Microcontrollers and Embedded Systems

Shyama P Das
Professor
Dept of Electrical Engg
IIT Kanpur

Lecture 1 : Introduction to Microcontrollers & Microprocessors

What is a Microcontroller?
A Microcontroller is a programmable digital processor with necessary peripherals. Both microcontrollers and microprocessors are complex sequential digital circuits meant to carry out job according to the program / instructions. Sometimes analog input/output interface makes a part of microcontroller circuit of mixed mode(both analog and digital nature).
A microcontroller can be compared to a Swiss knife with multiple functions incorporated in the same IC.

Fig. 1.1 A Microcontroller compared with a Swiss knife

Microcontrollers Vs Microprocessors

1. A microprocessor requires an external memory for program/data storage. Instruction execution requires movement of data from the external memory to the microprocessor or vice versa. Usually, microprocessors have good computing power and they have higher clock speed to facilitate faster computation.
2. A microcontroller has required on-chip memory with associated peripherals. A microcontroller can be thought of a microprocessor with inbuilt peripherals.
3. A microcontroller does not require much additional interfacing ICs for operation and it functions as a stand alone system. The operation of a microcontroller is multipurpose, just like a Swiss knife.
4. Microcontrollers are also called embedded controllers. A microcontroller clock speed is limited only to a few tens of MHz. Microcontrollers are numerous and many of them are application specific.

Development/Classification of microcontrollers (Invisible)
Microcontrollers have gone through a silent evolution (invisible). The evolution can be rightly termed as silent as the impact or application of a microcontroller is not well known to a common user, although microcontroller technology has undergone significant change since early 1970's. Development of some popular microcontrollers is given as follows.

Intel 4004	4 bit (2300 PMOS trans, 108 kHz)	1971
Intel 8048	8 bit	1976
Intel 8031	8 bit (ROM-less)	.
Intel 8051	8 bit (Mask ROM)	1980

Microchip PIC16C64	8 bit	1985
Motorola 68HC11	8 bit (on chip ADC)	.
Intel 80C196	16 bit	1982
Atmel AT89C51	8 bit (Flash memory)	.
Microchip PIC 16F877	8 bit (Flash memory + ADC)	.

Development of microprocessors (Visible)

Microprocessors have undergone significant evolution over the past four decades. This development is clearly perceptible to a common user, especially, in terms of phenomenal growth in capabilities of personal computers. Development of some of the microprocessors can be given as follows.

Intel 4004	4 bit (2300 PMOS transistors)	1971
Intel 8080 8085	8 bit (NMOS) 8 bit	1974
Intel 8088 8086	16 bit 16 bit	1978
Intel 80186 80286	16 bit 16 bit	1982
Intel 80386	32 bit (275000 transistors)	1985
Intel 80486 SX DX	32 bit 32 bit (built in floating point unit)	1989
Intel 80586 I MMX Celeron II III IV	64 bit	1993 1997 1999 2000
Z-80 (Zilog)	8 bit	1976
Motorola Power PC 601 602 603	32-bit	1993 1995

Lecture 2 : Basic Architectures of Microcontrollers

We use more number of microcontrollers compared to microprocessors. Microprocessors are primarily used for computational purpose, whereas microcontrollers find wide application in devices needing real time processing / control.
Application of microcontrollers are numerous. Starting from domestic applications such as in washing machines, TVs, airconditioners, microcontrollers are used in automobiles, process control industries , cell phones, electrical drives, robotics and in space applications.

Microcontroller Chips

Broad Classification of different microcontroller chips could be as follows:

- Embedded (Self -Contained) 8 - bit Microcontroller
- 16 to 32 Microcontrollers
- Digital Signal Processors

Features of Modern Microcontrollers

- Built-in Monitor Program
- Built-in Program Memory
- Interrupts
- Analog I/O
- Serial I/O
- Facility to Interface External Memory
- Timers

