UNIT 1

1. INTRODUCTION TO MICROCONTROLLERS AND EMBEDDED SYSTEMS

Microcontrollers have only been with us for a few decades but their impact (direct or indirect) on our lives is profound. Usually these are supposed to be just data processors performing exhaustive numeric operations. But their presence is unnoticed at most of the places like

- At supermarkets in Cash Registers, Weighing Scales, etc.
- At home in Ovens, Washing Machines, Alarm Clocks, etc.
- At play in Toys, VCRs, Stereo Equipment, etc.
- At office in Typewriters, Photocopiers, Elevators, etc.
- In industry in Industrial Automation, safety systems, etc.
- On roads in Cars, Traffic Signals, etc.

What inside them makes these machines smart? The answer is microcontroller. Creating applications for the microcontrollers is different than any other development job in electronics and computing. Before selecting a particular device for an application, it is important to understand what the different options and features are and what they can mean with regard to developing the application. The microcontroller incorporates all the features that are found in microprocessor. The microcontroller has built in ROM, RAM, Input Output ports, Serial Port, timers, interrupts and clock circuit. A microcontroller is an entire computer manufactured on a single chip. Microcontrollers are usually dedicated devices embedded within an application. For example, microcontrollers are used as engine controllers in automobiles and as exposure and focus controllers in cameras. In order to serve these applications, they have a high concentration of on-chip facilities such as serial ports, parallel input output ports, timers, counters, interrupt control, analog-to-digital converters, random access memory, read only memory, etc. The I/O, memory, and on-chip peripherals of a microcontroller are selected depending on the specifics of the target application. Since microcontrollers are powerful digital processors, the degree of control and programmability they provide significantly enhances the effectiveness of the application. The 8051 is the first microcontroller of the MCS-51 family introduced by Intel Corporation at the end of the 1970s. The 8051 family with its many enhanced members enjoys the largest market share, estimated to be about 40%, among the various microcontroller architectures.

1.1 Microcontrollers

- Microcontroller (MC) may be called computer on chip since it has basic features of microprocessor with internal ROM, RAM, Parallel and serial ports within single chip. Or we can say microprocessor with memory and ports is called as microcontroller. This is widely used in washing machines, vcd player, microwave oven, and robotics or in industries.
- Microcontroller can be classified on the basis of their bits processed like 8bit MC, 16bit MC.
- 8 bit microcontroller means it can read, write and process 8 bit data. Ex. 8051 microcontroller. Basically 8 bit specifies the size of data bus. 8 bit microcontroller means 8 bit data can travel on the data bus or we can read, write process 8 bit data.

1.2 Embedded Controller

Simply an embedded controller is a controller that is embedded in a greater system. One can define an embedded controller as a controller (or computer) that is embedded into some device for some purpose other than to provide general purpose computing. Is an embedded controller is the same as a microcontroller? The answer is definitely no. One can state devices such as 68000, 32032, x86, Z80, and so on that are used as embedded controllers but they aren’t microcontrollers.

We might be correct by stating that an embedded controller controls something (for example controlling a device such as a microwave oven, car braking system or a cruise missile). An embedded controller may also embed the on-chip resources like a microcontroller. Microcontrollers and microprocessors are widely used in embedded systems. Though, microcontrollers are preferred over microprocessors for embedded systems due to low power consumption.

1.2.1.1 Criteria for Selection of a Microcontroller in Embedded System

Criteria for selection of microcontroller in any embedded system are as following:

a) Meeting the computing needs of task at hand efficiently and cost effectively
   - Speed of operation
   - Packing
   - Power consumption
   - Amount of RAM and ROM on chip
   - No. of I/O pins and timers on chip
   - Cost

b) Availability of software development tools such as compiler, assembler and debugger.

1.3 EMBEDDED AND EXTERNAL MEMORY MICROCONTROLLERS

Embedded Microcontrollers

When an embedded system has a microcontroller unit that has all the functional blocks (including program as well as data memory) available on a chip is called an embedded microcontroller. For example, 8051 having Program & Data Memory, I/O Ports, Serial Communication, Counters and Timers and Interrupt Control logic on the chip is an embedded microcontroller.

External Memory Microcontrollers

When an embedded system has a microcontroller unit that has not all the functional blocks available on a chip is called an external memory microcontroller. In external memory microcontroller, all or part of the memory units are externally interfaced using an interfacing circuit called the glue circuit. For example, 8031 has no program memory on the chip is an external memory microcontroller.

1.4 MICROCONTROLLERS AND MICROPROCESSORS

A controller is used to control some process. At one time, controllers were built exclusively from logic components, and were usually large, heavy boxes. Later on, microprocessors were used and the entire controller could fit on a small circuit board. This is still common– one can find many controllers powered by one of the many common microprocessors (including Zilog Z80, Intel 8088, Motorola 6809, and others).

As the process of miniaturization continued, all of the components needed for a controller were built right onto one chip. A one chip computer or microcontroller was born. A CPU built into a single VLSI chip is called microprocessor. The simplified block diagram of the CPU is shown in the Fig. 2. It contains arithmetic and logic unit (ALU), Instruction decodes and control unit, Instruction register, Program counter (PC), clock circuit (internal or external), reset circuit (internal or external) and registers. For example, Intel 8085 is 8-bit microprocessor and Intel 8086/8088 is 16-bit microprocessor. Microprocessor is general-purpose digital computer central processing unit (CPU). The microprocessor is general-purpose device and additional external circuitry is added to make it microcomputer.

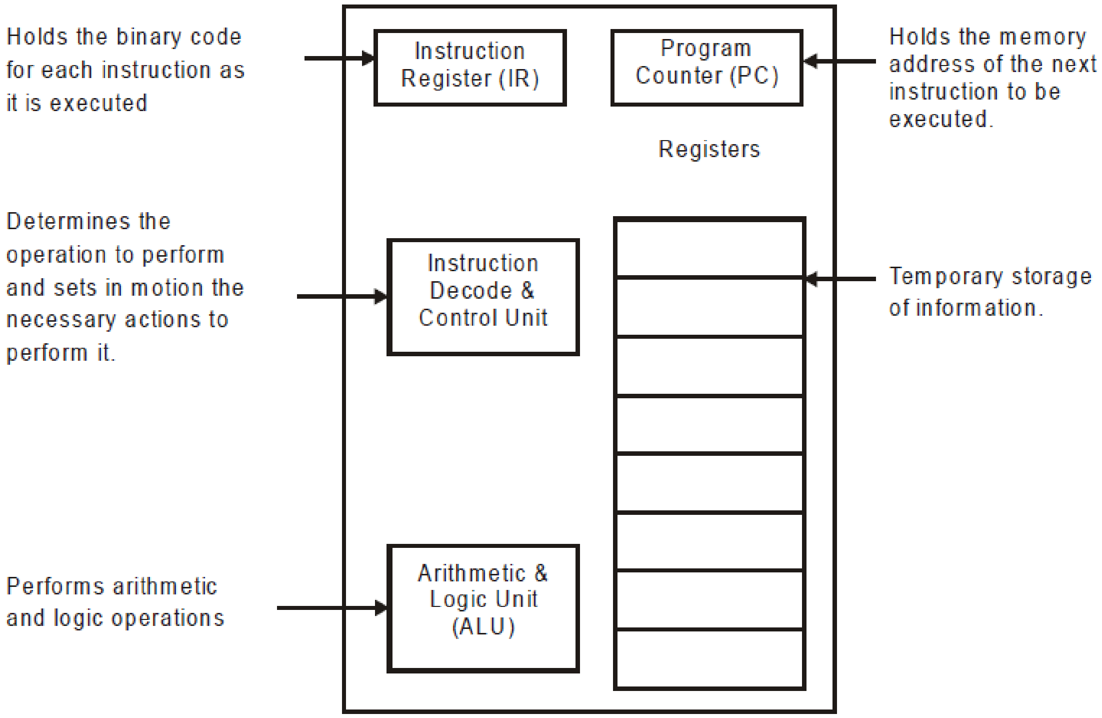

Holds the binary code for each instruction as it is executed
Instruction Register (IR)
Program Counter (PC)
Holds the memory address of the next instruction to be executed.
Registers
Determines the operation to perform and sets in motion the necessary actions to perform it.
Instruction Decode & Control Unit
Temporary storage of information.
Performs arithmetic and logic operations
Arithmetic & Logic Unit (ALU)

Figure 1.1 General block diagram of CPU (Microprocessor)

A digital computer having microprocessor as the CPU along with I/O devices and memory is known as microcomputer. The block diagram in the Fig. 3 shows a microcomputer.

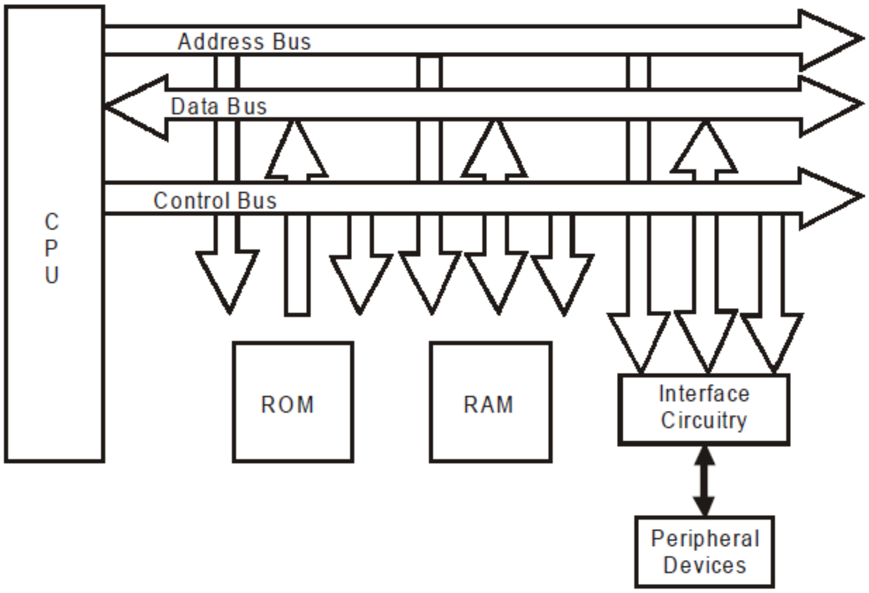


Figure 1.2 Microcomputer block diagram

A microcontroller is a highly integrated chip, which includes on single chip, all or most of the parts needed for a controller. The microcontroller typically includes: CPU (Central Processing Unit), RAM (Random Access Memory), EPROM/PROM/ROM (Erasable Programmable Read Only Memory), I/O (input/output) – serial and parallel, timers, interrupt controller. For example, Intel 8051 is 8-bit microcontroller and Intel 8096 is 16-bit microcontroller.

