EMBEDDED SYSTEMS INTERNSHIP

Microcontroller-Based Washing Machine Simulation Using Picsimlab

Team Members:

R.GANESH
Sk.ARIEF
K.CHANDANA
P.GURUSAI

Project Mentor:

JAYALAXMI N.DHANYAL

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING(ECE)
3rd years of SRI VENKATESWARA ENGINEERING COLLEGE, Tirupati,517507.

USE OF C LANGUAGE

1.C programming is the Basic language.

2.Used in software Development.

3.Used in Embedded software Development.

4.Efficiency is high while comparing to remaining languages.

5.OS kernel Development.

Code Structure

```
#include<stdio.h>                     ⇐Preprocessor Directive

int main()                            ⇐The start of program

{

    /* To Display Hello World */      ⇐comment

    printf(“ HelloWorld \n ”);        ⇐statement

    return 0;                         ⇐return statement

}
```

Number Systems conversions

1.A number is generally represented as

- ➢ Decimal
- ➢ Octal
- ➢ Hexadecimal
- ➢ Binary

Type	Range (8 bits)
Decimal	0 - 255
Octal	000 - 0377
hexadecimal	0x00 - 0xFF
Binary	0b00000000 - 0b11111111

Data Representation

1.Bit Representation: High & Low states such as 1 & 0.

2.Byte Representation: A unit of Digital Information.

Commonly 1 byte = 8 bits

3.Word Representation: An Amount of data that a machine can fetch and process at one time.

Example: An integer no of bytes is 1,2,4,8.

For 32 bit chip has a 32 bits(4 bytes) word size

4.Integer Number - positive

Mathematically: $-k=2^n - k$

1st bit in 8 bits represents positive or negative integer.

What is an Embedded System ?

1.It is the combination of Hardware and software which is designed to perform a specific task.

2.Hardware components has become cheap.

3.They are available in abundance.

4.Components have been miniaturized.

Examples for Embedded system: Washing Machine

Microwave oven

Smart Refrigerators

WIFI modem

Wearables & Hospital Equipments: Smart watches, Fitness devices, ECG.