Internal Structure of a Microcontroller

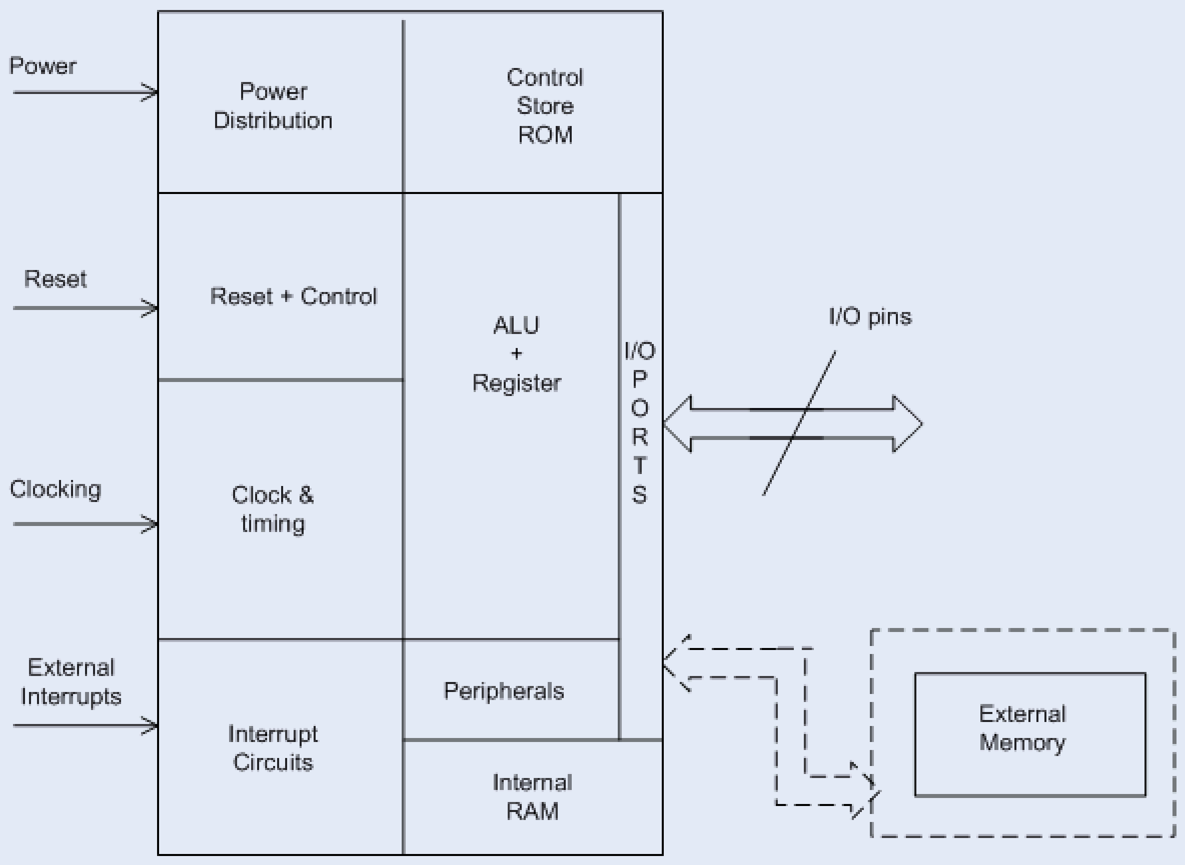


Fig. 2.1 Internal Structure of a Microcontroller

At times, a microcontroller can have external memory also (if there is no internal memory or extra memory interface is required). Early microcontrollers were manufactured using bipolar or NMOS technologies. Most modern microcontrollers are manufactured with CMOS technology, which leads to reduction in size and power loss. Current drawn by the IC is also reduced considerably from 10mA to a few micro Amperes in sleep mode(for a microcontroller running typically at a clock speed of 20MHz).

Harvard vs. Princeton Architecture

Many years ago, in the late 1940's, the US Government asked Harvard and Princeton universities to come up with a computer architecture to be used in computing distances of Naval artillery shell for defense applications. Princeton suggested computer architecture with a single memory interface. It is also known as Von Neumann architecture after the name of the chief scientist of the project in Princeton University John Von Neumann (1903 - 1957 Born in Budapest, Hungary).

Harvard suggested a computer with two different memory interfaces, one for the data / variables and the other for program / instructions. Although Princeton architecture was accepted for simplicity and ease of implementation, Harvard architecture became popular later, due to the parallelism of instruction execution.

Princeton Architecture (Single memory interface)

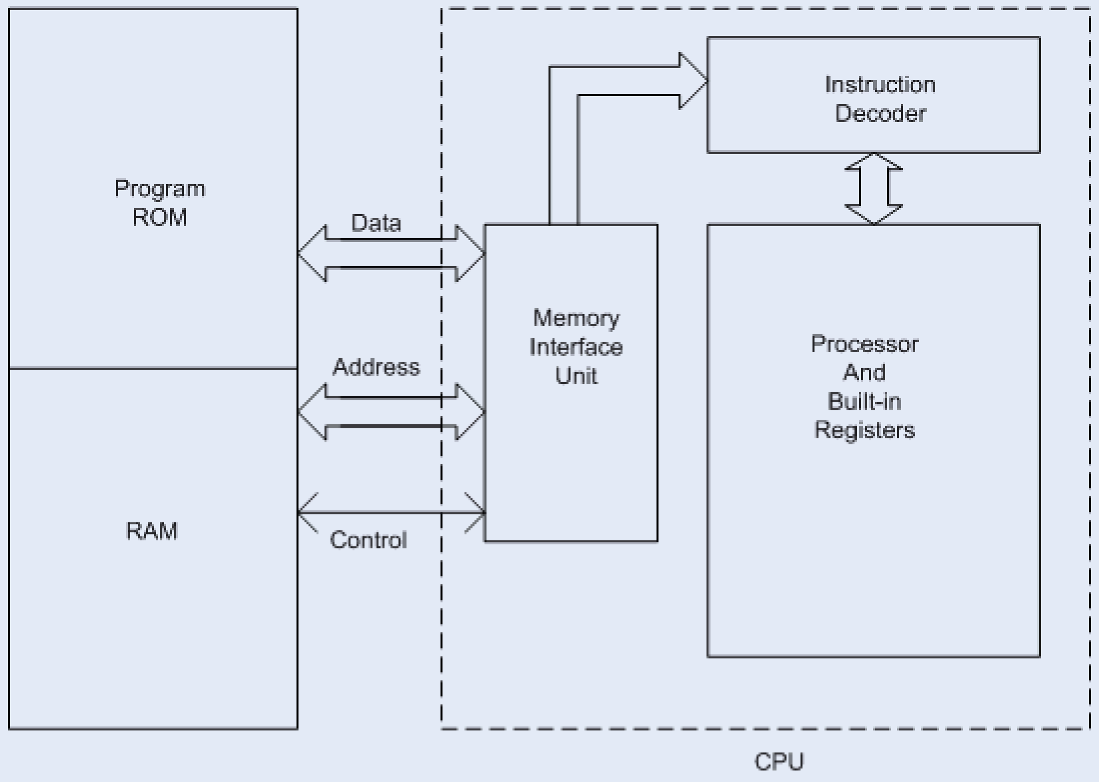


Fig. 2.2 Princeton Architecture

Example : An instruction "Read a data byte from memory and store it in the accumulator" is executed as follows: -

Cycle 1 - Read Instruction
Cycle 2 - Read Data out of RAM and put into Accumulator

Harvard Architecture (Separate Program and Data Memory interfaces)