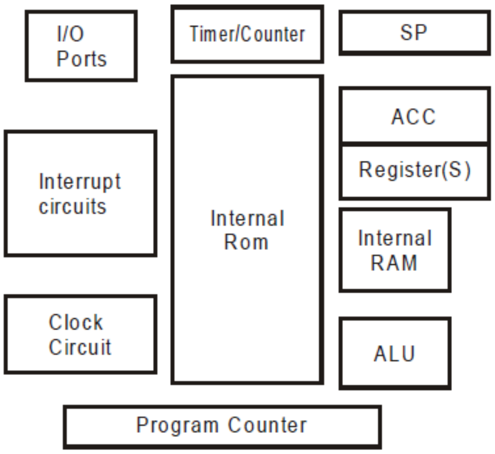
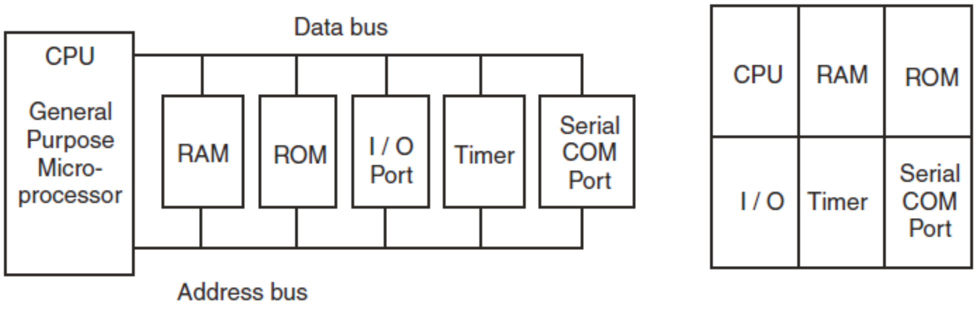


Figure 1.3 A block diagram of a microcontroller

By only including the features specific to the task (control), cost is relatively low. A typical microcontroller has bit manipulation instructions, easy and direct access to I/O (input/output), and quick and efficient interrupt processing. Figure 1.3 shows the block diagram of a typical microcontroller.

1.3.1 Comparing Microcontroller and Microprocessor



(a) General-purpose Microprocessor System

(b) Microcontroller

Fig. 1.4 Structure of microprocessor and microcontroller

It is very clear from figure that in microprocessor we have to interface additional circuitry for providing the function of memory and ports, for example we have to interface external RAM for data storage, ROM for program storage, programmable peripheral interface (PPI) 8255 for the Input Output ports, 8253 for timers, USART for serial port. While in the microcontroller RAM, ROM, I/O ports, timers and serial communication ports are in built. Because of this it is called as “system on chip”. So in micro-controller there is no necessity of additional circuitry which is interfaced in the microprocessor because memory and input output ports are inbuilt in the microcontroller. Microcontroller gives the satisfactory performance for small applications. But for large applications the memory requirement is limited because only 64 KB memory is available for program storage. So for large applications we prefer microprocessor than microcontroller due to its high processing speed.

- Microprocessor is a single chip CPU, microcontroller contains, a CPU and much of the remaining circuitry of a complete microcomputer system in a single chip.
- Microcontroller includes RAM, ROM, serial and parallel interface, timer, interrupt schedule circuitry (in addition to CPU) in a single chip.
    - ➢ RAM is smaller than that of even an ordinary microcomputer, but enough for its applications.
    - ➢ Interrupt system is an important feature, as microcontrollers have to respond to control oriented devices in real time. E.g., opening of microwave oven’s door cause an interrupt to stop the operation. (Most microprocessors can also implement powerful interrupt schemes, but external components are usually needed).
- Microprocessors are most commonly used as the CPU in microcomputer systems. Microcontrollers are used in small, minimum component designs performing control-oriented activities.
- Microprocessor instruction sets are “processing intensive”, implying powerful addressing modes with instructions catering to large volumes of data. Their instructions operate on nibbles, bytes, etc. Microcontrollers have instruction sets catering to the control of inputs and outputs. Their instructions operate also on a single bit. E.g., a motor may be turned ON and OFF by a 1-bit output port.

Before going in to details of microcontrollers it will be beneficial to go through common and frequently used terminology encountered in the description of microcontrollers.

A. Central Processing Unit (CPU)

CPU is the brain of the computer system, administers all activity in the system and performs all operations on data. It continuously performs two operations: fetching and executing instructions. It understands and executes instructions based on a set of binary codes called the instruction set.

Machine Cycle

To execute an instruction–the processor must:

1. Fetch the instruction from memory
2. Decode the instruction
3. Execute the instruction
4. Store the result back in the memory. These four steps refer to Machine Cycle.

Generally one machine cycle = X clock cycles (_X_ depends on the particular instruction being executed). Shorter the clock cycle, lesser the time it takes to complete one machine cycle, so instructions are executed faster hence, faster the processor.

B. Fetching And Executing An Instruction

Fetching involves the following steps:

(a) Contents of PC are placed on address bus.

(b) READ signal is activated.

(c) Data (instruction opcode) are read from RAM and placed on data bus.

(d) Opcode is latched into the CPU's internal instruction register.

(e) PC is incremented to prepare for the next fetch from memory.

While execution involves decoding the opcode and generating control signals to gate internal registers in and out of the ALU and to signal the ALU to perform the specified operation.

C. The Buses: Address, Data, And Control

A BUS is a collection of wires carrying information with a common purpose. For each read or write operation, the CPU specifies the location of the data or instruction by placing an address on the address bus, then activates a signal on the control bus indicating whether the operation is read or write.

- READ OPERATIONS retrieve a byte of data from memory at the location specified and place it on the data bus. CPU reads the data and places it in one of its internal registers.
- WRITE OPERATIONS put data from CPU on the data bus and store it in the location specified.

Address Bus carries the address of a specified location. For n address lines, 2n locations can be accessed. E.g., A 16-bit address bus can access 216 = 65,536 locations or 64K locations (210 = 1024 = 1K, 26 = 64).

Data Bus carries information between the CPU and memory or between the CPU and I/O devices.

Control Bus carries control signals supplied by the CPU to synchronize the movement of information on the address and data bus.

D. Control/Monitor (Input/output) Devices

Control Devices are outputs, or actuators, that can affect the world around them when supplied with a voltage or current.

Monitoring Devices are inputs, or sensors, that are stimulated by temperature, pressure, light, motion, etc. and convert this to voltage or current read by the computer. Note: The interface circuitry converts the voltage or current to binary data, or vice versa.

2. TYPES OF MICROCONTROLLERS

Microcontrollers can be classified on the basis of internal bus width, architecture, memory and instruction set. Figure 1.4 shows the various types of microcontrollers.

2.1 The 8, 16 and 32-Bit Microcontrollers

THE 8-BIT MICROCONTROLLER

When the ALU performs arithmetic and logical operations on a byte (8-bits) at an instruction, the microcontroller is an 8-bit microcontroller. The internal bus width of 8-bit microcontroller is of 8-bit. Examples of 8-bit microcontrollers are Intel 8051 family and Motorola MC68HC11 family.

THE 16-BIT MICROCONTROLLER

When the ALU performs arithmetic and logical operations on a word (16-bits) at an instruction, the microcontroller is an 16-bit microcontroller. The internal bus width of 16-bit microcontroller is of 16-bit. Examples of 16-bit microcontrollers are Intel 8096 family and Motorola MC68HC12 and MC68332 families. The performance and computing capability of 16 bit microcontrollers are enhanced with greater precision as compared to the 8-bit microcontrollers.

THE 32-BIT MICROCONTROLLER

When the ALU performs arithmetic and logical operations on a double word (32- bits) at an instruction, the microcontroller is an 32-bit microcontroller. The internal bus width of 32-bit microcontroller is of 32-bit. Examples of 32-bit microcontrollers are Intel 80960 family and Motorola M683xx and Intel/Atmel 251 family. The performance and computing capability of 32 bit microcontrollers are enhanced with greater precision as compared to the 16-bit microcontroller

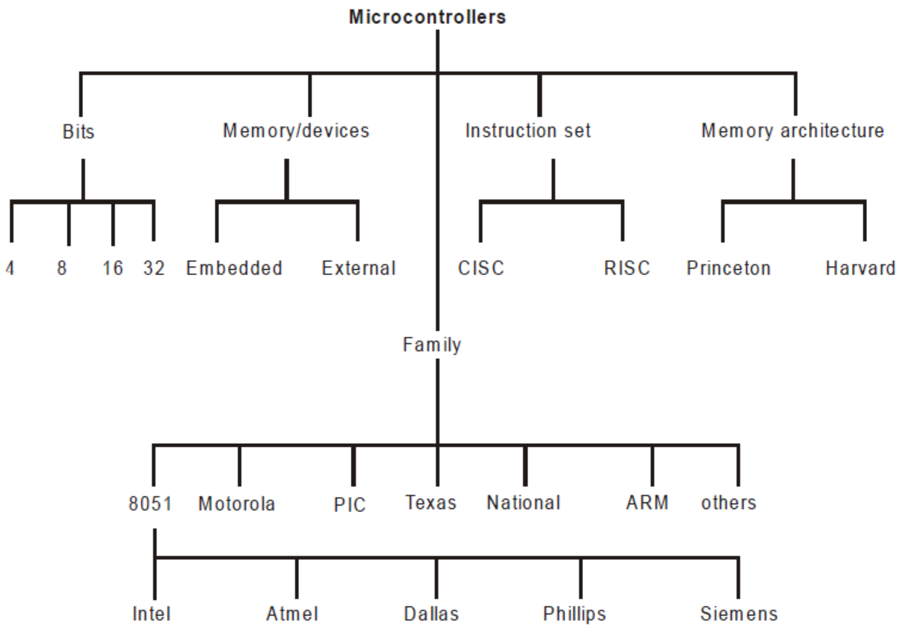


Figure 1.5 Types of microcontrollers

3. MICROCONTROLLER 8051 ARCHITECTURE

It is 8-bit microcontroller, means MC 8051 can Read, Write and Process 8 bit data. This is mostly used microcontroller in the robotics, home appliances like mp3 player, washing machines, electronic iron and industries. Mostly used blocks in the architecture of 8051 are as follows:

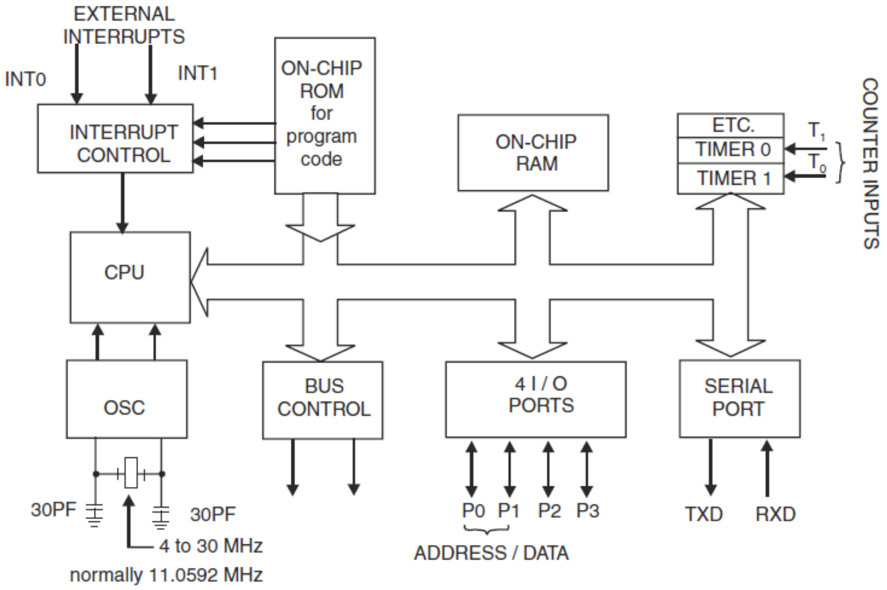
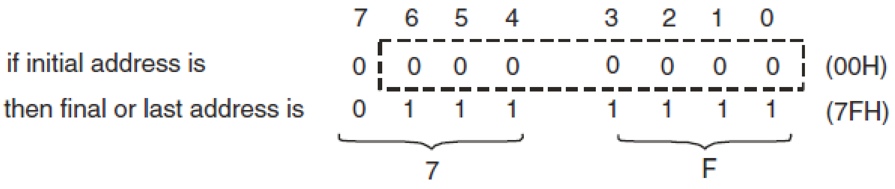


Fig. 1.6 8051 architecture

3.1 128 Byte RAM for Data Storage

MC 8051 has 128 byte Random Access memory for data storage. Random access memory is non volatile memory. During execution for storing the data the RAM is used. RAM consists of the register banks, stack for temporary data storage. It also consists of some special function register (SFR) which are used for some specific purpose like timer, input output ports etc. Normally microcontroller has 256 byte RAM in which 128 byte is used for user space which is normally Register banks and stack. But other 128 byte RAM which consists of SFRs. We will discuss the RAM in detail in next section. Now what is the meaning of 128 byte RAM. What are address range which is provided for data storage. We will discuss here. We know that 128 byte = 27 byte



Since 27 bytes so last 7 bits can be changed so total locations are from 00H to 7F H. This procedure of calculating the memory address is called as ***"memory mapping"*.** We can save data on memory locations from 00H to 7FH. Means total 128 byte space from 00H to 7FH is provided for data storage.