Commonly Seen Components

1.ROM - Read Only Memory / Primary storage device

2. RAM - Random Access Memory / Secondary storage device

3.Microprocessor

4.Microcontroller

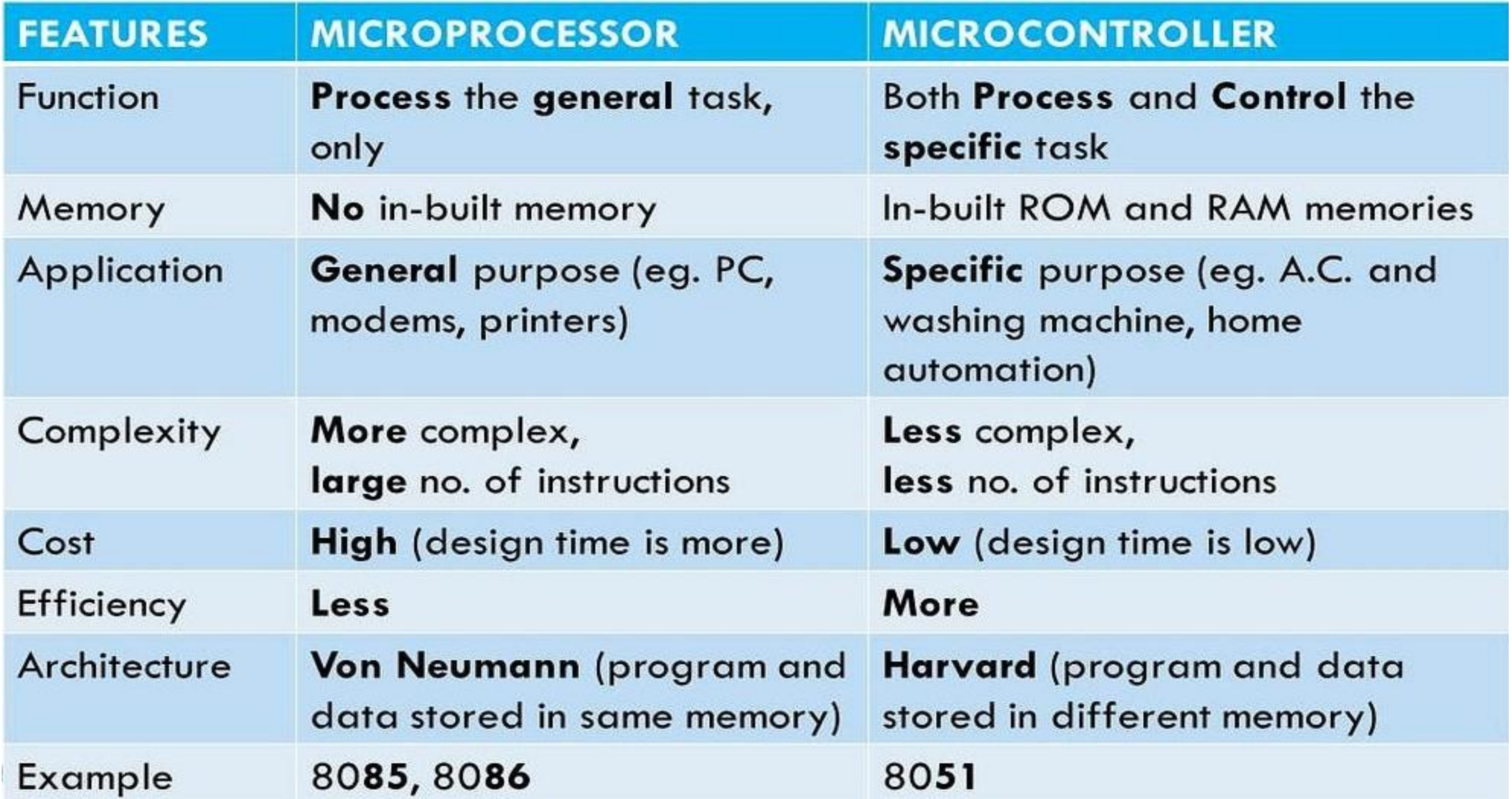
5.Actuators

6.Sensors

7.Power supply

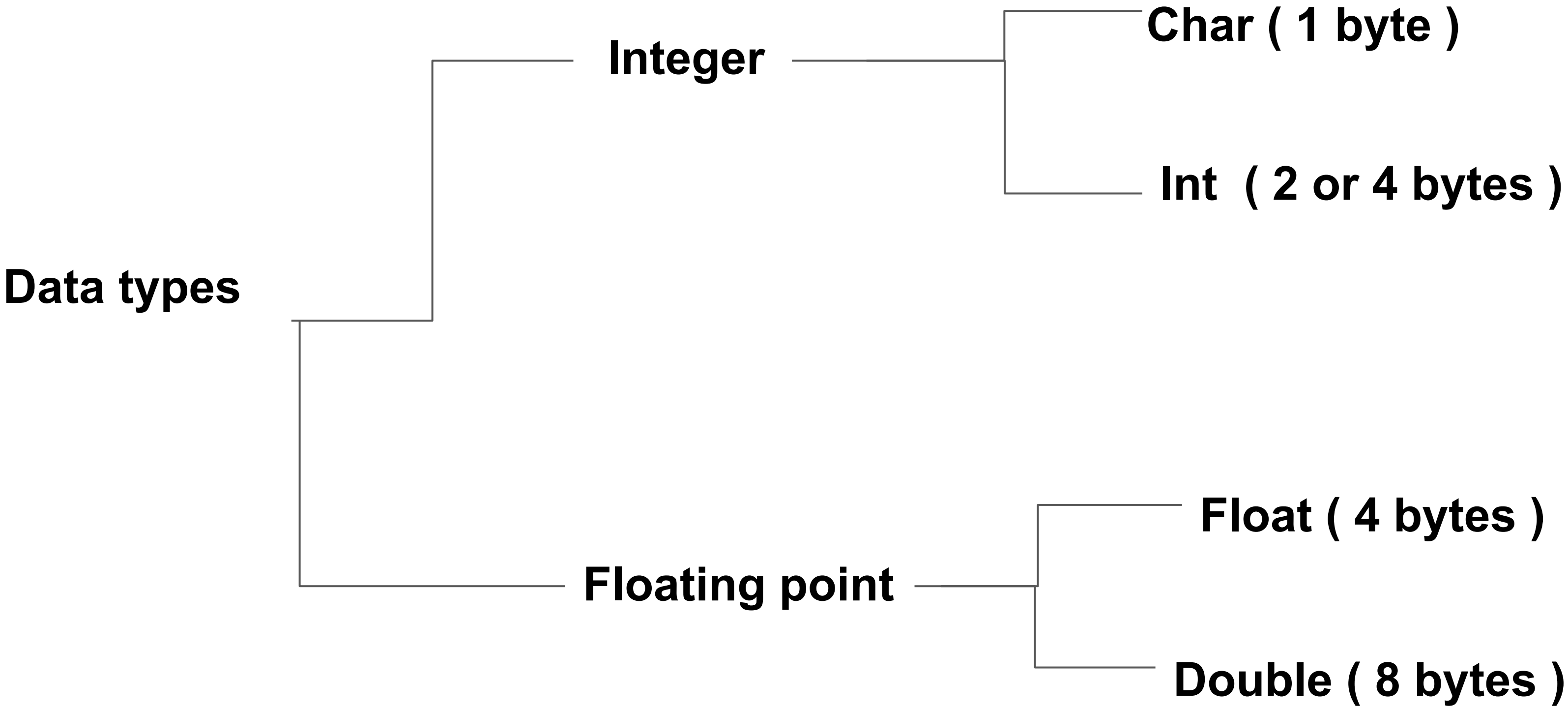
8.Motors

Difference Between Microprocessor & Microcontroller



FEATURES	MICROPROCESSOR	MICROCONTROLLER
Function	**Process** the **general** task, only	Both **Process** and **Control** the **specific** task
Memory	**No** in-built memory	In-built ROM and RAM memories
Application	**General** purpose (eg. PC, modems, printers)	**Specific** purpose (eg. A.C. and washing machine, home automation)
Complexity	**More** complex, **large** no. of instructions	**Less** complex, **less** no. of instructions
Cost	**High** (design time is more)	**Low** (design time is low)
Efficiency	**Less**	**More**
Architecture	**Von Neumann** (program and data stored in same memory)	**Harvard** (program and data stored in different memory)
Example	8085, 8086	8051

Basic Data types



Data types

Integer

Char (1 byte)

Int (2 or 4 bytes)

Floating point

Float (4 bytes)

Double (8 bytes)

Conditional Constructs

1. Conditional constructs are classified into 2 types

- Single iteration
- Multi iteration

2.Single iteration

- If
- If else
- Nested if else
- Switch case

3.Multi iteration

- For
- While
- Do while

Single iteration

1.If statement syntax:

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

2.If else statement syntax:

```
if (condition) {
  // block of code to be executed if the condition is true
}
Else{
    //block of code to be executed if the condition is false
}
```

Single iteration

3.Else if statement syntax:

```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2 is true
} else {
  // block of code to be executed if the condition1 is false and condition2 is false
}
```

Single iteration

4. Switch case statement syntax:

```
switch (expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
```

Multi iteration

1.For loop statement syntax:

```
for (expression 1; expression 2; expression 3) {
  // code block to be executed
}
```

2.while loop statement syntax:

```
while (condition) {
  // code block to be executed
}
```

Multi iteration

3.do while loop statement syntax:

```
do {
  // code block to be executed
}
while (condition);
```

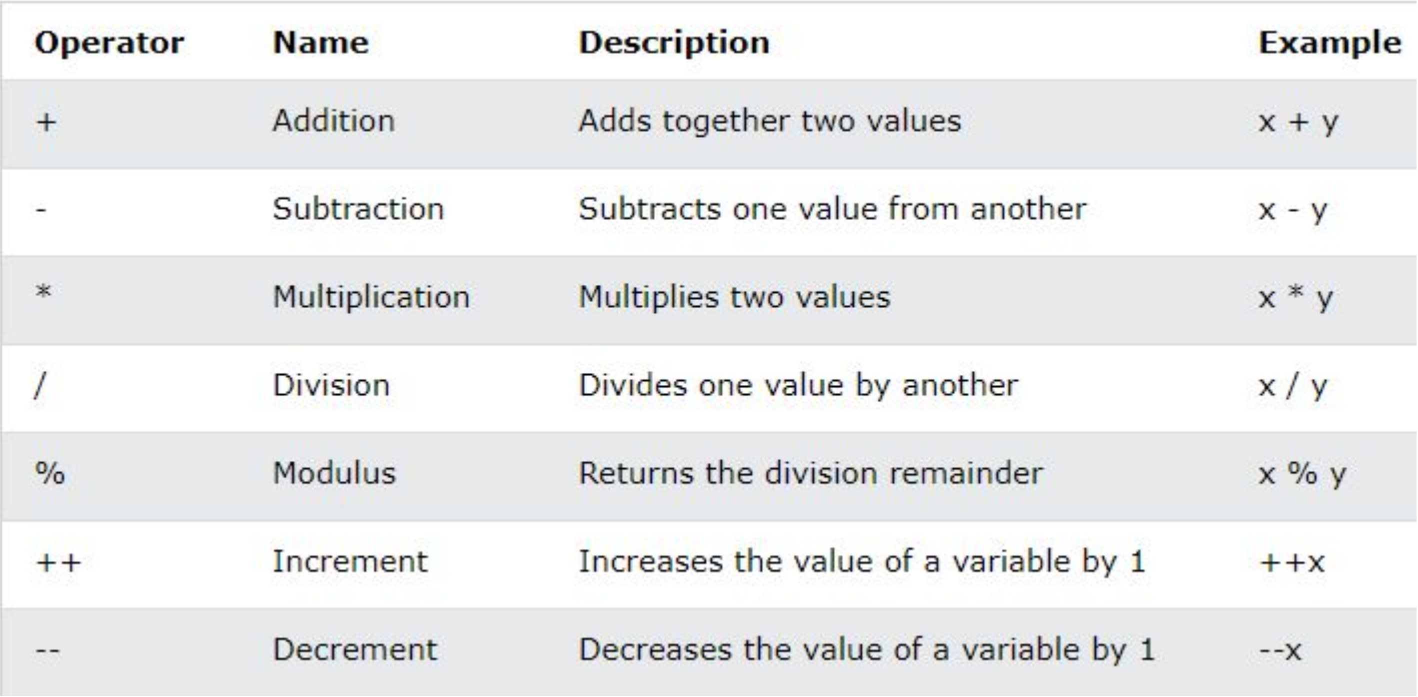
Operators

C divides the operators into the following groups:

- **Arithmetic operators**
- **Assignment operators**
- **Comparison operators**
- **Logical operators**
- **Bitwise operators**
- **Ternary operator**

