MODULE 1: 8051 Microcontroller Basics

Structure

1.1 Inside the Computer

1.2 Microcontrollers and Embedded Processors

1.3 Block Diagram of 8051

1.4 PSW and Flag Bits

1.5 8051 Register Banks and Stack

1.6 Internal Memory Organization of 8051

1.7 IO Port Usage in 8051

1.8 Types of Special Function Registers and their uses in 8051

1.9 Pins Of 8051

1.10 Memory Address Decoding, 8031/51 Interfacing With External ROM and RAM

1.11 8051 Addressing Modes

Objectives

1. To explain the internal organization and working of Computers, microcontrollers and embedded processors.
2. Compare and contrast the various members of the 8051 family.
3. To explain the registers of the 8051 microcontrollers

1.1 Inside the computer

1. A computer is a multipurpose programmable machine that reads binary instructions from its memory, accepts binary data as input ,processes the data according to those instructions and provides results as output.
2. It is a programmable device made up of both hardware and software. The various components of the computer are called hardware.
3. A set of instructions written for the computer to solve a specific task is called program and collection of programs is called software.
4. The computer hardware consists of four main components. The central processing unit which acts as computer's brain.
5. Input unit through which program and data can be entered to computer, output unit on which the results of the computations can be displayed. Memory in which data and program are stored

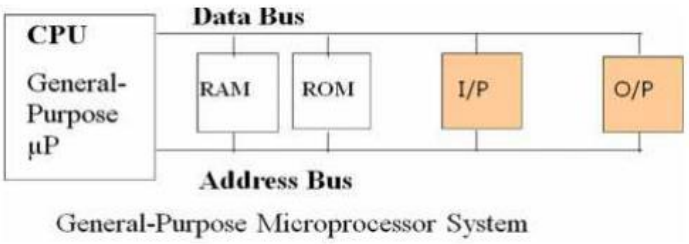


Fig.1: Block diagram of a microcomputer

6. A computer that is designed using a microprocessor as its CPU , is known as a microcomputer. Microprocessor or 'Computer on Chip' first became a commercial reality in 1971 with the introduction of the 4 bit 4004 by Intel.
7. A by product of Microprocessor development was Microcontroller. The same fabrication technology and programming concept that make the general purpose microprocessor also yielded the Microcontroller.

1.1.1 Computer Software

1. A set of instructions written in a specific sequence for the computer to solve a specific task is called a program and software is a collection of such programs.
2. The program stored in the computer memory in the form of binary numbers is called machine instructions. The machine language program is called **object code.**
3. An assembly language is a mnemonic representation of machine language. Machine language and assembly language are low level languages and are processor specific.
4. The assembly language program the programmer enters is called source code. The source code (assembly language) is translated to object code (machine language) using assembler.

5. Programs can be written in high level languages such as C, C++ etc. High level language will be converted to machine language using **compiler or interpreter.**
6. Compiler reads the entire program and translate into the object code and then it is executed by the processor.
7. Interpreter takes one statement of the high level language as input and translate it into object code and then executes.

1.2 Microcontrollers and Embedded Processors

Microprocessors

1. A microprocessor is a general purpose digital computer central processing unit (CPU). Although known as a 'Computer on Chip' the Microprocessor in no sense a complete digital computer.
2. Block diagram of a Microprocessor CPU which contains ALU; Program counter (PC), a stack pointer (SP) ,some working registers , a clock timing circuit and interrupt circuit s is shown in the following Fig.1.1

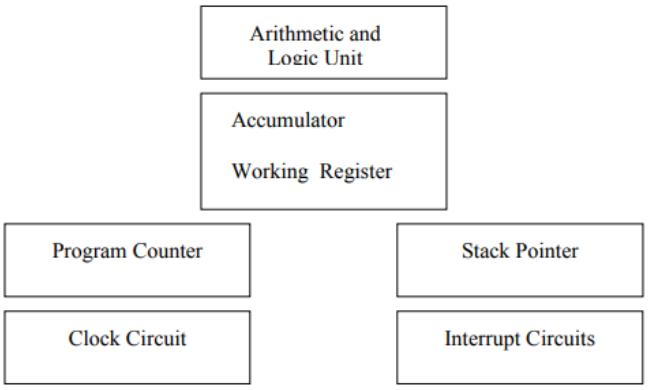


Fig.1.1: Block diagram of microprocessor

3. To make a computer microcomputer one must add memory usually RAM and ROM, memory decoders, an oscillator and a number of Input, Output devices such as serial and parallel ports.
4. In addition special purpose devices such as interrupt handler and counters may be added to relieve the CPU from time consuming counting or timing cores. When the Microcomputer is equipped with mass storage devices, I/O peripherals such as a key board and a display CRT it yields a small computer that can be applied to a range of general purpose applications.
5. The hardware design of a microprocessor is arranged such that a very small or very large system can be configured around the CPU as the application demands as shown in Fig1.
6. The prime use of the Microprocessor is to read data , perform extensive calculations on that data, and store those calculations in a mass storage device or display the results for human use. The programs used by microprocessor are stored in the mass

storage device and loaded into RAM as user directs. A few microprocessor program are stored in ROM. The ROM based programs are primarily small fixed programs that operate peripherals and other fixed devices that are connected to the system.