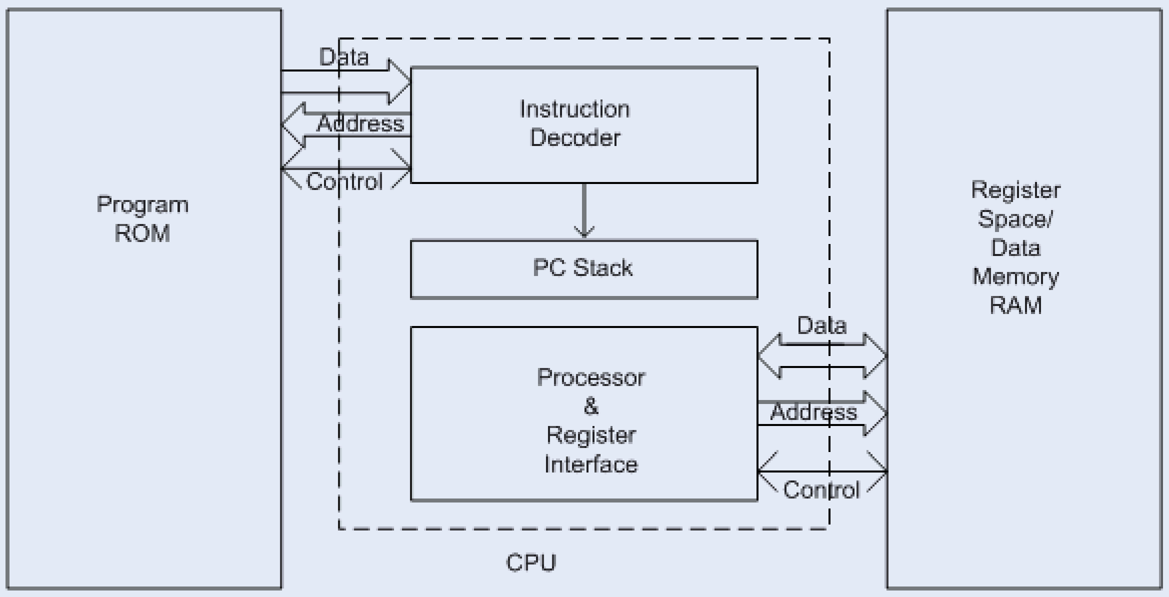


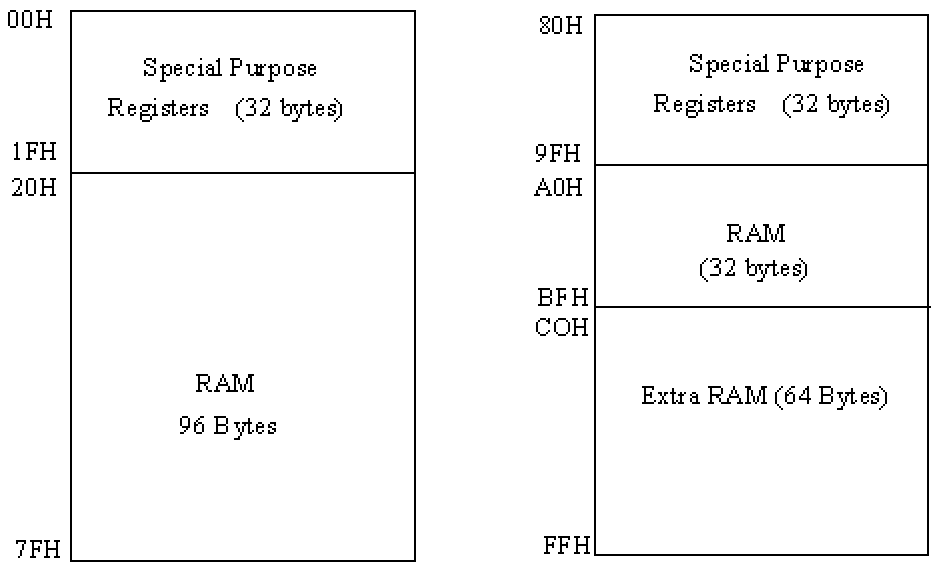
Fig. 2.3 Harvard Arcitecture

The same instruction (as shown under Princeton Architecture) would be executed as follows:

Cycle 1
- Complete previous instruction
- Read the "Move Data to Accumulator" instruction

Cycle 2
- Execute "Move Data to Accumulator" instruction
- Read next instruction

Hence each instruction is effectively executed in one instruction cycle, except for the ones that modify the content of the program counter. For example, the "jump" (or call) instructions takes 2 cycles. Thus, due to parallelism, Harvard architecture executes more instructions in a given time compared to Princeton Architecture.



Specifications of some popular PIC microcontrollers are as follows:

Device	Program Memory (14bits)	Data RAM (bytes)	I/O Pins	ADC	Timers 8/16 bits	CCP (PWM)	USART SPI / I2C
16C74A	4K EPROM	192	33	8 bits x 8 channels	2/1	2	USART SPI / I^2C
16F877	8K Flash	368 (RAM) 256 (EEPROM)	33	10 bits x 8 channels	2/1	2	USART SPI / I^2C

Device	Interrupt Sources	Instruction Set
16C74A	12	35
16F877	15	35

PIC Microcontroller Clock

Most of the PIC microcontrollers can operate upto 20MHz. One instructions cycle (machine cycle) consists of four clock cycles.

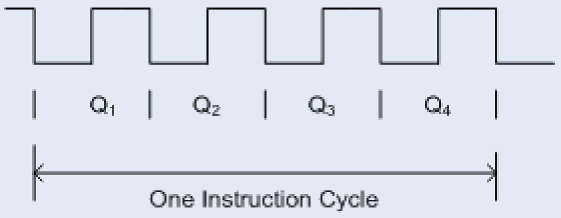


Fig 3.1 Relation between instruction cycles and clock cycles for PIC microcontrollers

Instructions that do not require modification of program counter content get executed in one instruction cycle.

Although the architectures of various midrange 8 - bit PIC microcontroller are not the same, the variation is mostly interns of addition of memory and peripherals. We will discuss here the architecture of a standard mid-range PIC microcontroller, 16C74A. Unless mentioned otherwise, the information given here is for a PIC 16C74A microcontroller Chip.