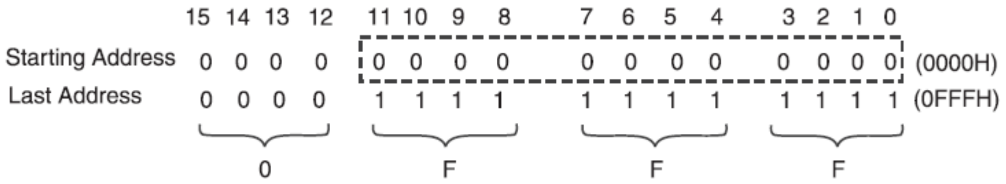
3.2 4KB ROM

In 8051, 4KB read only memory (ROM) is available for program storage. This is used for permanent data storage. Or the data which is not changed during the processing like the program or algorithm for specific applications.

- This is volatile memory; the data saved in this memory does not disappear after power failure.
- We can interface up to 64KB ROM memory externally if the application is large. These sizes are specified different by their companies.
- **Address Range of PC:** Address range of PC means program counter (which points the next instruction to be executing) can be moved between these locations or we can save the program from this location to this location. The address range can be calculated in the same way just like the RAM which is discussed in previous section.

  4KB = 2^2. 2^{10} B (since 1KB = 210 B)

  = 212 Byte


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Starting Address 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0000H)
Last Address 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 (0FFFH)
0
F
F
F

Address range of PC is 0000H to 0FFFH means total 4KB locations are available from 0000H to 0FFFH. At which we can save the program.

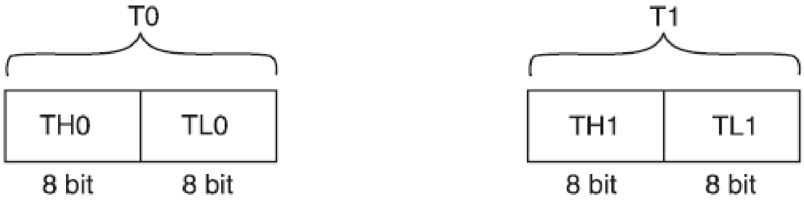
Difference between RAM and ROM

- RAM is used for data storage while ROM is used for program storage.
- Data of RAM can be changed during processing while data of ROM can’t be changed during processing.
- We can take an example of calculator. If we want to perform addition of two numbers then we type the two numbers in calculator, this is saved in the RAM, but the Algorithms by which the calculation is performed is saved in the ROM. Data which is given by us to calculator can be changed but the algorithm or program by which calculation is performed can’t be changed.

3.3 Timers and Counters

Timer means which can give the delay of particular time between some events. For example, on or off the lights after every 2 sec. This delay can be provided through some assembly program but in microcontroller two hardware pins are available for delay generation. These hardware pins can be also used for counting some external events. How much times a number is repeated in the given table is calculated by the counter.

- In MC8051, two timer pins are available T0 and T1, by these timers we can give the delay of particular time if we use these in timer mode.
- We can count external pulses at these pins if we use these pins in counter mode.
- 16 bits timers are available. Means we can generate delay between 0000H to FFFFH.
- Two special function registers are available.



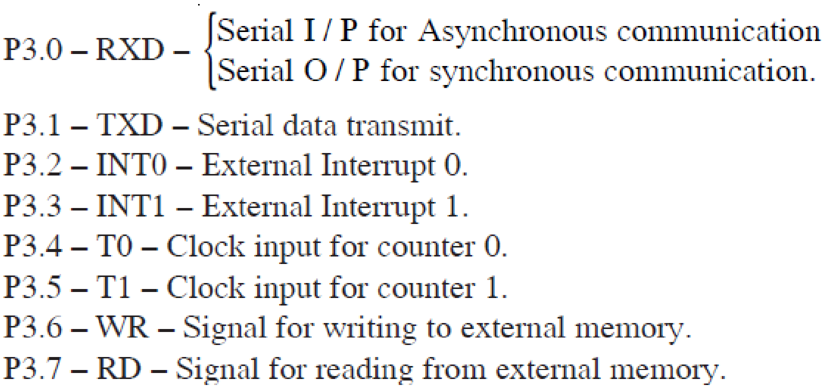
- If we want to load T0 with 16 bit data then we can load separate lower 8 bit in TL0 and higher 8 bit in TH0.
- In the same way for T1. TMOD, TCON registers are used for controlling timer operation.

3.4 Serial Port

- There are two pins available for serial communication TXD and RXD.
- Normally TXD is used for transmitting serial data which is in SBUF register, RXD is used for receiving the serial data.
- SCON register is used for controlling the operation.

3.5 Input Output Ports

- There are four input output ports available P0, P1, P2, P3.
- Each port is 8 bit wide and has special function register P0, P1, P2, P3 which are bit addressable means each bit can be set or reset by the Bit instructions (SETB for high, CLR for low) independently.
- The data at any port which is transmitting or receiving is in these registers.
- The port 0 can perform dual works. It is also used as Lower order address bus (A0 to A7) multiplexed with 8 bit data bus P0.0 to P0.7 is AD0 to AD7 respectively the address bus and data bus is demultiplex by the ALE signal and latch which is further discussed in details.
- Port 2 can be used as I/O port as well as higher order address bus A8 to A15.
- Port 3 also have dual functions it can be worked as I/O as well as each pin of P3 has specific function.



P3.0 – RXD – {Serial I / P for Asynchronous communication
Serial O / P for synchronous communication.

P3.1 – TXD – Serial data transmit.
P3.2 – INT0 – External Interrupt 0.
P3.3 – INT1 – External Interrupt 1.
P3.4 – T0 – Clock input for counter 0.
P3.5 – T1 – Clock input for counter 1.
P3.6 – WR – Signal for writing to external memory.
P3.7 – RD – Signal for reading from external memory.

When external memory is interfaced with 8051 then P0 and P2 can't be worked as I/O port they works as address bus and data bus, otherwise they can be accessed as I/O ports.

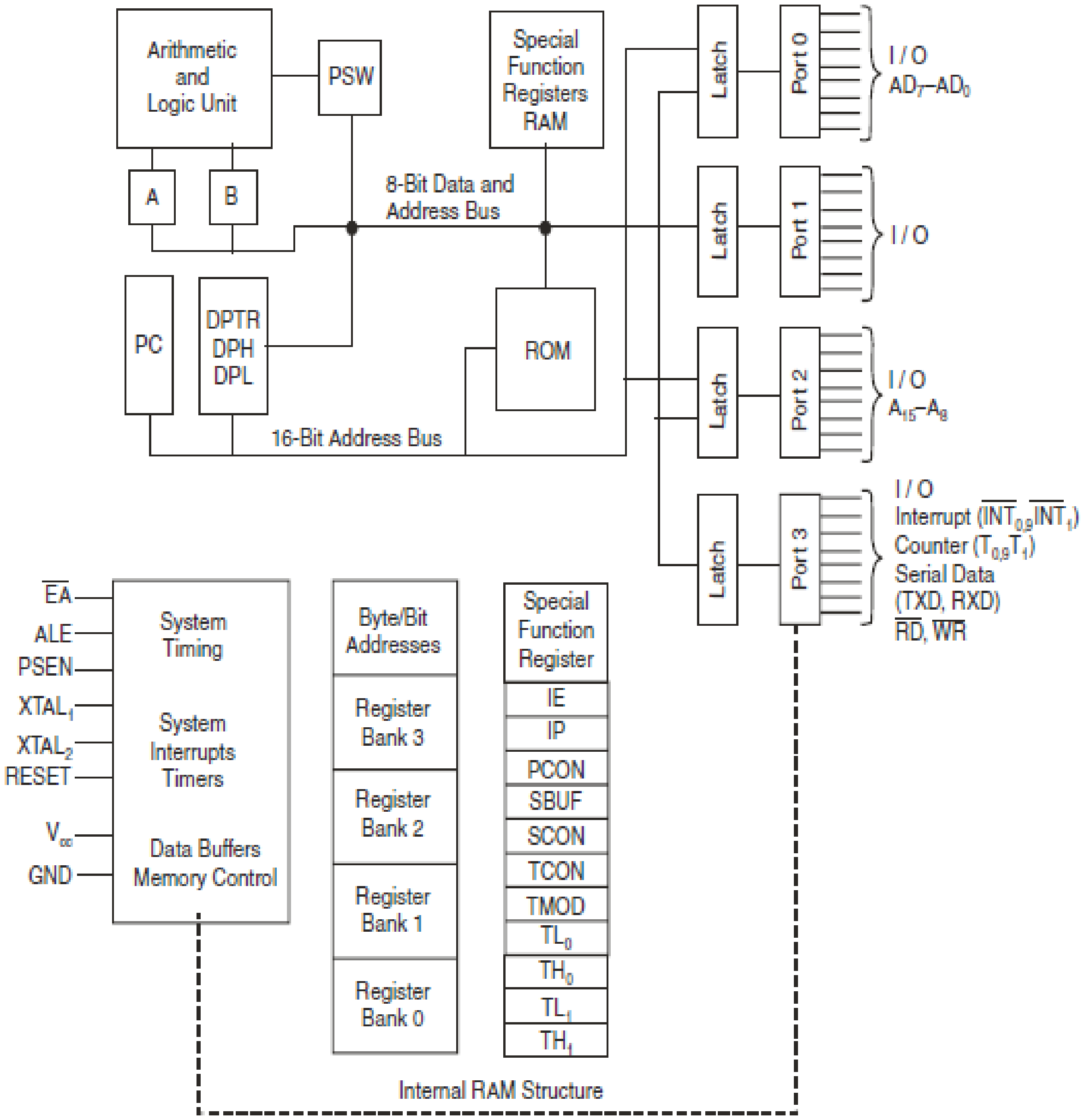
3.6 Oscillator

- It is used for providing the clock to MC8051 which decides the speed or baud rate of MC.
- We use crystal which frequency varies from 4MHz to 30 MHz, normally we use 11.0592 MHz frequency.

3.7 Interrupts

- Interrupts are defined as requests because they can be refused (masked) if they are not used, that is when an interrupt is acknowledged. A special set of events or routines are followed to handle the interrupts. These special routines are known as interrupt handler or interrupt service routines (ISR). These are located at a special location in memory.
- INT0 and INT1 are the pins for external interrupts.