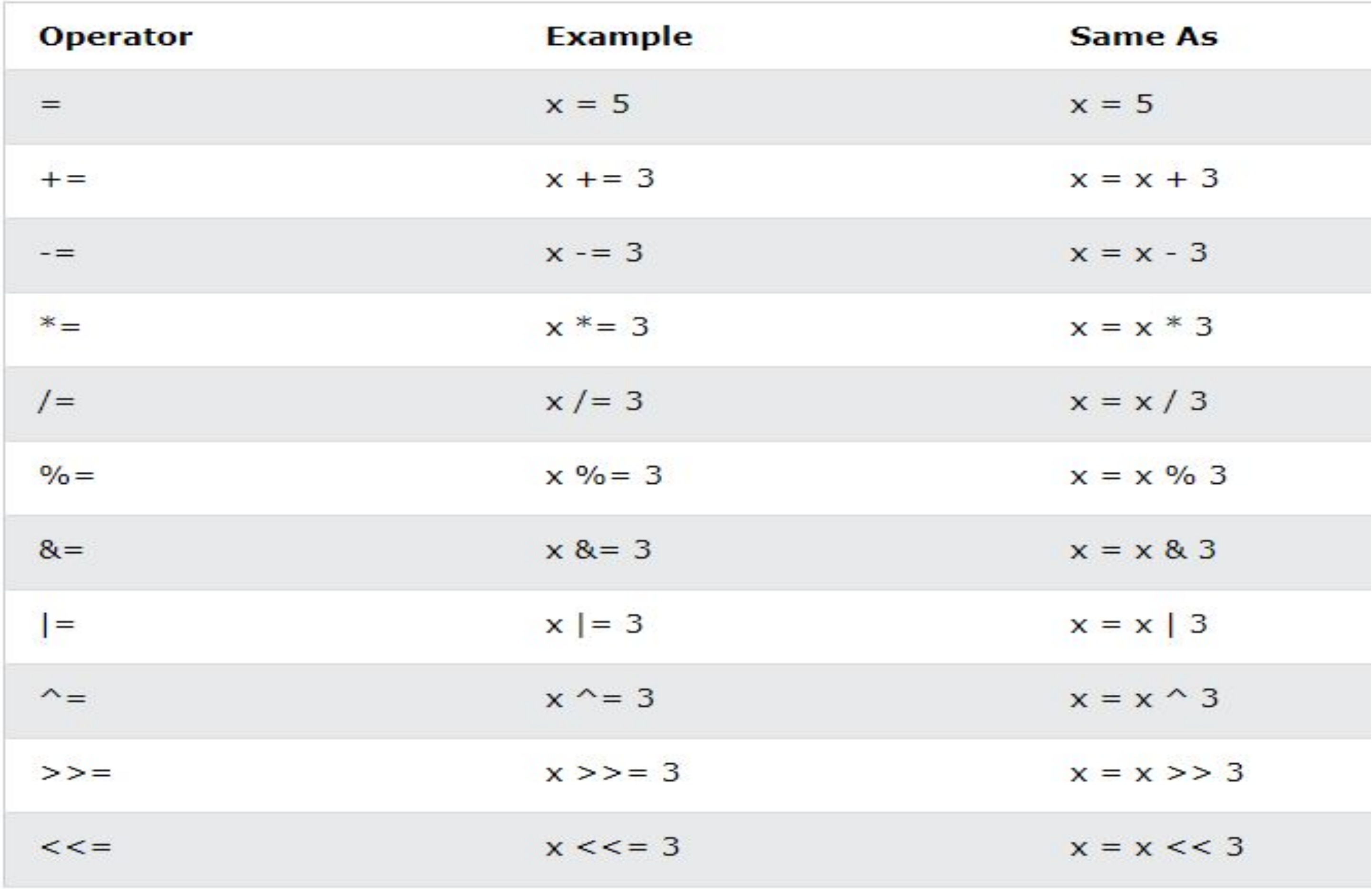
Arithmetic operators

Arithmetic operators are used to perform common mathematical operations.



Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

Assignment operators



Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

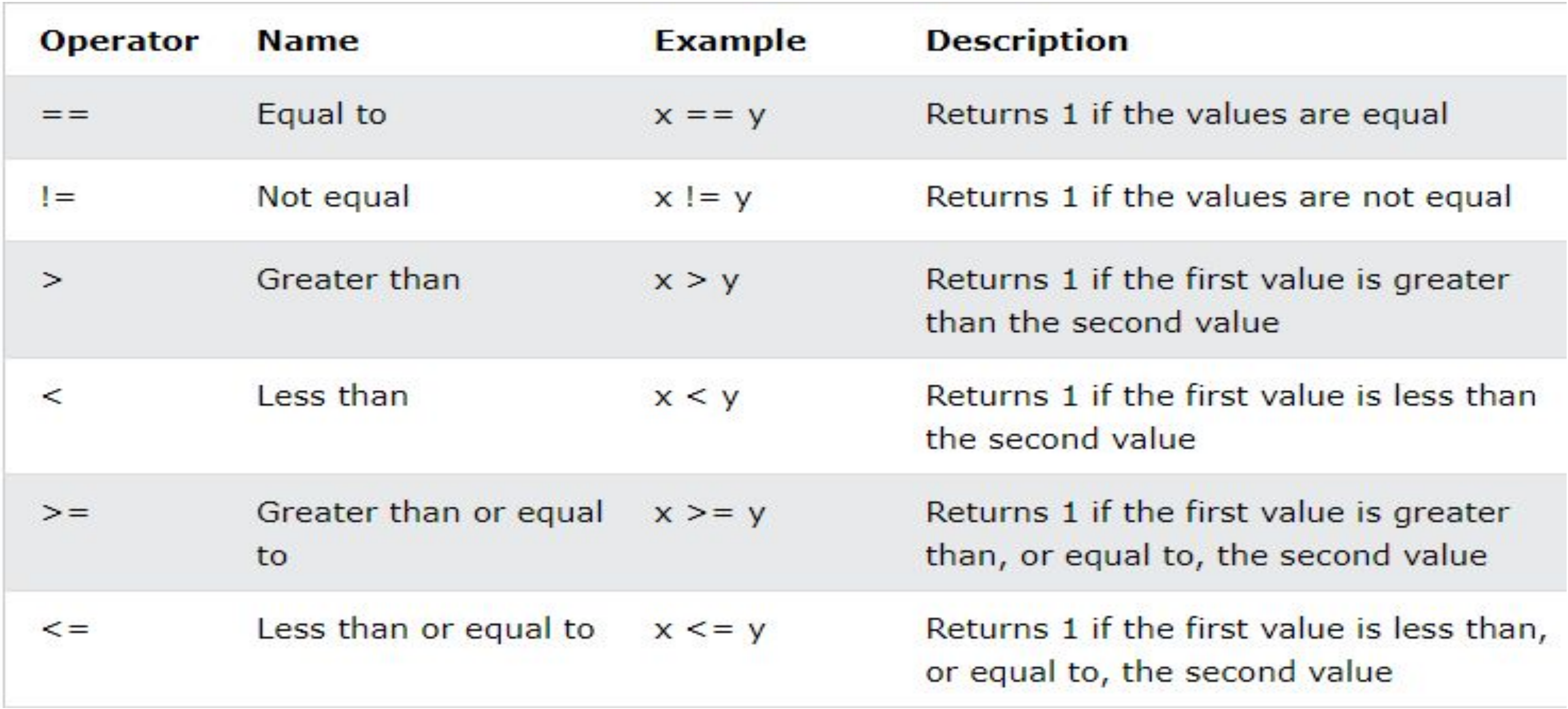
Comparison Operators

1.Comparison operators are used to compare two values (or variables).

2.This is important in programming, because it helps us to find answers and make decisions.

3.The return value of a comparison is either 1 or 0, which means true (1) or false (0).

