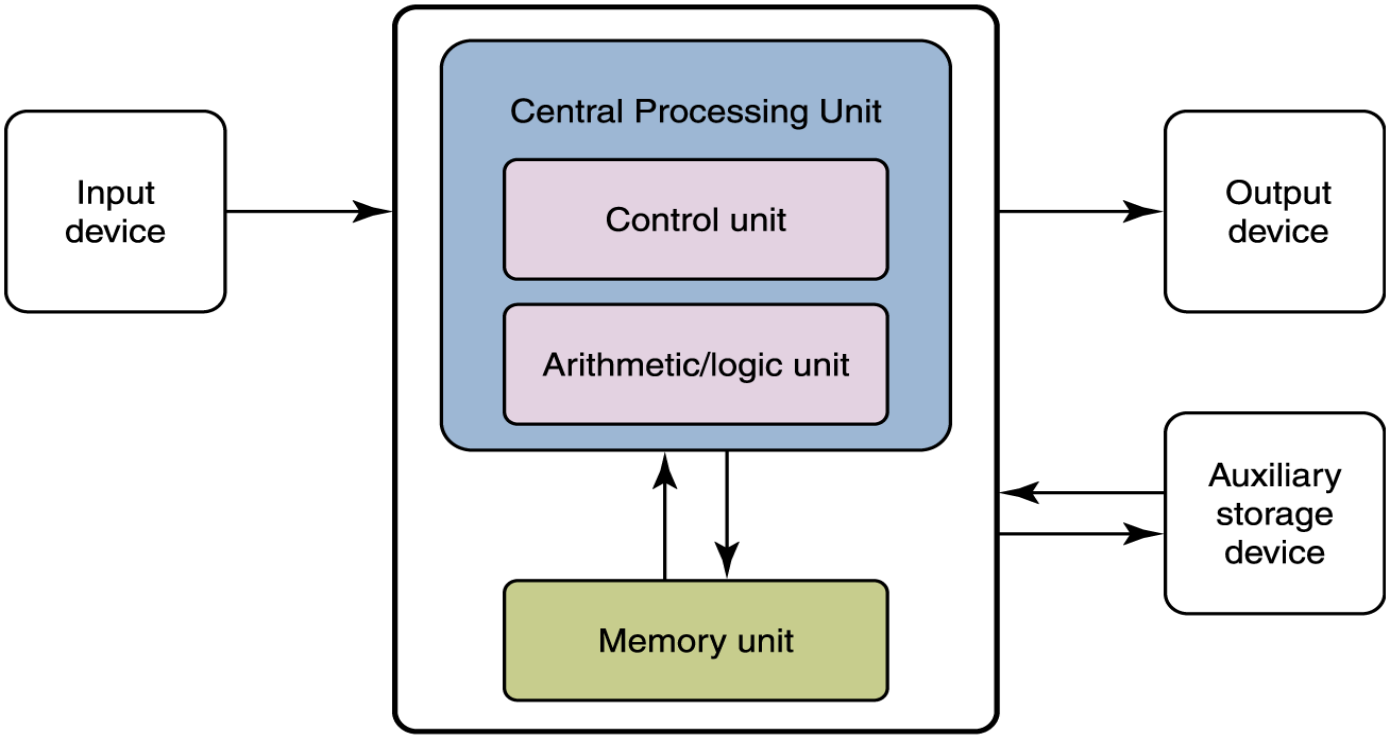
Devices that comprise a computer system



What Does A Computer Do?

Computers can perform four general operations, which comprise the information processing cycle.

- Input
- Process
- Output
- Storage



Why Is A Computer So Powerful?

- The ability to perform the information processing cycle with amazing speed.
- Reliability (low failure rate).
- Accuracy.
- Ability to store huge amounts of data and information.
- Ability to communicate with other computers.

Data Representation

Bit - the smallest unit of data that a computer uses. It can be used to represent two states of information, such as Yes or No.

Byte - is equal to 8 Bits. A Byte can represent 256 states of information, for example, numbers or a combination of numbers and letters. 1 Byte could be equal to one character.

Bits on Bytes

1 byte = 8 bits

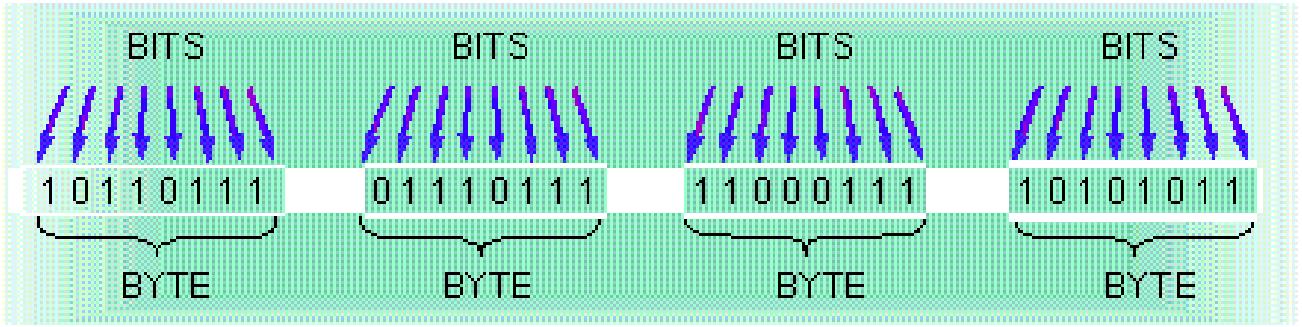
1 kilobyte (K / KB) = 1,024 bytes

1 megabyte (M / MB) = 1,048,576 bytes (1024KB)

1 gigabyte (G / GB) = 1,073,741,824 bytes (1024 MB)

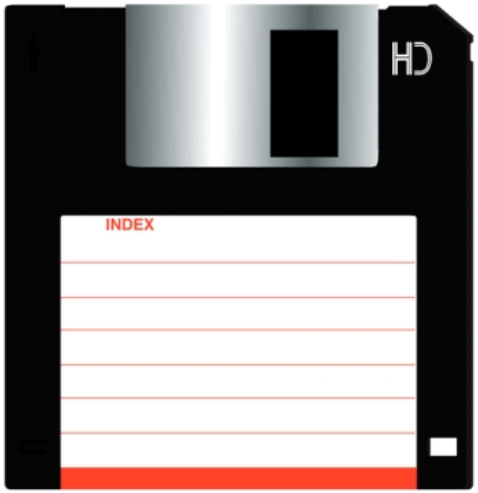
1 terabyte (T / TB) = 1,099,511,627,776 bytes (1024 GB)

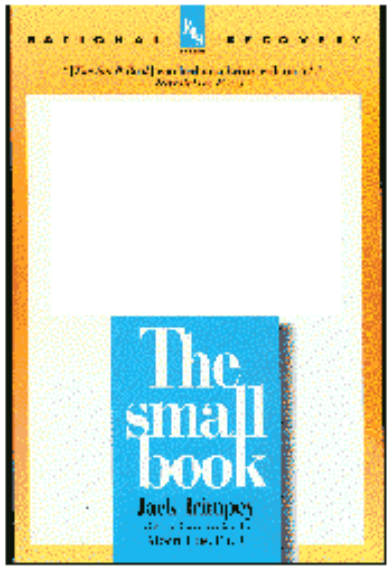
- Information is moved in bytes, or multiple bytes called words.
  - **Words** are the fundamental units of information.
  - The number of bits per word may vary per computer.
  - A word length for most computers is 32 bits:



- Fact 1: Megabyte

  One of those old 3-1/2 inch floppy disks can hold 1.44 Megabytes or the equivalent of a small book.

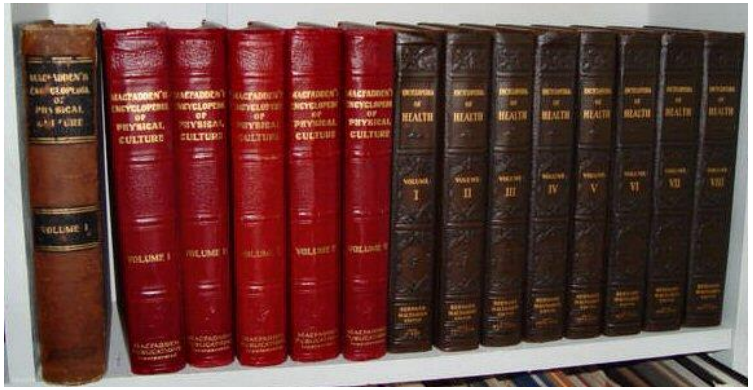




650 Megabytes is about the amount of data that will fit on a CD-ROM disk.

- Fact 2: Megabyte

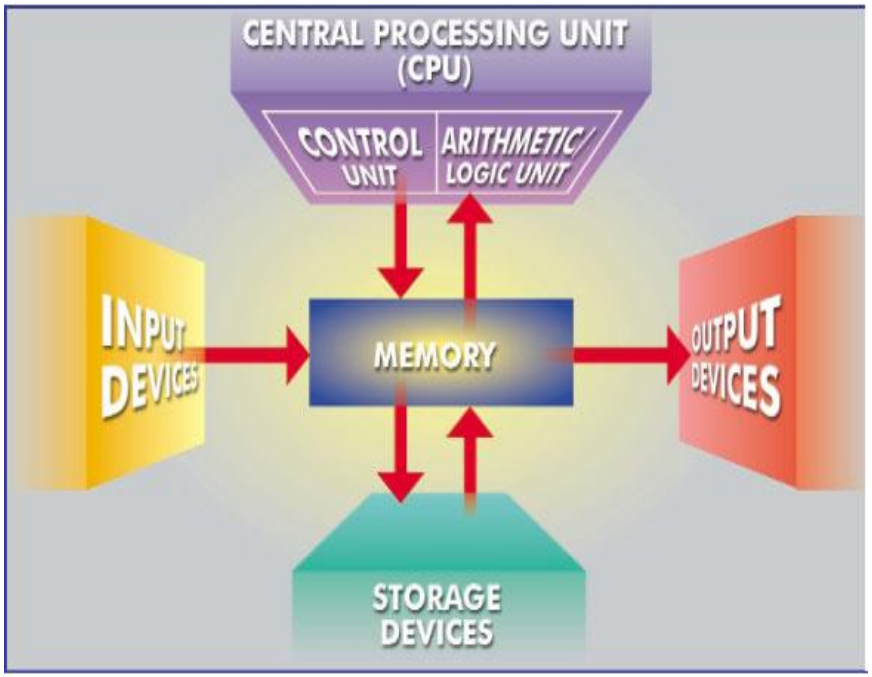
  650 Megabytes might hold a couple volumes of Encyclopedias.



× 6 =

What Are The Primary Components Of A Computer ?

- Input devices.
- Central Processing Unit (containing the control unit and the arithmetic/logic unit).
- Memory.
- Output devices.
- Storage devices.



Input Devices

- **Input** - getting data into the computer

Input Devices

– enable users to get data into the computer for processing

- Keyboard.
- Mouse.

Output Devices

Output devices make the information resulting from the processing available for use. The two output devices more commonly used are the **printer** and the computer **screen**.

The printer produces a hard copy of your output, and the computer screen produces a soft copy of your output.

A Look Inside

A Look Inside…

- Identify all the major components:
  - Power Supply
  - Motherboard
  - Memory
  - Card Slots
  - Cards (sound, video, network)
  - CPU, heatsink and fan
  - Drives (floppy, hard and CD-ROM)

What these components do.

- Power Supply – supplies DC power to all circuits and devices in computer system.
- Motherboard – acts as a manager for everything on the computer – connects all the other components together.
- CPU – Central Processing Unit – Represent the brain--this does all the work of computing.

What these components do..

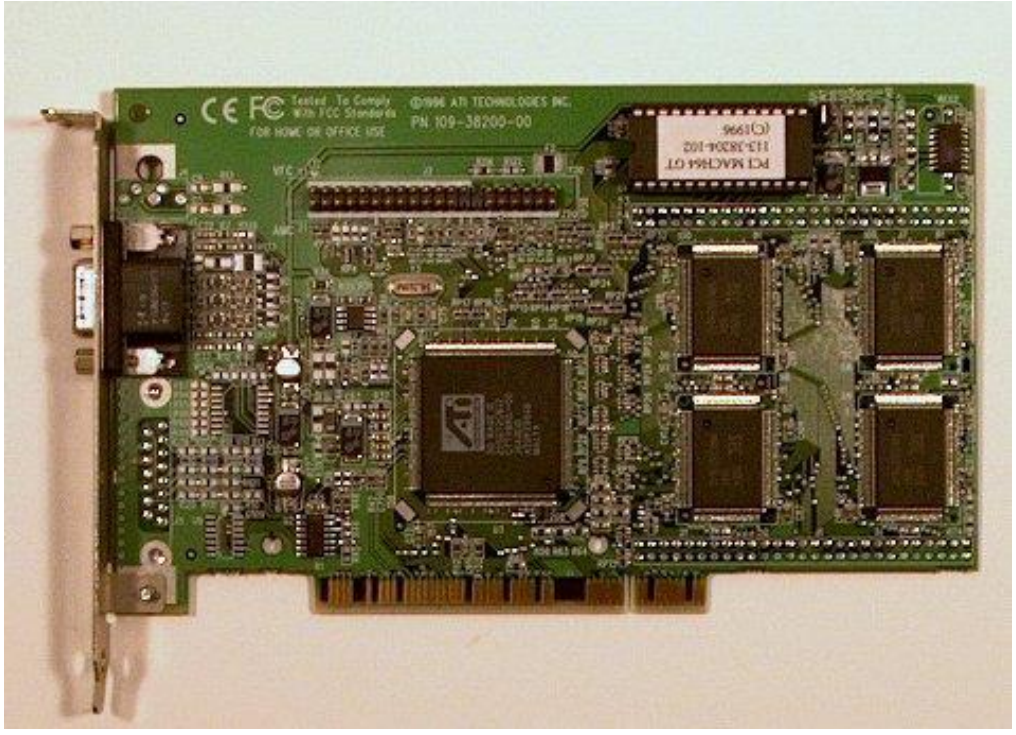
- RAM – Random Access Memory holds data and program instructions that the computer is currently using.
- Hard Drive – holds all types of data that needs to be stored between uses of the computer.

Floppy and CD-ROM drives – store data on portable media and allow to give data to the computer (Read) or take data away from the computer (Write).

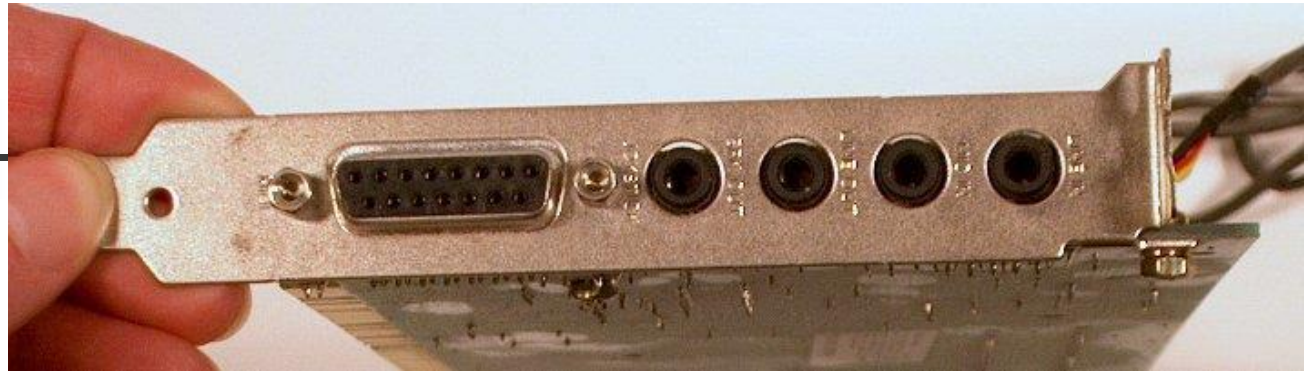
What these components do..