4. ARCHITECTURE OF 8051



Architectural block diagram of microcontroller 8051

Each block will be discussed step by step:

4.1 ALU — Arithmetic Logical Unit

This unit is used for the arithmetic calculations.

4.2 A-Accumulator

This register is used for arithmetic operations. This is also bit addressable and 8 bit register.

4.3 B-Register

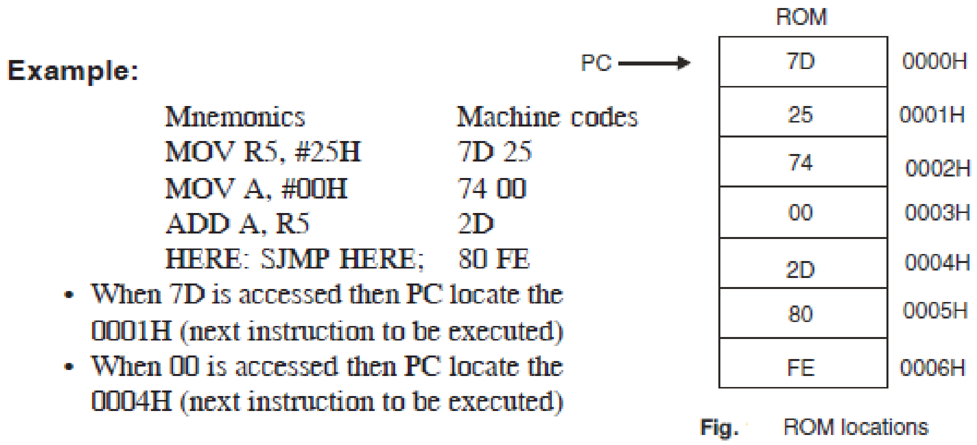
This register is used in only two instructions MUL AB and DIV AB. This is also bit addressable and 8 bit register.

4.4 PC-Program Counter

- Points to the address of next instruction to be executed from ROM
- It is 16 bit register means the 8051 can access program address from 0000H to FFFFH. A total of 64KB of code. 16 bit register means.

Initial value	0000	0000	0000	0000	(0000H)
Final value	1111	1111	1111	1111	(FFFFH)

- Initially PC has 0000H
- ORG instruction is used to initialize the PC ORG 0000H means PC initialize by 0000H
- PC is incremented after each instruction.



Example:

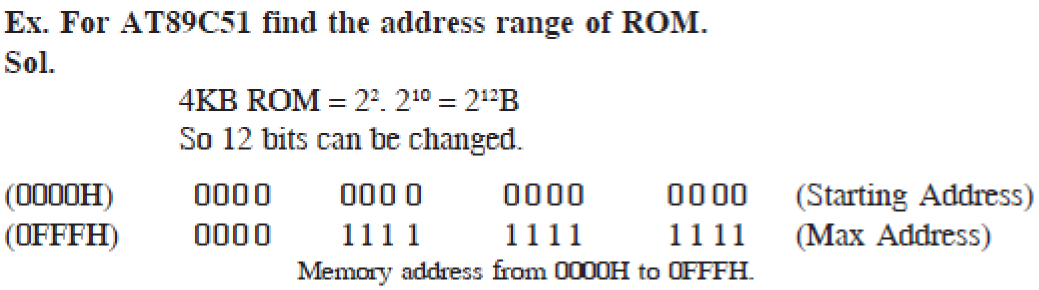
Mnemonics	Machine codes
MOV R5, #25H	7D 25
MOV A, #00H	74 00
ADD A, R5	2D
HERE: SJMP HERE;	80 FE

- When 7D is accessed then PC locate the 0001H (next instruction to be executed)
- When 00 is accessed then PC locate the 0004H (next instruction to be executed)

Fig. ROM locations

4.5 ROM Memory Map in 8051

- 4KB, 8KB, 16KB, 32KB, 64KB on chip ROM is available.
- Max ROM space is 64 KB because 16 bit address line is available in 8051.
- Starting address for ROM is 0000H (because PC which points the ROM is 16 bit wide).



Ex. For AT89C51 find the address range of ROM.

Sol.

4KB ROM = $2^2 . 2^{10} = 2^{12}$B

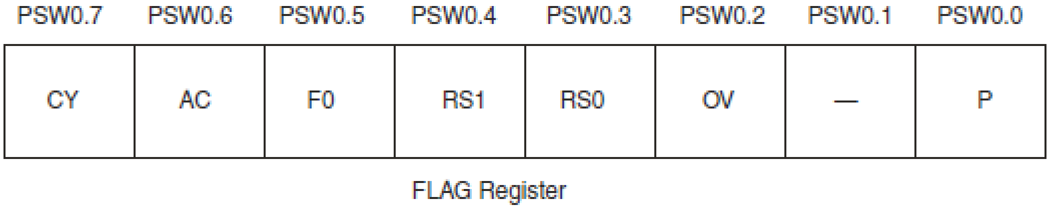
So 12 bits can be changed.

(0000H)	0000	0000	0000	0000	(Starting Address)
(0FFFH)	0000	1111	1111	1111	(Max Address)

Memory address from 0000H to 0FFFH.

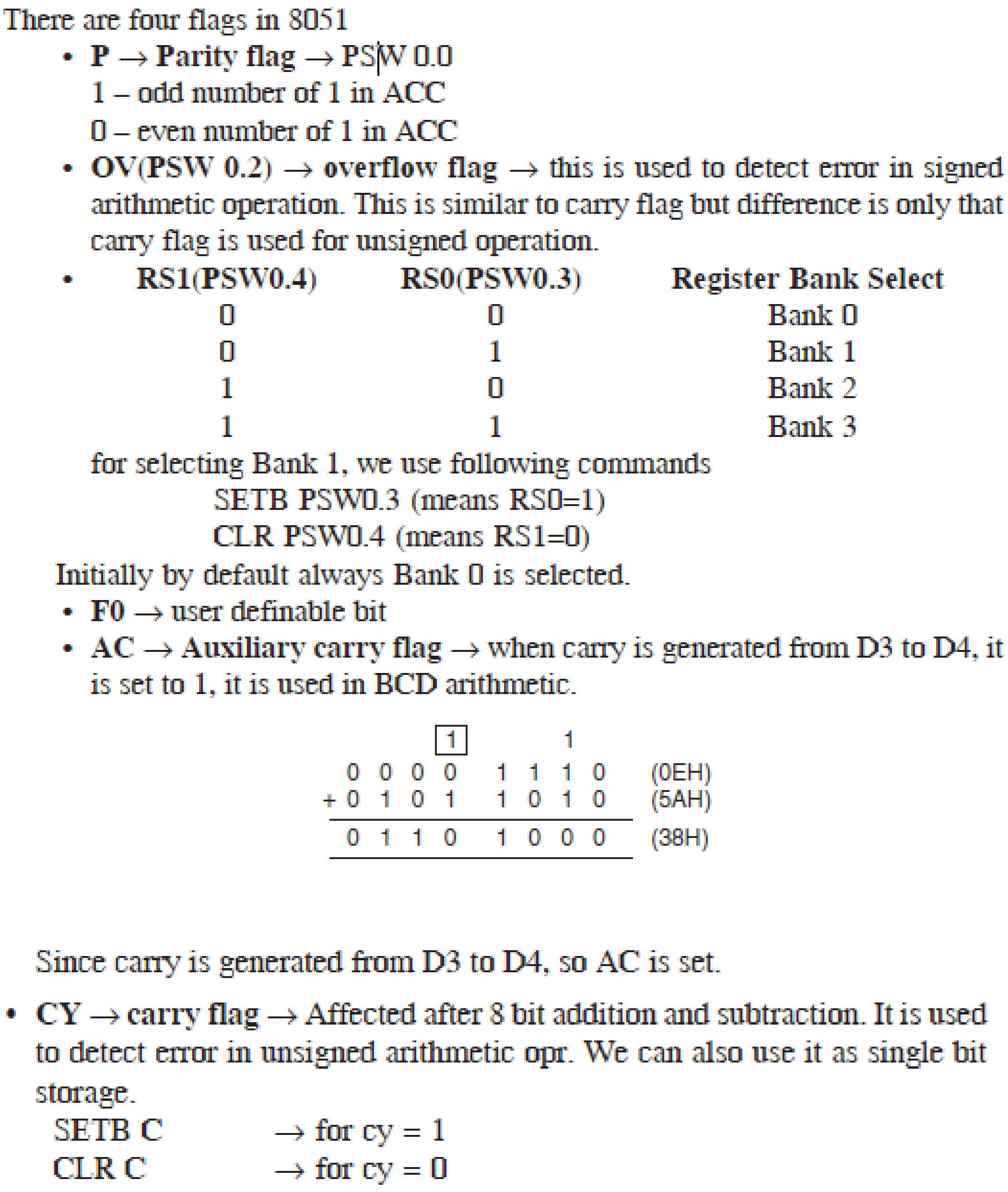
4.6 8051 Flag Bits and PSW Register (Special Function Registers)

- Used to indicate the Arithmetic condition of ACC.
- Flag register in 8051 is called as program status word (PSW). This special function register PSW is also bit addressable and 8 bit wide means each bit can be set or reset independently.



PSW0.7	PSW0.6	PSW0.5	PSW0.4	PSW0.3	PSW0.2	PSW0.1	PSW0.0
CY	AC	F0	RS1	RS0	OV	—	P

FLAG Register



There are four flags in 8051

- **P → Parity flag →** PSW 0.0
  1 – odd number of 1 in ACC
  0 – even number of 1 in ACC
- **OV(PSW 0.2) → overflow flag →** this is used to detect error in signed arithmetic operation. This is similar to carry flag but difference is only that carry flag is used for unsigned operation.
-

RS1(PSW0.4)	RS0(PSW0.3)	Register Bank Select
0	0	Bank 0
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

for selecting Bank 1, we use following commands

SETB PSW0.3 (means RS0=1)
CLR PSW0.4 (means RS1=0)

Initially by default always Bank 0 is selected.

- **F0 →** user definable bit
- **AC → Auxiliary carry flag →** when carry is generated from D3 to D4, it is set to 1, it is used in BCD arithmetic.

$$\begin{array}{rcccccccccl} & & & & \boxed{1} & & & 1 & & & \\ & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & & (0EH) \\ + & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & & (5AH) \\ \hline & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & & (38H) \\ \hline \end{array}$$

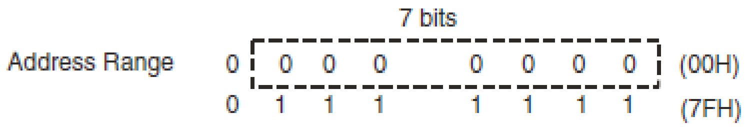
Since carry is generated from D3 to D4, so AC is set.

- **CY → carry flag →** Affected after 8 bit addition and subtraction. It is used to detect error in unsigned arithmetic opr. We can also use it as single bit storage.

SETB C → for cy = 1
CLR C → for cy = 0

4.7 Structure of RAM or 8051 Register Bank and Stack

- 128 byte RAM is available in 8051
- 128 byte = 2^7 B



Address range of RAM is 00H to 7FH.