Comparison Operators

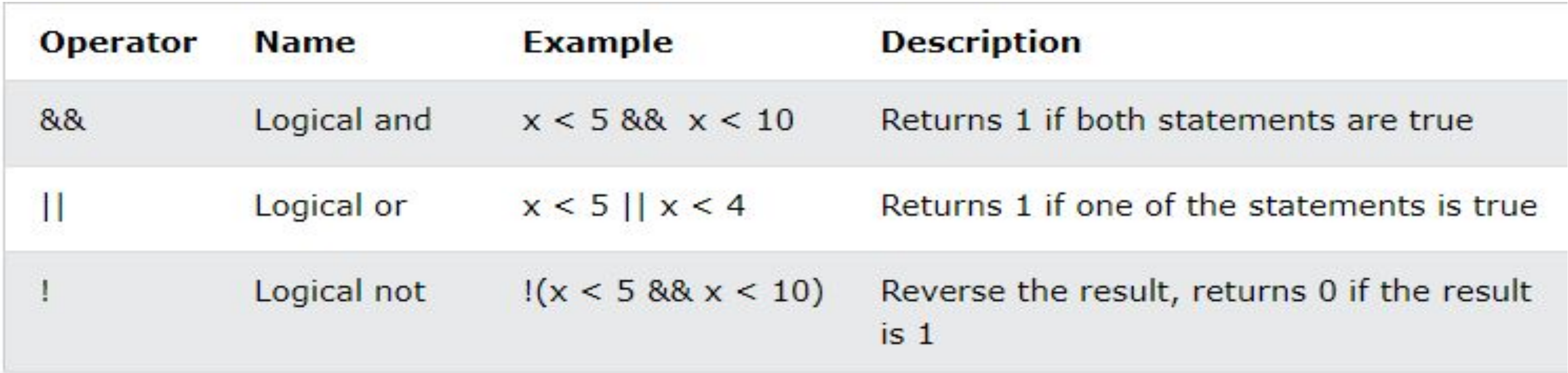


Operator	Name	Example	Description
==	Equal to	x == y	Returns 1 if the values are equal
!=	Not equal	x != y	Returns 1 if the values are not equal
>	Greater than	x > y	Returns 1 if the first value is greater than the second value
<	Less than	x < y	Returns 1 if the first value is less than the second value
>=	Greater than or equal to	x >= y	Returns 1 if the first value is greater than, or equal to, the second value
<=	Less than or equal to	x <= y	Returns 1 if the first value is less than, or equal to, the second value

Logical Operators

1.You can also test for true or false values with logical operators.

2.Logical operators are used to determine the logic between variables or values:



Operator	Name	Example	Description
&&	Logical and	x < 5 && x < 10	Returns 1 if both statements are true
||	Logical or	x < 5 || x < 4	Returns 1 if one of the statements is true
!	Logical not	!(x < 5 && x < 10)	Reverse the result, returns 0 if the result is 1

Bitwise operator

- 6 operators are bitwise operators (also known as bit operators as they work at the bit-level).

1. The & (bitwise AND) in C takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1.

2.The | (bitwise OR) in C takes two numbers as operands and does OR on every bit of two numbers. The result of OR is 1 if any of the two bits is 1.

3.The ^ (bitwise XOR) in C takes two numbers as operands and does XOR on every bit of two numbers. The result of XOR is 1 if the two bits are different.

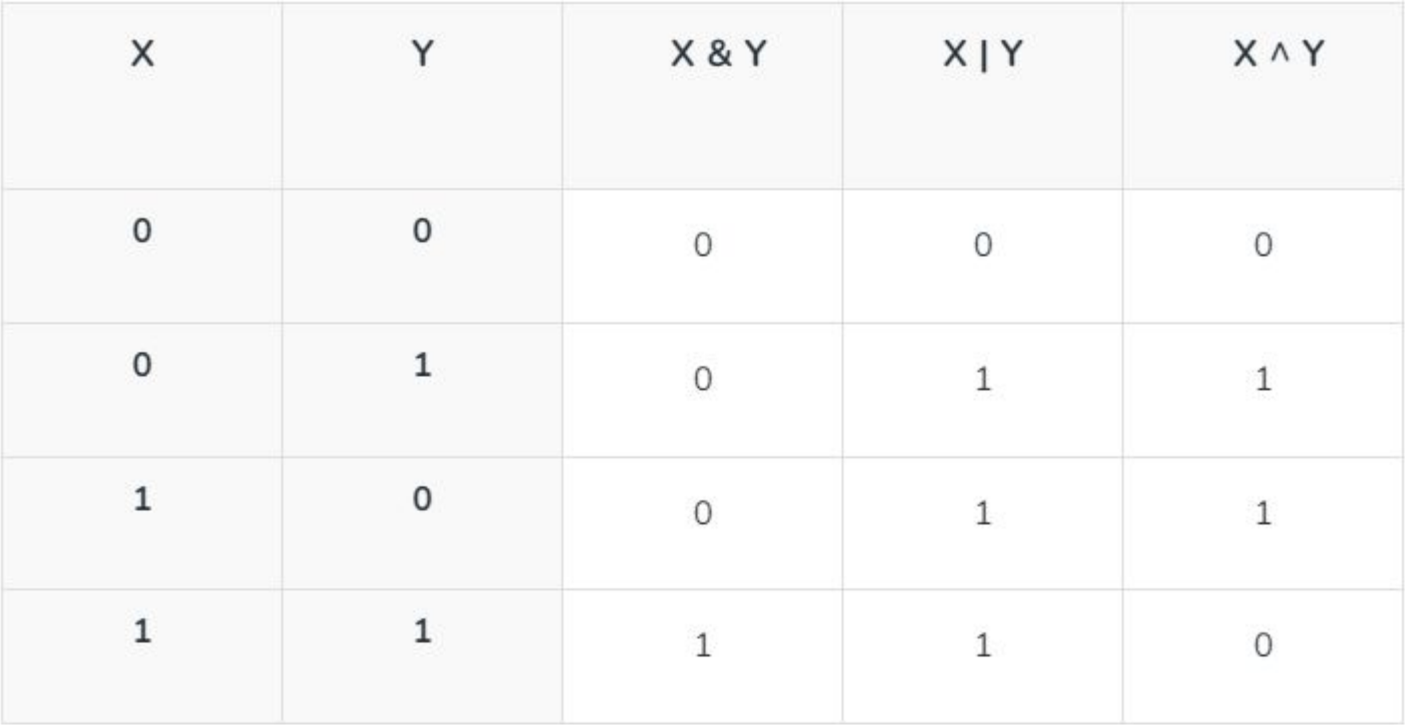
Bitwise operator

4.The << (left shift) in C takes two numbers, the left shifts the bits of the first operand, and the second operand decides the number of places to shift.

5.The >> (right shift) in C takes two numbers, right shifts the bits of the first operand, and the second operand decides the number of places to shift.

6.The ~ (bitwise NOT) in C takes one number and inverts all bits of it.

Bitwise operator



X	Y	X & Y	X | Y	X ^ Y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Jump Statements

The jump statements are continue & Break.

1.Break syntax

```
int i;
for (i = 0; i < 10; i++) {
  if (i == 4) {
    break;
  }
  printf("%d\n", i);
}
```

Jump Statements

2.Continue syntax:

```
int i;
for (i = 0; i < 10; i++) {
 if (i == 4) {
   continue;
 }
 printf("%d\n", i);
}
```

Arrays

1.Arrays is the collection of Homogeneous elements.

2.Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

3.To create an array, define the data type (like int) and specify the name of the array followed by square brackets [ ].

4.Syntax of an Array is Data_type array_name[size_of_an_array]

Example program:

```
int myNumbers[] = {25, 50, 75, 100};

printf("%d", myNumbers[0]);

// Outputs 25
```

Pointers

1.A pointer is a variable that stores the memory address of another variable as its value.

2.Syntax for pointer: datatype * *ptr*;

Example program:

```
#include <stdio.h>
int main(){
    int x;
    int *ptr;
    x=5;
    *ptr=5;
    return 0;
}
```

Functions

1.A function is a block of code which only runs when it is called.

2.You can pass data, known as parameters, into a function.

3.Functions are used to perform certain actions, and they are important for

4.reusing code: Define the code once, and use it many times.