Architecture of PIC16C74A

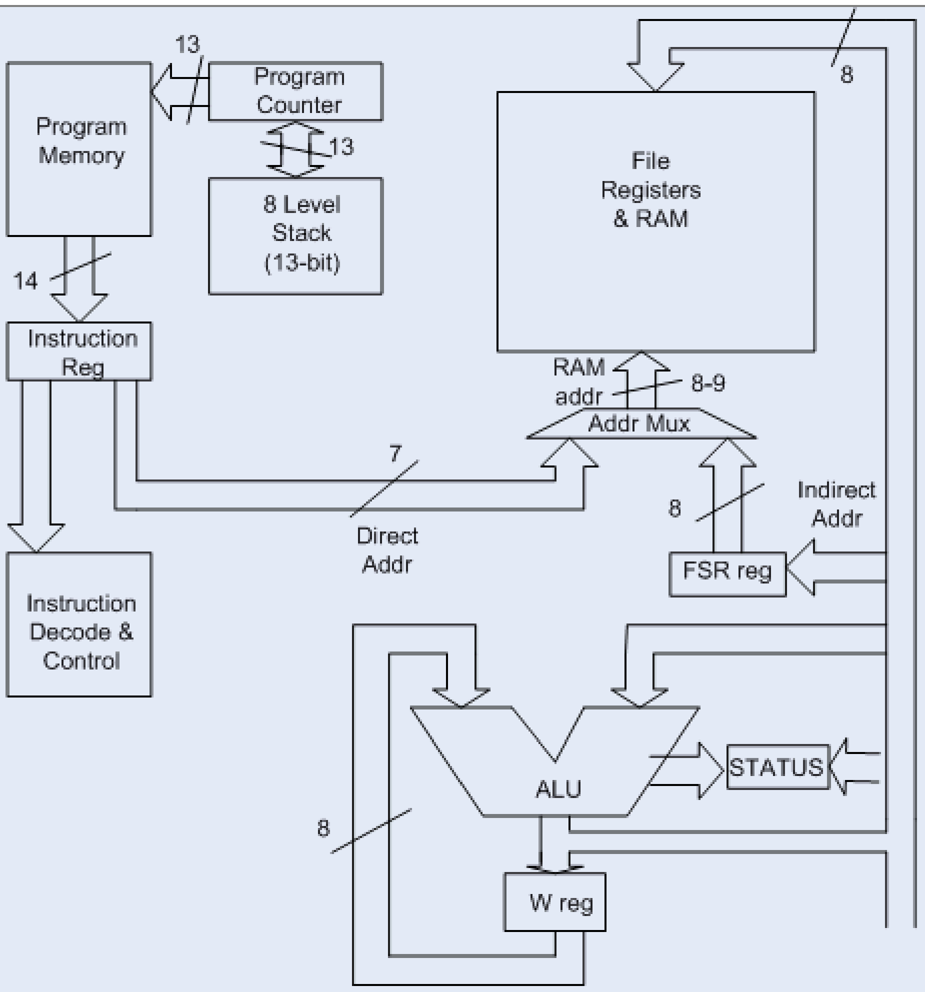


Fig 3.2 Basic Architecture of PIC 16C74A

The basic architecture of PIC16C74A is shown in fig 17.2. The architecture consists of Program memory, file registers and RAM, ALU and CPU registers. It should be noted that the program Counter is 13 - bit and the program memory is organised as 14 - bit word. Hence the program Memory capacity is 8k x 14 bit. Each instruction of PIC 16C74A is 14 - bit long. The various CPU registers are discussed here.

CPU registers (registers commonly used by the CPU)

W, the working register, is used by many instructions as the source of an operand. This is similar to accumulator in 8051. It may also serve as the destination for the result of the instruction execution. It is an 8 – bit register.

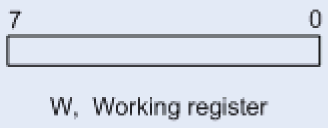


Fig 3.3 W register

STATUS Register

The STATUS register is a 8-bit register that stores the status of the processor. This also stores carry, zero and digit carry bits.

STATUS – address 03H, 83H

7	6	5	4	3	2	1	0
0	0	RP0	NOT_TO	NOT_PD	Z	DC	C

Fig 3.4 STATUS register

C = Carry bit

DC = Digit carry (same as auxiliary carry)

Z = Zero bit

NOT_TO and NOT_PD – Used in conjunction with PIC's sleep mode
RP0– Register bank select bit used in conjunction with direct addressing mode.

FSR Register

(File Selection Register, address = 04H, 84H)
FSR is an 8-bit register used as data memory address pointer. This is used in indirect addressing mode.

INDF Register

(INDirect through FSR, address = 00H, 80H)
INDF is not a physical register. Accessing INDF access is the location pointed to by FSR in indirect addressing mode.

PCL Register

(Program Counter Low Byte, address = 02H, 82H)
PCL is actually the lower 8-bits of the 13-bit program counter. This is a both readable and writable register.

PCLATH Register

(Program Counter Latch, address = 0AH, 8AH)
PCLATH is a 8-bit register which can be used to decide the upper 5bits of the program counter. PCLATH is not the upper 5bits of the program counter. PCLATH can be read from or written to without affecting the program counter. The upper 3bits of PCLATH remain zero and they serve no purpose. When PCL is written to, the lower 5bits of PCLATH are automatically loaded to the upper 5bits of the program counter, as shown in the figure.