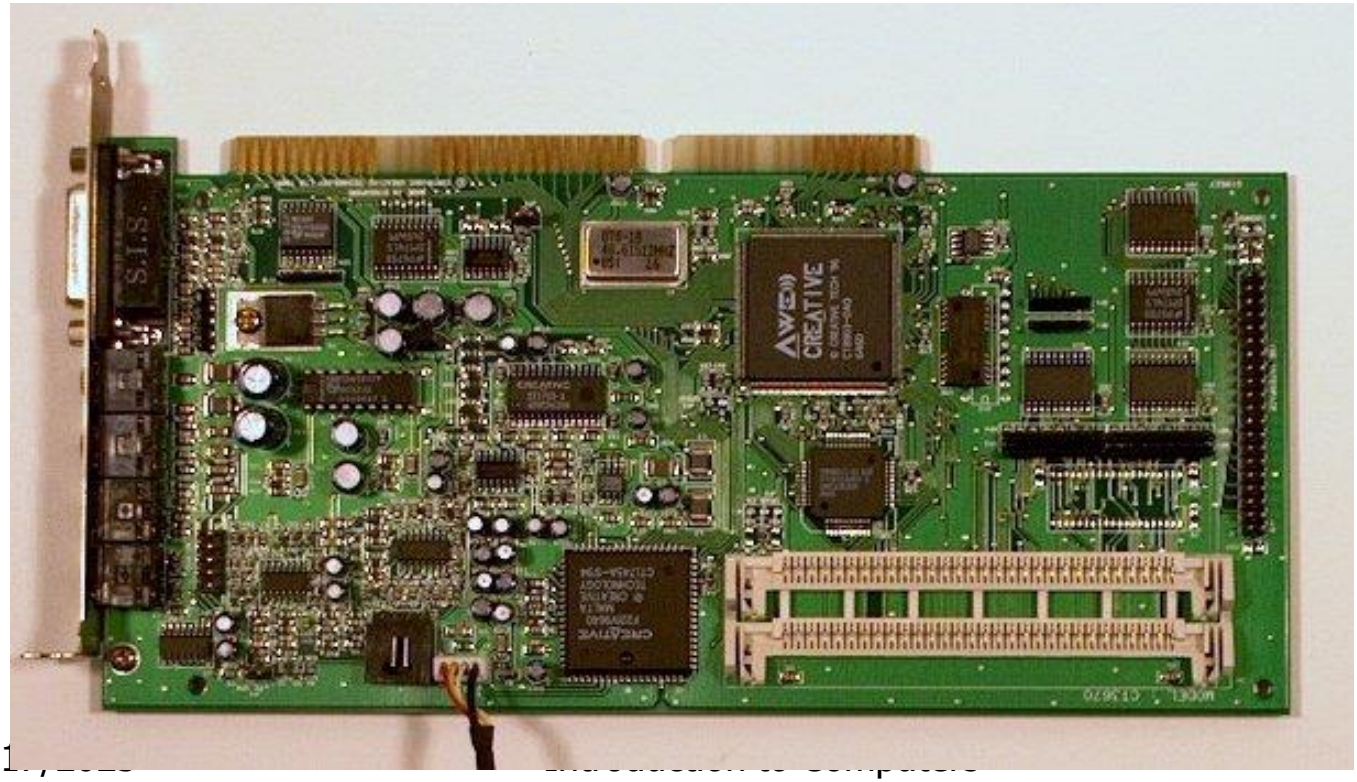
- Card Slots – Allows other components (Cards) to be added to the computer.
- Video card – Increase the processing of data (photo and videos) (media) quickly.
- Sound card – Increase the processing of data audios quickly.
- Network Card – allows computer to connect to other computers over a wire or wireless.

Video Card

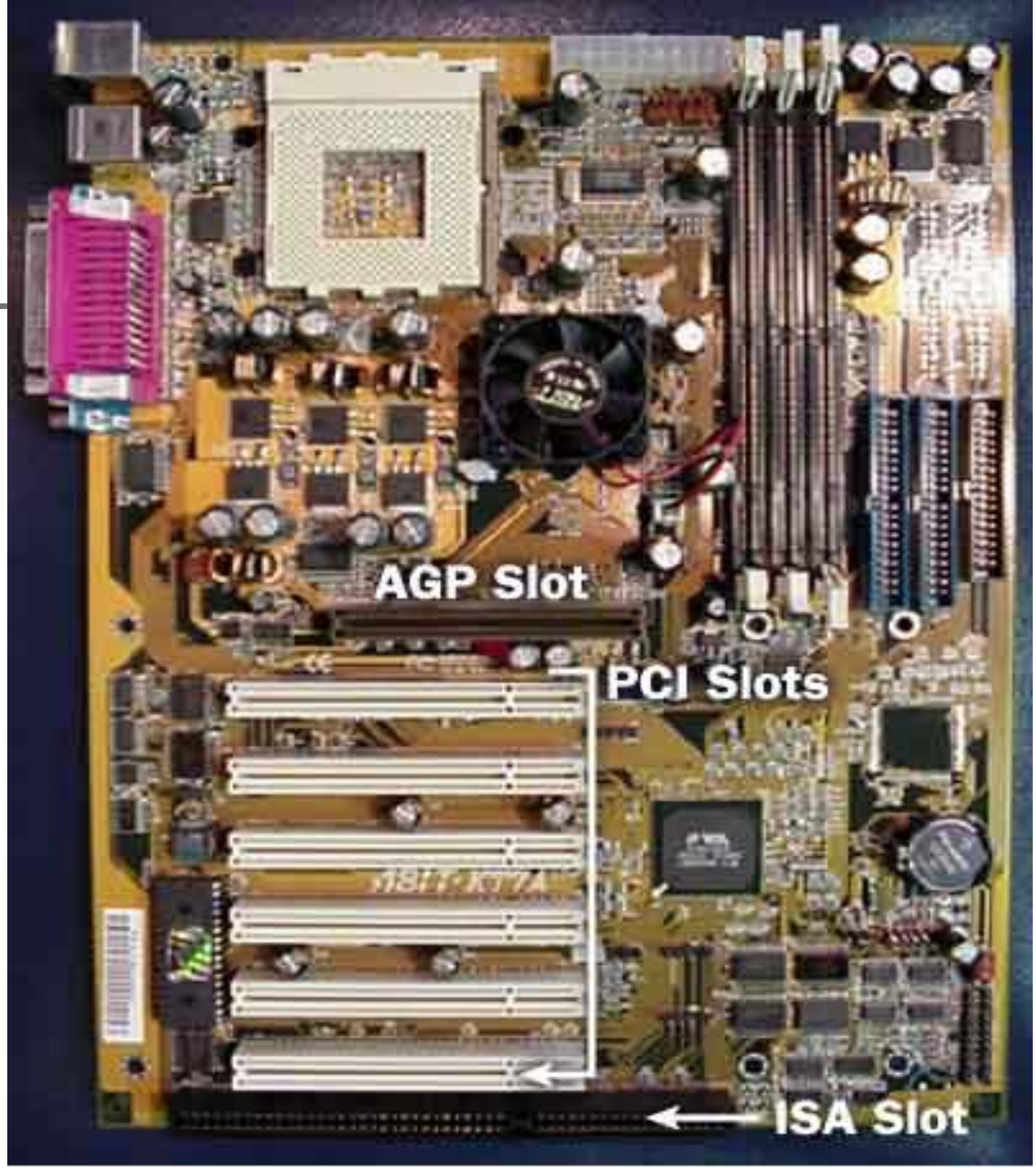

Tested To Comply With FCC Standards
FOR HOME OR OFFICE USE
©1996 ATI TECHNOLOGIES INC.
P/N 109-38200-00

Sound Card



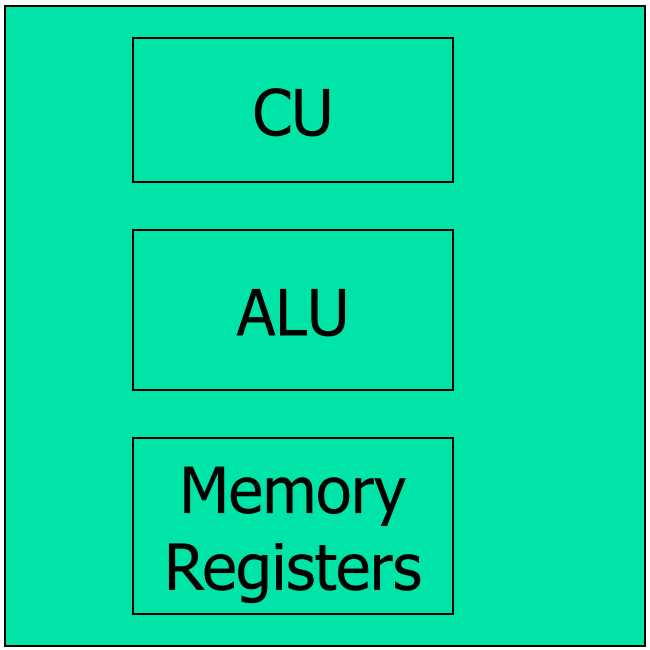


Motherboard



CPU

- **Processing** - transforming data into information



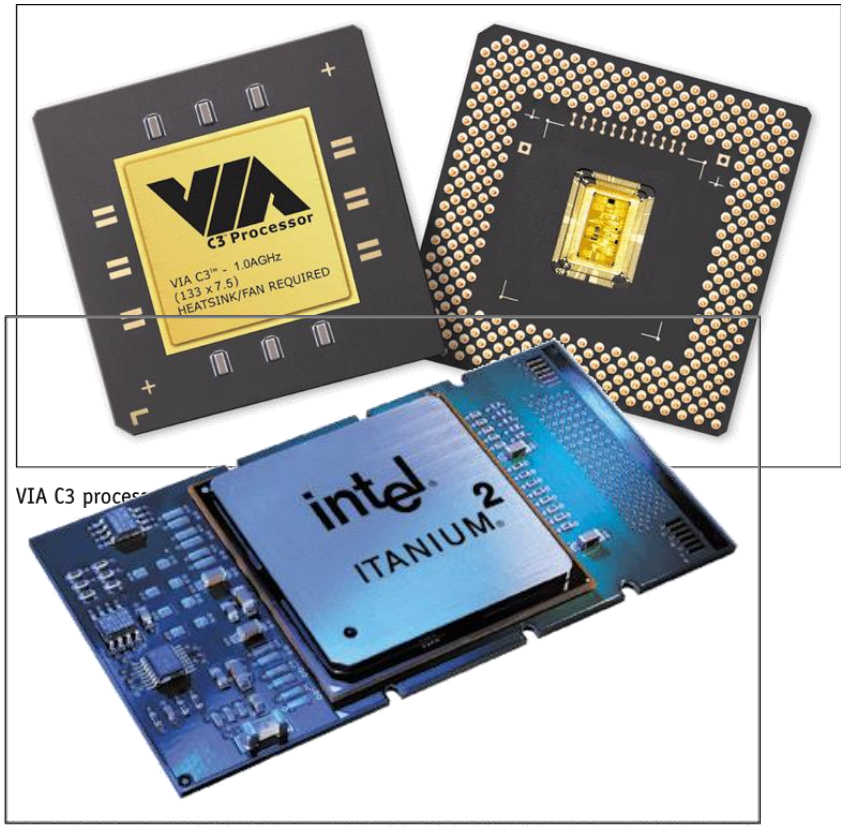


Figure 5-8 VIA C3 proces

Figure 5-9 The Itanium 2

The Central processing Unit

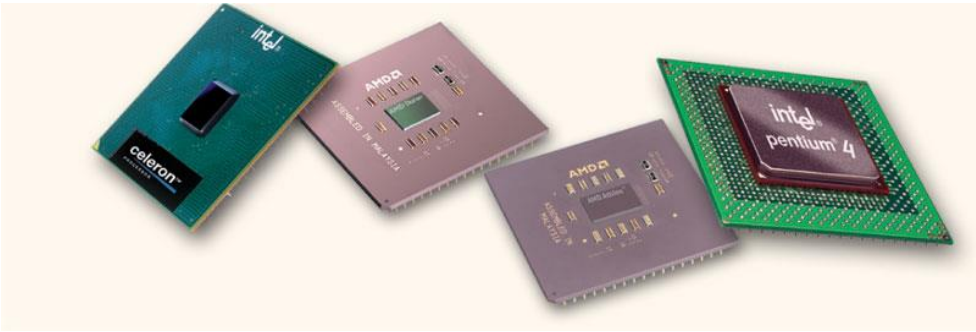
The central processing unit (CPU) contains electronic circuits of millions transistor for processing the data.

1-The CPU interprets instructions to the computer.

2-Performs the logical and arithmetic processing operations.

3-Causes the input and output operations to occur.

4-It is considered the “brain” of the computer.



Memory

RAM Memory, Random Access Memory or (temporary memory) is the main memory of the computer.

1-It consists of electronic components that store data including numbers, letters of the alphabet, graphics and sound.

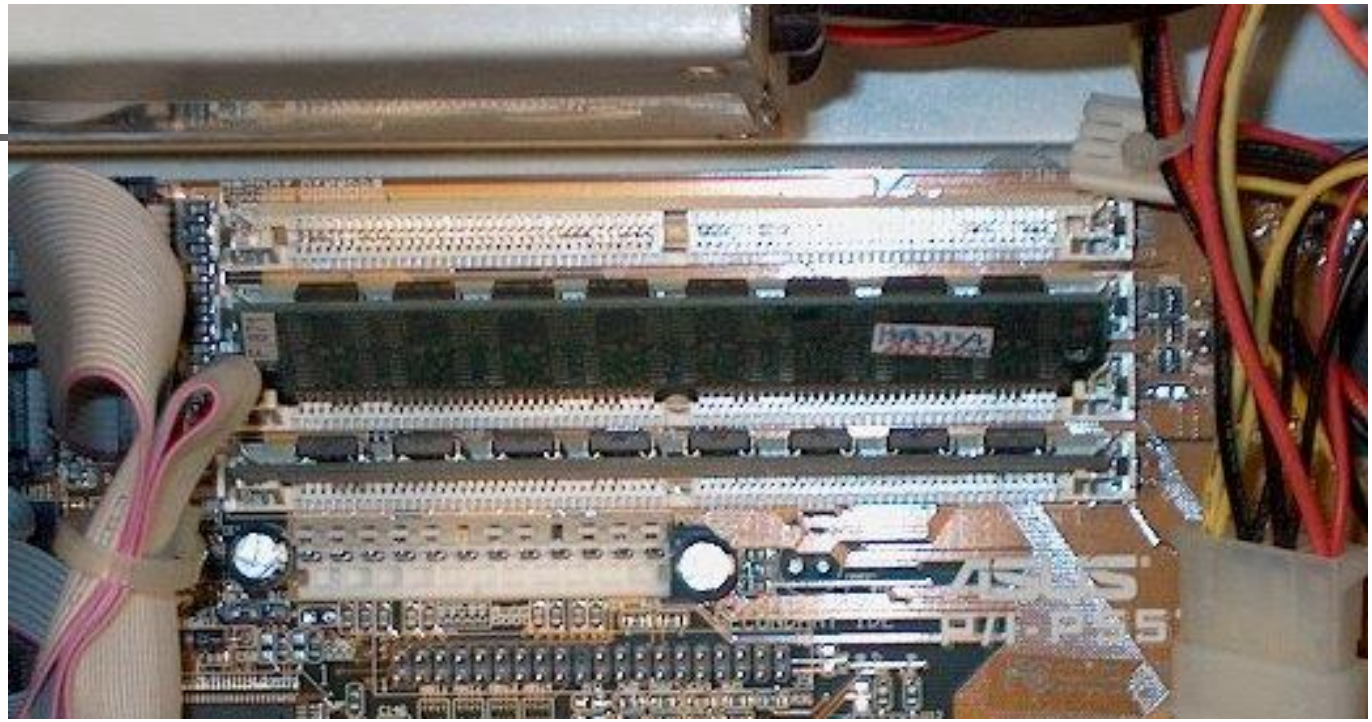
2- Any information stored in RAM is lost when the computer is turned off.

