MICROPROCESSOR AND MICROCONTROLLER TECHNOLOGY

Today's world is comprised of manmade sophisticated gadgets that make our lives easier and more comfortable.

Ex: shophisticated human-like robots
smart e-bicycles

An embedded contoller and several communication modules are included in a general smart e-bicylcle. With the help of embedded processing power and distributed processing, they can make the riding experience easier and more comfortable by adjusting the riding conditions according to the situation.

Several microcontrollers or microprocessors are used in Robots, if small commercial popular robot ‘vector’ is interact with house hold components.

Ex: Fire alarms, Auto control fuse, fans, lights, auto closing doors through some interfacing devices.

* It can be used to closed doors, start water pump, whenever the tank get empty, clean the floor etc.

Such automations make our lives easier. All such machine, gadgets consists of number of microprocessor based tiny devices called microprocessors.

Although, we initially need to know the architecture of minicomputers associated processor peripheral devices for our simplicity. Specially in our laboratory works give us very good experience and employability related to automation industry. Therefore we have to refer the MPLAB, PICkit 2, Proteus, C language version used for instructing the instructor. (The basic way to familiarize with this programming language is, use help files.)

MICROCONTROLLER

A microcontroller is a VLSI (very large scale Integrated) Circuit. It is called as a computer-

on-chip or a single-chip-computer. Since the microcontroller and its supporting circuitry are often embedded in the device it controls, also called as an Embedded-controller.

A microcontroller is a chip optimized to control electronic devices. It is stored in a single integrated circuit which is dedicated to performing a particular task and execute one specific application.
It is specially designed circuits for embedded applications and is widely used in automatically controlled electronic devices. It contains memory, processor, and programmable I/O.

MICROPROCESSOR

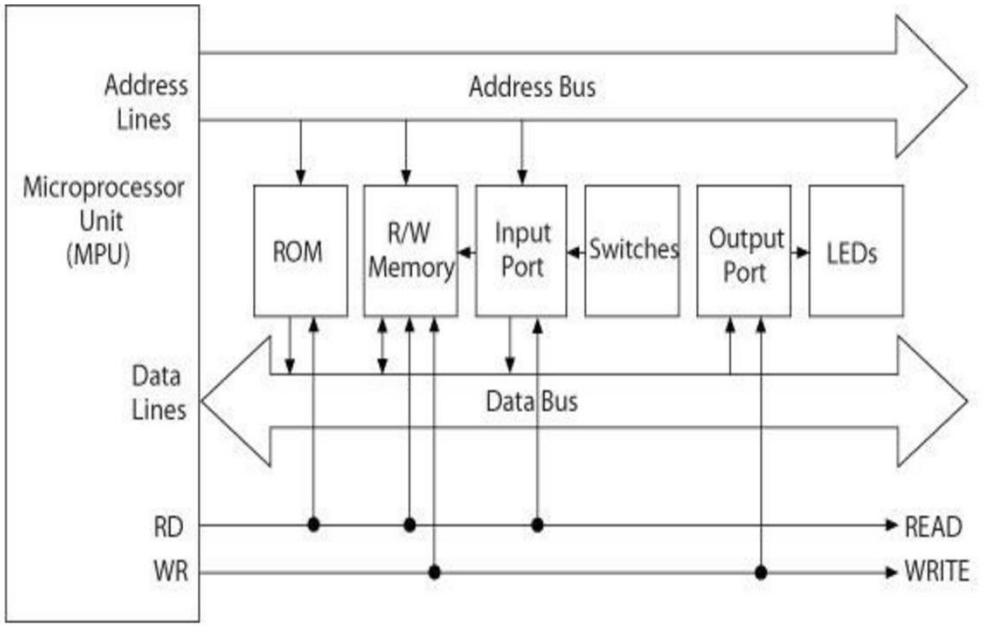
A microprocessor is a controlling unit of a micro-computer enclosed inside a small chip. It performs Arithmetic Logical Unit (ALU) operations and communicates with the other devices connected with it. It is a single Integrated Circuit in which several functions are combined

What is an embedded system?

it is a microcontroller or
microprocessor based system which is designed to perform a specific task. For example, a fire alarm is an embedded system; it will sense only smoke. An embedded system has three components − It has hardware. It has application software.

MICROPROCESSOR-BASED SYSTEM WITH BUSES; ADDRESS, DATA, AND CONTROL

Microprocessor-Based System



MICROCONTROLLERS VS MICROPROCESSORS

- Microprocessor consists of only a Central Processing Unit, whereas
- Micro Controller contains a CPU, Memory, I/O all integrated into one chip.
- □ Microprocessor is used in Personal Computers whereas Micro Controller is used in an embedded system.
- □ Microprocessor uses an external bus to interface to RAM, ROM, and other

peripherals, on the other hand, Microcontroller uses an internal controlling bus.

- Microprocessors are based on Von Neumann model Micro controllers are based on Harvard architecture

- Microprocessor is complicated and expensive, with a large number of instructions to process but Microcontroller is inexpensive and

- straightforward with fewer instructions to process.

MICROPROCESSORS	MICROCONTROLLERS
▪ Microprocessor is the heart of Computer system. ▪ It is only a processor, so memory and I/O components need to be connected externally ▪ Memory and I/O has to be connected externally, so the circuit becomes large. ▪ You can't use it in compact systems ▪ Cost of the entire system is high ▪ Due to external components, the total power consumption is high. Therefore, it is not ideal for the devices running on stored power like batteries. ▪ Most of the microprocessors do not have power saving features. ▪ It is mainly used in personal computers. ▪ Microprocessor has a smaller number of registers, so more operations are memory-based. ▪ Microprocessors are based on Von Neumann model ▪ It is a central processing unit on a single	▪ Micro Controller is the heart of embedded systems. ▪ Micro Controller has a processor along with internal memory and I/O components. ▪ Memory and I/O are already present internally.internal circuit is small. ▪ You can use it in compact systems ▪ Cost of the entire system is low ▪ As external components are low, total power consumption is less. So it can be used with devices running on stored power like batteries. ▪ Most of the microcontrollers offer power saving mode. ▪ It is used mainly in washing machines, Mp3 Players, and embedded systems. ▪ Microcontroller has more register. Hence programs are easier to write. ▪ Micro controllers are based on Harvard architecture ▪ It is a byproduct of the development of

siliconbased integrated chip. ▪ It has no RAM, ROM, Input-Output units, timers, and other peripherals on the chip. ▪ It uses an external bus to interface to RAM, ROM, and other peripherals. ▪ Microprocessor-based systems can run at a very high speed because of the technology involved. ▪ It's used for general purpose ▪ applications that allow you to handle loads of data. ▪ It's complex and expensive, with a large number of instructions to process.	microprocessors with a CPU along with other peripherals. ▪ It has a CPU along with RAM, ROM, and other peripherals embedded on a single chip. ▪ It uses an internal controlling bus. ▪ Microcontroller based systems run up to 200MHz or more depending on the architecture. ▪ It's used for application-specific systems. ▪ It's simple and inexpensive with less number of instructions to process.

Features of Microprocessor

Here are some important features of Microprocessor:

- Offers built-in monitor/debugger program with interrupt capability
- Large amount of instructions each carrying out a different variation of the same operation
- Offers Parallel I/O
- Instruction cycle timer
- External memory interface

Features of Microcontroller

Here are some important features of Microcontroller:

- Processor reset
- Program and Variable Memory (RAM) I/O pins

- Device clocking central processor
- Instruction cycle timers

Applications of Microprocessor

Microprocessors are mainly used in devices like:

- Calculators
- Accounting system
- Games machine
- Complex industrial controllers
- Traffic light
- Control data
- Military applications
- Defense systems
- Computation systems

Applications of Microcontroller

Microcontrollers are mainly used in devices like:

- Mobile phones
- Automobiles
- CD/DVD players
- Washing machines

- Cameras
- Security alarms
- Keyboard controllers
- Microwave oven
- Watches
- Mp3 players

What is the Difference Between a Microcontroller and Microprocessor?

The key difference between a Microprocessor and a Microcontroller is the Microprocessor consists of only a Central Processing Unit, whereas the Microcontroller contains a CPU, Memory, I/O all integrated into one chip. A microcontroller is an inexpensive, straightforward, and small number of instructions to process, whereas a Microprocessor is complex and expensive, with many instructions.

Which is Better Microcontroller or Microprocessor?

Both of these processes are good. However, which one you should use depends upon your requirements. Microcontrollers are mainly used for small applications like washing machines, Cameras, Security alarms, Keyboard controllers, etc., Whereas Microprocessor is used in Personal Computers, Complex industrial controllers, Traffic light, Defense systems, etc.

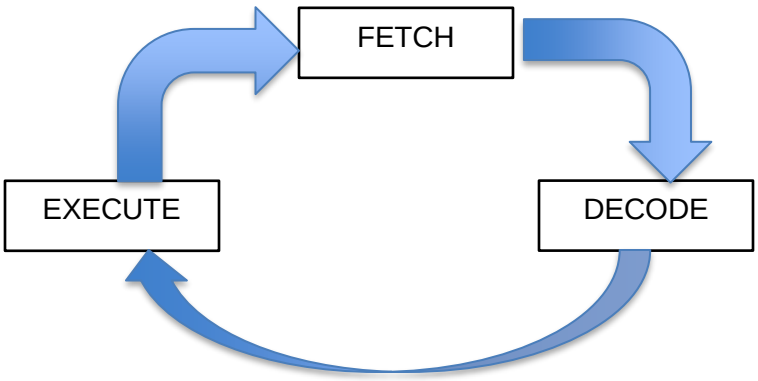
Which is Faster Microprocessor or Microcontroller?

Microprocessors are much faster than microcontrollers. The clock speed of a microprocessor is above 1 GHz. While in the case of the Microcontroller, the clock speed is 200MHz or more, depending on the architecture.

WHAT IS A COMPUTER?

An electronic device for storing and processing data, typically in binary form to instructions given to it in a variable program.

WHAT DOES A PROCESSOR DO?



TYPES OF DATA

1. Text
2. Video
3. Graphic

BUS ARCHITECTURE

DATA BUS

Provides transportation only for data.
Bidirectional.

ADDRESS BUS

Carries only addresses.
Unidirectional.

CONTROL BUS

Uses to process data that is what to do with selected memory by generating timing and control signals.

MICROCONTROLLERS
EMBEDDED SYSTEMS

System – An arrangement in while all its units assemble and work according to a set of rules.

Embedded system - A special-purpose computer system a perform one or a few dedicated functions often with Embedded systems have 3 components,

- Hardware
- Application Software
- Real-time operating system (RTOS)

(Can be based on a microcontroller or a microprocessor)

Applications of Embedded Systems:

• Smart homes ⟶ Home security Systems, Digital camera, Microwave oven, Television, Refrigerator

• Transportation → Anti - Clock - Barking System (ABS)

Air conditioning control

Rain sensing wipers

• Health care → Blood pressure monitors, scanner,
Heart beat monitors

• Industrial world → Industrial machinery & control monitoring BD printers

WHAT IS OPCODE AND OPERAND

OPCODE - A part of the instruction that tells the processor what should be done.

OPERAND- A part of the instruction that contains the data to be acted on, or the memory location of the data in a register.

ASSEMBLY LANGUAGE

It is not difficult to conclude that software development in machine language as extremely hard.

i) Program entering - Must use the binary patterns of every machine instructions.
ii) Program debugging - Whenever a program does not perform as expected the programmer will have a hard time to identify the instruction that causes the problem.
iii) Program maintenance - Most program will need to be maintained in the long run.

ELEMENTS OF AN ASSEMBLY LANGUAGE STATEMENT

An assembly language instruction consists of four fields;

[label]	[mnemonic or opcode]	[operands]	[;comment]

- Brackets indicates that a field is optional and not all lines have them

a) LABEL field

Alphanumeric names used to define the starting location of a block of statements. Must be unique in the executable file otherwise assembler will generate an error.

b) OPCODE(MNEMONIC) field

Using the mnemonic, it can decide what operation you want to perform on the operand.

c) OPERAND field

Might contain different number of operands and operands for instructions or arguments for assembler directives.

d) COMMENT field

Added for documentation purpose.