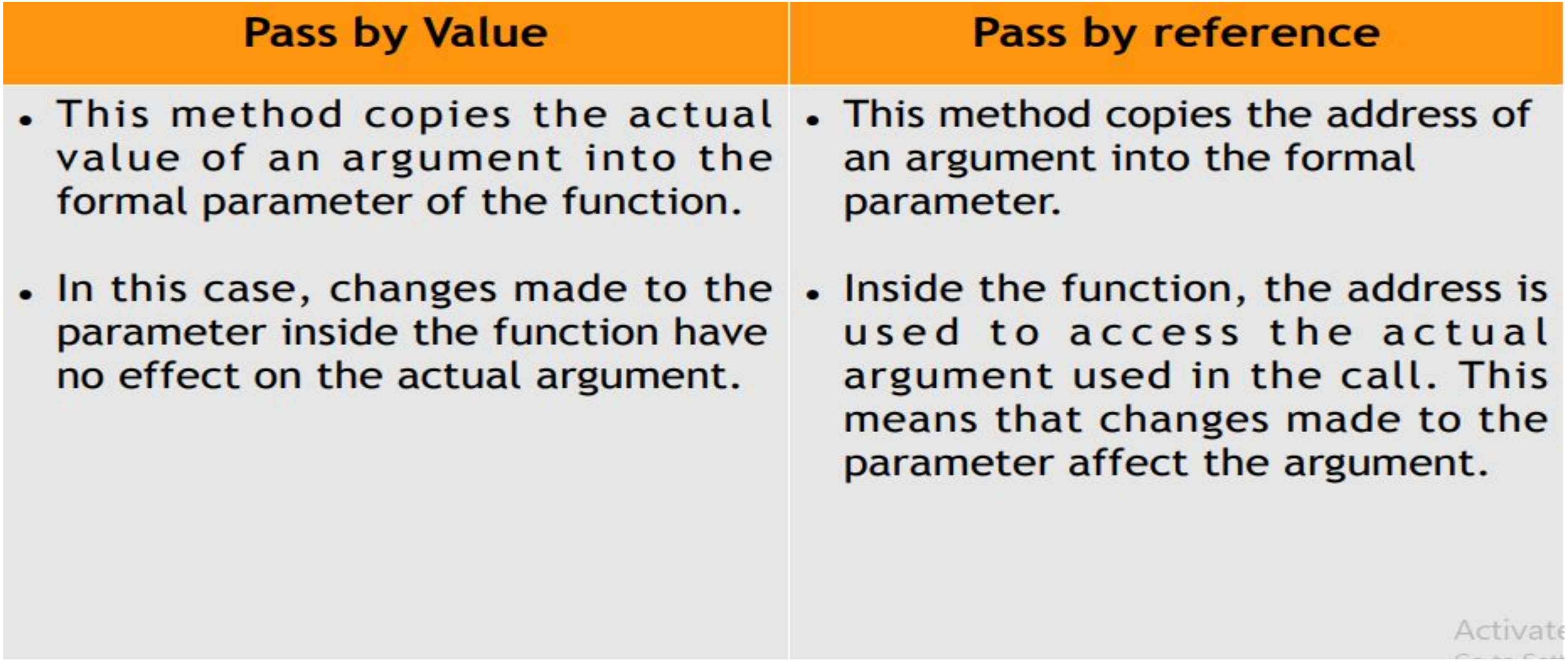
5.Re usability Functions can be stored in library & re-used When some specific code is to be used more than once, at different places, functions avoids repetition of the c.

```
Syntax:void myFunction() {

          // code to be executed

          }
```

Difference Between Call by value & Call by Reference



Pass by Value	Pass by reference
• This method copies the actual value of an argument into the formal parameter of the function. • In this case, changes made to the parameter inside the function have no effect on the actual argument.	• This method copies the address of an argument into the formal parameter. • Inside the function, the address is used to access the actual argument used in the call. This means that changes made to the parameter affect the argument.

Recursive Function

1.Recursion is the technique of making a function call itself.

2.This technique provides a way to break complicated problems down into simple problems which are easier to solve.

Example program:

```
#include<stdio.h>

int sum(int k);

int main() {

  int result = sum(3);

  printf("%d", result);

  return 0;

 }
```

Recursive Function

```
int sum(int k) {
  if (k > 0) {
    return k + sum(k - 1);
  } else {
    return 0;
  }
}
// output is 6
  3*2*1= 6.
```

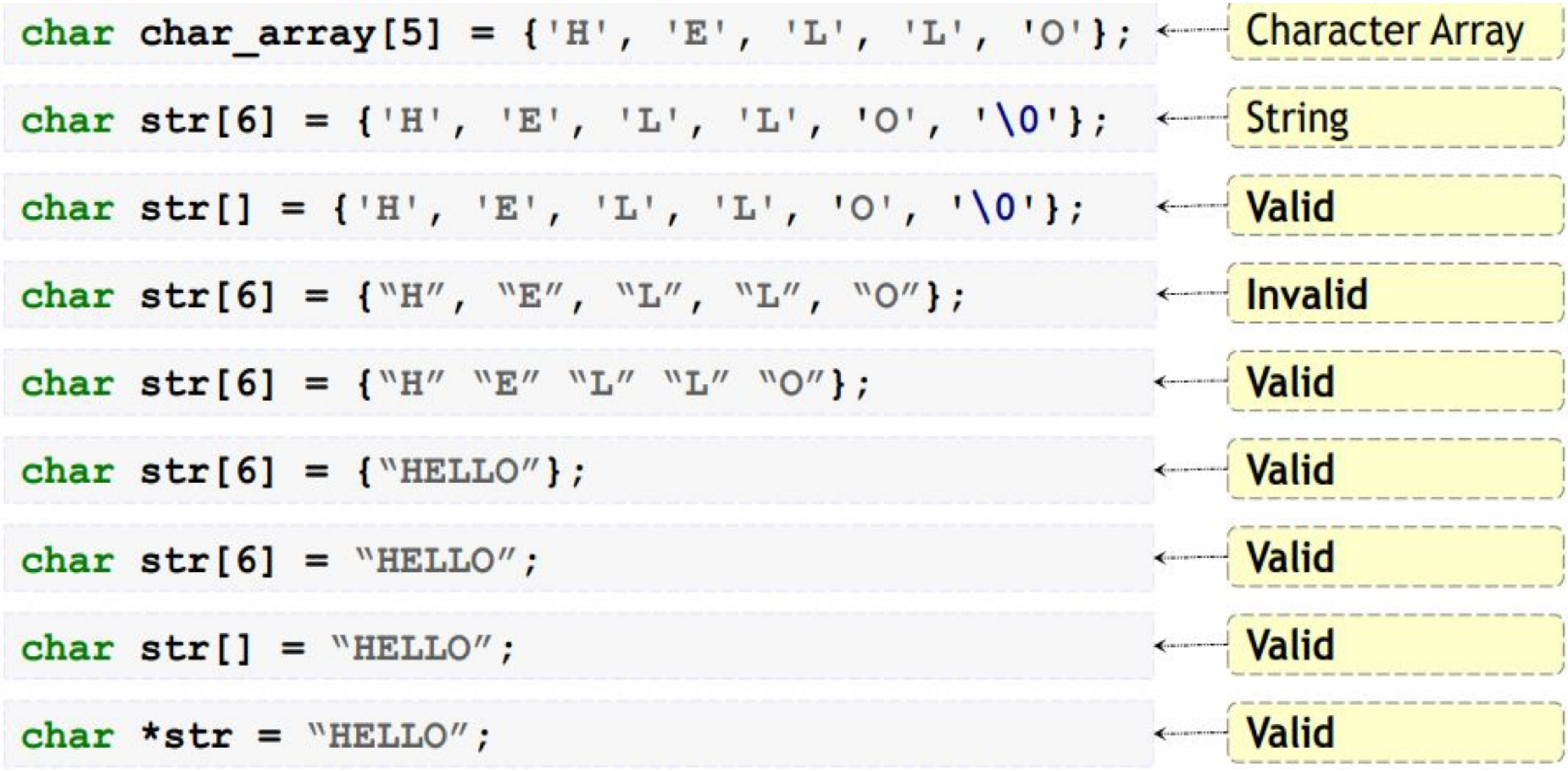
Strings

1.A String is a word in which group of letters.

2.Strings are used for storing text/characters.