- In MC8051, 128 byte visible or user accessible RAM is available which is shown in figure. Extra 128B RAM which is not user accessible. 80H to FFH used for storage of SFR (special function register)

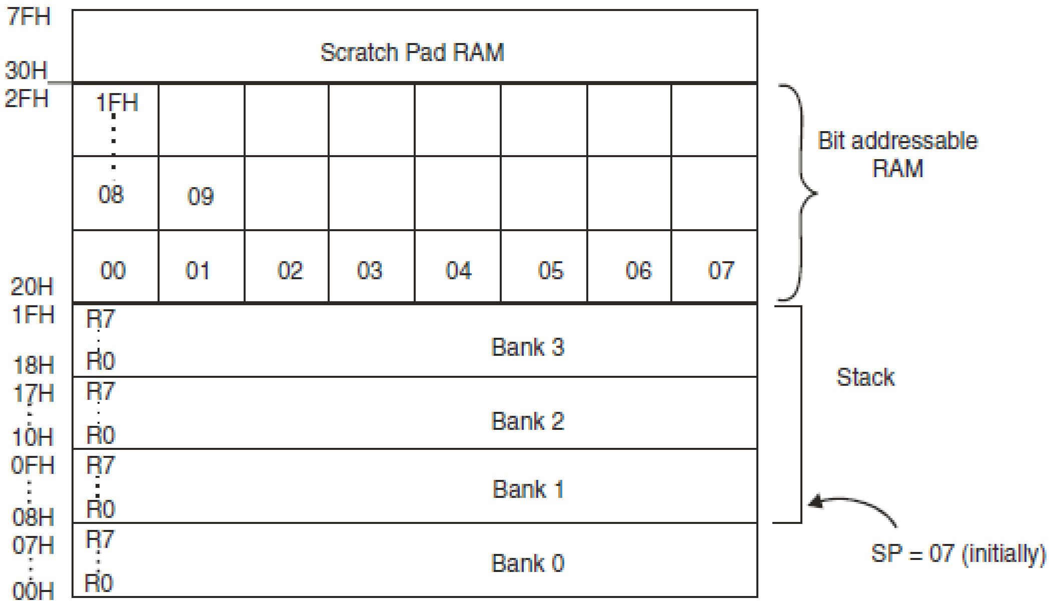
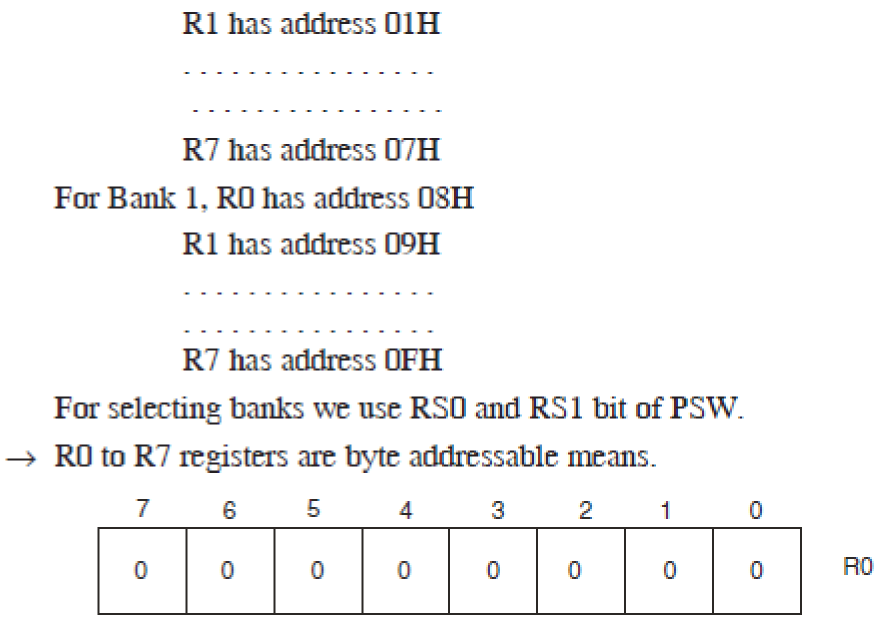


Fig. Structure of RAM

Four Register Banks

- There are four register banks, in each register bank there are eight 8 bit register available from R0 to R7
- By default Bank 0 is selected.

  For Bank 0, R0 has address 00H



R1 has address 01H

. . . .

. . . .

R7 has address 07H

For Bank 1, R0 has address 08H

R1 has address 09H

. . . .

. . . .

R7 has address 0FH

For selecting banks we use RS0 and RS1 bit of PSW.

→ R0 to R7 registers are byte addressable means.

7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	R0

If we want to set the bit 3 of R0 then we can't use SETB R0.3

We use MOV R0, #08H;

For changing single bit we can modify all the other bits of R0.

- Locations 20H to 2FH is bit addressable RAM means each bit from 00H to FFH in this we can set or reset CF rather than changing whole byte.
- Locations 30H to 7FH is used as scratch pad means we can use this space for data reading and writing or for data storage.

4.8 Stack in 8051

✓ RAM locations from 08H to 1FH can be used as stack. Stack is used to store the data temporarily. Stack is last in first out (LIFO)

✓ Stack pointer (SP) →

- 8bit register
- It indicate current RAM address available for stack or it points the top of stack.
- Initially by default at 07H because first location of stack is 08H.
- After each PUSH instruction the SP is incremented by one while in MC after PUSH instruction SP is decremented.
- After each POP instruction the SP is decremented.

Example:

MOV R6,#25H;

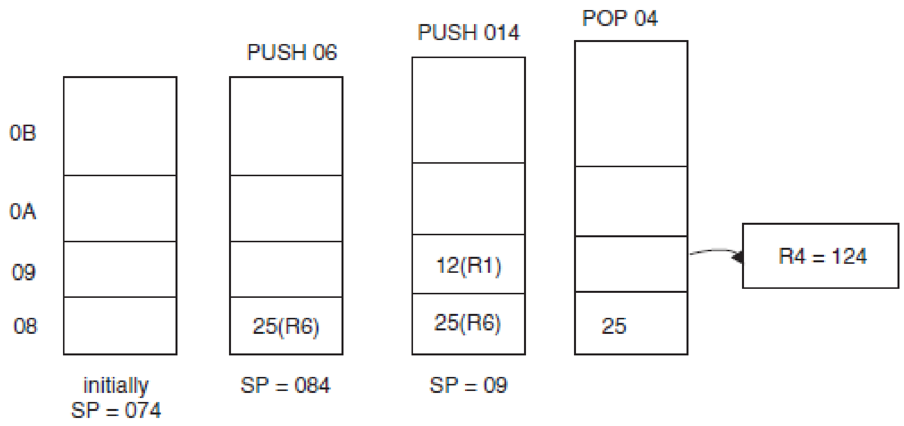
MOV R1,#12H;

MOV R4,#OF3H;

PUSH 06H;

PUSH 01H;

POP 04H;



→ if we want to use more than 24byte (08H to 1FH) of stack. We can change SP to point RAM address 30H to 7FH by MOV SP, #XX

└→ Any value from 30 to 7FH

4.8.1 *Conflicting of Register Banks and Stack*

- We know locations from 08H to 1FH is used as stack and it is also used as register bank.
- If in the program, we use the Register Bank 1 to 3 and also use the stack then conflicts exist and error can be possible.

  For removing this situation we use the stack from location 30H to 7FH by shifting SP to 2FH.

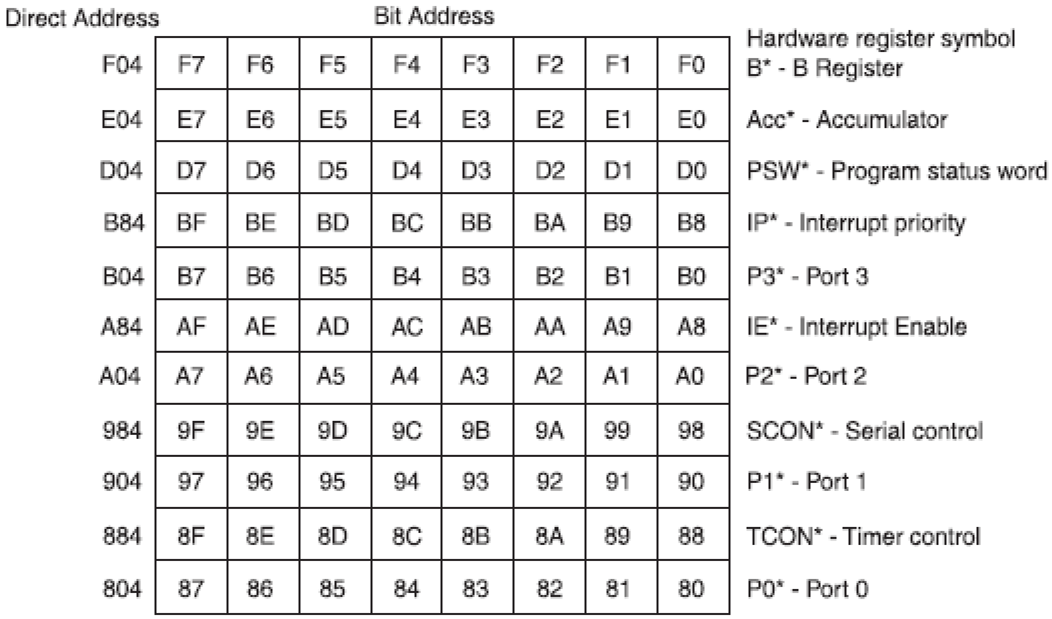
MOV SP,#2FH;

4.9 DPTR → Data Pointer in 8051

- 16 bit register, it is divided into two parts DPH and DPL.
- DPH for Higher order 8 bits, DPL for lower order 8 bits.
- DPTR, DPH, DPL these all are SFRs in 8051.

4.10 Special Function Register

- (See in the following figure) RAM scratch pad, there is extra 128 byte RAM which is used to store the SFRs
- Following figure shows special function bit address, all access to the four I/O ports CPU register, interrupt control register, timer/counter, UART, power control are performed through registers between 80H and FFH.



Direct Address	Bit Address								Hardware register symbol
F04	F7	F6	F5	F4	F3	F2	F1	F0	B* - B Register
E04	E7	E6	E5	E4	E3	E2	E1	E0	Acc* - Accumulator
D04	D7	D6	D5	D4	D3	D2	D1	D0	PSW* - Program status word
B84	BF	BE	BD	BC	BB	BA	B9	B8	IP* - Interrupt priority
B04	B7	B6	B5	B4	B3	B2	B1	B0	P3* - Port 3
A84	AF	AE	AD	AC	AB	AA	A9	A8	IE* - Interrupt Enable
A04	A7	A6	A5	A4	A3	A2	A1	A0	P2* - Port 2
984	9F	9E	9D	9C	9B	9A	99	98	SCON* - Serial control
904	97	96	95	94	93	92	91	90	P1* - Port 1
884	8F	8E	8D	8C	8B	8A	89	88	TCON* - Timer control
804	87	86	85	84	83	82	81	80	P0* - Port 0

Special Function Register

Byte Addressable SFR with byte address

SP:	Stack printer – 81H
DPTR:	Data pointer 2 bytes
DPL:	Low byte – 82H
DPH:	High byte – 83H
TMOD:	Timer mode control – 89H
TH0:	Timer 0 Higher order bytes – 8CH
TL0:	Timer 0 Low order bytes – 8AH
TH1:	Timer 1 High bytes = 80H
TL1:	Timer 1 Low order byte = 86H
SBUF:	Serial data buffer = 99H
PCON:	Power control – 87H.