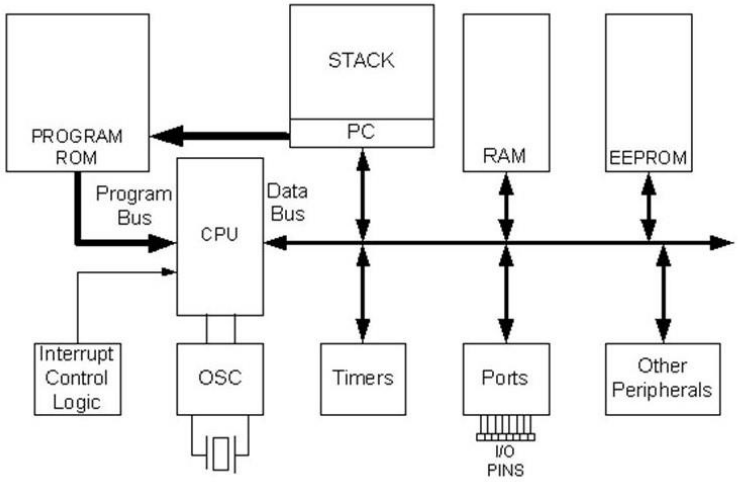
CHARACTERISTICS OF EMBEDDED SYSTEMS

1. Task specified

2. Low cost
3. Time specific
4. Require less power
5. Highly stable
6. High reliability
7. High efficiency
8. Minimal user interface

MICROCONTROLLERS

SIMPLIFIED VIEW OF PIC MICROCONTROLLERS



CISC vs RISC

CISC Complex Instruction Set Computer	RISC Reduced Instruction Set Computer
• Instruction can take several clock cycles	• Instruction can take single clock cycles
• Emphasis on hardware	• Emphasis on software

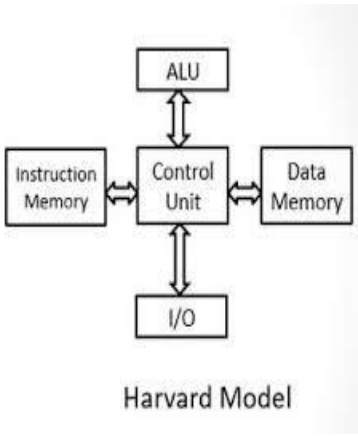
• More efficient use of RAM than RISC	• Heavy use of RAM
• Complex and variable length instructions	• Simple standardize instructions
• Large number of instructions	• Small number of fixed length instructions
• Compound addressing mode	• Limited addressing mode
• Pipeline is difficult	• Pipeline is easy

MICROCONTROLLER ARCHITECTURE

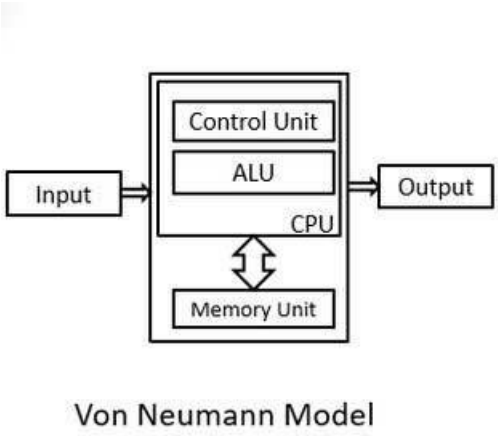
There are two types of microcontroller architecture.

1. Harvard Architecture
2. Von – Neumann Architecture

HARWARD ARCHITECTURE vs VON – NEUMANN ARCHITECTURE

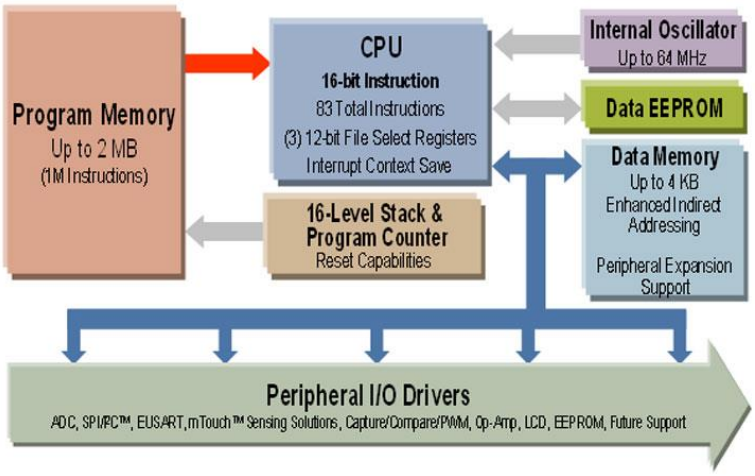


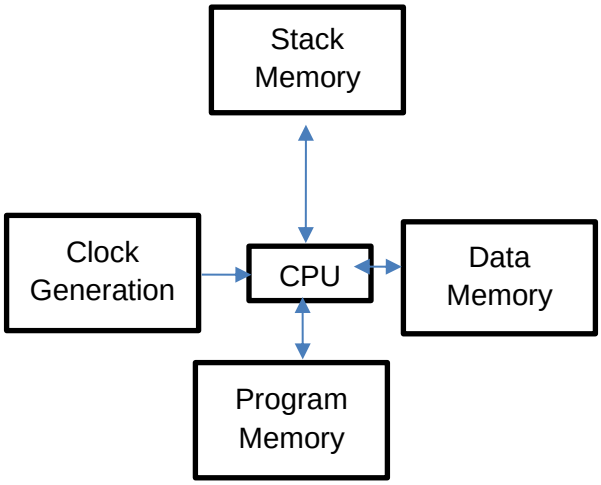
Harvard Model



Von Neumann Model

Basis of Comparison	Von Neumann	Harvard Architecture
Definition	The Von Neumann architecture is a style of computer architecture that is straightforward and makes use of a single memory connection.	The Harvard Architecture is the current design standard, and it features RAM and ROM that are kept completely independent.
Design	The layout is straightforward and makes use of the same path to both store data and take instructions.	When compared to the Von Neumann architecture, this design is more complicated because it utilizes separate connections for RAM and ROM.
Hardware	When compared to Harvard Architecture, the hardware requirements are significantly lower.	When compared to the Von Neumann Architecture, the Harvard Architecture places a greater emphasis on the use of hardware.
Speed	In comparison to the Harvard Architecture, the speeds of the processors are significantly lower.	Harvard Architecture is faster than the others. A computer modelled are significantly lower. after the Harvard Architecture calls for an increase in the available space.
Physical Space	When compared to the Harvard Architecture computers, the Von Neumann computers have a smaller footprint in terms of the required amount of physical space.	In Harvard Architecture, the requirement for the actual space is increased.
Internal Memory	Because the memory and the programs share the same space, there is no unused space in the internal memory.	Because the instruction memory and the data memory cannot share the same space, some of Harvard's internal memory is going to waste somewhere.
Running Instructions	The instructions for running can either be taken from the program that has been stored or they can be given explicitly. As a result, the two cannot be considered together.	Due to the fact that the input and the program instructions that are stored in the program are taken simultaneously, the running instructions are somewhat complicated and somewhat slow.

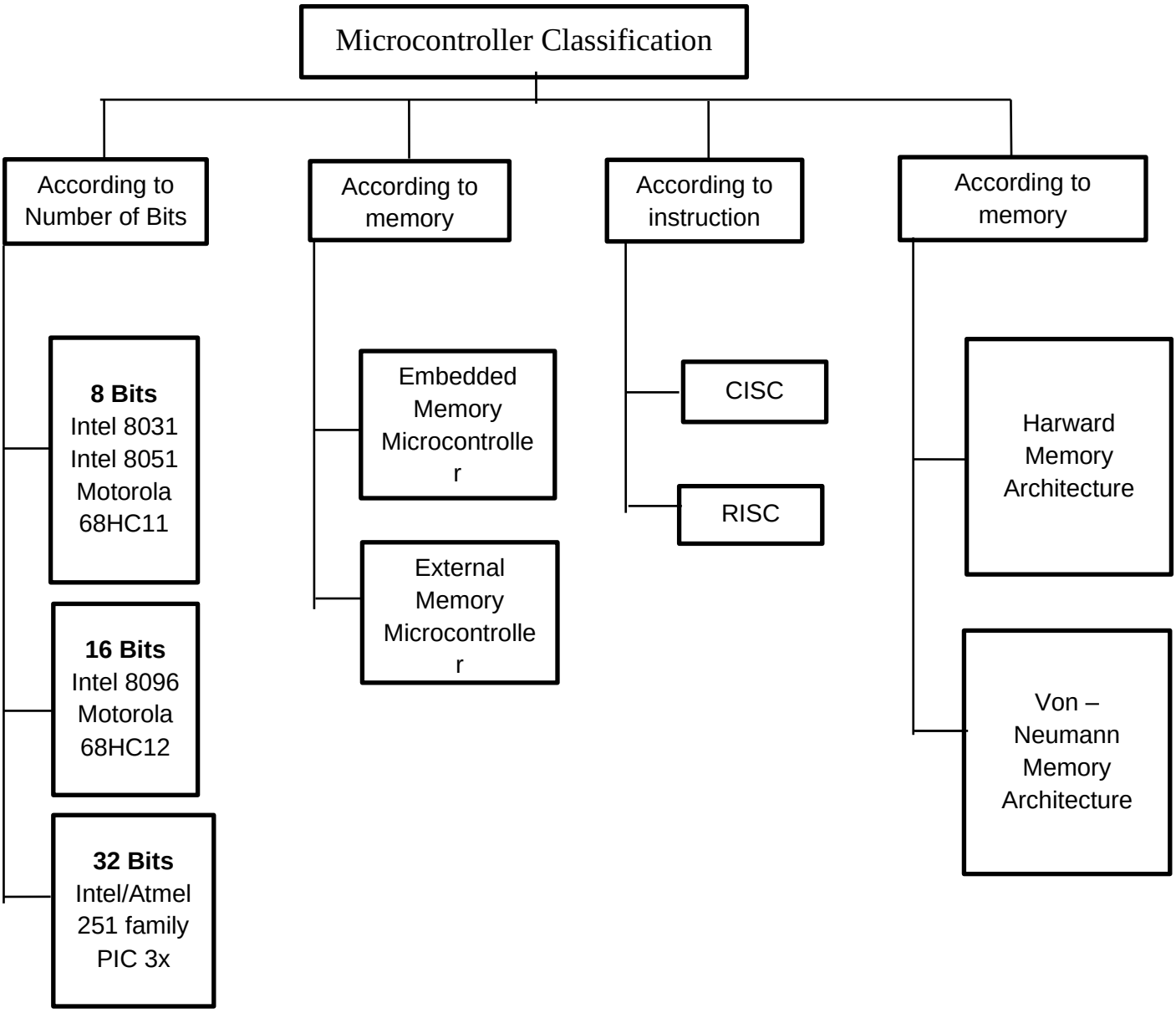




CPU – Stack Memory: Program Address Bus

CPU – Data Memory: Data Bus

CPU – Program Memory: Instruction Bus



PIC MICROCONTROLLERS

Why PIC 18 Popular?

- Low cost, wide availability with high dock speed.
- Availability of low cost or free development tools.

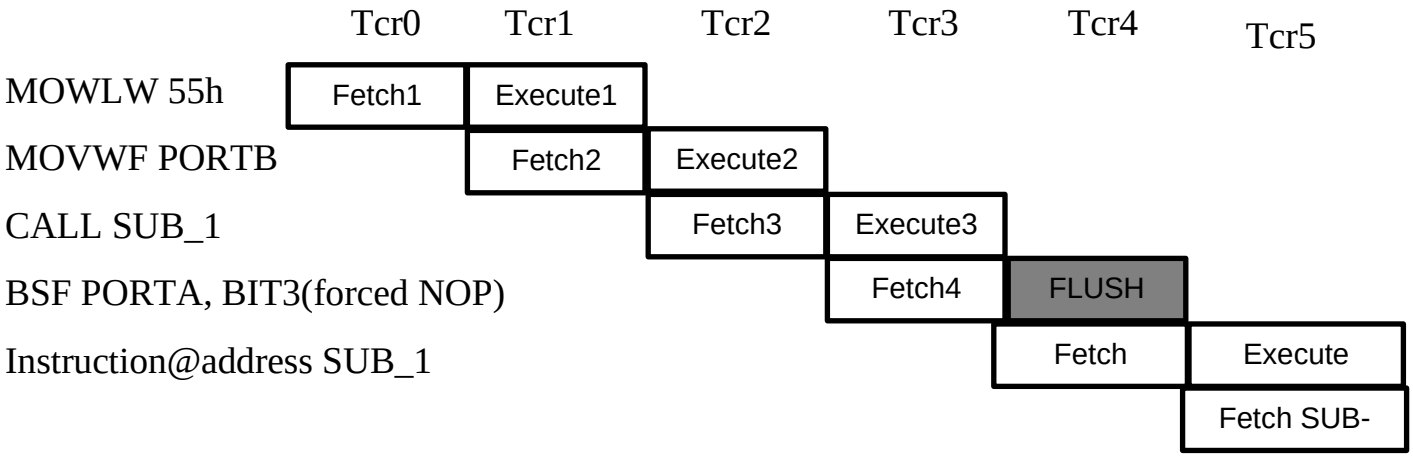
- Only 35-77 instructions to remember.
- Serial programming and re-programming with flash memory capability.
- Its code is extremely efficient, allowing the PIC to run with typically less program memory than its larger competitors.
- PIC is very small and very easy to implement for non-complex problems and usually accompanies to the microprocessors as an interface.