ROM Memory, Read Only Memory is permanent memory that has start-up software for direction the computer.

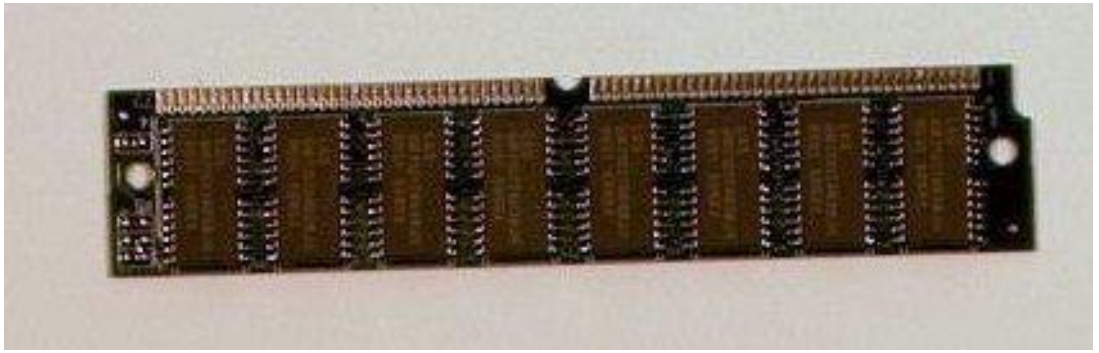
CASH Memory, built in CPU to support high speed of CPU for processing data.

RAM Module

RAM Slot



RAM Module



Storage Devices

Auxiliary storage devices are used to store data when they are not being used in memory.

The most common types of auxiliary storage used on personal computers are:

floppy disks,

hard disks

CD-ROM drives.

Floppy Disks

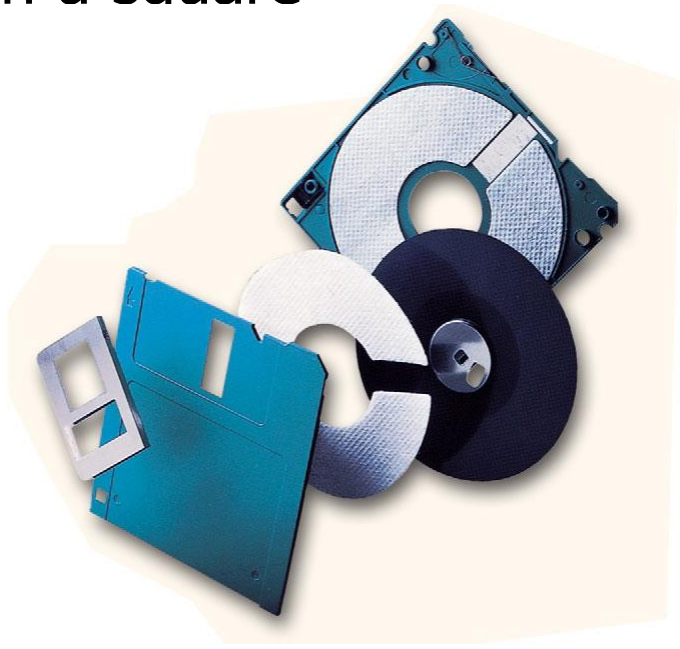
A floppy disk is a

portable,

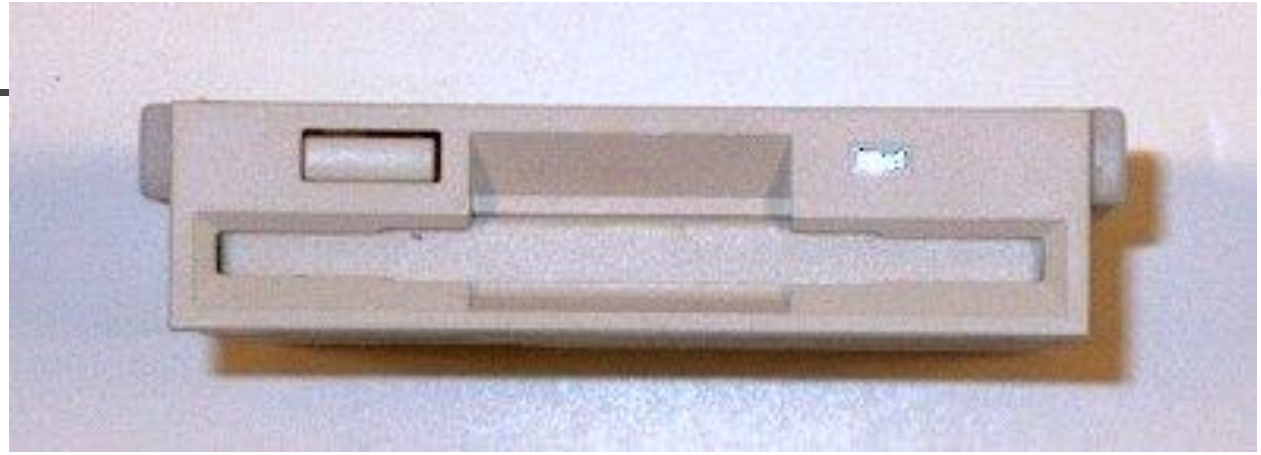
inexpensive storage medium

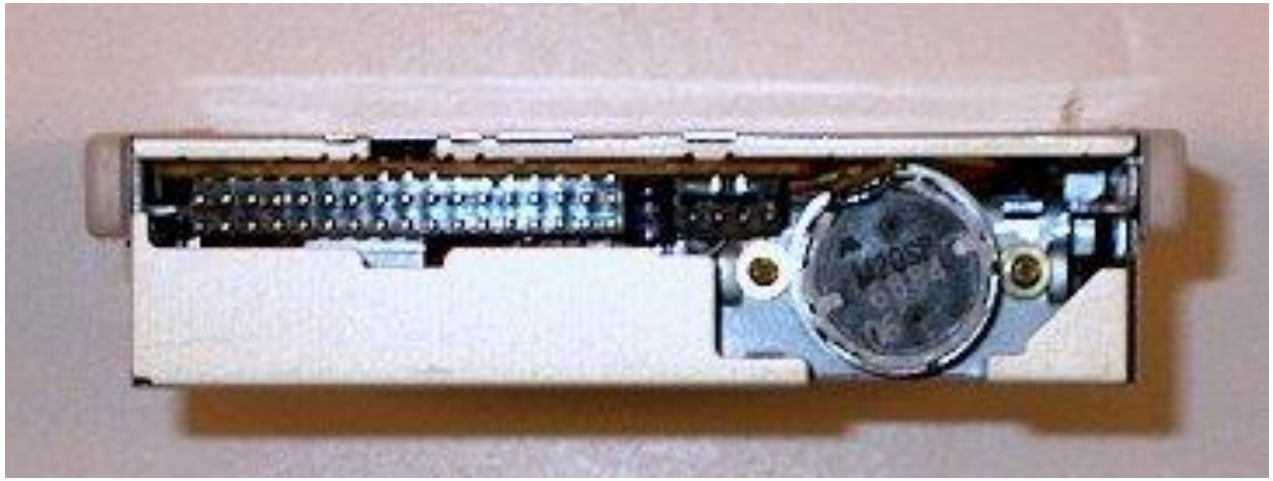
consists of a thin, circular, flexible plastic disk with a magnetic coating enclosed in a square-shaped plastic shell.

Store 1.44MB or 2.25MB



Floppy Drive

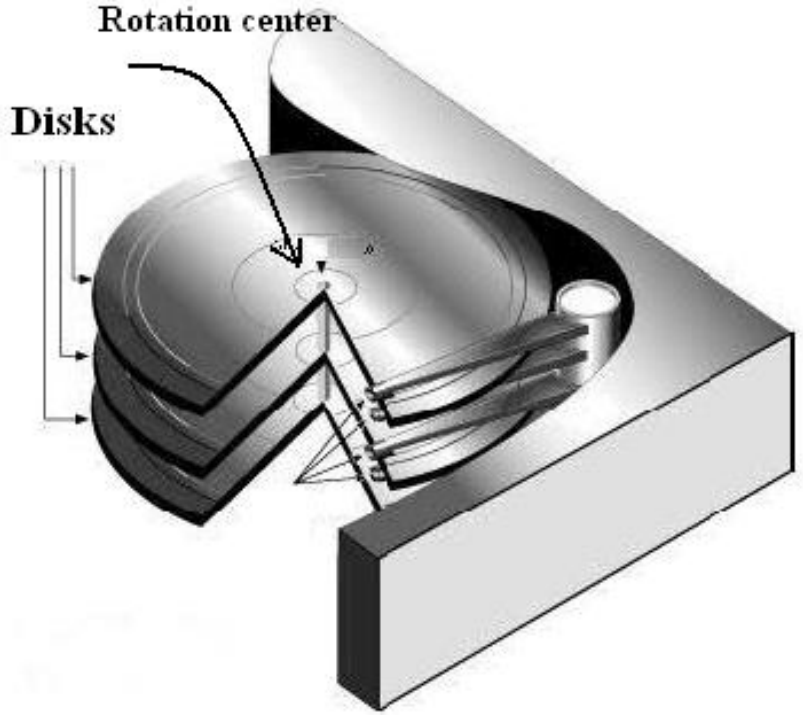




Hard Disks

- Another form of auxiliary storage is a hard disk.
- A hard disk consists of one or more sold metal plates coated with a metal oxide material that allows data to be magnetically recorded on the surface of the platters.
- The hard disk platters spin at a high rate of speed, by revolutions per minute (RPM).
- Storage capacities of hard disks for personal computers range from 10 GB to 1 TB.

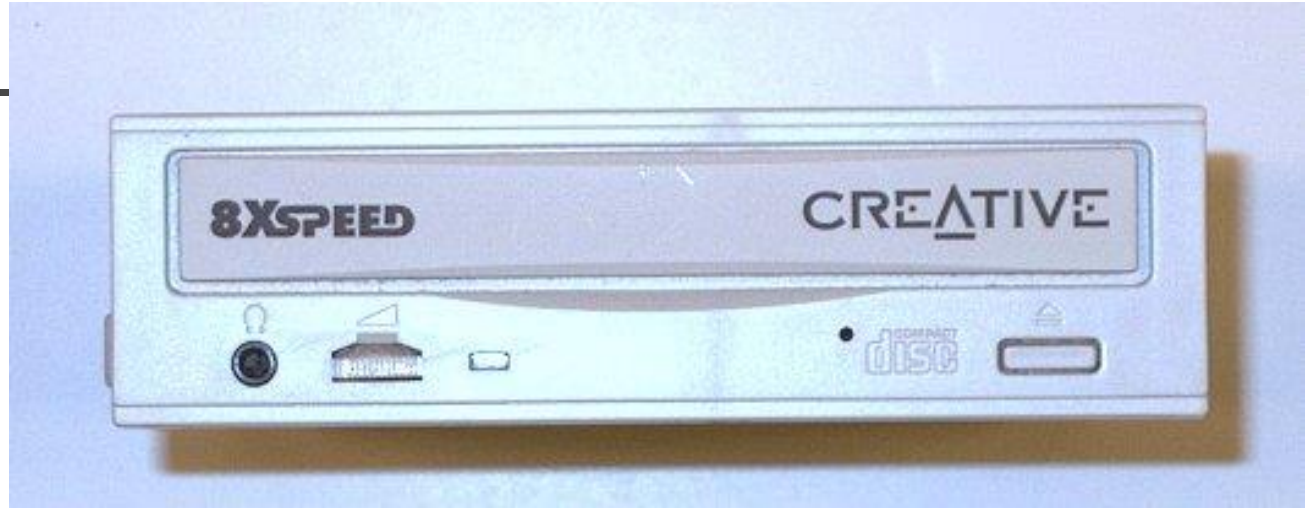

Quantum
POWER REQ: 5/12V 200/210mA
3.5 SERIES
S40AT 214124-001
CT:1RH2RDEA2VZ


Rotation center
Disks

Compact Discs

- A compact disk (CD), also called an optical disc, is a flat round, portable storage medium.
- A CD-ROM (read only memory), is a compact disc uses laser technology for recording data.
- Store different types of data such as text, graphics, audios and videos.
- The capacity of a CD-ROM is 650 MB of data.
- A DVD (Digital Video Disc) looks just like a CD, but holds much more video data

CD-ROM Drive



Flash Drive

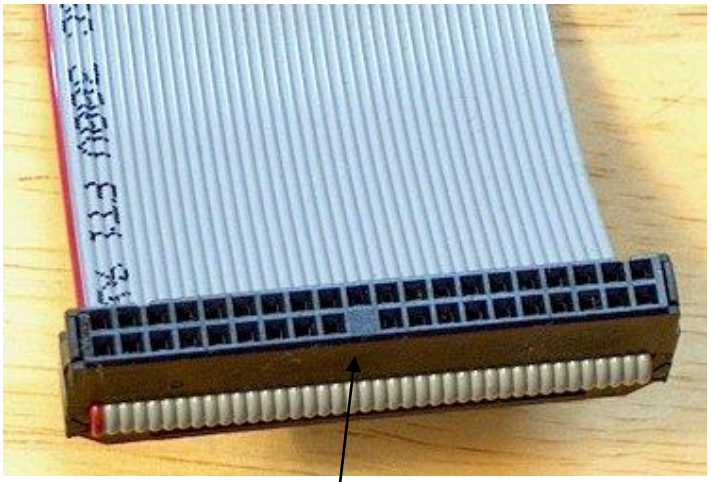
Store data permanently (ROM)

It's like a mini, portable hard drive!