5. PIN DIAGRAM OF 8051

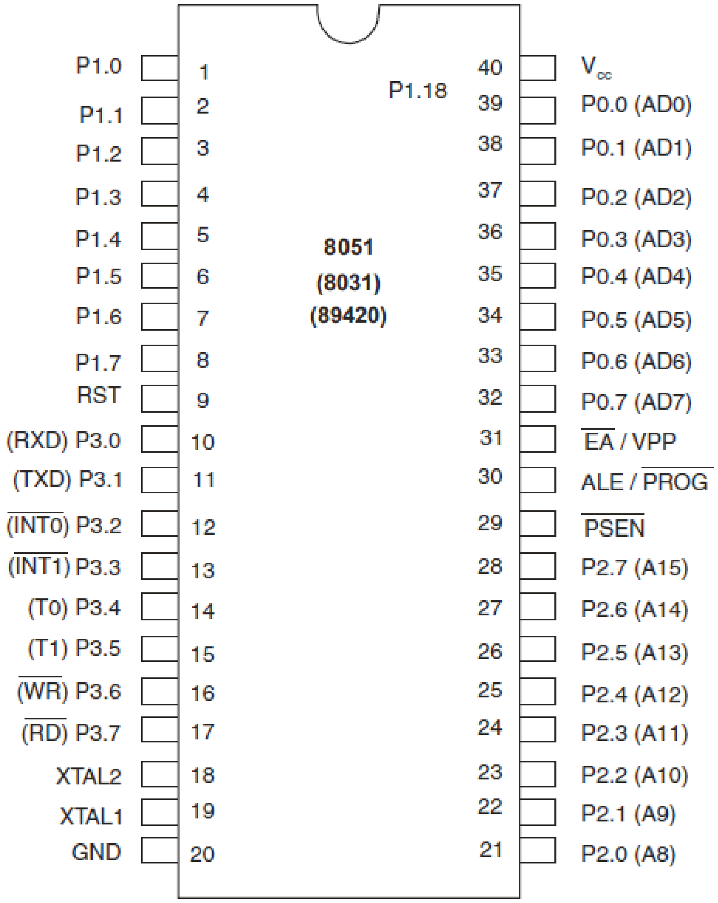


Fig. 1.7 Pin diagram of MC 8051

Description of each pin is discussed here:

- VCC → 5V supply
- VSS → GND
- XTAL2/XTALI are for oscillator input
- Port 0 – 32 to 39 – AD0/AD7 and P0.0 to P0.7
- Port 1 – 1 to 8 – P1.0 to P1.7
- Port 2 – 21 to 28 – P2.0 to P2.7 and A 8 to A15
- Port 3 – 10 to 17 – P3.0 to P3.7
- P 3.0 – RXD – Serial data input – SBUF
- P 3.1 – TXD – Serial data output – SBUF
- P 3.2 – INT0 – External interrupt 0 – TCON 0.1
- P 3.3 – INT1 – External interrupt 1 – TCON 0.3

- P 3.4 – T0 – External timer 0 input – TMOD
- P 3.5 – T1 – External timer 1 input – TMOD
- P 3.6 –WR – External memory write cycle – Active LOW
- P 3.7 – RD – External memory read cycle – Active LOW
- RST – for Restarting 8051
- ALE – Address latch enable

  1 – Address on AD 0 to AD 7

  0 – Data on AD 0 to AD 7
- PSEN – Program store enable

6. MICROCONTROLLER ARCHITECTURAL FEATURES

There are mainly two categories of processors, namely, Von-Neuman (or Princeton) architecture and Harvard Architecture. These two architectures differ in the way data and programs are stored and accessed.

6.1 VON-NEUMAN ARCHITECTURE

Microcontrollers based on the Von-Neuman architecture have a single data bus that is used to fetch both instructions and data. Program instructions and data are stored in a common main memory. When such a controller addresses main memory, it first fetches an instruction, and then it fetches the data to support the instruction. The two separate fetches slows up the controller's operation. Figure 1.7 shows the Von-Neuman Architecture. The Von-Neuman architecture's main advantage is that it simplifies the microcontroller design because only one memory is accessed. In microcontrollers, the contents of RAM can be used for data storage and program instruction storage. For example, the Motorola 68HC11 microcontroller Von- Neuman architecture.

Example: An Instruction "Read a byte from memory and store it in the accumulator" as follows:

Cycle 1:- Read instrution

Cycle 2:- Read data out of RAM and put into Accumulator

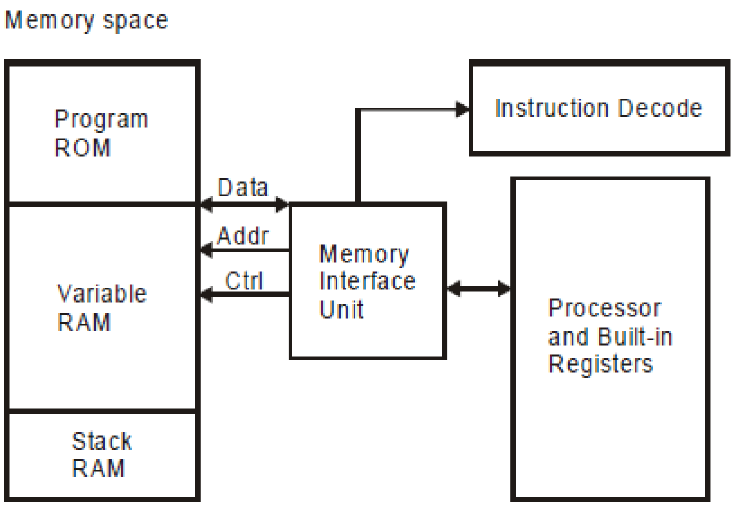


Figure 1.7 Von-neuman architecture block diagram

6.2 HARVARD ARCHITECTURE

Microcontrollers based on the Harvard Architecture have separate data bus and an instruction bus. This allows execution to occur in parallel. As an instruction is being "pre-fetched", the current instruction is executing on the data bus. Once the current instruction is complete, the next instruction is ready to go. This pre-fetch theoretically allows for much faster execution than Von-Neuman architecture, on the expense of complexity. Figure 1.8 shows the Harvard Architecture.The Harvard Architecture executes instructions in fewer instruction cycles than the Von-Neuman architecture. For example,the intel MCS-51 family of microcontrollers and PIC microcontrollers uses Harvard Architecture.

The same instruction (as shown under Von-Newman architecture) would be executed as follows:

Cycle 1: Complete previous instructio

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.

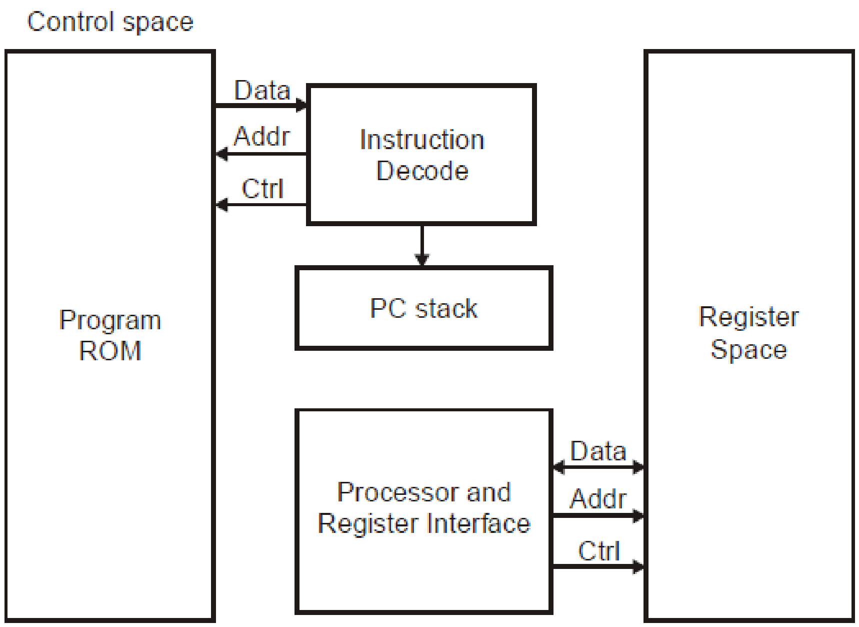


Figure 1.8 Harvard architecture block diagram

6.3 CISC (COMPLEX INSTRUCTION SET COMPUTER) ARCHITECTURE MICROCONTROLLERS

Almost all of today's microcontrollers are based on the CISC (Complex Instruction Set Computer) concept. When a microcontroller has an instruction set that supports many addressing modes for the arithmetic and logical instructions, data transfer and memory accesses instructions, the microcontroller is said to be of CISC architecture. The typical CISC microcontroller has well over 80 instructions, many of them very powerful and very specialized for specific control tasks. It is quite common for the instructions to all behave quite differently. Some might only operate on certain address spaces or registers, and others might only recognize certain addressing modes. The advantages of the CISC architecture are that many of the instructions are macro like, allowing the programmer to use one instruction in place of many simpler instructions.

An example of CISC architecture microcontroller is Intel 8096 family.

6.4 RISC (REDUCED INSTRUCTION SET COMPUTER) ARCHITECTURE MICROCONTROLLERS

The industry trend for microprocessor design is for Reduced Instruction Set Computers (RISC) designs. When a microcontroller has an instruction set that supports fewer addressing modes for the arithmetic and logical instructions and for data transfer instructions, the microcontroller is said to be of RISC architecture.

The benefits of RISC design simplicity are a smaller chip, smaller pin count, and very low power consumption.

Some of the typical features of a RISC processor- Harvard architecture are

a) Allows simultaneous access of program and data.

b) Overlapping of some operations for increased processing performance.

c) Instruction pipelining increases execution speed.

d) Orthogonal (symmetrical) instruction set for programming simplicity.

e) Allows each instruction to operate on any register or use any addressing mode.

6.5 SISC (SPECIFIC INSTRUCTION SET COMPUTER)

Actually, a microcontroller is by definition a Reduced Instruction Set Computer. It could really be called a Specific Instruction Set Computer (SISC). The basic idea behind the microcontroller was to limit the capabilities of the CPU itself, allowing a complete computer (memory, I/O, interrupts, etc) to fit on the single chip. At the expense of the more general purpose instructions that make the standard microprocessors (8088, 68000, 32032) so easy to use, the instruction set was designed for the specific purpose of control (powerful bit manipulation, easy and efficient I/O, and so on).