Disadvantages of PIC MICROCONTROLLERS

- the length of the program will be big because of using RISC (35 or 77 instructions).
- Program memory is not accessible and only one single accumulator is present.

PIPELINE IN PIC / INSTRUCTION FLOW

The instruction fetches and execute are pipelined in such a manner that a fetch takes one instruction cycle, while the decode and execute take another instruction cycle. However, due to the pipelining each instruction effectively executes in one cycle.



All instructions are single cycle, except for any program branches. These takes two cycles since the fetch instructions is “FLUSHED” from the pipeline while the new instruction is being fetched and executed.

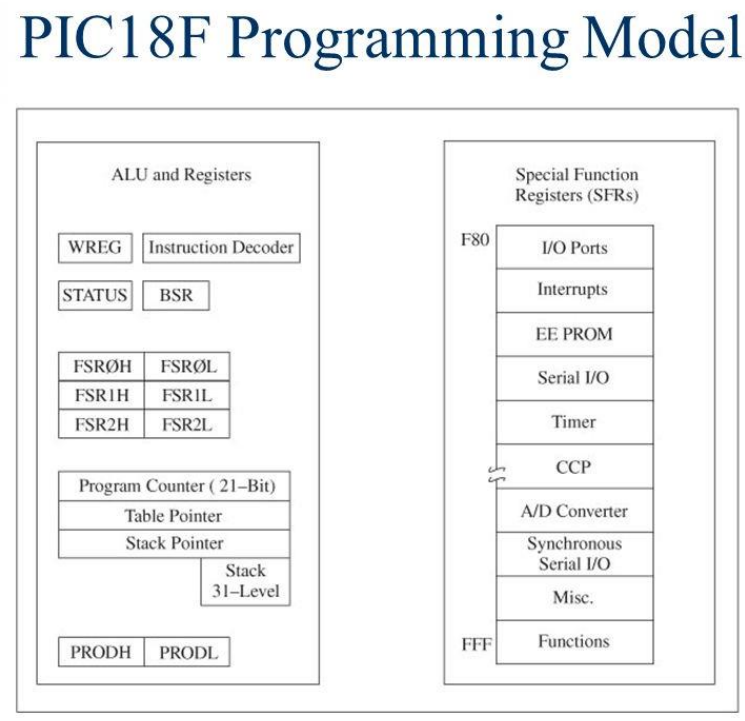
PIC18F DATA MEMORY

- Implemented in SRAM and consists of GPRS (General Purpose Registers) and SFRs (Special Function Registers). Both are referred to as Data Registers.
- A PIC 18 MCU may have up to 4096 bytes of data memory.
- Data memory is divided into banks. Each bank has 256 bytes.
- GPR s used to hold Dynamic data.
- SARS is used to control the operation of peripheral functions.
- only one bank is active at any time. The access bank 18 is specified by BSR register.
- Bank switching is an overhead and can be error prone.
- PIC 18f implements the access bank to reduce the problem caused by bank switching
- Access bank consists of the lowest 96 bytes and the highs 16 bytes of the data memory space.

PIC 18F PROGRAM MEMORY

- The Program Counter is 21-bit long, which enables the user program to access upto 2mB of program memory.
- The PIC 18 has a 31-entry return address stack to hold the return address for a subroutine call.
- After power on, the PIC 18 starts to execute instructions from address o
- The location at address ox18 is reversed for low-priority interrupt service routine (ISR). upto 128KB (at present) of program memory is inside the MCU chip.
- Part of the program memory is located outside the MCU chip.

PIC 18f PROGRAMMING MODEL



- WREG - Working Registers
- BSR - Bank Select Registers
- TB – Table Pointer
- Stack
- SFR – special function registers
- STATUS - status Registers/Flag Registers
  → C - Carey / Borrow flag
  → DC-Digital Carry flag
  → Z - Zero Flag
  → Ov- overflow flag
  → N- Negative flag .

- FSR-file select Registers
- PC - Program Counter.
- SP - Stack Pointer
- TP – Table pointer
- Stack
- SFR – Special function Register

PIC18F INSTRUCTION SET

01) 31-byee oriented file register operations

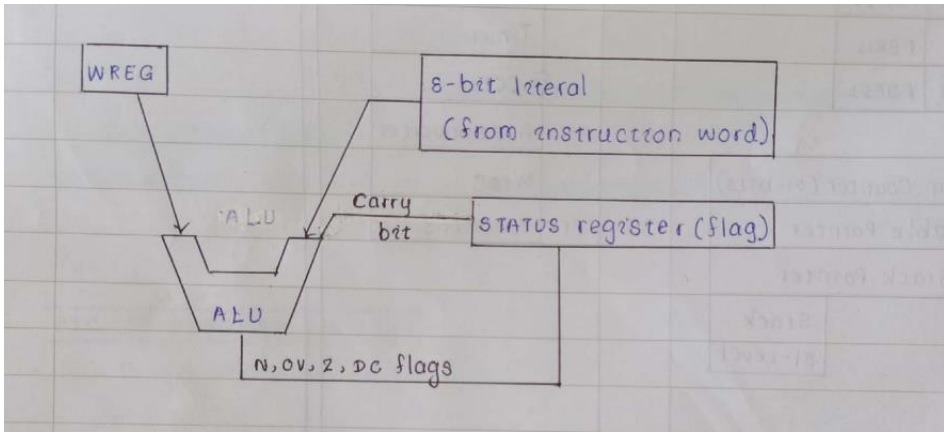
02) 5 bit oriented file register operations

05) 23-bit control instructions

04) 10-bit lateral instructions

05) 8- bit data memory program memory operations.

• A diagram to represent data flow when using literal value with WREGON ALU WREG



? Write a program that adds the 3 numbers stored in data registers at Ox 20, 0x80 and 0x40 and places the sum in data register at ox50.

Step 01 - Load the number scored at Ox 20 into the WREG register.
Step 02 - Add the number stored ac ox30 and the number inches WREG register and leave the sum in the WREG register.
Step 03 - Add the number stored at ox40 and the number in the WREG and leave in the sum in the WREG.
Step 04 - Store the content in WREG Zn memory location of ox5o.

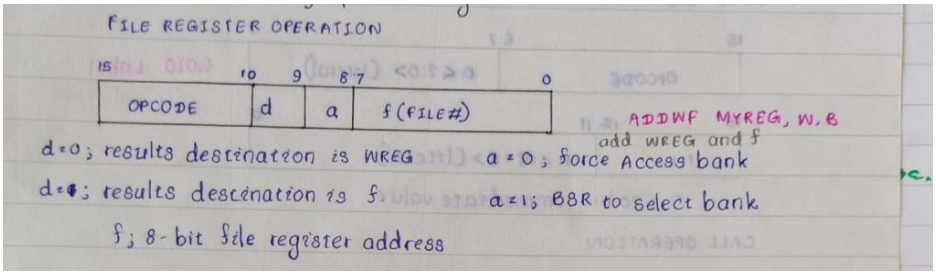
PIC 18F452 INSTRUCTION SET

- Some instructions with alternate results destinations. The default destinations for the result of an operation is the file register, but the WREG, W is sometimes an option.

General format for instructions

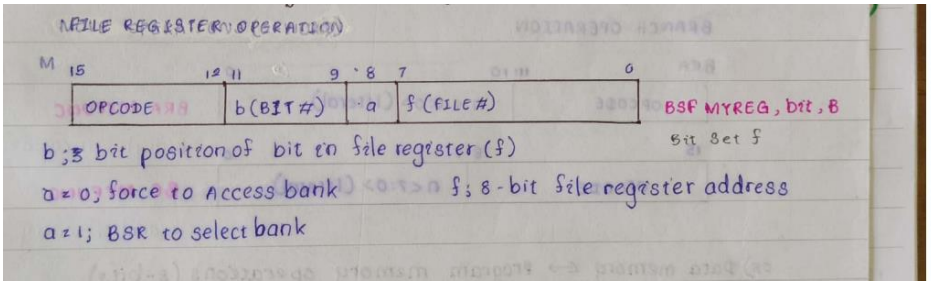
01) Byte oriented file register operation (51 bytes)

01. The file register specified by "f"
02. The destination of the result specified by "d"(if d=o; result placed in WREG).
03. The accessed memory specified by “a?



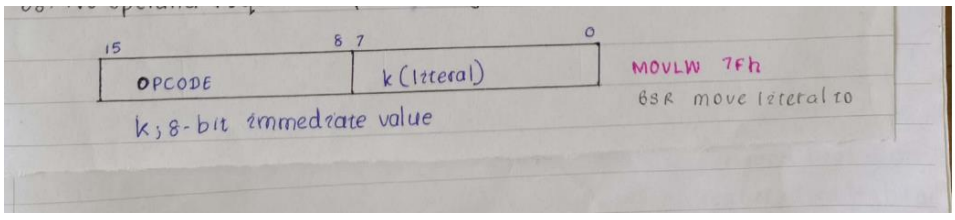
02) Bit oriented file register operation (s bits)

01. The file register specified by " f "
02) The destination of the result specified by "d"
03. The accessed memory specified by "a"



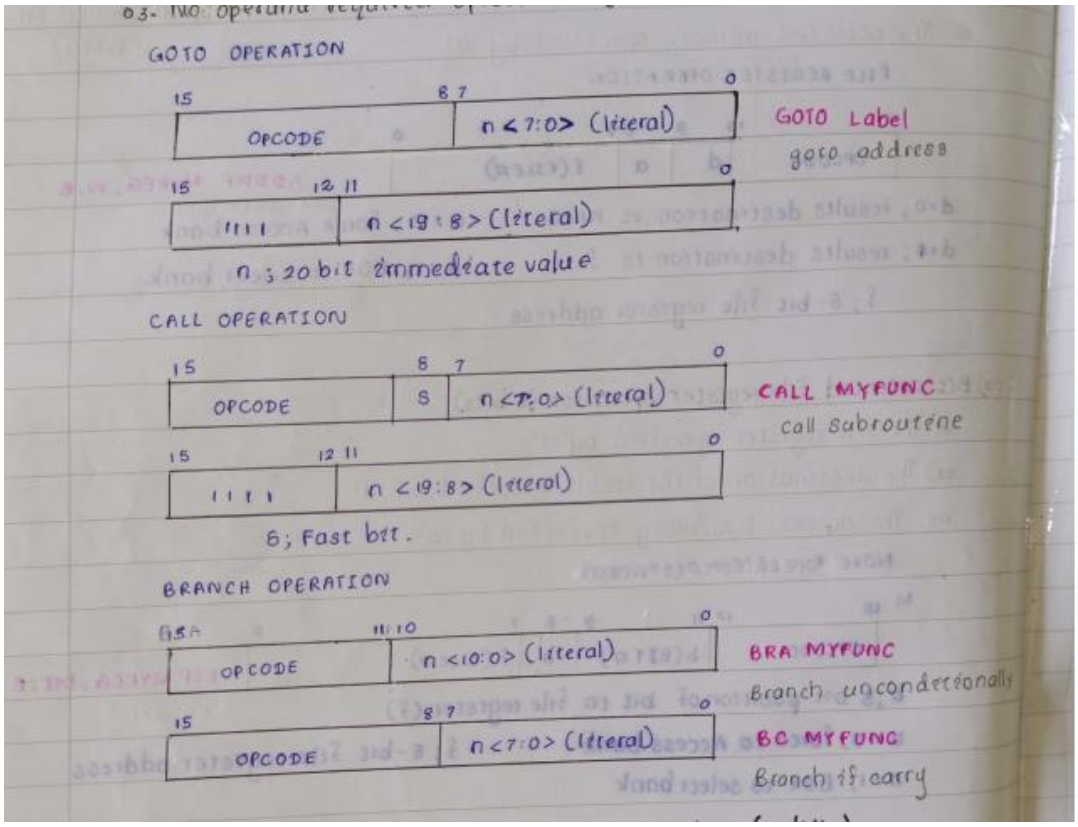
03) Literal operation (23 bits)

01. A literal value to be loaded into a file register specified by "K"
02. The desired FSR register to load the literal value in to f
03. No operand required (specified by “-")



04) Control operations (10 bits)

01. A program memory address specified by "n"
02. The mode of the CALL or RETURN instructions specified by "s"
03. No operand required specified by "_"

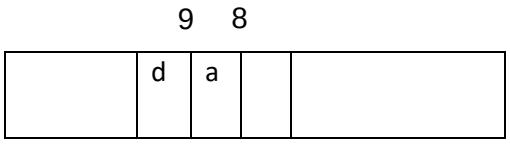


05) Data memory o program memory operations (8-bits)

TBLRD* - Table read
TBLRD*+ - Table read with post increment
TBLR D.+* - Table read with pre increment
TBLWT* - Table write with post decrement

The instruction set can also organized by operational groups.