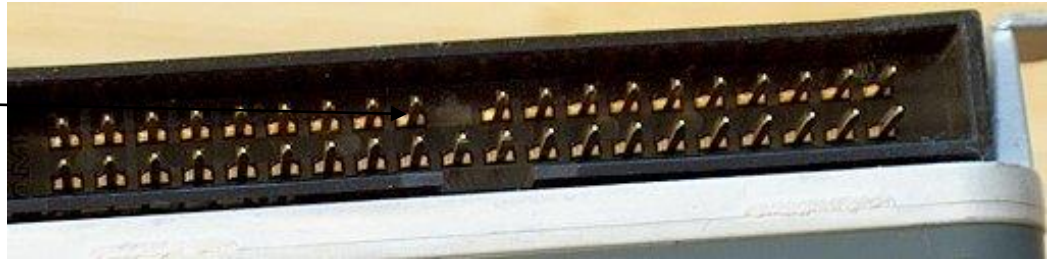
Plug into the USB port (Universal Serial Bus) on the front or rear of computers

Ribbon Cables

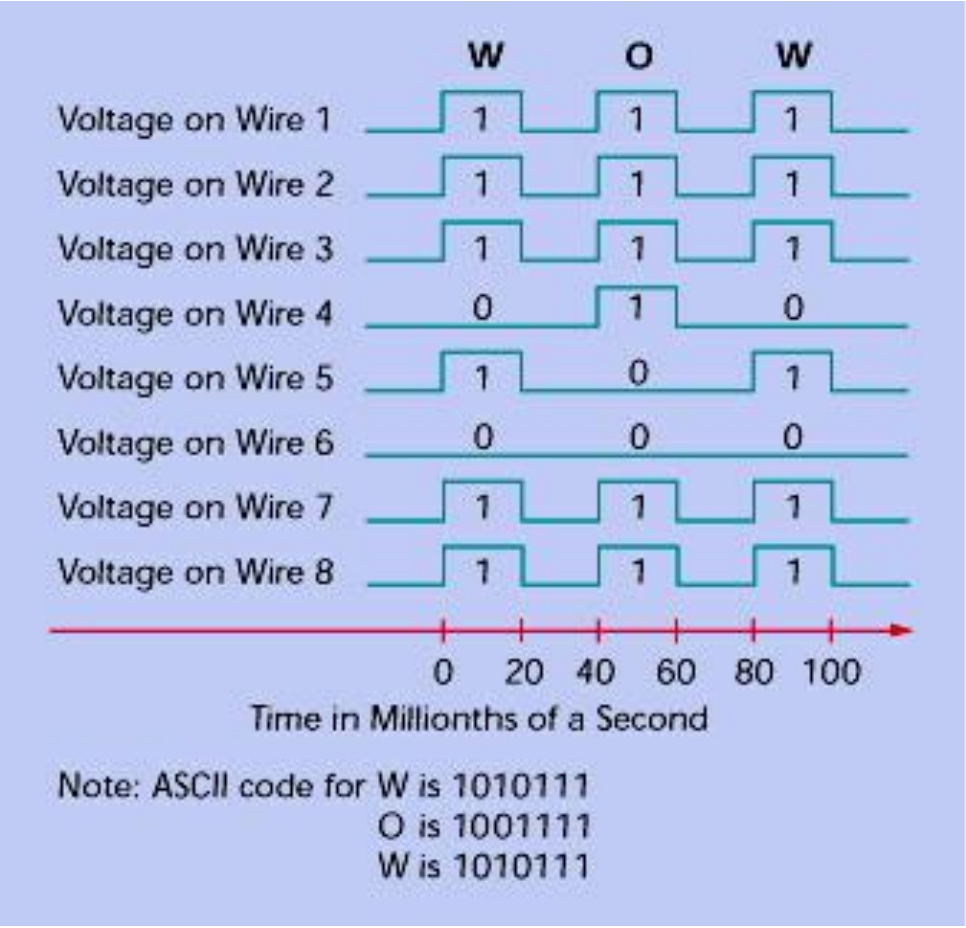
Connect between drivers and motherboard



polarized



- Example of sending the word WOWover the ribbon cable
  - Voltage pulses corresponding to the ASCII codes would pass through the cable



Computer Software

Computer software is the key to productive use of computers. Software can be categorized into two types:

- Operating system software
- Application software.

Operating System Software

Operating system software tells the computer how to perform the functions of :-

Loading application.

Storing application.

Executing application.

transferring data.

GUI (Graphical user interface): type of operating system provides visual clues such as icon symbols to help the user. Like Windows, Linux and Macintosh.

DOS (Disk Operating System) is an older operating system uses text-based command.

Application Software

Application Software consists of programs that tell a computer how to produce information. Some of the more commonly used packages are:

- **Word processing**
- **Electronic spreadsheet**
- **Database**
- **Presentation graphics**

Introduction to the C Programming Language

Introduction

- Program is a set of instructions to direct CPU like widows and application
- **Why programming??**

1- For processing a lot of data (text, audio and video) in rapid speed (Arithmetically or logically)

2- For controlling hardware like printer, scanner, etc…

- **Type of programming language**

1- Low-level language

a- Machine language

b- Assembly language (commands with understanding words)

2- high-level language

- Fortran 1956…Basic 1963…Pascal 1970….C 1990

Levels of Programming Languages

High-level program

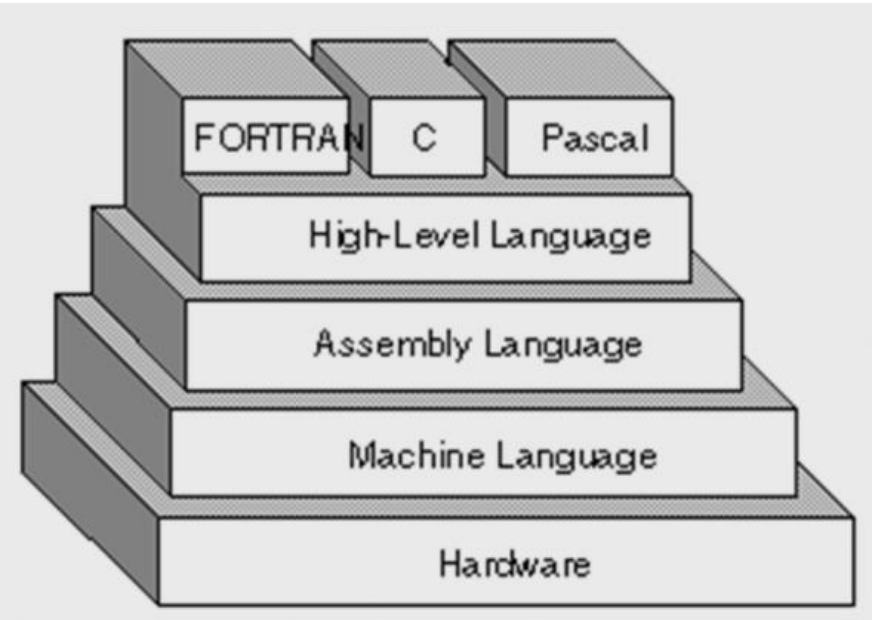
```
class Triangle {
  ...
  float surface()
    return b*h/2;
  }
```

Low-level program

```
LOAD r1,b
LOAD r2,h
MUL r1,r2
DIV r1,#2
RET
```

Executable Machine code

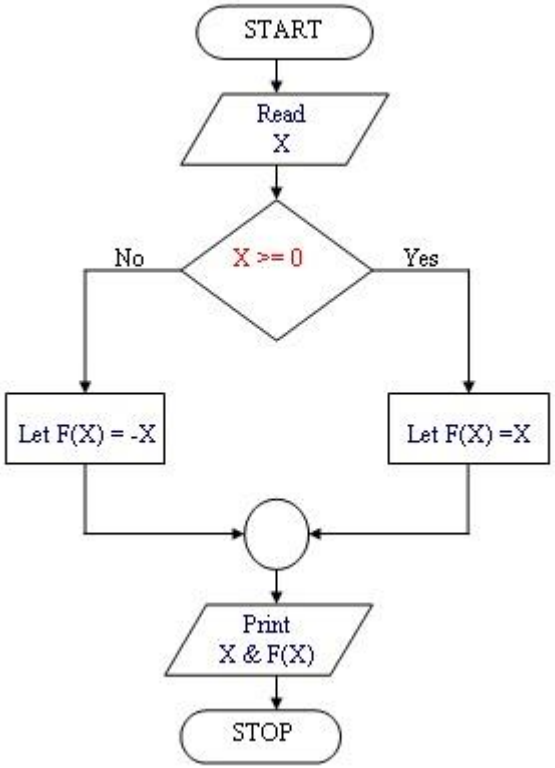
```
0001001001000101
0010010011101100
10101101001...
```

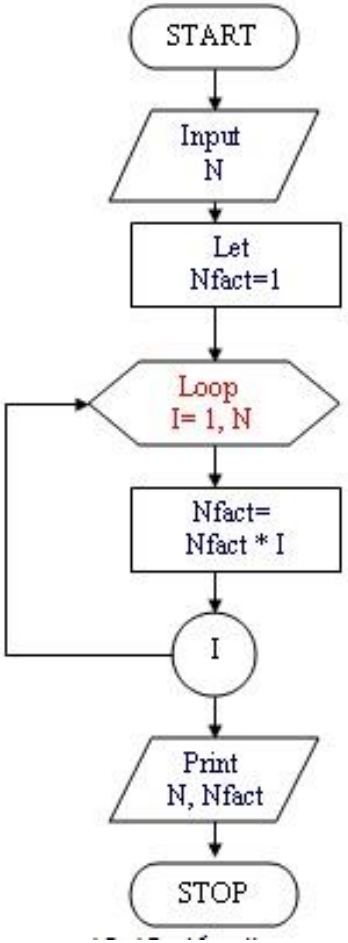


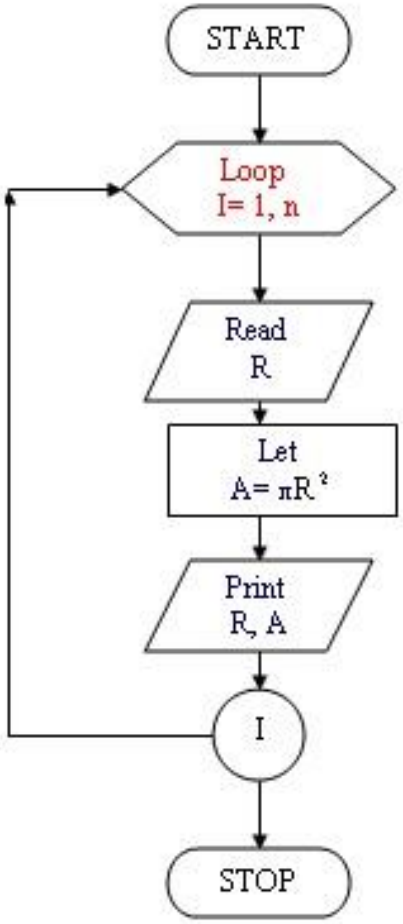
Standard C

- Standardized in 1989 by ANSI (American National Standards Institute) known as ANSI C
- International standard (ISO) in 1990 which was adopted by ANSI and is known as *C89*
- As part of the normal evolution process the standard was updated in 1995 (*C95*) and 1999 (*C99*)
- C++ and C
  - C++ extends C to include support for Object Oriented Programming and other features that facilitate large software development projects
  - C is not strictly a subset of C++, but it is possible to write "*Clean C*" that conforms to both the C++ and C standards.

Algorithms


START
Read
X
X >= 0
No
Yes
Let F(X) = -X
Let F(X) =X
Print
X & F(X)
STOP


START
Input
N
Let
Nfact=1
Loop
I= 1, N
Nfact=
Nfact * I
I
Print
N, Nfact
STOP


START
Loop
I= 1, n
Read
R
Let
A= πR²
Print
R, A
I
STOP

C Program Structure

- Preprocessor directives
- Function main
- The curly braces { }
- Statements
- functions

C program skeleton

- In short, the basic skeleton of a C program looks like this:

```
#include <stdio.h>

void main(void)
{
    statement(s);
}

Int   function_name( )
{
}
```

Preprocessor directives

Function main

Start of segment

End of segment

Preprocessor directives

- a C program line begins with # provides an instruction to the C preprocessor
- It is executed **before** the actual compilation is done.
- Two most common directives :
  - #include<stdio.h>…
  - #include<math.h>
  - #include<stdlib.h>
  - #include<conio.h>
  - #define
- In our example (#include<stdio.h>) identifies the ***header*** file for standard input and output needed by the printf().

Function main

- Identify the start of the program
- Every C program has a main ( )
- 'main' is a C ***keyword***. We **must not** use it for any other variable.
- 4 common ways of main declaration

```
int main(void)
{

return 0;
}
```

```
void
main(void)
{
}
```

```
main(void)
{

}
```

```
main( )
{

}
```

The curly braces { }

- Identify a ***segment / body*** of a program
  - The start and end of a function
  - The start and end of the selection or repetition block.

- Since the opening brace indicates the **start** of a segment while the closing brace indicating the **end** of a segment, **there must be just as many opening braces as closing braces** (this is a common mistake of beginners)

Statements

- A specification of an action to be taken by the computer as the program executes.
- Each statement in C needs to be terminated with semicolon (;)
- Example:

```
#include <stdio.h>

void main(void)
{
  printf("I love programming\n");
  printf("You will love it too once ");
  printf("you know the trick\n");
}
```

statement

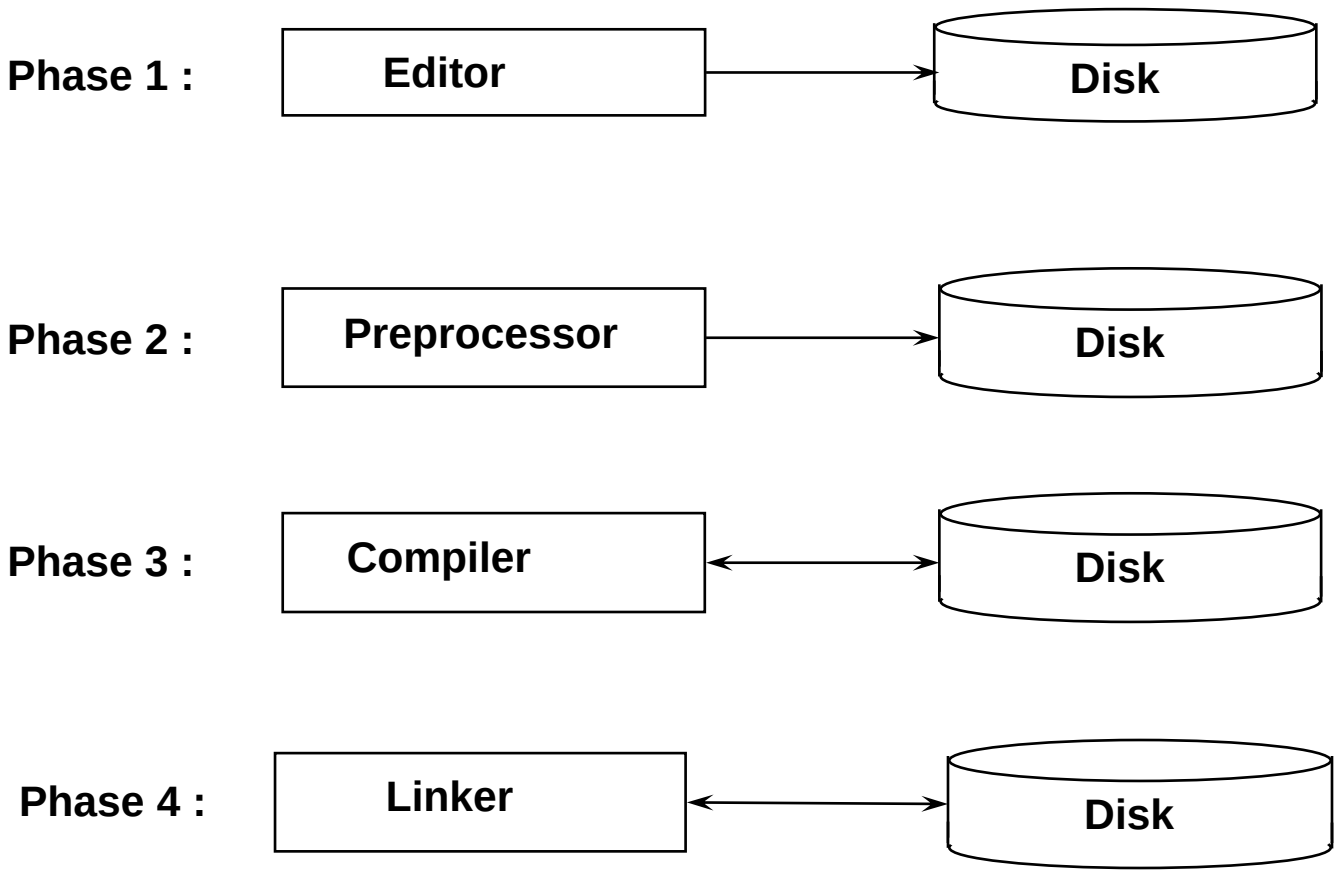
statement

statement

Statement cont…

- Statement has two parts :
  - Declaration
    - The part of the program that tells the compiler the names of memory cells in a program
  - Executable statements
    - Program lines that are converted to machine language instructions and executed by the computer