7. 8051 ADDRESSING MODES

The CPU can access data in various ways, which are called addressing modes:

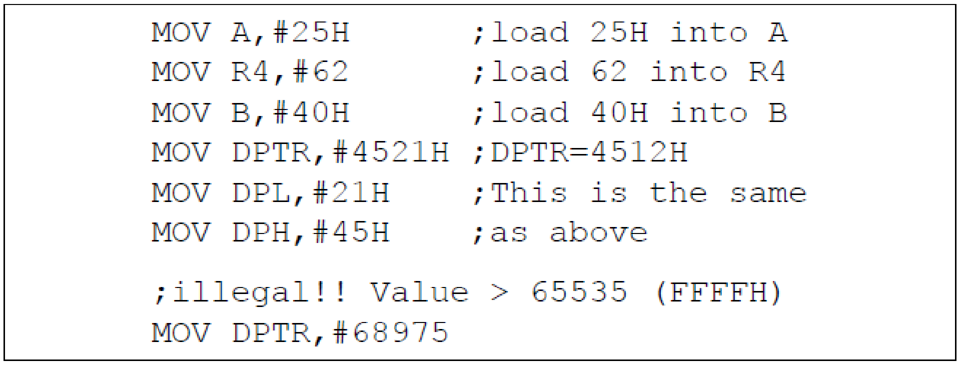
- Immediate
- Register
- Direct
- Register indirect
- Indexed

Accessing Memories

1. Immediate Addressing Mode:

✓ The source operand is a constant

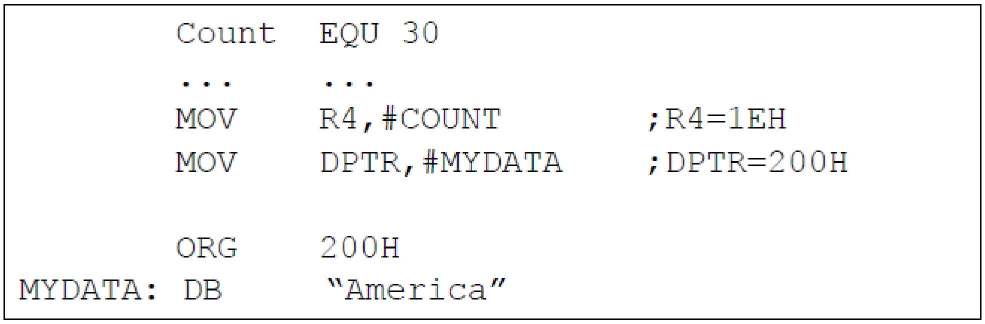
- The immediate data must be preceded by the pound sign, “#”
- Can load information into any registers, including 16-bit DPTR register
  - ➢ DPTR can also be accessed as two 8-bit registers, the high byte DPH and low byte DPL



```
MOV A,#25H        ;load 25H into A
MOV R4,#62        ;load 62 into R4
MOV B,#40H        ;load 40H into B
MOV DPTR,#4521H ;DPTR=4512H
MOV DPL,#21H      ;This is the same
MOV DPH,#45H      ;as above

;illegal!! Value > 65535 (FFFFH)
MOV DPTR,#68975
```

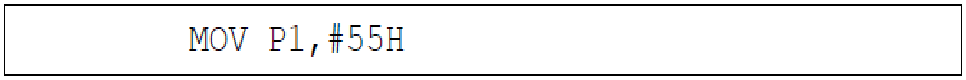
- ✓ We can use EQU directive to access immediate data



```
        Count   EQU 30
        ...     ...
        MOV     R4,#COUNT       ;R4=1EH
        MOV     DPTR,#MYDATA    ;DPTR=200H

        ORG     200H
MYDATA: DB      "America"
```

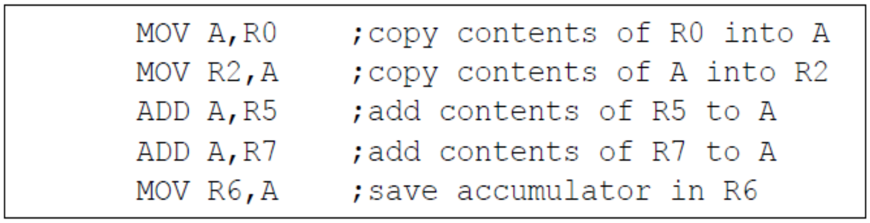
- ✓ We can also use immediate addressing mode to send data to 8051 ports



```
MOV P1,#55H
```

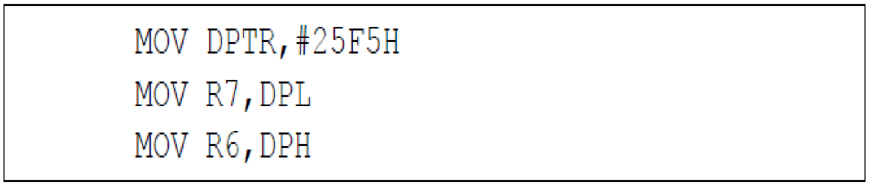
2. Register Addressing Mode

- ✓ Use registers to hold the data to be manipulated



```
MOV A,R0     ;copy contents of R0 into A
MOV R2,A     ;copy contents of A into R2
ADD A,R5     ;add contents of R5 to A
ADD A,R7     ;add contents of R7 to A
MOV R6,A     ;save accumulator in R6
```

- ✓ The source and destination registers must match in size
  - MOV DPTR,A will give an error



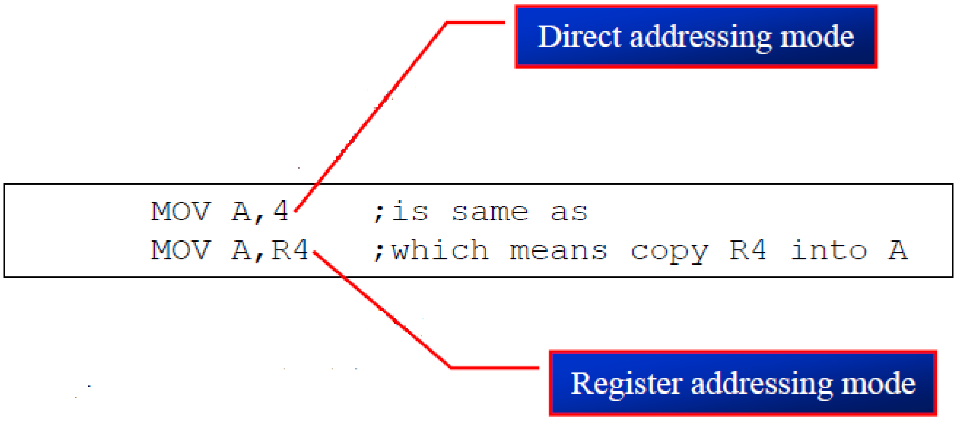
```
MOV DPTR,#25F5H
MOV R7,DPL
MOV R6,DPH
```

- ✓ The movement of data between Rn registers is not allowed
  - MOV R4, R7 is invalid

ACCESSING MEMORY

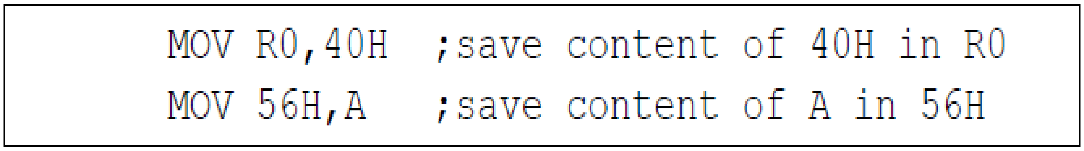
3. Direct Addressing Mode

- ✓ It is most often used the direct addressing mode to access RAM locations 30 – 7FH
  - The entire 128 bytes of RAM can be accessed
  - The register bank locations are accessed by the register names



```
MOV  A,4      ;is same as
MOV  A,R4     ;which means copy R4 into A
```

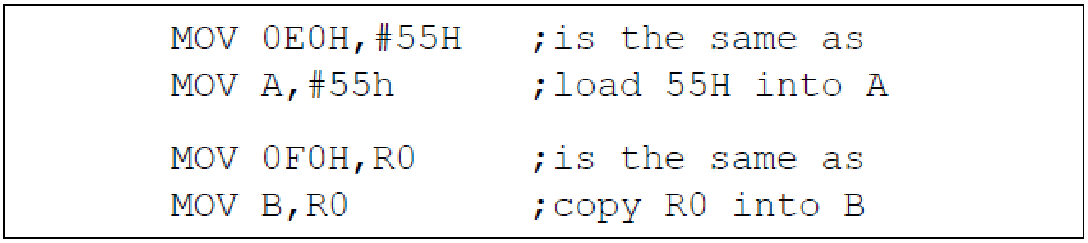
- ✓ Contrast this with immediate addressing mode
  - There is no “#” sign in the operand



```
MOV R0,40H   ;save content of 40H in R0
MOV 56H,A    ;save content of A in 56H
```

4. SFR Registers and Their Addresses

- ✓ The SFR (Special Function Register) can be accessed by their names or by their addresses

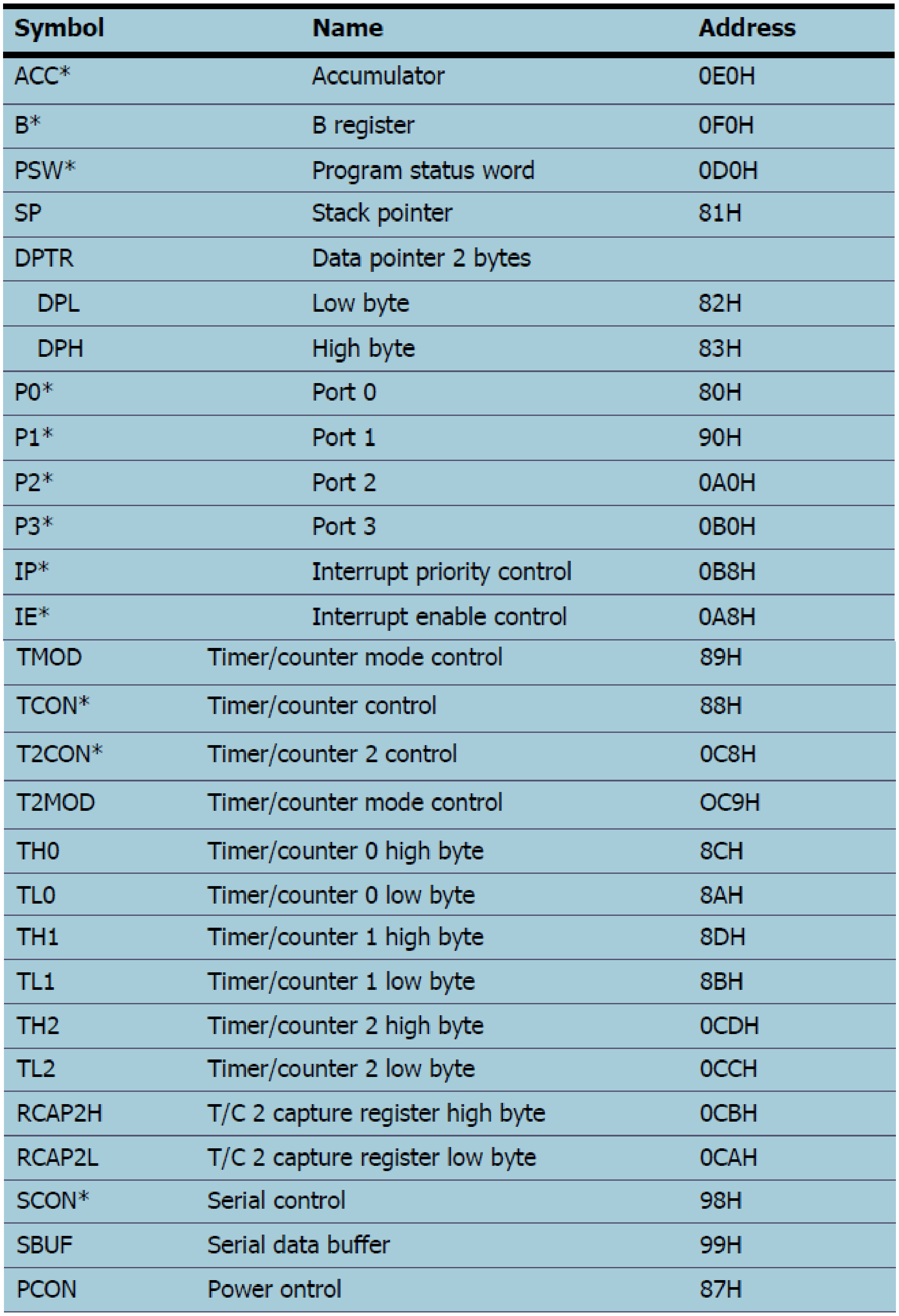


```
MOV  0E0H,#55H    ;is the same as
MOV  A,#55h       ;load 55H into A

MOV  0F0H,R0      ;is the same as
MOV  B,R0         ;copy R0 into B
```