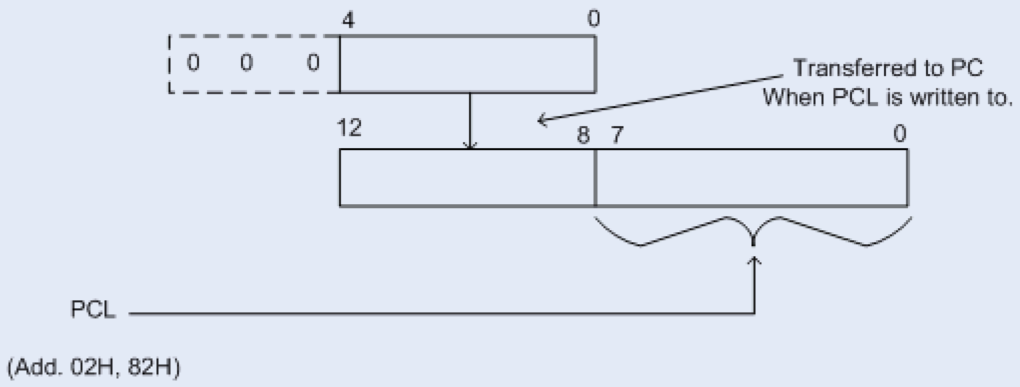
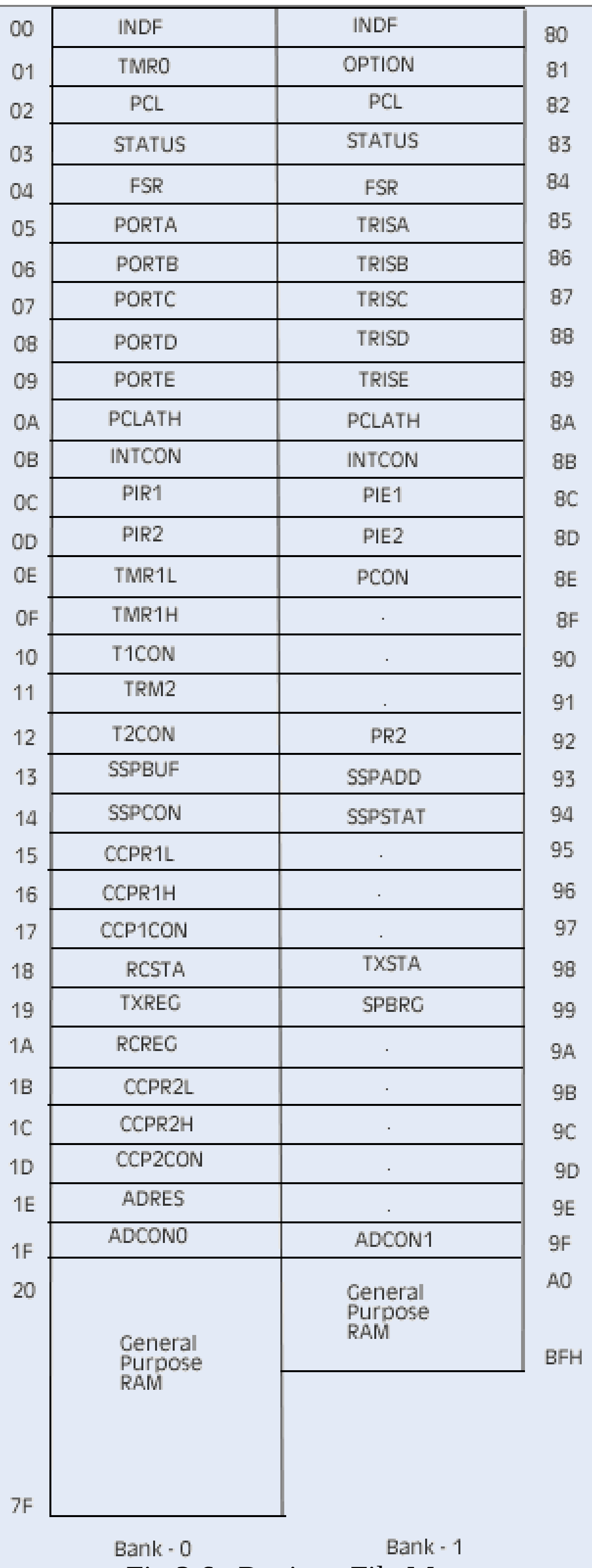


Fig 3.5 Schematic of how PCL is loaded from PCLATH

Program Counter Stack

An independent 8-level stack is used for the program counter. As the program counter is 13bit, the stack is organized as 8x13bit registers. When an interrupt occurs, the program counter is pushed onto the stack. When the interrupt is being serviced, other interrupts remain disabled. Hence, other 7 registers of the stack can be used for subroutine calls within an interrupt service routine or within the mainline program.

Register File Map



00	INDF	INDF	80
01	TMR0	OPTION	81
02	PCL	PCL	82
03	STATUS	STATUS	83
04	FSR	FSR	84
05	PORTA	TRISA	85
06	PORTB	TRISB	86
07	PORTC	TRISC	87
08	PORTD	TRISD	88
09	PORTE	TRISE	89
0A	PCLATH	PCLATH	8A
0B	INTCON	INTCON	8B
0C	PIR1	PIE1	8C
0D	PIR2	PIE2	8D
0E	TMR1L	PCON	8E
0F	TMR1H	.	8F
10	T1CON	.	90
11	TRM2	.	91
12	T2CON	PR2	92
13	SSPBUF	SSPADD	93
14	SSPCON	SSPSTAT	94
15	CCPR1L	.	95
16	CCPR1H	.	96
17	CCP1CON	.	97
18	RCSTA	TXSTA	98
19	TXREG	SPBRG	99
1A	RCREG	.	9A
1B	CCPR2L	.	9B
1C	CCPR2H	.	9C
1D	CCP2CON	.	9D
1E	ADRES	.	9E
1F	ADCON0	ADCON1	9F
20 – 7F	General Purpose RAM	General Purpose RAM	A0 – BFH
	Bank - 0	Bank - 1	

Fig 3.6 Register File Map

It can be noted that some of the special purpose registers are available both in Bank-0 and Bank-1. These registers have the same value in both banks. Changing the register content in one bank automatically changes its content in the other bank.

Port Structure and Pin Configuration of PIC 16C74A

As mentioned earlier, there is a large variety of PIC microcontrollers. However, the midrange architectures are widely used. Our discussion will mainly confine to PIC16C74A whose architecture has most of the

required features of a mid-range PIC microcontroller. Study of any other mid-range PIC microcontroller will not cause much variation from the basic architecture of PIC 16C74A ..
PIC 16C74A has 5 I/O Ports. Each port is a bidirectional I/O port. In addition, they have the following alternate functions.

Port	Altenative uses of I/O pins	No.of I/O pins
Port A	A/D Converter inputs	6
Port B	External interrupt inputs	8
Port C	Serial port, Timer I/O	8
Port D	Parallel slave port	8
Port E	A/D Converter inputs	3
	Total I/O pins	33
	Total pins	40

In addition to I/O pins, there is a Master clear pin (MCLR) which is equivalent to reset in 8051. However, unlike 8051, MCLR should be pulled low to reset the micro controller. Since PIC16C74Ahas inherent power-on reset, no special connection is required with MCLR pin to reset the micro controller on power-on.