C Development Environment



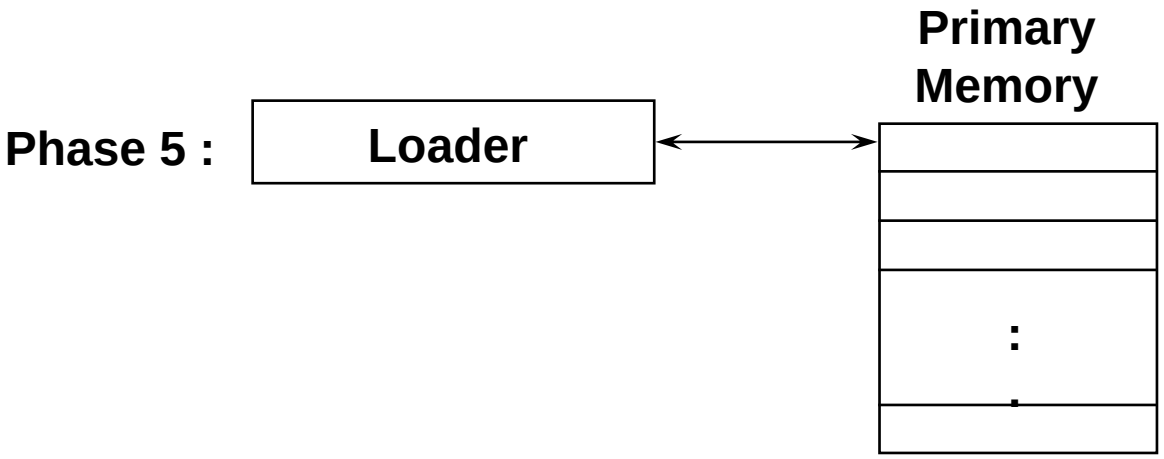
Program is created using the Editor and stored on Disk.

Preprocessor program processes the code.

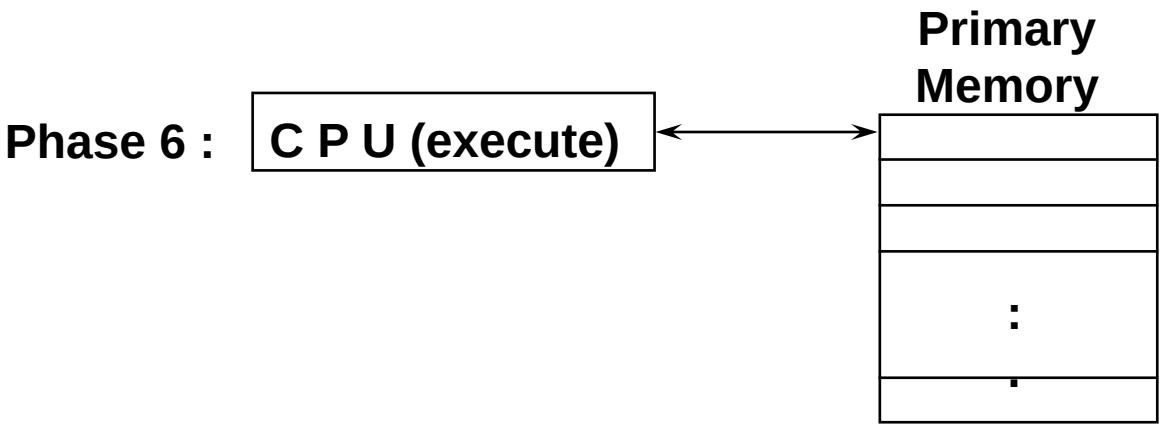
Compiler creates object code and stores it on Disk.

Linker links object code with libraries, creates **a.out** and stores it on Disk

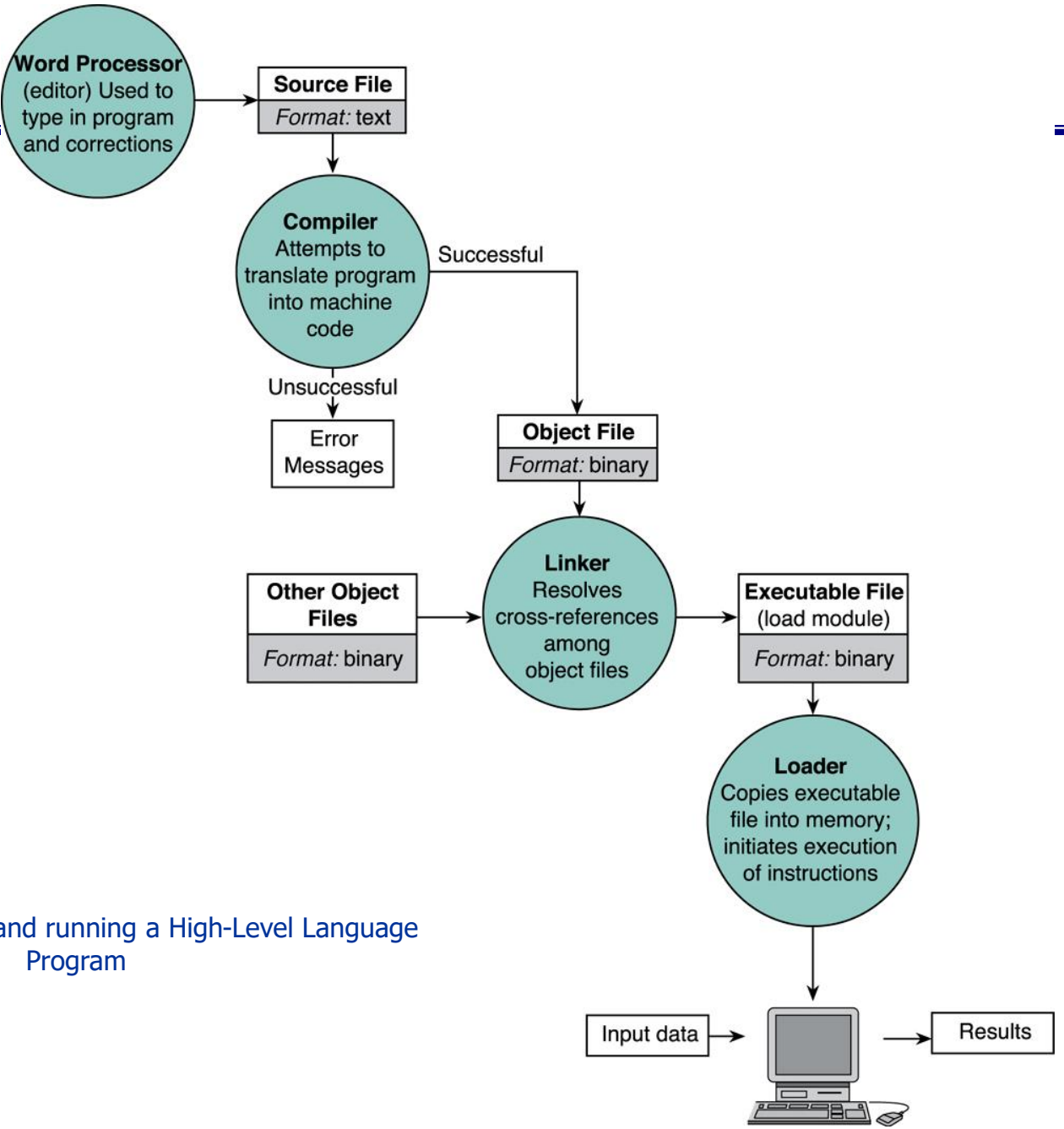
C Development Environment cont



Loader puts Program in Memory



CPU takes each instruction and executes it, storing new data values as the program executes.



Entering, translating, and running a High-Level Language Program

Identifiers

- Words used to represent certain program entities (variables, function names, etc).
- Example:
  - int my_name;
    - my_name is an identifier used as a program variable
  - void CalculateTotal(int value)
    - CalculateTotal is an identifier used as a function name

Rules for naming identifiers

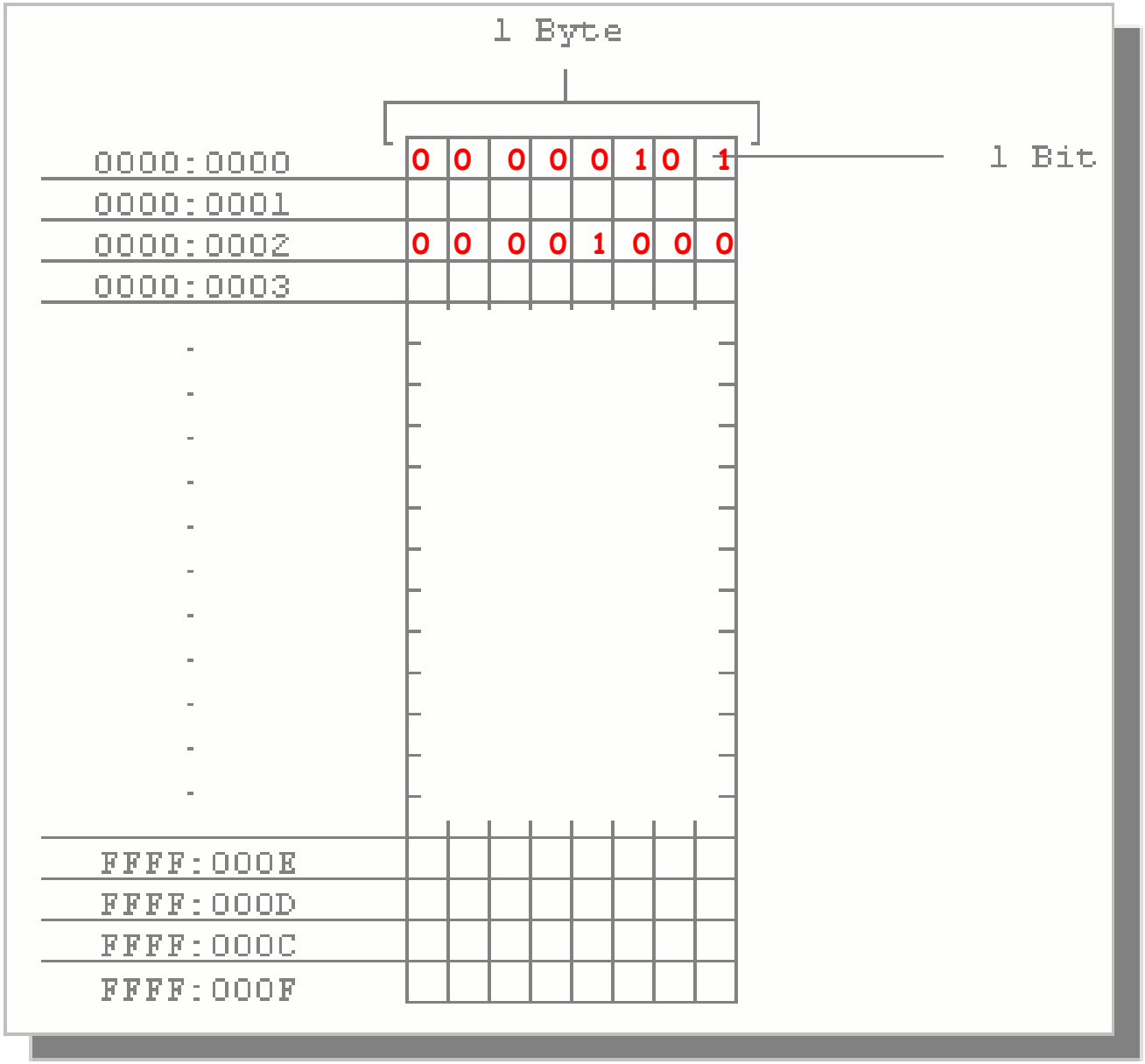
Rules	Example
Can contain a mix of characters and numbers. However it **cannot** start with a number	H2o
First character must be a letter or underscore	Number1; _area
Can be of mixed cases including underscore character	XsquAre my_num
Cannot contain any arithmetic operators	R*S+T
… or any other punctuation marks…	#@x%!!
Cannot be a C keyword/reserved word	struct; printf;
Cannot contain a space	My height
… identifiers are **case sensitive**	Tax != tax

Variables

- **Variable** → a name associated with a memory cell whose value can change
- Memory is addresses in hexadecimal, for each address variable value equal to one byte.

- **Variable Declaration**: specifies the type of a variable
  - Example: `int num;`
  - `int deg = 8;`

- **Variable Definition**: ***assigning*** a value to the declared ***variable***
  - Example: `num = 5;`



Memory addresses

Basic Data Types

- There are 4 basic ***data types*** :
  - int
  - float
  - double
  - char
- **int**
  - used to declare numeric program variables of integer type (2 Byte) (65536)
  - whole numbers, positive and negative
  - keyword: int

```
int number, number2;
number = 12;
Number=number2;
```

Basic Data Types cont…

- **float**
  - fractional parts, positive and negative(4 Byte)
  - keyword: float

```
float height;
height = 1.72;          1.2E-38  to 3.4E+38
```

- **double**
  - used to declare floating point variable of higher precision or higher range of numbers
  - exponential numbers, positive and negative(8 Byte)
  - keyword: double

```
double valuebig;
valuebig = 2.3E-308 to 1.7E+308;
```

Basic Data Types cont…

- **char**
  - equivalent to 'letters' in English language
  - Example of characters:(1 Byte) (256)
    - Numeric digits: 0 - 9
    - Lowercase/uppercase letters: a - z and A - Z
    - Space (blank)
    - Special characters: , . ; ? " / ( ) [ ] { } * & % ^ < > etc
  - single character
  - keyword: char

```
char my_letter;
my_letter = 'U';
```

The declared character must be enclosed within a single quote!