- ✓ The SFR registers have addresses between 80H and FFH
  - Not all the address space of 80 to FF is used by SFR
  - The unused locations 80H to FFH are reserved and must not be used by the 8051 programmer

Special Function Register (SFR) Addresses



Symbol	Name	Address
ACC*	Accumulator	0E0H
B*	B register	0F0H
PSW*	Program status word	0D0H
SP	Stack pointer	81H
DPTR	Data pointer 2 bytes	
DPL	Low byte	82H
DPH	High byte	83H
P0*	Port 0	80H
P1*	Port 1	90H
P2*	Port 2	0A0H
P3*	Port 3	0B0H
IP*	Interrupt priority control	0B8H
IE*	Interrupt enable control	0A8H
TMOD	Timer/counter mode control	89H
TCON*	Timer/counter control	88H
T2CON*	Timer/counter 2 control	0C8H
T2MOD	Timer/counter mode control	OC9H
TH0	Timer/counter 0 high byte	8CH
TL0	Timer/counter 0 low byte	8AH
TH1	Timer/counter 1 high byte	8DH
TL1	Timer/counter 1 low byte	8BH
TH2	Timer/counter 2 high byte	0CDH
TL2	Timer/counter 2 low byte	0CCH
RCAP2H	T/C 2 capture register high byte	0CBH
RCAP2L	T/C 2 capture register low byte	0CAH
SCON*	Serial control	98H
SBUF	Serial data buffer	99H
PCON	Power ontrol	87H

*** Bit addressable**

Example 1

Write code to send 55H to ports P1 and P2, using
(a) Their names (b) their addresses

Solution:

a)

```
MOV A,#55H   ;A=55H
MOV P1,A     ;P1=55H
MOV P2,A     ;P2=55H
```

b) From above Table , P1 address= 90H; P2 address= 0A0H

```
MOV A,#55H   ;A=55H
MOV 90H,A    ;P1=55H
MOV 0A0H,A   ;P2=55H
```

5. Stack and Direct Addressing Mode

- ✓ Only direct addressing mode is allowed for pushing or popping the stack
  - PUSH A is invalid
  - Pushing the accumulator onto the stack must be coded as PUSH 0E0H

Example 2

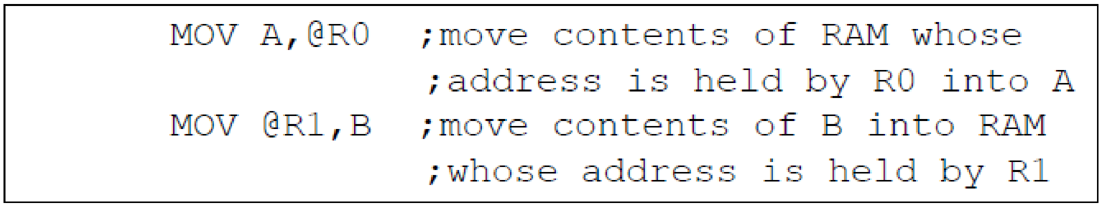
Show the code to push R5 and A onto the stack and then pop them back them into R2 and B, where B = A and R2 = R5

Solution:

```
PUSH 05      ;push R5 onto stack
PUSH 0E0H    ;push register A onto stack
POP 0F0H     ;pop top of stack into B
             ;now register B = register A
POP 02       ;pop top of stack into R2
             ;now R2=R6
```

6. Register Indirect Addressing Mode

- ✓ A register is used as a pointer to the data
  - Only register R0 and R1 are used for this purpose
  - R2 – R7 cannot be used to hold the address of an operand located in RAM
- ✓ When R0 and R1 hold the addresses of RAM locations, they must be preceded by the "@" sign



```
MOV A,@R0   ;move contents of RAM whose
            ;address is held by R0 into A
MOV @R1,B   ;move contents of B into RAM
            ;whose address is held by R1
```

Example 3

Write a program to copy the value 55H into RAM memory locations 40H to 41H using
(a) direct addressing mode, (b) register indirect addressing mode without a loop, and (c) with a loop

Solution:

```
   a)   MOV A,#55H       ;load A with value 55H
        MOV 40H,A        ;copy A to RAM location 40H
        MOV 41H.A        ;copy A to RAM location 41H
   b)   MOV A,#55H       ;load A with value 55H
        MOV R0,#40H      ;load the pointer. R0=40H
        MOV @R0,A        ;copy A to RAM R0 points to
        INC R0           ;increment pointer. Now R0=41h
        MOV @R0,A        ;copy A to RAM R0 points to
   c)   MOV A,#55H       ;A=55H
        MOV R0,#40H      ;load pointer.R0=40H,
        MOV R2,#02       ;load counter, R2=3
AGAIN:  MOV @R0,A        ;copy 55 to RAM R0 points to
        INC R0           ;increment R0 pointer
        DJNZ R2,AGAIN    ;loop until counter = zero
```

- ✓ The advantage is that it makes accessing data dynamic rather than static as in direct addressing mode
  - Looping is not possible in direct addressing mode

Example 4

Write a program to clear 16 RAM locations starting at RAM address 60H

Solution:

```
          CLR A           ;A=0
          MOV R1,#60H     ;load pointer. R1=60H
          MOV R7,#16      ;load counter, R7=16
AGAIN:    MOV @R1,A       ;clear RAM R1 points to
          INC R1          ;increment R1 pointer
          DJNZ R7,AGAIN   ;loop until counter=zero
```

Example 5

Write a program to copy a block of 10 bytes of data from 35H to 60H

Solution:

```
          MOV R0,#35H     ;source pointer
          MOV R1,#60H     ;destination pointer
          MOV R3,#10      ;counter
BACK:     MOV A,@R0       ;get a byte from source
          MOV @R1,A       ;copy it to destination
          INC R0          ;increment source pointer
          INC R1          ;increment destination pointer
          DJNZ R3,BACK    ;keep doing for ten bytes
```

- ✓ R0 and R1 are the only registers that can be used for pointers in register indirect addressing mode
- ✓ Since R0 and R1 are 8 bits wide, their use is limited to access any information in the internal RAM
- ✓ Whether accessing externally connected RAM or on-chip ROM, we need 16-bit pointer
  - In such case, the DPTR register is used

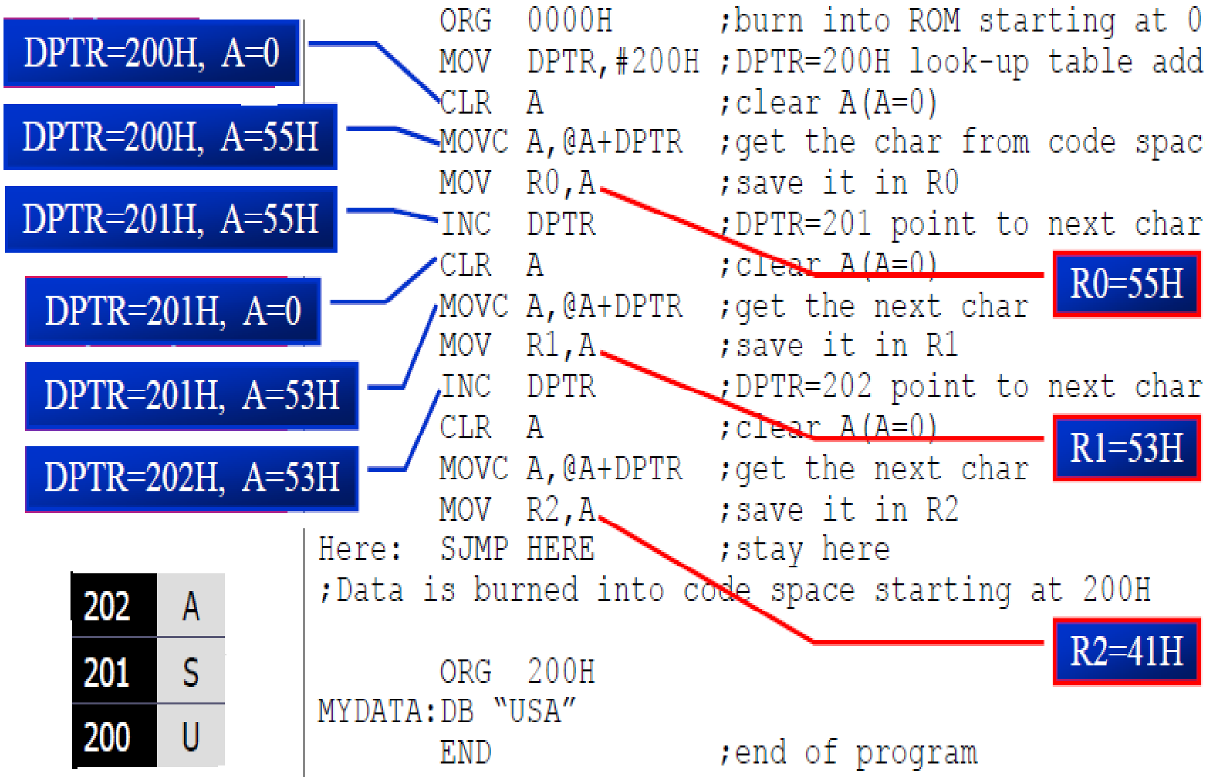
7. Indexed Addressing Mode and On-chip ROM Access

- ✓ Indexed addressing mode is widely used in accessing data elements of look-up table entries located in the program ROM
- ✓ The instruction used for this purpose is **`MOVC A,@A+DPTR`**
  - Use instruction MOVC, "C" means code
  - The contents of A are added to the 16-bit register DPTR to form the 16-bit address of the needed data

Example 6

In this program, assume that the word "USA" is burned into ROM locations starting at 200H. And that the program is burned into ROM locations starting at 0. Analyze how the program works and state where "USA" is stored after this program is run.

Solution:



```
       ORG   0000H      ;burn into ROM starting at 0
       MOV   DPTR,#200H ;DPTR=200H look-up table addr
       CLR   A          ;clear A(A=0)
       MOVC  A,@A+DPTR  ;get the char from code space
       MOV   R0,A       ;save it in R0
       INC   DPTR       ;DPTR=201 point to next char
       CLR   A          ;clear A(A=0)
       MOVC  A,@A+DPTR  ;get the next char
       MOV   R1,A       ;save it in R1
       INC   DPTR       ;DPTR=202 point to next char
       CLR   A          ;clear A(A=0)
       MOVC  A,@A+DPTR  ;get the next char
       MOV   R2,A       ;save it in R2
Here:  SJMP HERE        ;stay here
;Data is burned into code space starting at 200H

       ORG   200H
MYDATA:DB "USA"
       END              ;end of program
```

- ✓ The look-up table allows access to elements of a frequently used table with minimum operations.

Example 5-8

Write a program to get the x value from P1 and send x2 to P2, continuously

Solution:

```
          ORG  0
          MOV  DPTR,#300H     ;LOAD TABLE ADDRESS
          MOV  A,#0FFH        ;A=FF
          MOV  P1,A           ;CONFIGURE P1 INPUT PORT
BACK:     MOV  A,P1           ;GET X
          MOV  A,@A+DPTR      ;GET X SQAURE FROM TABLE
          MOV  P2,A           ;ISSUE IT TO P2
          SJMP BACK           ;KEEP DOING IT
          ORG  300H
XSQR_TABLE:
     DB        0,1,4,9,16,25,36,49,64,81
     END
```