There are two V_{DD} pins and two V_{SS} pins. There are two pins (OSC1 and OSC2) for connecting the crystal oscillator/ RC oscillator. Hence the total number of pins with a 16C74A is 33+7=40. This IC is commonly available in a dual-in-pin (DIP) package.

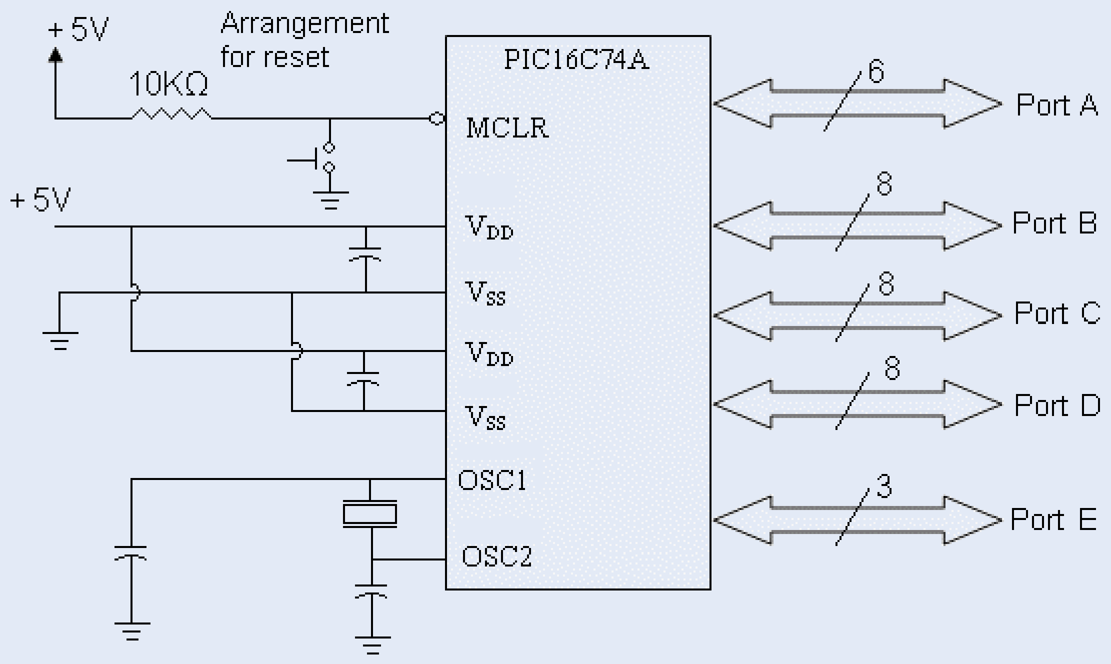


Fig 3.7 Pin configuration of PIC 16C74A

Lecture 4 : Instruction Set of PIC Microcontroller

Guidelines from Microchip Technology

For writing assembly language program Microchip Technology has suggested the following guidelines.

1. Write instruction mnemonics in lower case. (e.g., movwf)
2. Write the special register names, RAM variable names and bit names in upper case. (e.g., PCL, RP0, etc.)
3. Write instructions and subroutine labels in mixed case. (e.g., Mainline, LoopTime)

Instruction Set:

The instruction set for PIC16C74A consists of only 35 instructions. Some of these instructions are byte oriented instructions and some are bit oriented instructions.

The **byte oriented instructions** that require two parameters (For example, movf f, F(W)) expect the f to be replaced by the name of a special purpose register (e.g., PORTA) or the name of a RAM variable (e.g., NUM1), which serves as the source of the operand. ‘f' stands for file register. The F(W) parameter is the destination of the result of the operation. It should be replaced by:
F, if the destination is to be the source register.
W, if the destination is to be the working register (i.e., Accumulator or W register).
The **bit oriented instructions** also expect parameters (e.g., btfsc f, b). Here ‘f' is to be replaced by the name of a special purpose register or the name of a RAM variable. The ‘b' parameter is to be replaced by a bit number ranging from 0 to 7.
For example:

```
Z equ 2
btfsc STATUS, Z
```

Z has been equated to 2. Here, the instruction will test the Z bit of the STATUS register and will skip the next instruction if Z bit is clear.
The **literal instructions** require an operand having a known value (e.g., 0AH) or a label that represents a known value.
For example:

```
NUM equ 0AH ;            Assigns 0AH to the label NUM ( a constant )
movlw NUM ;              will move 0AH to the W register.
```

Every instruction fits in a single 14-bit word. In addition, every instruction also executes in a single cycle, unless it changes the content of the Program Counter. These features are due to the fact that PIC micro controller has been designed on the principles of RISC (Reduced Instruction Set Computer) architecture.

Instruction set:

Mnemonics	Description	Instruction Cycles
bcf f, b	Clear bit b of register f	1
bsf f, b	Set bit b of register f	1
clrw	Clear working register W	1

clrf f	Clear f	1
movlw k	Move literal ‘k' to W	1
movwf f	Move W to f	1
movf f, F(W)	Move f to F or W	1
swapf f, F(W)	Swap nibbles of f, putting result in F or W	1
andlw k	And literal value into W	1
andwf f, F(W)	And W with F and put the result in W or F	1
andwf f, F(W)	And W with F and put the result in W or F	1
iorlw k	inclusive-OR literal value into W	1
iorwf f, F(W)	inclusive-OR W with f and put the result in F or W	1
xorlw k	Exclusive-OR literal value into W	1
xorwf f, F(W)	Exclusive-OR W with f and put the result in F or W	1
addlw k	Add the literal value to W and store the result in W	1
addwf f, F(W)	Add W to f and store the result in F or W	1
sublw k	Subtract the literal value from W and store the result in W	1
subwf f, F(W)	Subtract f from W and store the result in F or W	1
rlf f, F(W)	Copy f into F or W; rotate F or W left through the carry bit	1
rrf f, F(W)	Copy f into F or W; rotate F or W right through the carry bit	1
btfsc f, b	Test ‘b' bit of the register f and skip the next instruction if bit is clear	1 / 2
btfss f, b	Test ‘b' bit of the register f and skip the next instruction if bit is set	1 / 2
decfsz f, F(W)	Decrement f and copy the result to F or W; skip the next instruction if the result is zero	1 / 2
incfcz f, F(W)	Increment f and copy the result to F or W; skip the next instruction if the result is zero	1 / 2
goto label	Go to the instruction with the label “label”	2
call label	Go to the subroutine “label”, push the Program Counter in the stack	2
retrun	Return from the subroutine, POP the Program Counter from the stack	2
retlw k	Retrun from the subroutine, POP the Program Counter from the stack; put k in W	2
retie	Return from Interrupt Service Routine and re-enable interrupt	2
clrwdt	Clear Watch Dog Timer	1
sleep	Go into sleep/ stand by mode	1
nop	No operation	1

Encoding of instruction:

As has been discussed, each instruction is of 14-bit long. These 14-bits contain both op-code and the operand. Some examples of instruction encoding are shown here.