- In addition, there are **void**, **short**, **long**, etc.

Basic Data Types cont…

- **Short**
  - used to declare numeric program variables of integer type (2 Byte) (65536)
  - whole numbers, positive and negative
  - keyword: short

```
short number;
number = 12;
```

- Long
  - used to declare numeric program variables of integer type (4 Byte)
  - whole numbers, positive and negative
  - keyword: long

```
Long number;
number = 12;
```

positive and negative numbers

- Signed int variable = positive and negative numbers
- Unsigned int variable = negative numbers

- 00001010 = 10
- 10001010 = -10

- Unsigned long variable = positive and negative numbers
- Unsigned long variable = negative numbers

Constants

- Entities that appear in the program code as fixed values.
- Any attempt to modify a CONSTANT will result in error.
- 4 types of constants:
  - Integer constants
    - Positive or negative whole numbers with no fractional part
    - Example:
      - const int MAX_NUM = 10;
      - const int MIN_NUM = -90;
  - Floating-point constants (float or double)
    - Positive or negative decimal numbers with an integer part, a decimal point and a fractional part
    - Example:
      - const double VAL = 0.5877e2; (stands for 0.5877×10^2)

Input/Output Operations

- Input operation
  - an instruction that copies data from an input device into memory

- Output operation
  - an instruction that displays information stored in memory to the output devices (such as the monitor screen)

Input/Output Functions

- A C function that performs an input or output operation
- A few functions that are pre-defined in the header file stdio.h such as :
  - printf()
  - scanf()
  - getchar() & putchar()

The printf function

- Used to send data to the standard output (usually the monitor) to be printed according to specific format.
- General format:
  - printf("string literal");
    - A sequence of any number of characters surrounded by double quotation marks.
  - printf("format string", variables);
    - Format string is a combination of text, conversion specifier and escape sequence.

The printf function cont…

- Example:
  - printf("Thank you");
  - printf ("Total sum is: %d\n", sum);
    - %d is a placeholder (conversion specifier)
      - marks the display position for a type integer variable
    - \n is an escape sequence
      - moves the cursor to the new line

Escape Sequence

Escape Sequence	Effect
\a	Beep sound
\b	Backspace
\f	Formfeed (for printing)
\n	New line
\r	Carriage return
\t	Tab
\v	Vertical tab
\\	Backslash
\”	“ sign
\o	Octal decimal
\x	Hexadecimal
\O	NULL

Placeholder / Conversion Specifier

No	Conversion Specifier	Output Type	Output Example
1	%d	Signed decimal integer	76
2	%i	Signed decimal integer	76
3	%o	Unsigned octal integer	134
4	%u	Unsigned decimal integer	76
5	%x	Unsigned hexadecimal (small letter)	9c
6	%X	Unsigned hexadecimal (capital letter)	9C
7	%f	Integer including decimal point	76.0000
8	%e	Signed floating point (using e notation)	7.6000e+01
9	%E	Signed floating point (using E notation)	7.6000E+01
10	%g	The shorter between %f and %e	76
11	%G	The shorter between %f and %E	76
12	%c	Character	‘7’
13	%s	String	‘76'

The scanf function

- Read data from the standard input device (usually keyboard) and store it in a variable.
- General format:
  - scanf("Format string", &variable);
- Notice ampersand (&) operator :
  - C address of operator
  - it passes the address of the variable instead of the variable itself
  - tells the scanf() where to find the variable to store the new value

The scanf function cont…

- Example :

```
int age;
printf("Enter your age: ");
scanf("%d", &age);
```

- Common Conversion Identifier used in printf and scanf functions.

	printf	scanf
int	%d	%d
float	%f	%f
double	%f	%lf
char	%c	%c
string	%s	%s

The scanf function cont…

- If you want the user to enter more than one value, you serialise the inputs.
- Example:

```
float height, weight;

printf("Please enter your height and weight:");
scanf("%f%f", &height, &weight);
```

getchar() and putchar()

- getchar() - read a character from standard input
- putchar() - write a character to standard output
- Example:

```
#include <stdio.h>
void main(void)
{
  char my_char;
  printf(“Please type a character: ”);
  my_char = getchar();
  printf(“\nYou have typed this character: ”);
  putchar(my_char);
}
```

getchar() and putchar() cont

- Alternatively, you can write the previous code using normal scanf and %c placeholder.
- Example

```
#include <stdio.h>
void main(void)
{
  char my_char;
  printf(“Please type a character: ”);
  scanf(“%c”,&my_char);
  printf(“\nYou have typed this character: %c ”, my_char);
}
```

Operators

- Arithmetic Operators
  - +, - , *, / and the modulus operator %.
  - + and – have the same precedence and associate left to right.

    3 - 5 + 7 = (3 - 5) + 7 ≠ 3 - (5 + 7)

    3 + 7 - 5 + 2 = ((3 + 7) - 5) + 2
  - *, /, % have the same precedence and associate left to right.
  - The +, - group has lower precendence than the *, / % group.

    3 - 5 * 7 / 8 + 6 / 2

    3 - 35 / 8 + 6 / 2

    3 - 4.375 + 6 / 2

    3 - 4.375 + 3

    -1.375 + 3

    1.625

Operators

- Arithmetic Operators
  - % is a modulus operator. x % y results in the remainder when x is divided by y and is zero when x is divisible by y.
  - Cannot be applied to float or double variables.
  - Example

int num=5;

R=5%2;

Printf("R=%d",R);

```
if ( num % 2 = = 0 )
 printf("%d is an even number\n", num)';
else
 printf("%d is an odd number\n", num);
```

Type Conversions

- The operands of a binary operator must have a the same type and the result is also of the same type.
- Integer division:

```
c = (9 / 5)*(f - 32)
```

The operands of the division are both int and hence the result also would be int. For correct results, one may write

```
c = (9.0 / 5.0)*(f - 32)
```

- In case the two operands of a binary operator are different, but compatible, then they are converted to the same type by the compiler. The mechanism (set of rules) is called Automatic Type Casting.

```
c = (9.0 / 5)*(f - 32)
```

- It is possible to force a conversion of an operand. This is called Explicit Type casting.

```
c = ((float) 9 / 5)*(f - 32)
```

Automatic Type Casting

1. char and short operands are converted to int
2. Lower data types are converted to the higher data types and result is of higher type.
3. The conversions between unsigned and signed types may not yield intuitive results.
4. Example

```
float f; double d; long l;
int i; short s;
d + f   f will be converted to double
i / s   s will be converted to int
l / i   i is converted to long; long result
```

Hierarchy
Double
float
long
Int
Short and char

Operators

- Assignment operators
  - The general form of an assignment operator is

```
v op= exp
```

  - Where v is a variable and op is a binary arithmetic operator. This statement is equivalent to

```
v = v op (exp)
```

`a = a + b`	can be written as	`a += b`
`a = a * b`	can be written as	`a *= b`
`a = a / b`	can be written as	`a /= b`
`a = a - b`	can be written as	`a -= b`

Operators

- Increment and Decrement Operators
  - The operators ++ and -- are called increment and decrement operators.

a++ and ++a are equivalent to a += 1.

a-- and --a are equivalent to a -= 1.

++a op b is equivalent to a ++; a op b;

a++ op b is equivalent to a op b; a++;

Example

```
Let b = 10 then
(++b)+b+b = 33
b+(++b)+b = 33
b+b+(++b) = 31
b+b*(++b) = 132
```

Logical Operator

- int d;
- d=2&4; AND gate

- int d;
- d=2 | 4; OR Gate

- int d;
- d=(2&4) | 6;

The If Statement

- Syntax: if (condition or conditions) statement;
- If the expression is true (not zero), the statement is executed. If the expression is false, it is not executed.
- You can group multiple expressions together with braces:

```
if (condition or conditions)
{
  statement 1;
  statement 2;
  statement 3;
}
```

The If/Else Statement

- Syntax:

```
 if (condition or conditions)
{
 statement_1;
}
 else
{
statement_2;
}
```

- If the expression is true, statement_1 will be executed, otherwise, statement_2 will be.
- **Example :**

```
if (myVal < 3) printf("myVal is less than 3.\n");
else printf("myVal is greater than or equal to 3.\n");
```

Operators

- Relational Operators

  < <= > >= == !=

  are the relational operators.

  The expression

  if(`operand1 relational-operator operand2`)

  takes a value of 1(int) if the relationship is true and 0(int) if relationship is false.

  - Example

```
int a = 25, b = 30, c, d;
c = a < b;
d = a > b;
```

  value of c will be 1 and that of d will be 0.

  If (c < b) printf("c > b");

Operators

- Logical Operators
  - `&&, ||` and `!` are the three logical operators.
  - `expr1 && expr2` has a value 1 if `expr1` and `expr2` both are nonzero.
  - `expr1 || expr2` has a value 1 if `expr1` and `expr2` both are nonzero.
  - `!expr1` has a value 1 if expr1 is zero else 0.
  - Example
  - `if ( marks >= 40 && attendance >= 75 ) grade = 'P'`
  - `If ( marks < 40 || attendance < 75 ) grade = 'N'`

The For Loop

- Syntax: for (initialization; condition; increment) {statements;}
- The for loop will first perform the initialization. Then, as long is test is TRUE, it will execute statements. After each execution, it will increment.

```
for (cntr = 0; cntr < 3; cntr = cntr + 1)
  printf(" Counter = %d\n", cntr);
```

```
Counter = 0;
Counter = 1;
Counter = 2;
```

```
For(      ;       ;    )
```

Logic Operator Examples

```
#include<stdio.h>
main()
{
int i;
i=10 < 5 || 5<1;
printf("i=%i",i);
}
```

```
#include<stdio.h>
main()
{
double i,k=10,n;
int h;
i=k/3;
n=10/3;
h=k/3;
printf("i=%f  n=%f  h=%i",i,n,h);
}
```

Few notes on C program…

- C is ***case-sensitive***
  - Word, word, WorD, WORD, WOrD, worD, etc are all different variables / expressions

    Eg. sum = 23 + 7
    - What is the value of Sum after this addition ?

- Comments (remember 'Documentation'; Chapter 2)
  - are inserted into the code using /* to start and */ to end a comment
  - Some compiler support comments starting with '//'
  - Provides supplementary information but is ignored by the preprocessor and compiler
    - `/* This is a comment */`
    - `// This program was written by Hanly Koffman`

Few notes on C program cont…

- Reserved Words
  - Keywords that identify language entities such as statements, data types, language attributes, etc.
  - Have special meaning to the compiler, cannot be used as identifiers (variable, function name) in our program.
  - Should be typed in lowercase.
  - Example: const, double, int, main, void,printf, while, for, else (etc..)

52

Few notes on C program cont…

- Punctuators (separators)
  - Symbols used to separate different parts of the C program.
  - These punctuators include:

    [ ] ( ) { } , ; “: * #
  - Usage example:

```
void main (void)
{
   int num = 10;
    printf (“% d”,num);
}
```


53

Common Programming Errors

- ***Debugging*** → Process removing errors from a program
- Three (3) kinds of errors :
  - Syntax Error
    - a violation of the C grammar rules, detected during program translation (compilation).
    - statement cannot be translated and program cannot be executed

54

Common Programming Errors cont…

- Run-time errors
  - An attempt to perform an invalid operation, detected during program execution.
  - Occurs when the program directs the computer to perform an illegal operation, such as dividing a number by zero.
  - The computer will stop executing the program, and displays a diagnostic message indicates the line where the error was detected

Common Programming Errors

- Logic Error/Design Error
  - An error caused by following an incorrect algorithm
  - Very difficult to detect - it does not cause run-time error and does not display message errors.
  - The only sign of logic error – incorrect program output
  - Can be detected by testing the program thoroughly, comparing its output to calculated results
  - To prevent – carefully desk checking the algorithm and written program before you actually type it

56

Q0

- Write a program in C language to calculate and print the factorial of number 5.

```
#include<stdio.h>
main()
{
int i,f=1;
for(i=1;i<=5;i++)
f=f*i;
printf("The Factorial of 5 is%i\n",f);
}
}
```

Q1

- Wire a program in C language to enter 10 numbers through the execution, test each number if even or odd and print exprition if the number is even or odd.

```
#include<stdio.h>
main()
{
int i,k,f=1,k1;
for(i=1;i<=10;i++)
{
printf("Enter the number\n");
scanf("%i",&k1);
k=k1%2;
if(k==0) printf("The number %i is Even\n",k1);
else printf("The number %i is Odd\n",k1);
}
}
```

Q2

- Wire a program in C language to calculate the primary number between 1 to 10.

```
#include<stdio.h>
main()
{
int i,j,c=0,s;
for(i=1;i<=100;i++)
{
for(j=1;j<=i;j++)
{
    s=i%j;
    if(s==0) c=c+1;
}
if(c==2 || c==1) printf("%i\n",i);
c=0;
}
}
```

Q3

- Write a program in C language to find the result of the following series, input the values of x and y through the execution

$$z = \frac{x^1 + y^2}{\sum_{i=1}^{2} i} + \frac{x^3 + y^4}{\sum_{i=1}^{4} i} + \cdots\cdots\cdots\cdots\cdots + \frac{x^9 + y^{10}}{\sum_{i=1}^{10} i}$$

- Pow(3,1/2)
- Sqrt(3)

Q3 Solution

```
1   #include<stdio.h>
2   main()
3   {
4   int i,j,sum=0;
5   float x,y,z=0;
6   printf("Enter the values of x and y\n"
7   scanf("%f %f",&x,&y);
8   for(i=1;i<=9;i=i+2)
9   {
10  for(j=1;j<=(i+1);j++)
11  sum=sum+j;
12  z=z+(pow(x,i)+pow(y,(i+1))/sum);
13  sum=0;
14  }
15  printf("The series Result  z=\n%f",z);
16  }
```

Q4

Write C program that can print in decimal the ASCII code for any key to be pressed fro
keyboard during execution of the program. Program should terminate only if user press th
Escape key, if you know that ASCII code for Escape key is (27) in decimal.

using While loop & getche()