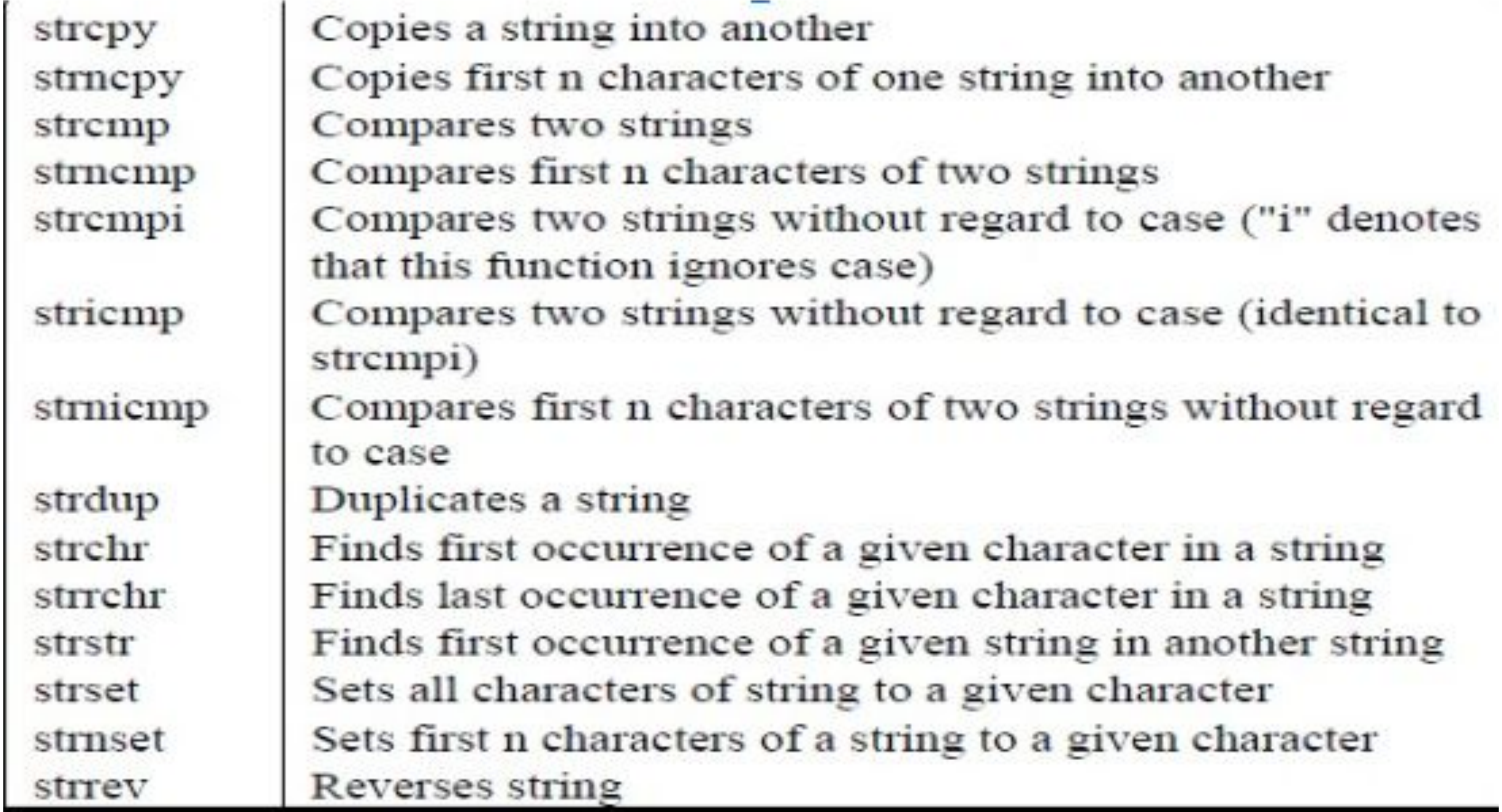
For example, "Hello World" is a string of characters.

Types of initialization of string



`char char_array[5] = {'H', 'E', 'L', 'L', 'O'};`	Character Array
`char str[6] = {'H', 'E', 'L', 'L', 'O', '\0'};`	String
`char str[] = {'H', 'E', 'L', 'L', 'O', '\0'};`	Valid
`char str[6] = {“H”, “E”, “L”, “L”, “O”};`	Invalid
`char str[6] = {“H” “E” “L” “L” “O”};`	Valid
`char str[6] = {“HELLO”};`	Valid
`char str[6] = “HELLO”;`	Valid
`char str[] = “HELLO”;`	Valid
`char *str = “HELLO”;`	Valid

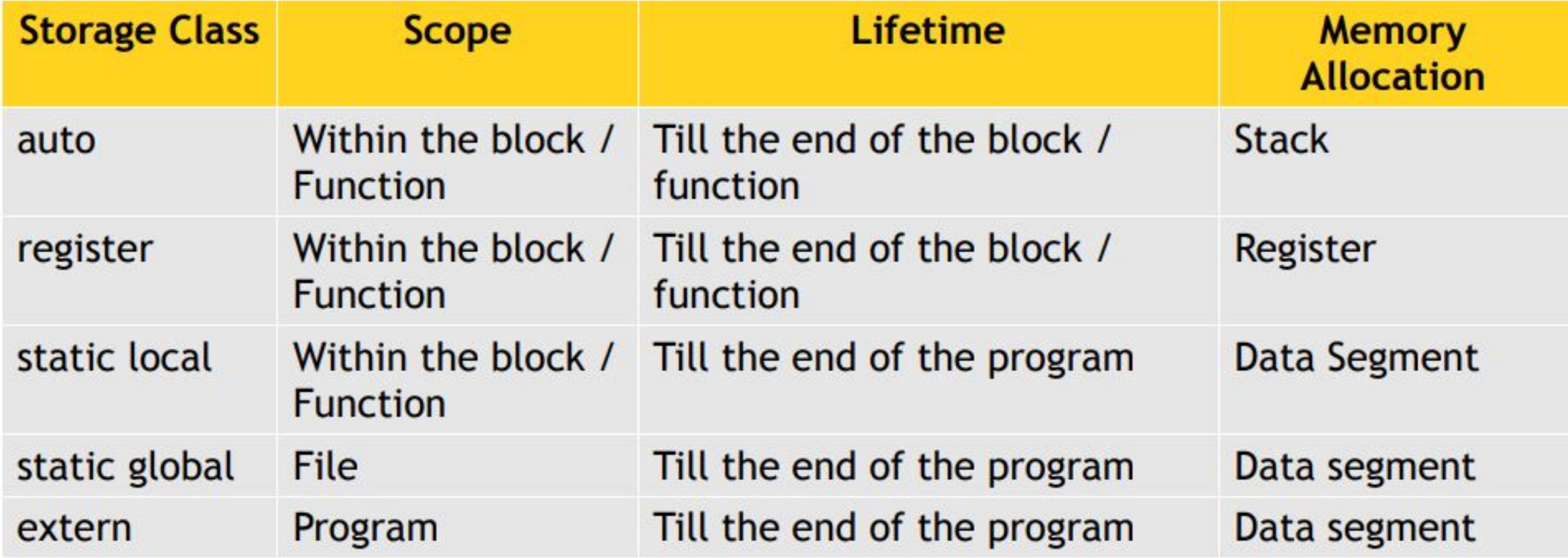
Library Functions of String



strcpy	Copies a string into another
strncpy	Copies first n characters of one string into another
strcmp	Compares two strings
strncmp	Compares first n characters of two strings
strcmpi	Compares two strings without regard to case ("i" denotes that this function ignores case)
stricmp	Compares two strings without regard to case (identical to strcmpi)
strnicmp	Compares first n characters of two strings without regard to case
strdup	Duplicates a string
strchr	Finds first occurrence of a given character in a string
strrchr	Finds last occurrence of a given character in a string
strstr	Finds first occurrence of a given string in another string
strset	Sets all characters of string to a given character
strnset	Sets first n characters of a string to a given character
strrev	Reverses string

Storage Classes

We have 4 types of storage classes, they are



Storage Class	Scope	Lifetime	Memory Allocation
auto	Within the block / Function	Till the end of the block / function	Stack
register	Within the block / Function	Till the end of the block / function	Register
static local	Within the block / Function	Till the end of the program	Data Segment
static global	File	Till the end of the program	Data segment
extern	Program	Till the end of the program	Data segment