There are 5 categories.



- 1 - Byte oriented instructions
  - F : file register or RAM
  - D : destination
    - D = 0 destination → w
    - D = 1 destination → File register
- Byte to Byte oriented.....
- 3 - Bit oriented instructions
  - F : Register file where the bit is located
  - B : Bit field
- 4 - Literal and control operations
  - K : 8 – bit constant
- Control operation

DATA ADDRESSING MODES OF PIC18F

- There are 4 addressing modes to PIC18f
  01) Register Direct Addressing mode
  02) Immediate Addressing Mode (literal Add: mode)

  03) Indirect Addressing Mode.

  04) Direct Addressing mode (ABSOLUTE ADDRESSING MODE)

01) DIRECT ADDRESSING MODE (ABSOLUTE ADDRESSING MODE)

- The 8-bit data in RAM memory location whose address 28 specified in instruction
- Used for accessing the RAM file register.

02)IMMEDIATE ADDRESSING MODE

- The immediate data is specified in the instruction
- Used to load the data into PIC registers and WREG registers
- It cannot use to load data into any of file registers
  Ex: MOVLW SOH
  ANDLW 40H

03)INDIRECT ADDRESSING MODE

- Allows address to be computed by adding an offset to a base address
- Very useful for array addressing
  Ex: movf POSTDE
  movF POSTINCO

04) REGISTER DIRECT ADDRESSING MODE

- No memory accesses.
- Very fast execution.
- Very limited address space.
- Multiple registers help performance.
  Ex: movwf 0x20

Speed OF PICs

Speed = 1/4 * clock frequency

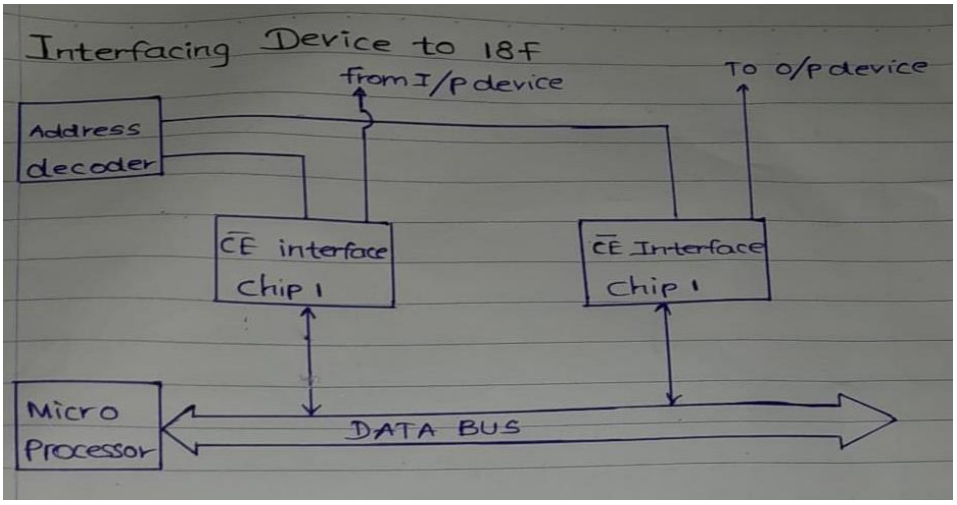
Clock frequency = clock speed / cycle per instruction

Tame per 1 instruction = 1/ speed

INTERFACING

- Peripheral_chips_are used to resolve the difference between the s/o devices and the CPU.
- The function of the interface chip is to synchronous the data transfer between the CPU and I/O devices.
- Resistors may be needed to limit the current flow whereas buffer chips may be needed to increase the current flow.

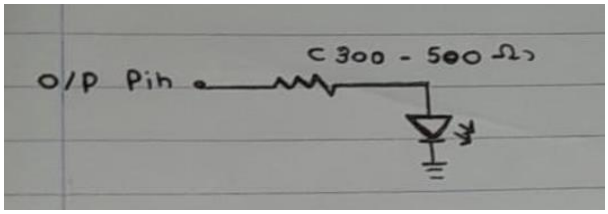
INTERFACING DEVICE TO 18F



- The address decoder in figure allows only one *i*nterface to exchange data with the microprocessor at a time.
- Data transfer can be proceeded in parallel or bet by bit (serial method).
- Serial method is meant to be used with slower 110 devices. Parallel method is meant to be used with high speed devices.

INTERPACING WITH LEDS

Simple as an LED, is often used to indicate if power is on, if operation is normal, and so on. An LED can illuminate when it is forward biased & sufficient current flowing through it. The current required to light and LED may range from a few more than 10mA. The voltage drops across the LED when it is forward bias can range from about 1.6V to more than 2.2V. When configured for output, a PIC 18 pin can drive & LED directly. A resistor (330ohm-500ohm) is required to limit the current flow. A typical circuit connected is shown below.



HD44180U LCD CONTROLLER

- Dot-matrix LCD controller and driver.
- Three control signals: R8 - Register select (RA3)
  R/W- Read / write (RA2)
  Enable (RA1)
- Three power connections: VCC-Power
  GND-Ground
  Variable resistor to control the brightness
- Can be interfaced either in the 8-bit mode or the 4-bit mode.
- 8 & 4- bit mode interfacing
  - In the 8-bit mode, all o data lan*es a*re connected for data transfer
  - In the 4-bit mode, only a data lines (DB7 - DB4or DB3 - DBO) are connected and transfer per character (or instructions) are needed.
- Driver HD 44780U has 2,8-bit internal registers.
  - IR - Instruction Register to write 20struction to set-up LCD
  - DR - Data Register to write data (ASCII characters).

LCD Operation:

- When the MPU writes on instruction to IR or DR, the controller
  - Sets the data line oB7 high as a flag indicating that the controller is busy completing the operation.
  - Sets the data line DB7 low after the completion of the operation.
- The MPU should always check whether DB7 is low before sending an instruction or data byte.
- After the power up, DB7 cannot be checked for the first two initialization instructions.

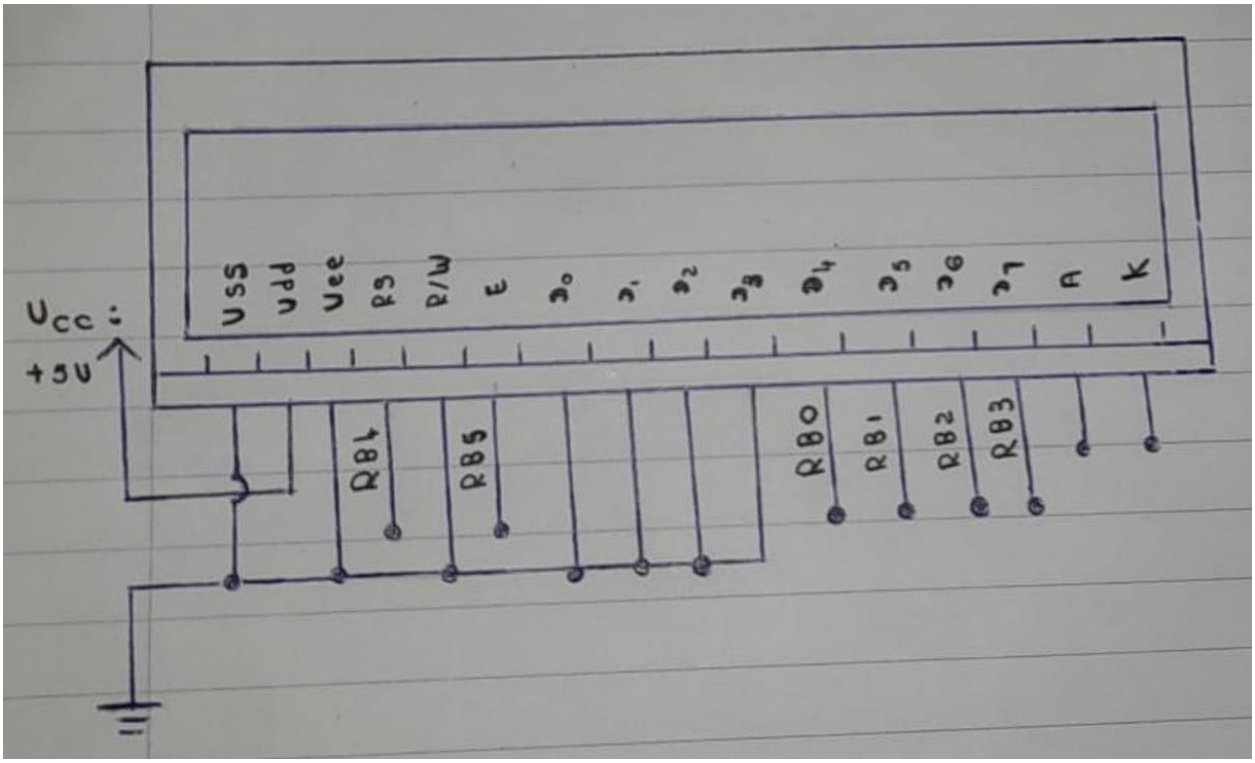
Resetting LCD:

- In 4-bit mode the data is sent in nibble.
  - First send the higher nibble and then the lower nibble
- To enable the 4-bit mode of LCD.
  It needs to follow special sequence of initialization that tells the lcd o controller that user has selected 4-bit mode of operation.
  - Wait for about 20ms
  - Send second init value (0x30)
  - Wait for about 10ms
  -Send second init value(0x30)
  -Wait for about 1ms
  -Select bus width (0x30-for 8 bit and 0x20 for 4-bit)
  -Wait for 1ms

Interfacing LCD:

- Writing to or reading from LCD.
- The MPU: Declare
  -Assert R8 high to select DR
  -Reads from LCD by asserting the R/W signal high
  -Assert the E signal high and then low(toggle) to latch a data byte or an instruction.
  -Assert RS high to select DR
  -Writes into LCD by asserting the R/W signal low
  -Assert the E signal high and then low (toggle) to latch a data byte or an instruction.

  Home work
  Write and draw the circuit diagram of interfacing LCD with PIC microcontroller and initialize code

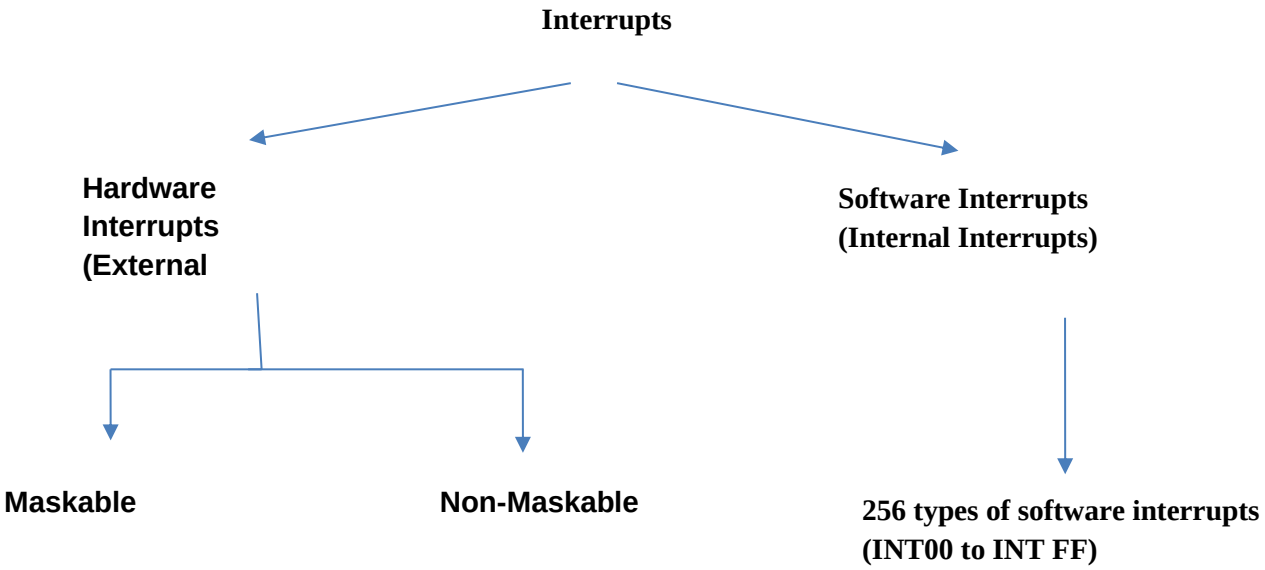


Sbit LCD – D7 – Direction at TRISES
//End of use. LCD module connection
Void main ( ) {
ANSELD =0;//Configure PORTB pins a
Digital
Lcd – Init( );//Initialize LCD
Lcd – Cmd (-LCD-CLEAR);//clear display
Lcd-Cmd (- LCD-CURSOR – OFF);//Cursor
}

```
// LCD module connection
Sbit LCD – RS at LATB4 – bit;
Sbit LCD-EN at LATB5-bit;
Sbit LCD – D 4 at LATBO – bit;
Sbit LCD – D5 at LATB1-bit;
Sbit LCD – D6 at LATB2-bit;
Sbit LCD – D7 at LATB3-bit;
Sbit LCD – RS- Direction at TRISB4-bit;
Sbit LCD – EN-Direction at TRISB5-bit;
Sbit LCD- D4 Direction at TRISBO – bit;
Sbit LCD – D5-Direction at TRISB1-bit;
Sbit LCD – D6-Direction at TRISB2 – bit;
```

INTERRUPTS

Interrupts are mechanism which enable instant response to events such as counter – overflow, pinchange, data received etc.
In normal mode microcontroller executes the main program as known as there are no accurances that would cause an interrupt.
Upon interrupt microcontroller stops the execution of main program and comments the special part of the program (ISR) which will analyze and handle the interrupt.