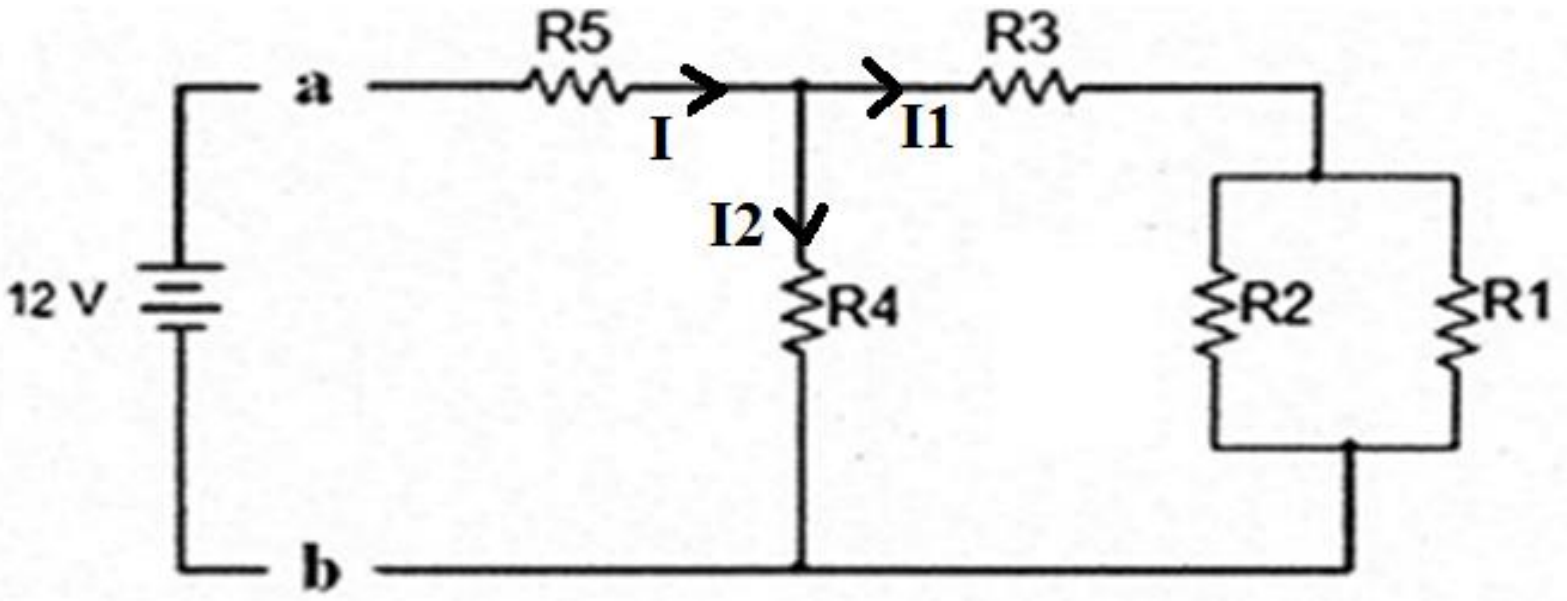
Solution Q4

```
1   #include<stdio.h>
2   main()
3   {
4   char ch;
5   printf("Press any key to see the Ascii code for it\n");
6   ch=getche();
7   while(ch!=27)
8   {
9   printf("\n\nThe Ascii code for %c is %d\n",ch,ch);
10  printf("\n\nPress any key to see the Ascii code for it\n");
11  ch=getche();
12  }
13  printf("\nExitting\n");
14  }
```

Q5

The circuit shown in Fig.(1), write a program to find ($R_{equ.}$) between **a** and **b**.

and find the currents I, I1, I2 in the circuit, make the program executed for infinite and stop the execution by pressing q letter



Q5 Solution

```c
#include<stdio.h>
main()
{
float R1,R2,R3,R4,R5,Re1,Re2,Re3,Rt,v=12,I,I1,I2;
char ch;
printf("Pree any key to contiue or Pree q to exit");
scanf("%c",&ch);
while(ch!='q')
{
Printf("Enter the value of R1 R2 R3 R4 R5\n");
scanf("%f %f %f %f %f",&R1,&R2,&R3,&R4,&R5);
Re1=(R1*R)/(R1+R2);
Re2=Re1+R3;
Re3=(Re2*R4)/(Re2+R4);
Rt=Re3+R5;
I=v/Rt;
I1=I*(R4/(R4+Re2));
I2=I*(Re2/(R4+Re2));
printf("The Req=%f\n",Rt);
printf("The total current=%f \n I1=%f \n I2=%f \n",I,I1,I2);
printf("Pree any key to contiue or Pree q to exit");
ch=getche();
}
}
```

Q6

- Wire a program in C language to calculate and print the primary numbers among 1 to 100, calculate the average of the primary numbers among 25 to 75 and print this average with explanation sentence, also calculate and print the summation of non primary numbers among 10 to 50

- Wire a program in C language to calculate and print the primary numbers among 1 to 100, calculate the average of the primary numbers among 25 to 75 and print this average with explanation sentence, also calculate and print the summation of non primary numbers among 10 to 50.

Q6 Solution

```c
#include<stdio.h>
main()
{
int i,j,c=0,s,c2=0;
float sum,A;
for(i=1;i<=100;i++)
{
for(j=1;j<=i;j++)
{
s=i%j;
if(s==0) c=c+1;
}
if(c<=2)
{
printf("%i\n",i);
if(i>=25  && i<=75)
{
sum=sum+i;
c2=c2+1;
}
}
c=0;
}
A=sum/c2;
printf("The average = %f\n",A);
}
```

The While LOOP

- Syntax:

```
while (condition)
{
statements;
}
```

Example :

```
int i=1;
While(i != 4)
{
Printf("%d",i);
i++;
}
```

```
#include<stdio.h>
Main()
{
Int num1,num2,sum;
Char ch;
Printf("Press + for summation) :");
Ch=getchar();
While(ch  == '+')
{
Printf("Enter num1 and num2");
Scanf("%d \n%d\n", &num1,&num2);
Sum=num1+num2;
}
Printf("\n Exitting….\n");
}
```

The Do….While LOOP

- Syntax:

```
do
{
statements;
}
while (condition)
```

Example :

```
int i=1;
do
{
Printf("%d",i);
i++;
}
While(i != 4)
```

Break ….Continue Instructions

- Break instruction stop the loop
- Continue instruction jump the execution of program to the start of loop

```c
#include<stdio.h>

main()
{
	int i;
	i = 1;

	while(1)
	{
		if(i<=3)
		{
			printf("\a%d ,", i);
			++i;
		}

		printf(" Go!\n");
		break;
	}

}
```

```c
#include<stdio.h>

main()
{
	int i;
	i = 1;

	while(1)
	{
		if(i<=3)
		{
			printf("\a%d ,", i);
			++i;
			continue;
		}

		printf(" Go!\n");
		break;
	}

}
```

LOOP using goto

- Syntax : goto place_name;
- Place_name:

```c
#include<stdio.h>

main()
{
begin:
        goto fin;
        printf("Begin!\n");

fin:
        printf("End!\n");
        goto begin;
}
```

```c
#include<stdio.h>

main()
{
        int i;
        i = 0;

        printf("Begin!\n");

begin:
        printf("%d ,", i);
        i++;
        if(i==10)
                goto fin;
        else
                goto begin;

fin:
    printf("\nEnd!\n");
}
```

Switch….case instruction

- Switch instruction using to test variable value with integer constant values, where each constant value consider as condition or comparison, switch like if…else
- Switch()
- {
- Case 1:
- Break;
- Case 2:
- Break;
- Case 3:
- Break;
- Default:
- }

```c
#include<stdio.h>

main()
{
	int num1, num2;
	char Char;

	printf("1:(+)\n2:(-)\n3:(/)\n4:(*)\nEnter a choice: ");
	scanf("%c", &Char);

	printf("Enter the first number: ");
	scanf ("%d", &num1);

	printf("Enter the second number: ");
	scanf("%d", &num2);

	if(Char == '+' || Char == '1')
		printf("%d + %d = %d\n", num1, num2, num1+num2);

	else if(Char == '-' || Char == '2')
		printf("%d - %d = %d\n", num1, num2, num1-num2);

	else if(Char == '/' || Char == '3')
		printf("%d / %d = %d\n", num1, num2, num1/num2);

	else if(Char == '*' || Char == '4')
		printf("%d * %d = %d\n", num1, num2, num1*num2);

	else
		printf("Error in choice!\nExiting...\n");
}
```

Q7

- Write a program in C language to enter two numbers through the execution, perform the following operation and print the result, the addition if press '+' sign, the subtraction if press '-', the multiplication if press '*' sign and the division if press '/' sign or print the error choice, repeated the execution for infinite and terminate by pressing 'q' using break…..continue and switch…case instructions.

Solution Q7

```c
#include<stdio.h>

main()
{
      int num1, num2;
      char Char;

      printf("1:(+)\n2:(-)\n3:(/)\n4:(*)\nEnter a choice: ");
      scanf("%c", &Char);

   printf("Enter the first number: ");
   scanf ("%d", &num1);

   printf("Enter the second number: ");
   scanf("%d", &num2);

   switch(Char)
   {
   case '+':
   case '1':
         printf("%d + %d = %d\n", num1, num2, num1+num2);
         break;

   case '-':
   case '2':
         printf("%d - %d = %d\n", num1, num2, num1-num2);
         break;

   case '/':
   case '3':
         printf("%d / %d = %d\n", num1, num2, num1/num2);
         break;
   case '*':
   case '4':
         printf("%d * %d = %d\n", num1, num2, num1*num2);
         break;

   default:
         printf("Error in choice!\nExiting...\n");
         break;
   }
```

Arrays

Syntax

Array_type Array_name[Row_size]……one dimension

Array_type Array_name[Row_size] [Colum_size]…two dimension

Array_type Array_name[Row_size] [Colum_size] [Item_size]…….Three dimension

Note : Need to loop for entering and loop for printing

```c
#include<stdio.h>

main()
{
	int arr1, arr2, arr3, arr4, arr5, ...,arr25;

	printf("Arr1: ");
	scanf("%d", arr1);

	printf("Arr2: ");
	scanf("%d", arr2);

	printf("Arr3: ");
	scanf("%d", arr3);

	printf("Arr4: ");
	scanf("%d", arr4);

	printf("Arr5: ");
	scanf("%d", arr5);
..
	...
..
	printf("Arr25: ");
	scanf("%d", arr25);
..
}
```

```c
#include<stdio.h>

main()
{
	int arr[24];
	int i;

	for(i=0;i<25;i++)
	{
		printf("Arr%d: ", i);
		scanf("%d", &arr[i]);
	}

	printf("***********- LIST -***********\n");

	for(i=0;i<25;i++)
	{
		printf("Arr%d: %d\n", i, arr[i]);
	}

}
```

Q8

- Write a program in C language to enter 10 numbers through the execution, arrange them descending and print these numbers before and after descending.

Solution Q8

```c
#include<stdio.h>
#include<math.h>
main()
{
int a[10],z=0,i,j,Temp;
for(i=0;i<=9;i++)
scanf("%d",&a[i]);

for(i=0;i<=9;i++)
for(j=i+1;j<=9;j++)
if(a[j]>a[i])
{
Temp=a[i];
a[i]=a[j];
a[j]=Temp;
}
printf("\n\n");
for(i=0;i<=9;i++)
printf("%d\n",a[i]);
getche();
}
```

Character Chain

```c
#include<stdio.h>

main()
{
	char text[14] = "Hello, World!";

	printf("%s\n", text);
}
```

- End of the chain must be '\0 '

```c
#include<stdio.h>

main()
{
	char text[14];

	text[0] = 'H';
	text[1] = 'e';
	text[2] = 'l';
	text[3] = 'l';
	text[4] = 'o';
	text[5] = ',';
	text[6] = ' ';
	text[7] = 'w';
	text[8] = 'o';
	text[9] = 'r';
	text[10] = 'l';
	text[11] = 'd';
	text[12] = '!';
	text[13] = '\0';

	printf("%s\n", text);
}
```

- char=getchar() - read a character from standard input
- getch()
- getchar()
-
- putchar('char') - write a character to standard output

- gets(string) read string from standard input

- puts("string") write string to standard output

Q9

- Write a program in C language to enter the names of 3 students with 3` degrees for each student, print the name of each student with its degrees in a table, calculate the average of each student, increase the average with 2 degrees if it is grater than 60, increase the average with 4 degrees if less than or equal 60.

Solution Q9

```c
#include<stdio.h>
main()
{
int d[3][5],i,j;
float a[3]={0.0,0.0,0.0};
char n[3][10];
for(i=0;i<3;i++)
{
printf("\nEnter the name of student_%d :",i+1);
scanf("%s",&]n[i]);
for(j=0;j<5;j++)
{
printf("\nEnter the degree %d of student %s =",j+1,n[i]);
scanf("%d",&d[i][j]);
a[i]=a[i]+d[i][j];
}
a[i]=a[i]/5;
if(a[i]>60) a[i]=a[i]+2;
if(a[i]<=60) a[i]=a[i]+4;
}
system("cls");
puts("\n\n");
for(i=0;i<3;i++)
{
printf("%s",n[i]);
for(j=0;j<5;j++)
{
printf("\t:   %d",d[i][j]);
}
printf("\t:   %f",a[i]);
puts("\n");
}
}
```

Q10

- Wire a program in C language to enter 10 numbers as an array through the execution, find the most repeated number among them and print this number with the repeated times.

Solution Q10

```c
#include<stdio.h>
#include<conio.h>
#include<windows.h>
main()
{
int a[10],z=0,i,j,Temp=0,Temp2,max=0;
for(i=0;i<=9;i++)
scanf("%d",&a[i]);

for(i=0;i<=9;i++)
{
Temp=0;
for(j=0;j<=9;j++)
if(a[j]==a[i])
{
Temp++;
}
if(Temp>1 && Temp>=max)
{
max=Temp;
Temp2=a[i];

}
}
printf("\n%50d\t%d",max,Temp2);
getche();
}
```

Q11

- **Wire a program in C language to calculate and print the following series using switch instruction, enter the values of X and Y through the execution.**

$$z = \frac{X^1 + Y^3}{\sum_{i=1}^{2} i} + \frac{X^3 + Y^5}{4!} + \frac{X^5 + Y^7}{\sum_{i=1}^{6} i} + \frac{X^7 + Y^9}{8!} + \frac{X^9 + Y^{11}}{\sum_{i=1}^{10} i}$$

Solution Q11

- Using switch statement

```c
#include<stdio.h>
main()
{
int i,j,k;
float x,y,z=0;
printf("Enter the values of x and y\n");
scanf("%f %f",&x,&y);
for(i=1;i<=9;i=i+2)
{
switch(i)
{
case 1:
case 5:
case 9:
k=0.0;
for(j=1;j<=(i+1);j++)
k=k+j;
printf("%d  +\n",i);
break;
case 3:
case 7:
k=1.0;
for(j=1;j<=(i+1);j++)
k=k*j;
printf("%d  *\n",i);
break;
}
z=z+(pow(x,i)+pow(y,(i+2))/k);
}
printf("The series Result  z=%f\n",z);
}
```

Solution Q11

- Using if statement

```c
#include<stdio.h>
main()
{
int i,j,k;
float x,y,z=0;
printf("Enter the values of x and y\n");
scanf("%f %f",&x,&y);
for(i=1;i<=9;i=i+2)
{
if(i==1 || i==5 || i==9)
{
k=0.0;
for(j=1;j<=(i+1);j++)
k=k+j;
printf("%d  +\n",i);
}
if(i==3 || i==7)
{
k=1.0;
for(j=1;j<=(i+1);j++)
k=k*j;
printf("%d  *\n",i);
}
z=z+(pow(x,i)+pow(y,(i+2))/k);
}
printf("The series Result  z=%f\n",z);
}
```

Multi_ Dimensional Array

Example : 3 Dimensional Array

int a[2][2][2]

1 4 9 3

6 7 5 2

```c
#include<stdio.h>

main()
{
	int arr3d[2][2][2];
	int i, j, k;

	for(i=0;i<=1;i++)
	{
		for(j=0;j<=1;j++)
		{
			for(k=0;k<=1;k++)
			{
				printf("arr3d[%d][%d][%d] : ", i, j, k);
				scanf("%d", &arr3d[i][j][k]);
			}
		}
	}

	for(i=0;i<=1;i++)
	{
		for(j=0;j<=1;j++)
		{
			for(k=0;k<=1;k++)
			{
				printf("arr3d[%d][%d][%d] = %d\n", i, j, k,
arr3d[i][j][k]);
			}
		}
	}

}
```

Function rand()

- rand()…..to generate random integer numbers within the limits.