Architecture

1.Von Neumann Architecture

2.Harvard Architecture

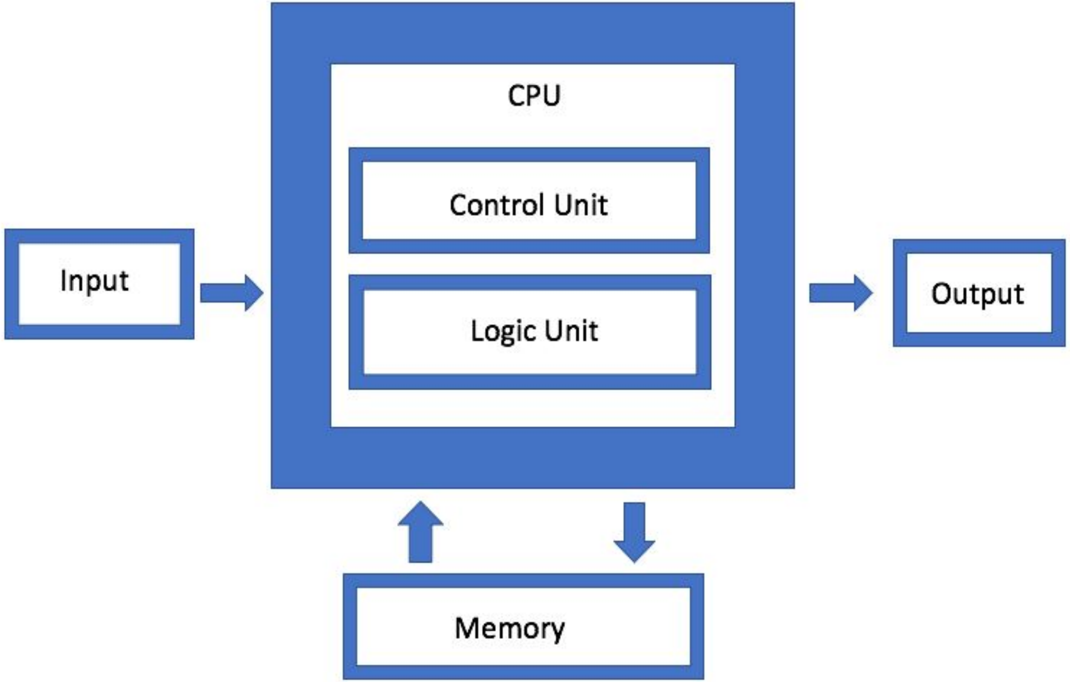


Fig-1: Von Neumann Architecture

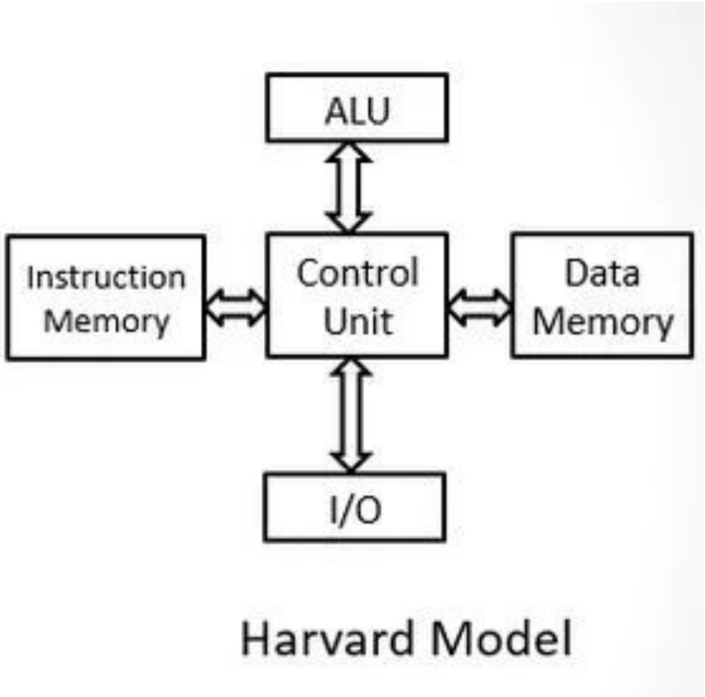


Fig-2: Harvard Architecture

Interfacing of LED to Microcontroller

1.We can interface the LED’s to the microcontroller

2.LED - Light Emitting Diode

3.Interfacing of LEDs are 2 types they are

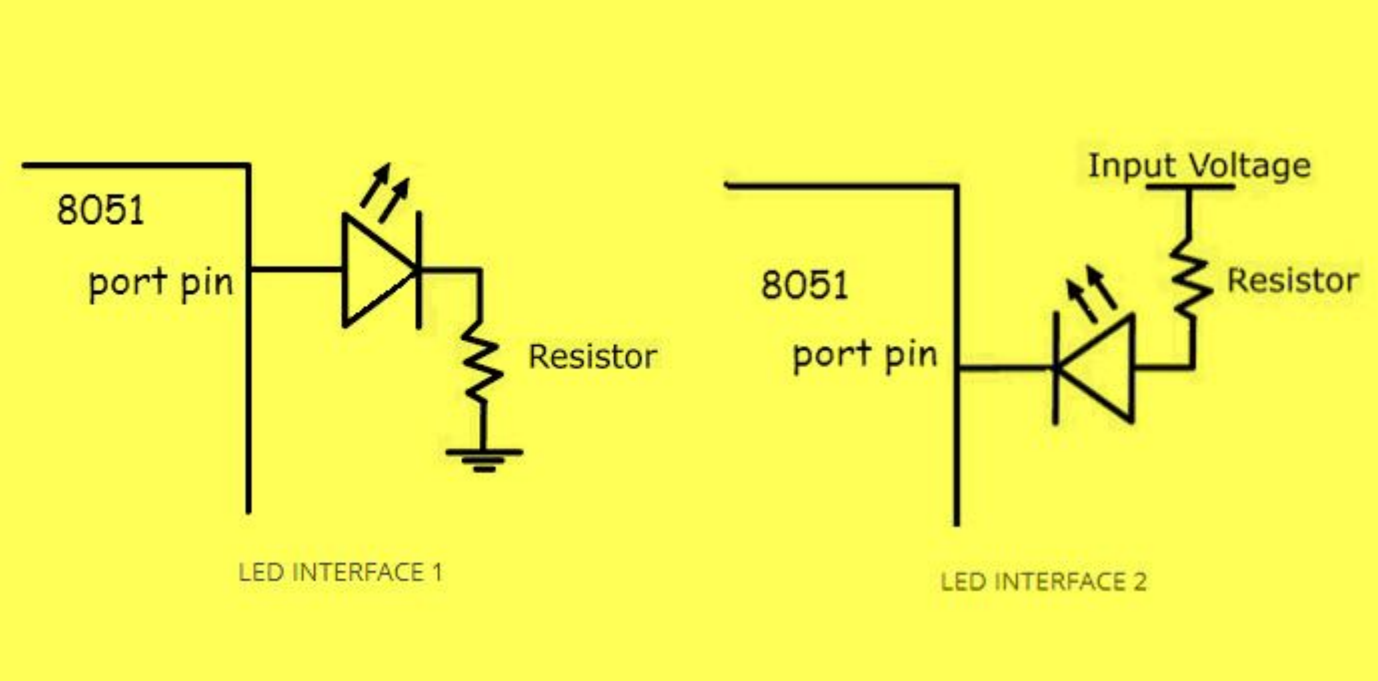
- Sourcing Circuit
- Sinking Circuit

4. Circuit follows Forward Bias to ON the circuit.

5. Circuit follows Reverse Bias to OFF the circuit.

Sourcing Circuit & Sinking Circuit

Pin	LED
1	ON
0	OFF



Pin	LED
1	OFF
0	ON

1. **Sourcing Circuit**

2. **Sinking Circuit**

Interfacing of Switch to Microcontroller

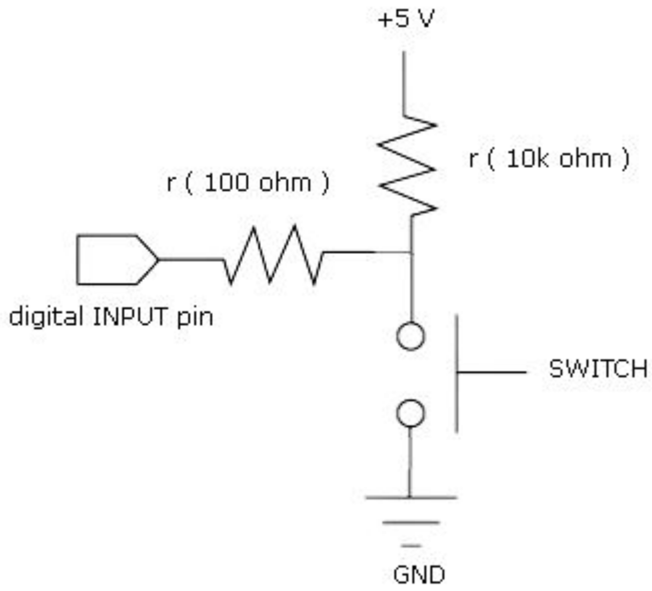
1.We can interface the switch to the microcontroller.

2.Interfacing of switch are 2 types they are

- Pull up circuit
- Pull Down circuit

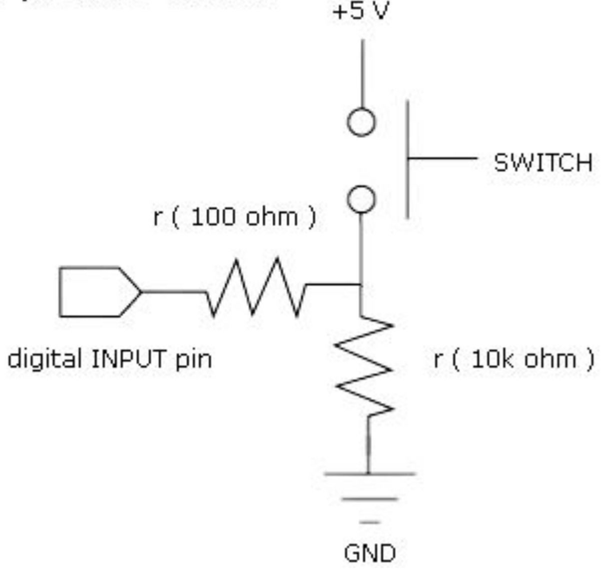
Pull up circuit & Pull Down circuit

Switch with "pull-up" resistor



Pin	Switch
0	Pressed
1	Released

Switch with "pull-down" resistor



Pin	Switch
0	Released
1	Pressed

Detection Type

1. Detection type are 2 types

- **Level triggering**
- **Edge triggering**

2.Level Triggering: To trigger the task based on the value.

Example: Volume button in the remote.

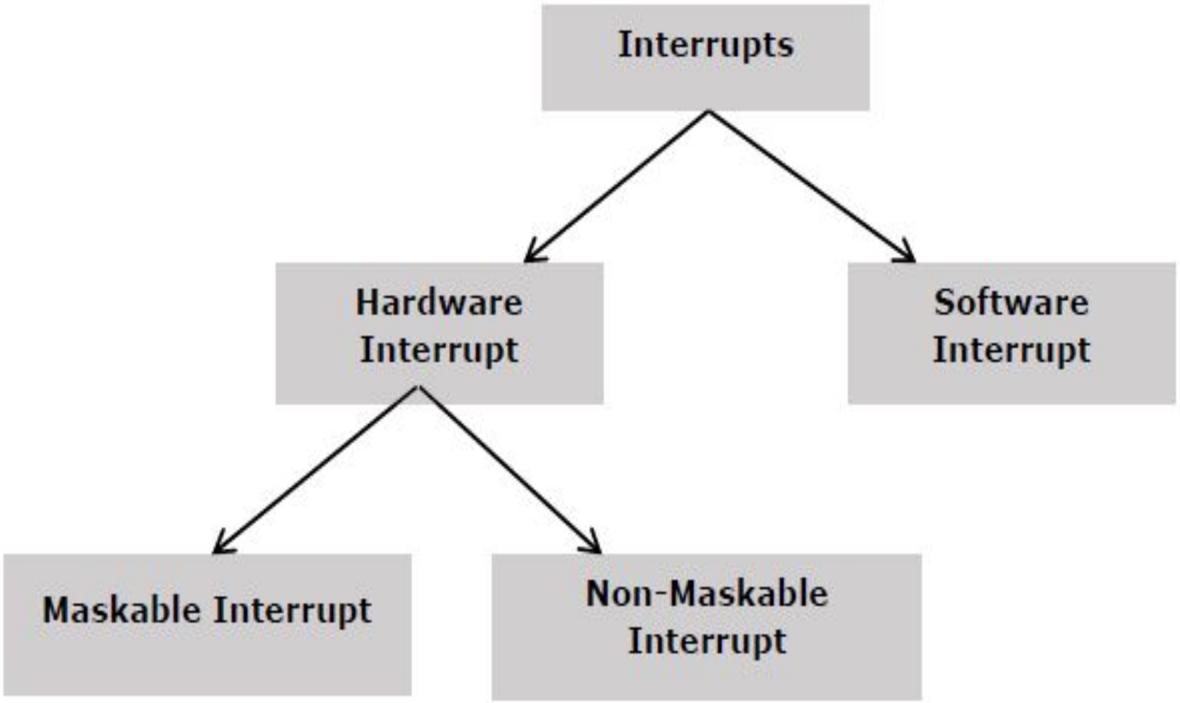
3.Edge Triggering: To trigger the task based on the change in the value.

Example: Power ON/OFF button in the remote.

Interrupts

1. An interrupt is a communication process setup in a microcontroller or microprocessor.