- Also, interrupts can be categorized as High Priority Interrupts and Low Priority Interrupts, based on ORIGIN

Hardware interrupts (Externa interrupts)

These interrupt requests are sent by external hardware devices connected to the certain pins of microcontroller

Software interrupts (Internal Interrupts)

It comes from a program that executed by a microcontroller or by internal peripherals of the µc.

High priority Interrupts

These interrupts cannot be interrupted. A high priority vector is located at 008h in the program memory.

Low priority interrupts

These interrupts it self could be interrupt by high priority interrupt and its interruptvector is located at 0018H

In pic 18f architecture the interrupts classified as high priority interrupts

Interrupt service routine – (ISR)

interrupt request associated with a particular code sequence is called as an ISR or interrupt vector.

- For every one interrupt, there must be an ISR or interrupt handler
- when an interrupt invoked MC runs the ISR
- For every interrupt there is a fixed location in memory that holds the address of this ISR generally.
- The groups of memory locations set aside to hold the addresses of ISRs is called interrupt vector table(IVT)

PIC 18F INTERRUPTS

There are two methods by which devices receive service from the Mc

1) Polling
2) Ioterrupts

01) polling methods

- MC Accesses at the exact time interval the external device and gets the required information
- The Mc continuously monitor the status of a a given device when the status condition met it it performs the service after that it moves on to monitor the next device until each one is serviced
- that time period is determined by you sir when using this method the information that it needed to be processed
- Cannot assign priority because it checks all devices in a round- Robin fashion(take turns)

- cannot ignore a device for service
- drawbacks

    Writing a program is waste of time of microcontroller

    microprocessor needs to you wait and check whether new information arrived.

02) INTERRUPT METHOD

- Whenever any device needs the microcontrollers service, the device notifies it by sending an interrupt signal.
- Interrupt is the signal sent to the microprocessor to mark the event that requires immediate action.
- Upon receiving an interrupt signal, the MC stops current program and serve the device(execute ISR)
- the program associated with the interrupt is called ISR interrupt Handler
- each device can get the attention of the microcontroller based on the priority assigned to it
- can ignore device request for service.

Steps in executing an interrupt

- Finish current instruction and saves the PC on stack
- Jumps to a fixed location in memory depend on type of interrupt
- starts to execute in the interrupt service routine(ISR) until return from interrupt(RETI)
- Upon executing the RETI the Mc returns to the place where it was interrupted. Get pop PC from stack
- In general each interrupt sources have following related bits co enable.

    Enable bit[XXIE(interrupt enable)]

    - 1-Enable
    - 0-Disable

    Flag bit[XXIF(interrupt Flag)]

    - 1-Enable
    - 0-disable

    Global interrupt enable bit[XXGIE(Global interrupt enable)]

    - 1-Enable
    - 0-disable

Interrupt handling

- PIC 18F can handle multiple interrupt request and treat them based on priority. The configuration and functioning of interrupt done with the help of the registers.
    - RCON
    - INTCON
    - INTCON2
    - INTCON3
    - PIRI,PIR2
    - PIE1,PIE2
    - IPR1,IPR2

The main functionality of these registers it to is configuration bits for each interrupts. these bits are:

- enable bits (IE)
- priority bits (IP)
- flag bits (IF)

INTERRUPT SOURCES OF PIC 18F

External

- INT0(RB0)
- INT1(RB1)
- INT2(RB2)

Internal

- PortB pins
- Timer 0/1/2/3 overflow interrupt
- Parallel salve port R/W interrupt
- A/D conversion complete interrupt
- USARI receive/transmit interrupt
- Synchronous serial port interrupt
- CCFI/CCP2 interrupt
- Comparator interrupt
- EEPROM/FLASH write interrupt

- Bus collision interrupt
- Low voltage detect interrupt
-

INTCON- INTERRUPT CONTROL REGISTORS

Intcon1

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
GIE/GIEH	PEIE/GIEL	IMROIE	INTOTE	RBIE	TMROIF	INTOIF	RBIF

GIE/GIEH- Global interrupt enable bit

- When IPEN is disabled. GIE enable all interrupts.
- When IPEN is it enabled. GIE enables all high priority interrupts

PEIE/GIEL- peripheral interrupt enable bit

- when IPEN is disabled. PEIE enables all peripheral interrupts
- when IPEN is enabled. GIE enables all low priority interrupts

TMROIE- TMRO overflow enable bit

INTOIE- INTO external interrupt enable bit

RBIE – RB port change interrupt enable bit

TMROIF- TMRO overflow interrupt flag bit

INTOIF – INTO external interrupt flag bit

RBIF- RB port change interrupt flag bit

Intcon2

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RBPU	INTEDG0	INTEDG1	INTEDG2	--	TMROIP	--	RBIP

INTEDGX- External interrupt x edge select bit

- SET interrupt flag rising edge
- CLEAR interrupt flag falling edge

TMROIP – TMRO Overflow interrupt priority bit

RBIP – RB Port change interrupt priority bit

Intcon3

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INT2IP	INT1IP	--	INT2IE	INT1IE	--	INT2IF	INT1IF

INTxIP- INTx external interrupt priority bit

INTxIE- INTx external interrupt enable bit

INTxIF- INTx external interrupt flag bit

PIR- PERIPHERAL INTERRUPT REQUEST

PIR1

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF

SPPIF - streaming parallel port read / write interrupt flag bit

ADIF- A/D convert interrupt flag bit

RCIF – EUSART receiver interrupt flag bit

TXIF – EUSART transmit interrupt flag bit

SSPIF- Master synchronous serial port interrupt flag bit

CCP1IF- CCP1 interrupt flag bit

TMR2IF – TMR2 interrupt flag bit

TMR1IF – TMR1 interrupt flag bit

PIE – PERIPHERAL INTERRUPT ENABLE

PIE1

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
--	ADIE	RC1IE	TXIE	SSP1IE	CCP1IE	TMR2IE	TMR1IE

ADIE – A/D Convert interrupt enable bit

RC1IE – EUSTAR Receive interrupt enable bit

TXIE – EUSTAR Transmit interrupt enable bit

SSP1IE- Master synchronous serial parallel port interrupt enable bit

TMRXIE – TMRX interrupt enable bit

IPR – PERIPHERAL INTERRUPT PRIORITY

IPR1

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP

SPPIP - Streaming parallel port R/W interrupt priority bit

ADIP – A/D converter interrupt priority bit

RCIP- EUSART receiver interrupt priority bit

TXIP- EUSART transmitter interrupt priority bit

SSPIP- Master synchronous serial port interrupt priority bit

CCP1IP- CCP1 interrupt priority bit

TMRxIP- TMRX interrupt priority bit

PRIORITY INTERRUPTS

HIGH PRIORITY

The contents of w, STATUS and BSR are automatically saved into respective shadow register

Al high-priority int*err*upts are directed to the interrupt vector location 000008H

LOW PRIORITY

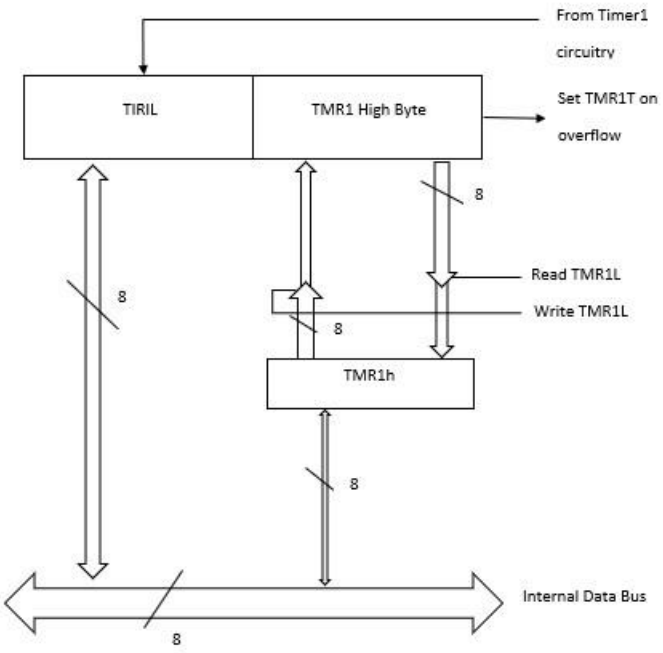
- These registers must be saved as a part of the ISR.
- A high-priority interrupt can interrupt a low priority interrupt in progress
- All low priority interrupt is directed to the interrupt vector location 000018H.

HOMEWORK

Draw the internal block diagram of t*im*ers and briefly explain the action.

Timers

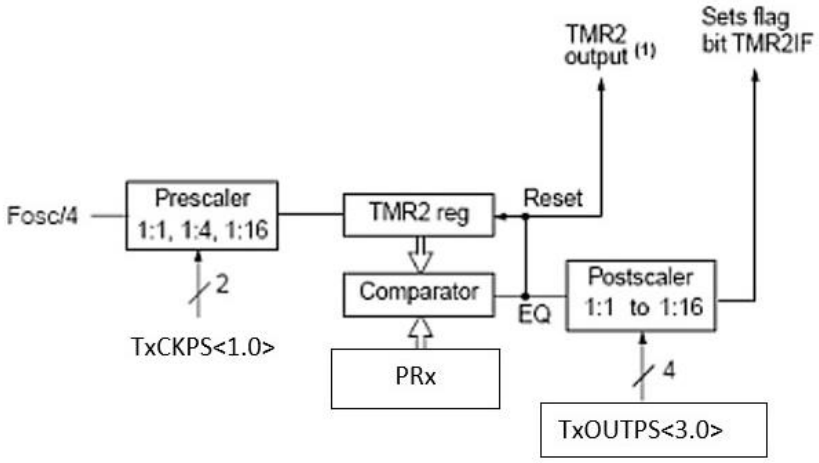
Timer 1/3/5



Operations

- A 16-bit incrementing counter which is accessed through the TMRxH register pair.
- Writes to TMRxL directly update the counter.
- When used with an internal clock source, the module is a timer and increments on every instruction cycle.
- When used with an external clock source, the module can be used as either a timer or counter and increments on every selected edge of the external source.
- Enabled by configuring
    - TMRxON(in TxCON register)
    - TMRxGE(in TxGCON registe

Timer 2/4/6



Operations:

- The system instruction clock (Fosc/4).
- A 4-bit counter/prescaler on the clock input allows direct input, devide-by-4 and divide by prescale option.
- The value of TMRx is compared to that of the PRx (Period Register) on each clock cycle.
- When the two values match the comparator generates a match signal as the timer output.
- Also, can generate an optional device interrupt.

REST(INTERRUPTS)

Special form of interrupt in that it causes the processor to stop what is doing and to rest the program.

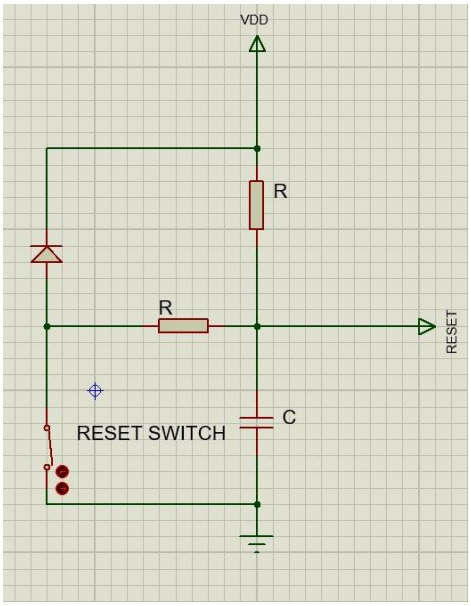
Why Interrupts

- Coordinating I/O activities and preventing the CPU from being tired up during the data transfer process.
- Performing time - critical applications.
- Providing a grateful way to exit from application when a software error occurs
- Reminding CPU to perform routine casks.