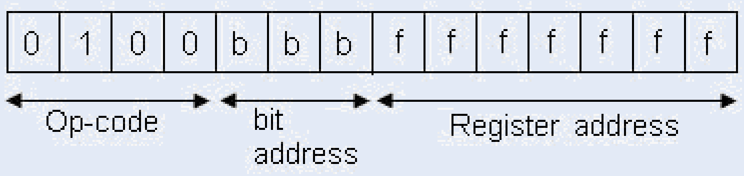
Example-1:

bcf f, b Clear ‘b' bit of register ‘f'

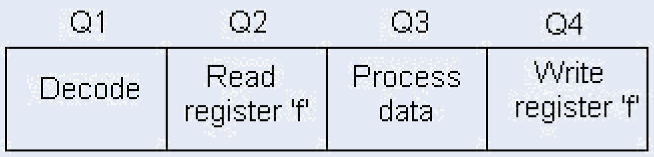
Operands: $0 \le f \le 127$

$0 \le b \le 7$

Encoding:



The instruction is executed in one instruction cycle, i.e., 4 clock cycles. The activities in various clock cycles are as follows.



Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write register 'f'

Example-2:

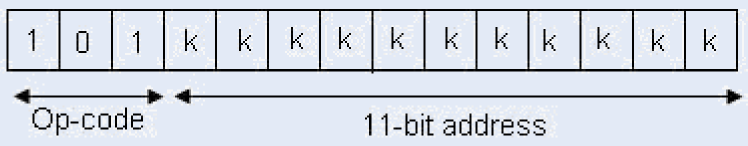
goto K Go to label ‘k' instruction

Operand: $0 \le K \le 2047$ (11-bit address is specified)

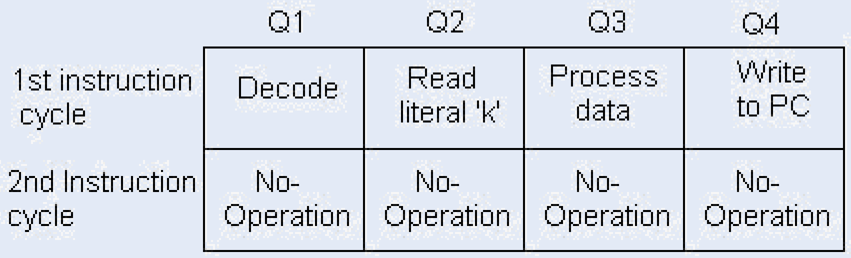
Operation: $K \longrightarrow PC <10:0>$

$PCLATH <4:3> \longrightarrow PC <12:11>$

Encoding:



Since this instruction requires modification of program Counter, it takes two instruction cycles for execution. Q-Cycle activities are shown as follows.



	Q1	Q2	Q3	Q4
1st instruction cycle	Decode	Read literal 'k'	Process data	Write to PC
2nd Instruction cycle	No-Operation	No-Operation	No-Operation	No-Operation

Experiment on PIC16F877A Microcontroller

Objectives:

a) Familiarisation with MPLAB IDE and PIC Microcontroller Programming
b) To light up an LED using PIC16F877A

Equipment:

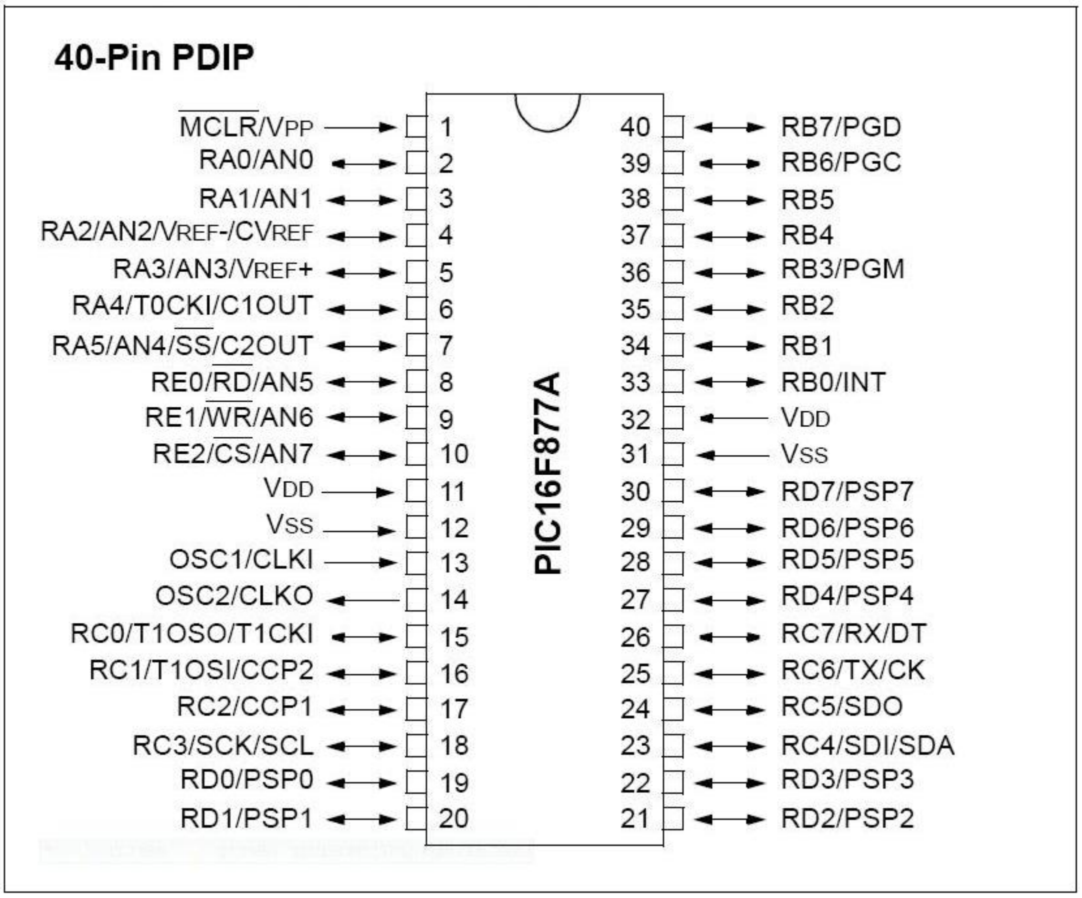
1) PIC16F877A IC
2) PIC Programmer
3) +5V DC Supply
4) 2 Capacitors: 22 pF & a 8 MHz Crystal
5) A 10 kOhm resistor for $\overline{MCLR}$ Pin
6) LED's (each with a 220 Ohm resistor)
7) Connecting Wires