2. An internal or External device requests the MPU to stop the processing.
3. The MPU acknowledge the request.
4. Attends to the request.
5. Goes back to processing where it was interrupted.
6. Interrupt sources - Internal & External Interrupts.
7. Interrupt Handlers - Timers

Interrupt Classification



Interrupts

Hardware Interrupt

Software Interrupt

Maskable Interrupt

Non-Maskable Interrupt

Timers

1.A device for indicating or measuring elapsed time, as a stopwatch.

2.Resolution: width of the timer register. EXample: 8 bit, 16 bit registers.

3.Tick: Up tick, Down Tick.

4.Quantum: Time taken by one tick.

5.System clock settings: Frequency = 20MHz

1 tick = 4 clock pulse = 4 *Time

Time = 4*1/f

Quantum = 4*1/20MHz

=0.2µsec

Time taken = no of ticks * Quantum

Time = 255*200nsec = 51000nsec.

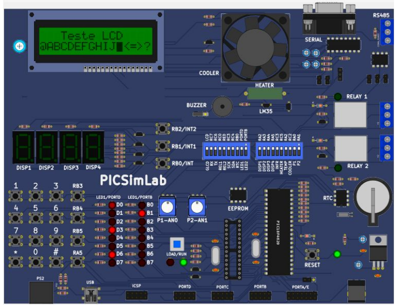
Washing machine Project Implementation

Applications Required

- MPLAB X IDE

- XC8 COMPILER

- PICSIMLAB



Peripherals required for the Project

- Switches
- Timers
- CLCD
- Buzzer
- Fan

Steps for Implementation

1.Press Key-5 to Power on the screen.

2.Power ON.

3.Washing Program screen, Press key-4 to scroll the options.

4.Long Press on Key-4 to select the option.

5.Water Level program, Press Key-4 to scroll the options.

6.Press key-5 to Start Screen.

7.Displays the washing time on CLCD.

8.Key-1 is for Door status screen in which means to know Door status.

9.Program completed Status it displays Program completed remove clothes.

THANK YOU

SUBMITTED BY

R.GANESH

SK.ARIEF

K.CHANDANA

P.GURUSAI