Power On Reset (POR)

- A circuit that provides a predictable, regulated voltage to a µP or µc with the initial applications of power.
- In ensures that the µP or µc will start in the same condition every time that is powered up.

- Can be a peripheral or integrated on main chip.
- Basically, can comprise a register and a capacitor connected together with values tailored.
- So that, when power is first applied the capacitor takes a predictable and constant time to charge up



CCP – Capture/Compare/PWM Modules

The CCP module is a peripheral which allows the user to time and control different events, and to generate PWM signals. In capture mode, the peripheral allows the timing of the duration of an event. The compare mode allows the user to trigger and external event when a predetermined amount of time has expired. The PWM mode can generate PWM modulated signal of varying frequency and duty cycle.

How many CCP modules in 16f and 18F?

In 16F

- 2 modules

1. CCPI
   a) A 16-bit register comprised of two 8-bit registers: (HIGH & LOW).
   b) CCP1CON controls the operation.

2. CCP2
   a) A 16bits register same as in CCPI
   b) CCP2CON controls the operation.

In 18F

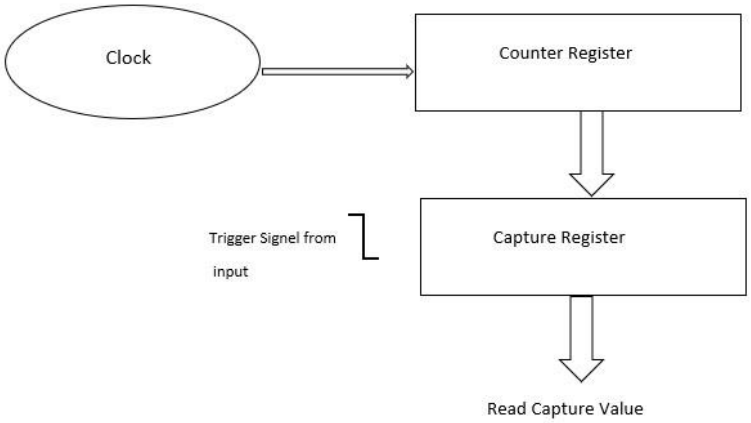
- 2 modules

1. CCP1
   a) Controls by ccp1con register
   b) Multiplexed with PORTC.2(RC2)

2. CCP2
   c) Controls by ccp2con register
   d) Multiplexed with PORTC.1(RC1)

Capture Mode

- Allows timing for the duration of an event
- Gives insight into cur*rent* state of a register which constantly changes its value.
- Makes use of the 16-bit Timer resources, Timer1, Timer2, and Timers3, and Timers.
- An event is defined as one of the following:
  - Every falling edges
  - Every rising edge
  - Every 4^{th} rising edge
  - Every 16^{th} rising edge



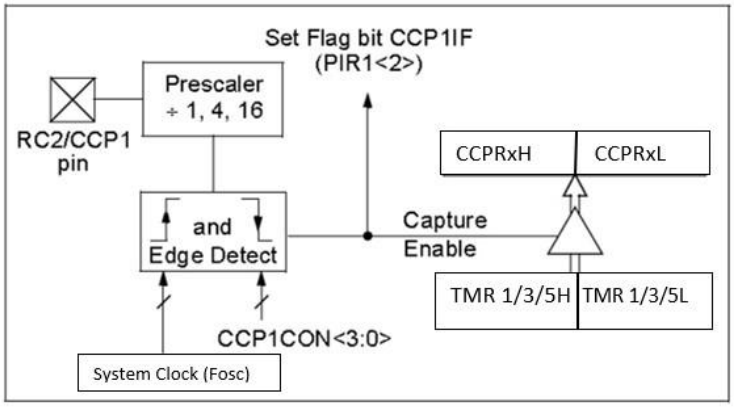
WORKING OF CAPTURE MODE

- Time1 or Time3 are initialized to count the value in the capture operation
- T3CON register is used for selecting the Timer1/Timer3 for input capture.

- When the rising or falling edge is detected at CCP1 pin the interrupt flog CCP!IF bit is set It is cleared through software

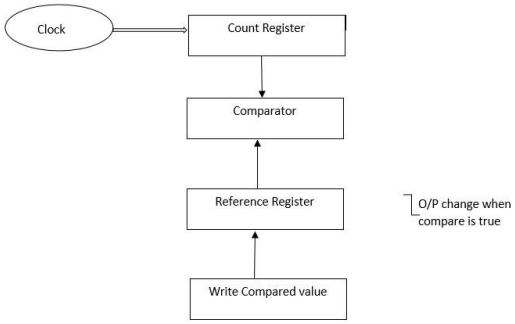
When an interrupt occurs the value of the timer is copied into CCP register CCPRI (CCPR1L and CCPR1H) from this value it well able to fine the arrival time of that event.

CAPTURE MODE OPERATION BLOCK DIAGRAM



COMPARE MODE

- COMPARES VALUES CONTAINED IN TWO REGISTERS AT SOME POINT.
- Makes use of the 16-bit TimerX resources, Timer1, Timer3, and Timers5.
- When a match occurs, one of the following events an occur.
    - driven high
    - Driven low
    - Toggle output
    - Remains unchanged

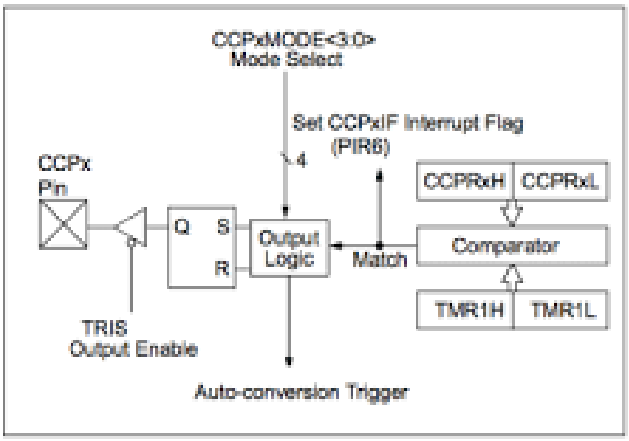


Simplified block diagram of operations

HOW TO USE COMPARE MODE?

- Makes a copy of the 16-bits timer values (timer 1 or Timer 3)
- Adds to this copy a delay count.
- Stores the sum in the CCPRxH : CCPRxL register pair.

COMPARE MODE OPERATION BLOCK DIAGRAM



PWM (PLUSE WIDTH MODULATION)

CCP1 1N PNW MODE

- Signals of varying frequency and duty cycle have a wide application mainly in automation.
- A scheme that provides power to a load by 8 witching quickly between fully ON and fully OFF states.
- The PWN signal resembles a square wave were,
  - The high portion of the signal → ON state
  - The low portion of the signal → OFF state
- Pulse width (high portion) can vary in time and defined in steps
- A larger number of steps applied which lengthens the pulse width
  →supplies more power to the load.
- A lower number of steps applied which lengthens the pulse width
  ➔ Supplies less power to the load.
- In PWM mode either Timer2 or Timer4(Timer6) may be used.

PWM Period

FiGure

PWM Period (T) (Duration of one Complete Cycle) $= t_{ON}$ **(ON time)** $+ t_{OFF}$ **(OFF time)**

PWM Resolution

The maximum number of steps that can be present in a single PWM period.

Ex- for 10 bit resolution, will result in 1024 (2^{10}) discrete duty cycles.

Duty Cycle

Figure

The ratio of the time T_c during which the O/P is high to the total time period T.

$$\text{Duty Cycle} = \frac{T_c}{T} \times 100$$

Duty Cycle α Power applied to the load

A power control circuit which is an example for PWM Mode.

Figure

Types of PWM Duty Cycles

1. ON-OFF PWM Duty Cycle

Time period: 16µs

Distributed PWM Cycle is more preferred compared to ON-OFF PWM Cycle.

2. Distributed PWM cycle

Time period: 16µs

PWM to generate an analog Voltage Level

In switched mode power supplies (SMPS) PWM is used to generate voltage levels, by changing the duty cycle of the PWM it can adjust the average voltage of the waveform. The PWM resolution should be selected to be greater than or equal to the required resolution of the power supply.

Ex:- A 5V power supply that can be adjusted to 1mV should use a PWM of resolution equal to or greater than 5000.

5V/5000 = 1mV

Figure

Any waveform can be generated by simply outputting a sequence of PWM waves to create analog voltage levels corresponding to different points in the waveform. When it uses more points, it can represent faster waveforms with greater accuracy. But it requires greater resolution PWM and heavier filtering.

Simplified PWM Block Diagram

Figure

Homework

- How do you the human hearing noise “Humming Sound” eradication with PWM of a DC motor in order to been used spike observation?

A PWM signal is at a fixed frequency. A very low PWM frequency will make the motor rotate in a sequence of ‘jerks’, a slightly higher frequency of a PWM motor signal should be 20kHz or more to ensure that the motor wine is outside the normal human hearing range[(20-20000)Hz].

Answer

PWM period = $(PR2+1) \times 4 \times Tosc \times TMR2$ pre scaler

$PR2 = [(PWM\ period \times Fosc)/ (4 \times Pre\ scaler)] - 1$

In this example it is desirable to set the PWM frequency above the range of human hearing. So that people are not bothered by the noise from the motors. The frequency chosen is, therefore;

Period = 1/25 kHz = 40µs

It is reasonable to set the pre scaler to 1 because the frequency needs to be, as high as possible. Hence the formula works ouCt as,

PR2 = [40 × 10-6s × 10 × 103 Hz / 4] – 1 = 99

ADC- Analog to Digital conversion
Basic of A/D Conversion

- Can convert only electrical voltage to digital voltage
- A transducer is needed to convert a non-electric quantity into an electrical voltage
- Different names of transducers are used for different physical quantities
- A data acquisition system is used those systems that perform A/D conversions

A/D Conversion Algorithms

1) Parallel (Flash) ADC
2) Slope and double slope ADC
3) Sigma-Delta ADC
4) Successive Approximation ADC

The PIC18 A/D Converter

- Has a 10-bit A/D Converter
- The number of analog inputs varies among different PIC18 devices
- Has the following registers
  - ✓ A/D Result High Register (ADRESH)
  - ✓ A/D Result Low Register (ADRESL)
  - ✓ A/D Control Register0 (ADCON0)
  - ✓ A/D Control Register1 (ADCON1)
  - ✓ A/D Control Register2 (ADCON2)

- The content of these registers vary with the PIC18 members

ADC Configuration

When configuring and using the ADC the following functions must be considered.

- ❖ Port configuration [ANSELX and TRISX registers configure the A/D Port Pins]
- ❖ Channel Selection
- ❖ ADC Voltage reference selection
- ❖ ADC Conversion clock source
- ❖ Interrupt Control
- ❖ Results Formatting

Serial Communication

Parallel communication easy but does not suitable for many communication devices. Since parallel communication needs many pins addition to 8 data pins available. Also we have limited number of pins to the parallel port to the interfacing and also need additional resources like a large number of connecting wires. So the serial communication is economical.

Start Bit and Stop Bit

PIC18F has a built-in asynchronous receiver-transmitter. Asynchronous means each character (data byte) is placed in between the start and stop bits. The scare bit is always (low) and the stop bit is always 1(HIGH).