PART A) Familiarisation with MPLAB IDE PIC Microcontroller Programming

Go through the tutorial of MPLAB IDE and PIC Programmer. Get familiarized with various features of MPLAB.

We use the MPLAB IDE Software for designing, editing & compiling codes. Look out for the MPLAB Tutorial for details, on how to begin with a new project & end up with a .hex file of the required code (after simulator testing.)

Read the PICPgm Tutorial to know, how to feed the code (.hex file) into the microcontroller.
Feed the Test File into the MicroController to understand the process.



As you can see in the PIN-Diagram of PIC16F877A, it has 5 Ports that can be used for Input/Output.
Port A : RA0 – RA5 (6 Bit)
Port B : RB0 – RB7 (8 Bit)
Port C : RC0 – RC7 (8 Bit)
Port D : RD0 – RD7 (8 Bit)
Port E : RE0 – RE2 (3 Bit)

Try to locate the pin nos. of these Ports, by scrutinizing the Pin Diagram.

PART B) To Light up an LED

Problem Statement:
Connect an LED to the RB0 of PORTB (Pin No. 33 of the IC), and write a code that will light up the given LED.

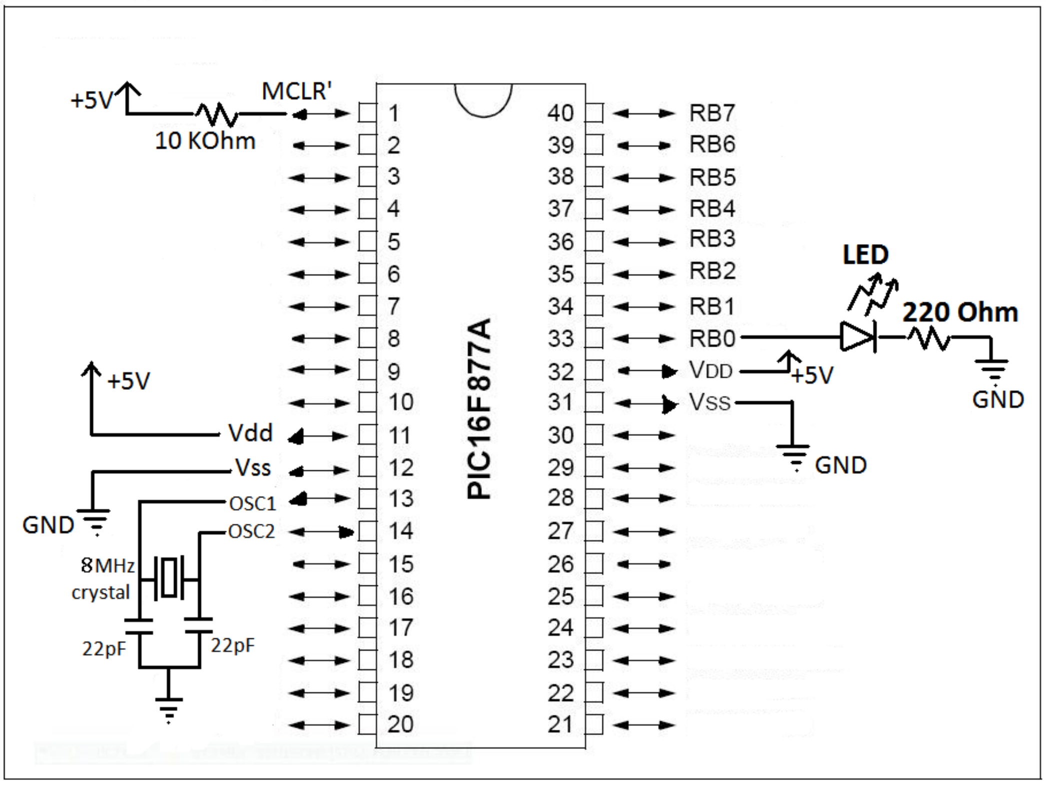
Hints for writing the program:
Write codes sequentially for each of the following

a) Select the Register Bank that contains the TRISB Register
b) Clear (equate to 0) the 0th Bit of TRISB Register (to configure RB0 as an Output Pin)
c) Select the Register Bank that contains the PORTB Register
d) Set (equate to 1) the 0th Bit of PORTB Register, so that RB0 outputs a High voltage and lights up the LED.

Compile the code, test it using the MPLAB Simulator, and feed the .hex file into the IC.

Circuit Diagram:

Connect the IC on the Bread-Board along with the circuit as shown.



PART C Blinking of an LED

Problem Statement
To blink an LED connected to PORTB pin RB0, continuously.

Procedure:
Continuously set and reset the bit RB0 (say) in an infinite loop with some delay between setting and resetting of RB0.