Rand() work with the function srand(time(NULL))

Random between two limits % (max-min)+min

```
1  #include<stdio.h>
2  #include<stdlib.h>
3
4
5  main(){
6        int i;
7
8        srand(time(NULL));
9
10       for(i=0;i<=10;i++){
11             printf("%d\n", rand()%100);
12       }
13 }
```

Clear function

- System("cls").....clear the results screen

Q12

- Write a program in C language to multiply two arrays 4x3, 7*(3x2), enter the values of the arrays randomly between 5 to 15, print the two arrays and the product of multiplication in a form of array.

```c
#include<stdio.h>
#include<stdlib.h>
main()
{
srand(time(NULL));
int a[4][3],b[3][2],p[4][2],i,j,k;
/////  Entering array a items
for(i=0;i<4;i++)
for(j=0;j<3;j++)
{
a[i][j]=rand()%(15-5)+5;
}
///////Entering array b items
for(i=0;i<3;i++)
for(j=0;j<2;j++)
{
b[i][j]=rand()%(15-5)+5;
}
//////// Set intial values = 0 to array p items
for(i=0;i<4;i++)
for(j=0;j<2;j++)
{
p[i][j]=0;
}
for(i=0;i<4;i++)
for(j=0;j<2;j++)
for(k=0;k<3;k++)
{
p[i][j]=p[i][j]+a[i][k]*b[j][k];
}
//////// print array a
for(i=0;i<4;i++)
{
for(j=0;j<3;j++)
{
printf("\t%d",a[i][j]);
}
puts("\n");
}
//////// print array b
for(i=0;i<3;i++)
{
for(j=0;j<2;j++)
{
printf("\t%d",b[i][j]);
}
puts("\n");
}
//////// print array p
for(i=0;i<4;i++)
{
for(j=0;j<2;j++)
{
printf("\t%d",p[i][j]);
}
puts("\n");
}
////////////////////////////////
}
```

Q13

- Write a program in C language to input 5 names through execution, find and print the number of names start by the letter 'm' with its location, also find and print the length of each name and the name itself, using a function.

```
{
if(a[i][0]=='m') c=c+1;
}
for(i=0;i<5;i++)
{
for(j=0;j<10;j++)
{
if(a[i][j]=='\0')
{
l[i]=l[i]+1;
}
}
}
//////////////  print the counter
printf("\n%d",c);
for(i=0;i<5;i++)
{
printf("%s \t Length=%d\n",a[i],l[i]);
}

}
```

Solution Q13

- Using function

```c
#include<stdio.h>
main()
{
int i,c=0;
char a[5][10];
char m;
for(i=0;i<5;i++)
scanf("%s",a[i]);
mm(a);
}

void mm(char g[5][10])
{
int i,c=0;
for(i=0;i<5;i++)
{
if(g[i][0]=='m') c=c+1;
}
printf("\n%d",c);
}
```

Q14

What will be the elements of array (W) after applying the following piece of C program:

```
int W[5][5] , i , j ;
  for ( i=0 ; i <5 ;i++ )
  for ( j=0 ; j <5 ;j++ ) W[i][j]=7;
  for ( i=0 ; i <4 ;i++ )
  for ( j=i+1 ; j <5 ;j++ )
    {
    W[i][j]= 2*i;
    W[j][i]=-2*j;
    }
```

Solution Q14

- 7 0 0 0 0
- -2 7 2 2 2
- -4 -4 7 4 4
- -6 -6 -6 7 6
- -8 -8 -8 -8 7

Functions

- Function is a secondary program executed just when write its name and call it.
- Function like main program where consist from several instructions (statements)….for….loop , if…else…printf
- Syntax of function

- Function_type Function_name (Function_variables)
- {
- Statements
- }

- Function_type int , float , short , long , void

```
#include<stdio.h>
main()
{
int  s;
s=sum(4,8);
printf("The summation=%d", s);
}
int sum(int b,int c)
{
int z;
z=b+c;
return z;
}
```

```c
#include<stdio.h>
float sum(int b);
int fac(int c);
main()
{
float  s;
s=sum(10.0)/fac(5)+sum(15.0)/fac(9);
printf("s=%f", s);
}
////// function to calculate summation
float sum(int b)
{
float z=0,i;
for(i=1;i<=b;i++)
z=z+i;
return z;
}
/////// function to claculate factorial
int fac(int c)
{
int i,f=1;
for(i=1;i<=c;i++)
f=f*i;
return f;
}
```

$$s = \frac{\sum_{i=1}^{10} i}{5!} + \frac{\sum_{i=1}^{15} i}{9!}$$

Some Mathematical instructions

pow(x,y) x^y

abs(x) $|x|$

sqrt(x) $\sqrt[2]{x}$

exp(x) e^x

- z = 5 & 6
- B = 5 << 2
- N = 2 | 6

Q15

- Write a program in C language to enter 5 student's name through the execution, arrange them alphabetically and print these names after arranging.

Solution Q15

```c
#include<stdio.h>
#include<math.h>
main()
{
char a[5][10],Temp[1][10];
int z=0,i,j,k;
for(i=0;i<5;i++)
scanf("%s",&a[i]);


for(i=0;i<5;i++)
for(j=i+1;j<5;j++)
{
if(a[j][0]<a[i][0])
{

for(k=0;k<10;k++)
{
if(a[i][k]!='\0') Temp[0][k]=a[i][k];
if(a[j][k]!='\0') a[i][k]=a[j][k];
if(Temp[0][k]!='\0') a[j][k]=Temp[0][k];
}
}
}
printf("\n\n");
for(i=0;i<5;i++)
printf("%s\n",a[i]);
getche();
}
```

Q16

Write a program in C language to sum two arrays 3x3, $\sqrt[2]{7}x(3x3)$ enter the values of the arrays randomly between 1 to 100, print the two arrays and the product of summation in a form of array, using function to enter and print the elements of the arrays.

Solution of Q16

```c
#include<stdio.h>
#include<math.h>
main()
{
int a[3][3],b[3][3],c[3][3],i,j;
////   enter the items of array a and b randomly using function
enter(a);
enter(b);
////   calculate the items of array c in main program
for(i=0;i<3;i++)
for(j=0;j<3;j++)
c[i][j]=a[i][j]+b[i][j]*sqrt(7);
///  print the array a, b and c using function
print(a);
puts("\n");
print(b);
puts("\n");
print(c);
}
//// function for printing an array
void print(int x[3][3])
{
int i,j;
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
printf("%d\t",x[i][j]);
puts("\n");
}
}
/////////  function for entering items of array
void enter(int k[3][3])
{
int i,j;
srand(time(NULL));
for(i=0;i<3;i++)
for(j=0;j<3;j++)
k[i][j]=rand()%(10-5)+5;
}
```

Q17

- **Write a program in C language to enter the items of two arrays a[10] and b[10] randomly between 5 to 10 using function, arrange the items of array a ascending and items of array b descending using same function then the main program will create a new array c by the adding the two arrays then calculate the summation of items of array c and b using a function then print the array a, b, c using a function and print the summation of items of array c and b in the main program.**

Solution of Q17

```c
#include<stdio.h>
int sum(int g[10]);
main()
{
int a[10],b[10],c[10],i,n,m;
////   enter the items of array a and b randomly using function
enter(a);
enter(b);
///   arrange the items of array a and b usinf function
arrange(a,1);
arrange(b,2);
////   calculate the items of array c in main program
for(i=0;i<10;i++)
c[i]=a[i]+b[i];
////  calculate the summation of items of array c and b using a function
n=sum(c);
m=sum(b);
//////////////
printf("Summation of array c items=%d\n",n);     /////print the summation of array c items in main program
printf("Summation of array b items=%d\n\n",m);     /////print the summation of array b items in main program
///  print the array a, b and c using function
print(a);
puts("\n");
print(b);
puts("\n");
print(c);
}
//// function for printing an array
void print(int x[10])
{
int i;
for(i=0;i<10;i++)
printf("%d\n",x[i]);
}
////// function for summation of two array
```

Complement of Solution Q17

```
36  int sum(int g[10])
37  {
38  int w=0,i;
39  for(i=0;i<10;i++)
40  w=w+g[i];
41  return(w);
42  }
43  /////////  function for entering items of array
44  void enter(int k[10])
45  {
46  int i;
47  srand(time(NULL));
48  for(i=0;i<10;i++)
49  k[i]=rand()%(10-5)+5;
50  }
51  ///// function for arranging of an array
52  void arrange(int h[10],int p)
53  {
54  int i,j,temp;
55  for(i=0;i<10;i++)
56  for(j=i+1;j<10;j++)
57  {
58  if(p==1)
59  if(h[i]>h[j])
60  {
61  temp=h[i];
62  h[i]=h[j];
63  h[j]=temp;
64  }
65  if(p==2)
66  if(h[i]<h[j])
67  {
68  temp=h[i];
69  h[i]=h[j];
70  h[j]=temp;
71  }
72  }
73  }
```

Calling function inside function

Q18

- Write a program in C language to calculate the value of S in main program using function(secondary program) to calculate the summation and factorial also use calling function inside function to calculate the factorial.

$$s = \frac{\sum_{i=1}^{10} i * 2!}{5!} + \frac{\sum_{i=1}^{15} i * 4!}{9!}$$

Solution Q18

```c
#include<stdio.h>
float sum(int b,int s);
int fac(int c);
main()
{
float  s;
s=sum(10.0,2)/fac(5)+sum(15.0,4)/fac(9);
printf("s=%f", s);
}
////// function to calculate summation
float sum(int b,int d)
{
float z=0,i;
int q;
q=fac(d);
for(i=1;i<=b;i++)
z=z+i*q;
return z;
}
/////// function to claculate factorial
int fac(int c)
{
int i,f=1;
for(i=1;i<=c;i++)
f=f*i;
return f;
}
```

Q19

Q2) Write a program in C language to enter the items of two arrays a[10] and b[10] randomly between 25 to 75, then the program will create a new array c as shown below:

C[a0 b9 a1 b8 a2b7 a3 b6 a4 b5 a5 b4 a6 b3 a7 b2 a8 b1 a9 b0]

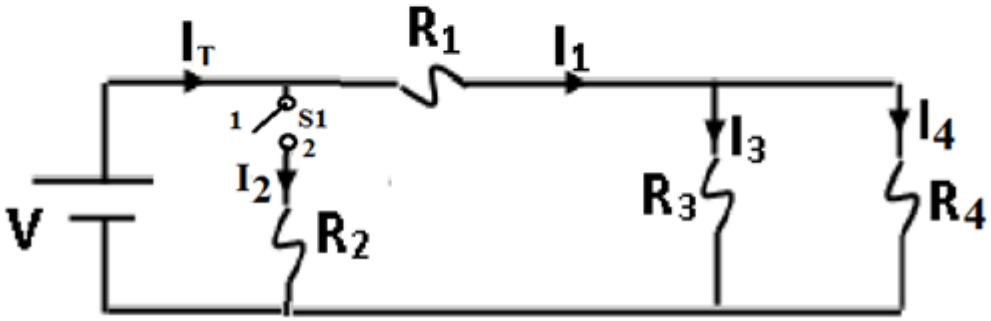
arrange the items c0 to c9 of array c ascending and the items c10 to c19 descending, then print the new array c.

Solution Q19

```c
#include<stdio.h>
main()
{
int a[10],b[10],c[20],i,j,k=0,m=9,max,min;
srand(time(NULL));
for(i=0;i<10;i++)
{
a[i]=rand()%(75-25)+25;
b[i]=rand()%(75-25)+25;
}
for(j=0;j<20;j=j+2)
{
c[j]=a[k];
c[j+1]=b[m];
k++;
m--;
}
for(i=0;i<20;i++)
for(j=i+1;j<20;j++)
{
if(i<=9)
if(c[i]>c[j])
{
min=c[i];
c[i]=c[j];
c[j]=min;
}
if(i>=10)
if(c[i]<c[j])
{
max=c[i];
c[i]=c[j];
c[j]=max;
}
}
for(i=0;i<20;i++)
printf("%d\n",c[i]);
}
```

Q20

Q4) For the following circuits below, write a program in C language to calculate I_2 ,I_3 ,I_4 , I_T, V when the switch S_1 at position 1 OR position 2, choose the case of switch s1 through the execution of program using switch…case instruction, enter the values of R_1, R_2, R_3 and R_4 through the execution if you know I_1 = 0.65 A then print the values of I_2 ,I_3 ,I_4 , I_T, V according to switch case.



Solution Q20

```c
#include<stdio.h>
main()
{
int S;
float R1,R2,R3,R4,I1=0.65,I2,I3,I4,It,V,i,j;
puts("Enter the values of R1,R2,R3 and R4");
scanf("%f %f %f %f",&R1,&R2,&R3,&R4);
puts("Enter the switch S1 case 1  or 2");
scanf("%d",&S);
switch(S)
{
case 1:
I2=0;
I3=I1*R4/(R4+R3);
I4=I1*R3/(R4+R3);
It=I1;
V=I1*R1+I3*R3;
break;
case 2:
I2=((R3*R4)/(R4+R3)+R1)*I1/R2;
I3=I1*R4/(R4+R3);
I4=I1*R3/(R4+R3);
It=I1+I2;
V=I2*R2;
break;
default :
puts("You enter the wrong case of switch");
}
printf(" I2=%f \n I3=%f \n I4=%f \n It=%f \n  V=%f",I2,I3,I4,It,V);}
```

Q21

B) What will be the elements of array w after applying the following piece of C program.

```
int  w[8] , k , n=3 ;
for( k=0 ; k<8 ; k++)
w[k] = n & 6 ;
for( k=0 ; k<8 ; k++)
{
if( k==n ) w[k-2] = n-- ;
if( k==2 ) w[k+1] = n++ ;
if( ( k==6 ) || ( k==4 ) ) w[k] = 13%5;
```

Solution Q21

First loop

2
2
2
2
2
2
2
2
2

second loop

The result

2
2
4
3
3
2
3
2

Q11

- **Wire a program in C language to calculate and print the following series, enter the values of X through the execution.**

$$z = \frac{X^1 * 4!}{\sum_{i=1}^{2} i} + \frac{X^5 * 8!}{\sum_{i=1}^{6} i} + \frac{X^9 * 12!}{\sum_{i=1}^{10} i} \cdots\cdots\cdots \frac{X^{17} * 20!}{\sum_{i=1}^{18} i}$$

- **Write a program in C language to enter an array a[10] through execution. Calculate and print the summation of array's when press 1 from keyboard. Find and print the maximum and minimum number among the array's items when press 2 with explanation expression. Exchange the number in the array with 1 if it is equal to 2 when press 3 using switch case where input the selection through execution.**
-

- Write a program in C language to input 5 names through execution, find and print the number of names start by the letter 'm' with its location, also find and print the length of each name and the name itself.