Data Transfer Rate / Baud Rate

- The rate of data transfer in serial data communication is stared in bps (bits per second)
- Another widely used terminology for bps is baud rate, means, number of changes in signal per second. (The signal is in bits)

Therefore,
Bit rate = baud rate

SPBRG Register and Baud Rate in PIC18F

- The baud rate is programmable.
- Loaded into the SPBRG decides the Baud rate.
- Depend on crystal frequency.

$$DesiredBaudRate = \frac{FOSC}{4\times16\times(SPBRGvalue+1)}$$

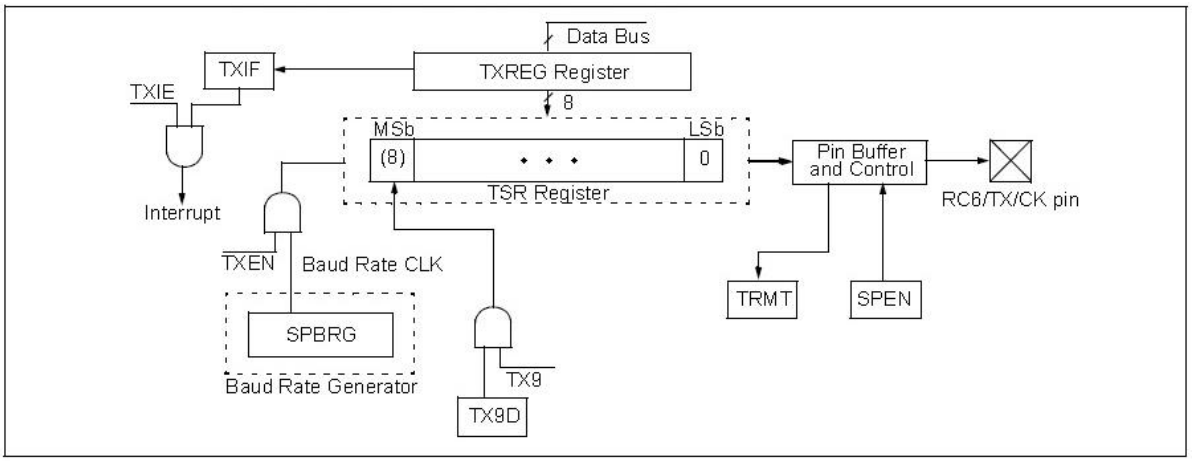
$$SPBRGvalue = \frac{FOSC}{4\times16\times DesiredBaudRate} - 1$$

USART

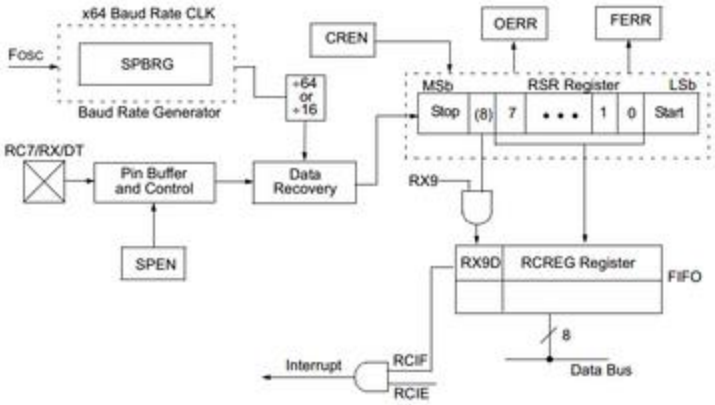
- To communicate with external components such as computers or microcontrollers, the PIC micro uses a component called USART (Universal Synchronous Asynchronous Receiver Transmitter)
  - This component can be configured as
    - A full-Duplex asynchronous system
    - Can communicate with peripheral devices, such as CRT terminals and personal computers.
  - A Half-Duplex synchronous system
    - Can communicate with peripheral devices, such as AKD or D/A JCS, Serial EEPROMs etc.
- To enable the serial communication with PIC micro it must set different parameters within two registers.
  - TXSTA – Transmit status and control Register
  - RXSTA – Receive Status and control Register

Homework
Draw the block diagrams of USART transmitter and receiver

USART TRANSMIT BLOCK DIAGRAM



USART RECEIVER BLOCK DIAGRAM



I2C Communication

- The incer-Intergrated circuit (I^2C) Bus is a multi-master serial data communication bus.
- Devices communicate in a master/slave environment where the master devices initiate the communication.
- A slave device is controlled through addressing.
- The I2C bus specifies two signal conditions.
    - Serial Clock (SCLX)
    - Serial Data (SDAX)

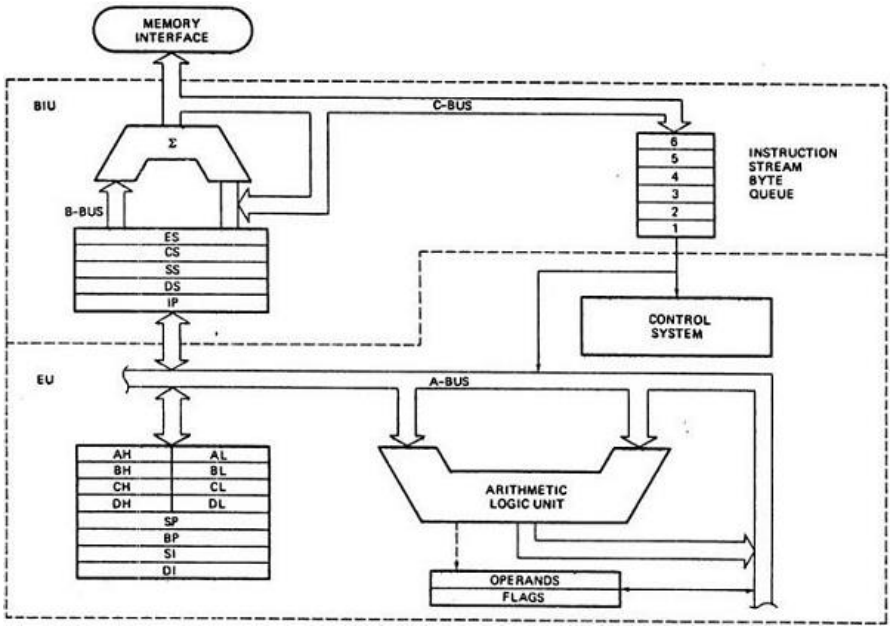
I2C Master

- Need to write to SSPCONI and SSPADD registers to configure MSSP module as I2C Master and to set the clock frequency of I2C communication respectively.

I2C Slave

- Need to write to SSPCONI and SSPADD registers to configure MSSP module as I2C slave and to set the clock frequency of I2C communication respectively.

Intel 8086 Microprocessor - Architecture



8086 μP is divided into two functional units.

1) EU (Execution unit)
2) BIU (Bus interface unit)

EU

Eu give instruction to BIO starting from where to fetch the data and the decode and execute those instructions.

ALU

Handle all arithmetic and logical operations

Flags

Conditional flags

It represents the result of the last arithmetic or logical instruction execution.

- Carry flag – Indicates an overflow condition for arithmetic operation.
- Auxiliary flag – To perform binary to BCD conversion.
- Parity flag – Indicates the parity of the results.
- Zero flag – Indicates whether the result is zero or not
- Sign flag – Holds sign of the result.
- Overflow flag – Indicates whether the system capacity is exceeded or nor

Control flag

Controls the operation of the EU

- Trap flag
- Interrupt flag
- Direction flag

General Purpose Registers (GPRs)

- Ax register (Accumulator Register) – Hold data result addresses etc.
- Bx register (Base Register) – Holds offset address of a memory location.
- Cx register (Counter Register) – Holds count for various instructions.
- Dx register (Data Register) – Holds data ports and results etc.

Special Function Registers (SFRs)

- SP (Stack Pointer) – Points to stack cop
- BP (Base Pointer) – Holds offset address of anylocation in the SS.
- SI (Source Index) – Holds offset address of DS during string operations.
- DI (Destination Index) – Holds offset address of ES during strong operations.

BIU

- Provides the interface of 8086 to external memory and I/O devices via the system Bus.
- Performs various machine cycles such as
    - Memory read
    - I/O read etc.

        To transfer data between memory and I/O devices.

6-Byte Pre-fetch Queue

- It is a 6-byte queue (FIFO)
- Fetching the next instruction while executing the current instruction is called pipelining.
- Gets flushed whenever a branch instruction occurs.

Segment Registers

Segment	***Offset***	***Purpose (Holds base address for)***
Code (CS)	IP	Code Segment
Stack (SS)	SP or BP	Stack Segment
Data (BS)	DI or SI	Data Segment
Extra (ES)	DI for Strings	Extra Segment

Instruction Pointer (IP) – Holds offset of the next instructions in the code segment.

INSRUCTIONS SET OF INTEL 8086

01.DATA TRANSFER INSTRUCTIONS

Used to transfer the data from the source operand to the destination operand.

-Instruction to a word transfer.

- MOV-Used to copy the byte or word from the provided source to the provided destination.
- POP-Used to get a word from the top of the stack to the provided location.

-Instructions for I/O port transfer.

Examples: In our.

-Instructions to transfer the address.

Examples: LEA,LDS,LES.

-Instructions to transfer flag registers.

Examples: POPF,PUSHF.

02.ARITHMETIC INSTRUCTIONS.

- Used to perform arithmetic operations.
  Examples:
  -ADD-Used to add the provided byte to byte/word to word.
  -SUB-Used to subtract the byte from the byte/word from a word.
  -DIV-Used to divide the unsigned word by byte or unsigned double word by word.
  -MOL-Used to multiply unsigned byte by byte/word by word.

03.BIT MANIPULATION INSTRUCTIONS.

- USED to perform operations where data bits are involved.
  -Instruction to perform logical operations
  Examples: NOT,AND,OR,XOR.
  -Instructions to perform shift operations.
  - SHL/SAL-Used to shift bits of a byte/word towards left and put o(s).
  - SHR-Used to shift bits of a byte/word towards the right and put o(s).
  - SAR-Used to shift bits of a byte/word towards the right and copy the old MSB into the new /MSB.

  -Instructions to perform rotate operations.
  - ROL-Used to rotate bits of byte/word towards the left.
  - ROR-Used to rotate bits of bits of byte/word towards the left.

04.STRING INSTRUCTIONS

- A group of byte/words and their memory is always allocated in a sequential order.

-MOVS/MOVSB/MOVSW-Used to move the byte/word from one string to another.

05.BRANCH AND LOOP INSTRUCTIONS

- Used to transfer/branch the instructions during an execution.
  -CALL-Used to call a procedure and save their return address to the stack.
  -RET-Used to return from the procedure to the main program.
  -JMP-Used to jump to the provided address to proceed to the next instruction.
  -Loop-Used to loop a group of instructions until che cond satisfies.

06.PROCESSOR CONTROL INSTRUCTIONS

- Used to control the processor action by setting/resetting the flags values.
  Examples: STC,CLO,CMC.

07.SUBROUTING AND INTERRUPT INSTRUCTION

- Used to call the interrupt during program execution.
  -INT-Used to interrupt the program during program execution.

  -INTO-Used to interrupt the program during execution.

ADDRESSING MODES OF INTEL 8086

01.IMMEDIATE ADDRESSING MODES

- The addressing mode in which the data operand is a part of the instruction itself is known as immediate addressing mode.
  Examples: MOV CX,4929H

  ADD AX,2387H

02.REGISTER ADDRESSING MODES

- The register is the source of an operand for an instruction.
  Examples: MOV CX,AX

  ADD BX,AX

03.DIRECT ADDRESSING MODES

- The effective address of the memory location is written directly in the instruction.
  Examples: MOV AX,[1592H]

  MOV AL,[0300H]

04.REGISTER INDIRECT ADDRESSING MODES

- Allows data to be addressed at any memory location through an offset address held in any of the following registers: BP,BX,DI and SI.
  Examples: MOV AX, [BX]

  ADD CX, [BX]

05.REGISTER RELATING ADDRESSING MODE

- The data in a segment of memory are addressed by adding the displacement to the contents of a base or an index register.
  Examples: MOV DL,[DI]

06.BASE-PLUS-INDEX ADDRESSING MODE

- Indirectly addresses memory data.
  Examples: MOV DL,[ENX+EBX]

07.BASE RELATIVE-PLUS-INDEX ADDRESSING MODE

- It adds a displacement, besides using a Base register and an Index register to form the memory address. This addresses a two dimensional array of memory data.
  Examples: MOV AX,[EX+SI+LOOH]

SEGMENTED MEMORY ARCHITECTURE

From Intel 8086/88 MCP onwards physical memory formation, Segmentation and paging techniques were intenduced in order to enhance the availability of up to speed up and specially to enhance the relocation techniques of the processor.

-A segment addresses 64k of memory.

-A segment register controls the starting location of segment.

-An offset of the distance, from the beginning of segment to a particular instruction or variable.

ADVANTAGES OF USING SEGMENT REGISTERS

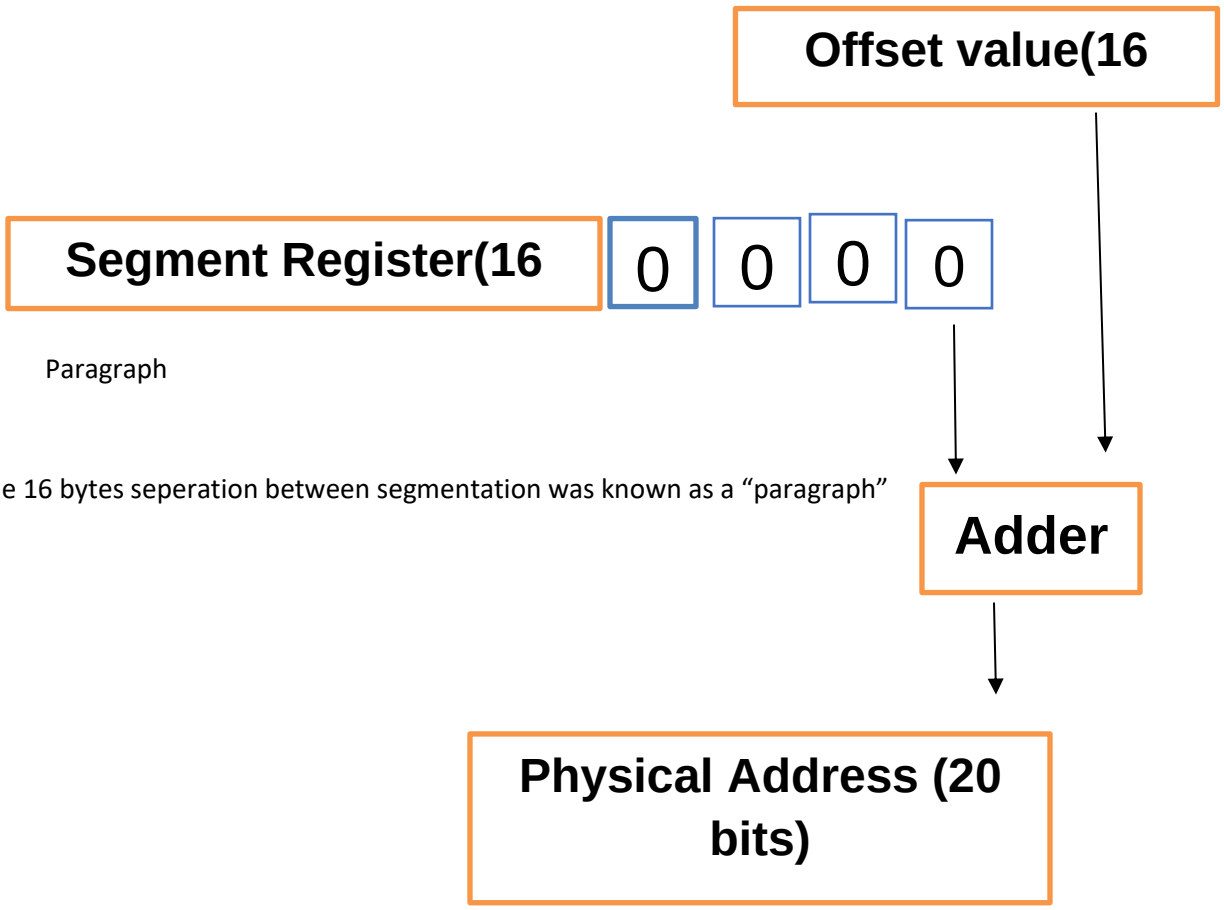
- Allow the memory capacity to be 1 Mb even through the addresses associated with the individual instructions are only 16 bits wide.
- Allow the instruction, data or stack porcino of a program to be more than 64 Kb long by using more than one code, data or stack segment.
- Facilitate the use of separate memory areas for a program, its data and the stack.
- Permit a program and/or its data to be put into different areas of memory each time the program is executed.

SEGMENT MEMORY

Segmented memory addressing absolute (linear) address is a combination of a 16-bit segment value added to a 16-bit offset.

MEMORY ADDRESS GENERATION

The BIU has a dedicated adder for determining physical memory addresses.



Paragraph

The 16 bytes seperation between segmentation was known as a “paragraph”

PHYSICAL ADDRESS CALCULATIONS

If DS=7FA2H and the offset is 438EH,

(a) Calculate physical Address.
7FA2HX10H+(OFFSET ADDRESS)H

=8BDAEH

(b) Calculate lower range.
7FA2HX10H+0000H

=7FA20H

(c) Calculate upper range.
7FA2HX10H+FFFFH

=83DAEH

(d) Write the logical address.
7FA2H:438EH

FOR PHYSICAL ADDRESS CALCULATION

Physical Address =Register Value(H)*10H+Offset Value(H)

Upper Level/Range =Register Value(H)*10H+FFFFH

Lower Level/Range =Register Value(H)*10H+0000H

-For Lower level
Physical Address = Register x 10H + 0000H
(PA) value (H)

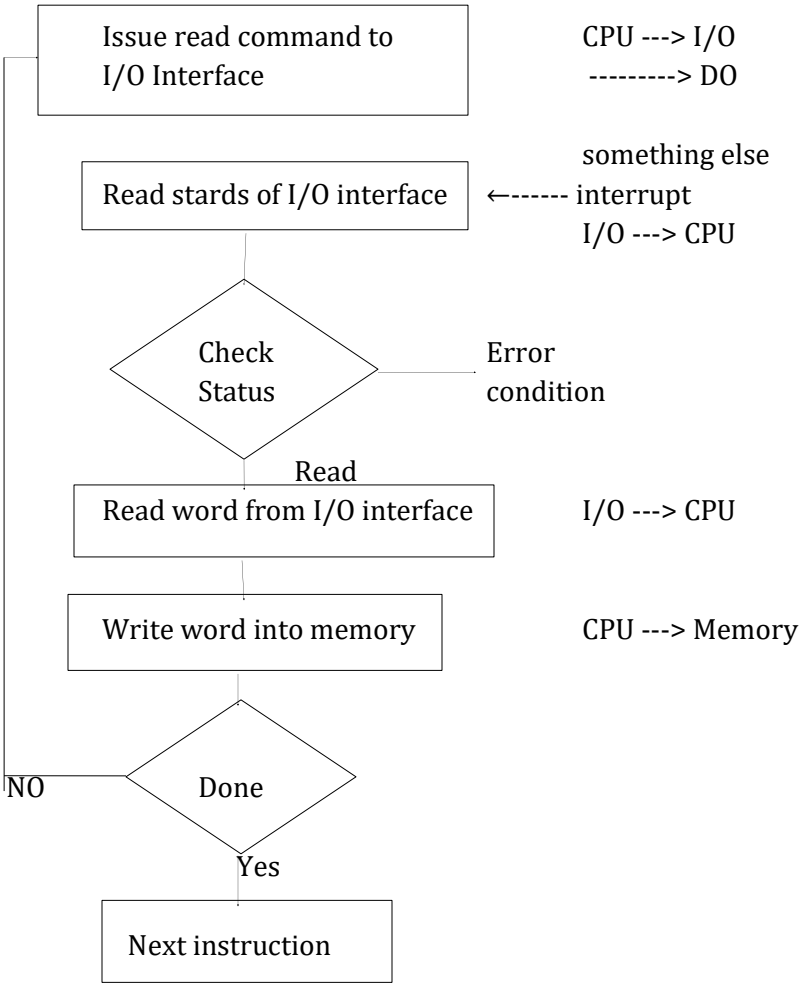
I/O INTERFACE

• I/O operations are accomplished through external devices that exchange data between the external environment and the computer.
• External devices (peripherals) attach to the computer by a link to an I/O device Interface
• The link exchange control, status and data between the I/O interface and external device

MULTITASKING OPERATIONS IN INTEL 8086

• Memory interfacing & I/O interfacing
• Parallel communication interface (8255 PPI)
• Serial communication interface (8251 USART)
• D/A and A/D interface (ADC 0800/0809, DAC 0800)
• Timer {or counter} - {8253/8254 Timer}
• Keyboard /display controller {8279}
• Interrupt controller {8239}
• DMA controller { 8037/8257}
• Programing and applications case studies.

INTERRUPT-DRIVEN I/O

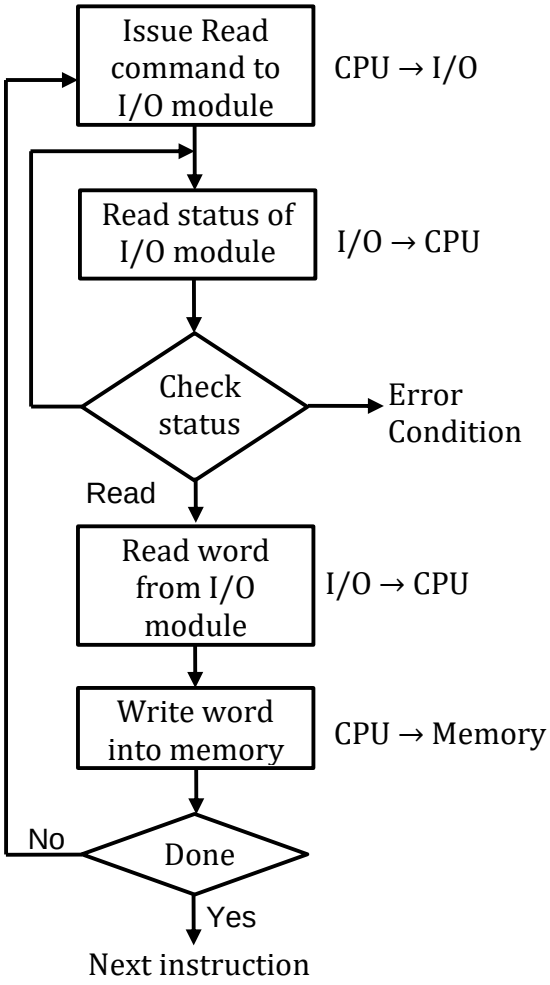


Home work

Draw the flow charts of three techniques for input of block of data.

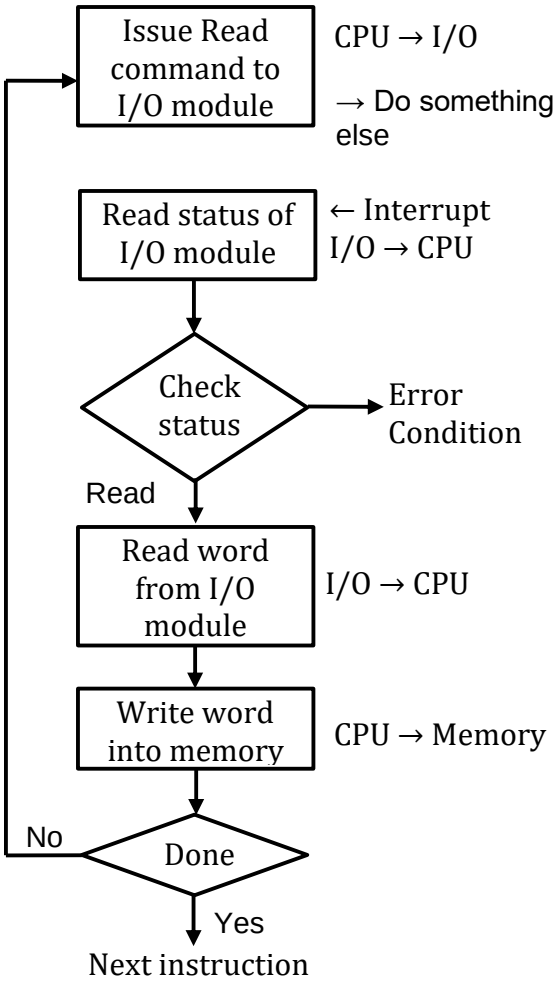
THREE TECHNIQUES FOR INPUT OF A BLOCK OF A DATA.

a) PROGRAMMED I/O



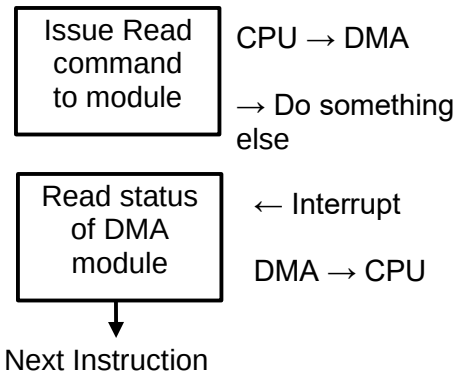
- Due to the result of the I/O instruction written in the program
  - I/O device doesn't have direct address to the memory unit.
- Requires the execution of several instructions by the CPU,
  - I/P instruction to transfer data from device to CPU.
  - Store instruction to transfer data from CPU to memory.
- CPU stays in the loop until the I/O unit indicates that is ready for data transfer.
- A time consuming process since it needlessly keeps CPU busy.

b) INTERRUPT DRIVEN



- To avoid keeping the CPU busy unnecessarily
- By using interrupt facility and special commands to inform the interface to issue an interrupt request whenever data is available from any device.
- In the meantime CPU can proceed for any other program execution.
- The interface meanwhile keep monitoring the device.
- When it determines the device is ready to transfer data, it initiate interrupt signal to CPU.
- Upon detection, of an external interrupt signal, CPU stops momentarily the task that et was already performing, branches to the service program to process the I/O transfer.
- Return to the task it was originally performing

c) DIRECT MEMORY ACCESS



- Data transfer between a fast storage media such as magnetic disk.
- Memory unit is limited by the speed of the CPU.
- Can allow the peripherals directly communicate with the each other using memory buses, removing the integration of the CPU the
- CPU is idle and it has no control over memory buses

COMPUTER MEMORY HIERARCHY