Microcontroller:

1. 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.
2. Sometimes analog input/output interface makes a part of microcontroller circuit as mixed mode(both analog and digital) in nature. A microcontroller can be compared to a Swiss knife with multiple functions incorporated in the same Integrated Circuits.
3. Block diagram of a typical Microcontroller which is a true computer on a chip is shown in Fig.1.2.
4. The design incorporates all the features found in microprocessor CPU: ALU,PC, SP and registers. It also has other features needed to make a complete computer: ROM, RAM, Parallel I/O, serial I/O, Counters and clock circuits.
5. Like the microprocessor, a microcontroller is a general purpose device, but one that is meant to read data, perform limited calculations on that data and control its environment based on those calculations.
6. The prime use of microcontroller is to control the operation of a machine using a fixed program that is stored in ROM and that does not change over the lifetime of the system.

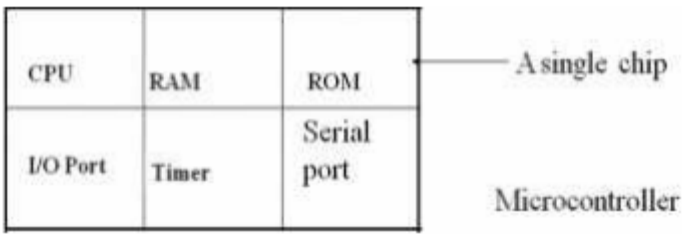
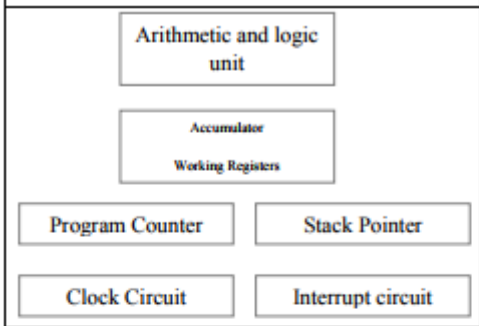
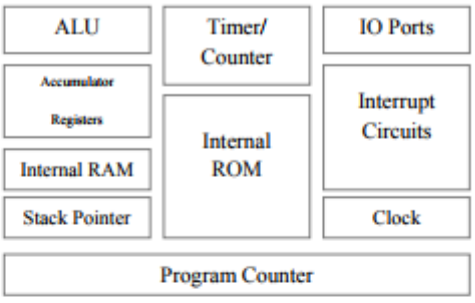


Fig.1.2: Block diagram of microcontroller

The comparison of a microprocessor and microcontroller is shown in Table 1.

Table 1: Comparison between microprocessor and microcontroller

Microprocessor	***Microcontroller***
	
Block diagram of microprocessor	*Block diagram of microcontroller*
Microprocessor contains ALU, General purpose registers, stack pointer, program counter, clock timing circuit, interrupt circuit	Microcontroller contains the circuitry of microprocessor, and in addition it has built in ROM, RAM, I/O Devices, Timers/Counters etc.
It has many instructions to move data between memory and CPU	It has few instructions to move data between memory and CPU
Few bit handling instruction	It has many bit handling instructions
Less number of pins are multifunctional	More number of pins are multifunctional
Single memory map for data and code (program)	Separate memory map for data and code (program)
Access time for memory and IO are more	Less access time for built in memory and IO.
Microprocessor based system requires additional hardware	It requires less additional hardwares
More flexible in the design point of view	Less flexible since the additional circuits which is residing inside the microcontroller is fixed for a particular microcontroller
Large number of instructions with flexible addressing modes	Limited number of instructions with few addressing modes

1.2.1 RISC AND CISC CPU ARCHITECTURES

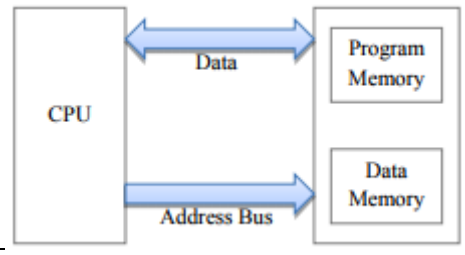
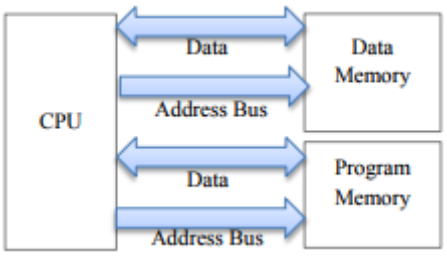
Microcontrollers with small instruction set are called reduced instruction set computer (RISC) machines and those with complex instruction set are called complex instruction set computer (CISC). Intel 8051 is an example of CISC machine whereas microchip PIC 18F87X is an example of RISC machine.

Table 2: Comparison between RISC and CISC Architectures

SL.No.	RISC	CISC
1	Instruction takes one or two cycles	Instruction takes multiple cycles
2	Only load/store instructions are used to access memory	In additions to load and store instructions, memory access is possible with other instructions also
3	Instructions executed by hardware	Instructions executed by the micro program
4	Fixed format instruction	Variable format instructions
5	Few addressing modes	Many addressing modes
6	Few instructions	Complex instruction set
7	Most of the have multiple register banks	Single register bank
8	Highly pipelined	Less pipelined
9	Complexity is in the compiler	Complexity in the micro program

1.2. 2 HARVARD & VON- NEUMANN CPU ARCHITECTURE

Table 3: Comparison between HARVARD & VON- NEUMANN CPU Architectures

SL.No.	Von-Neumann (Princeton architecture)	Harvard architecture
1		
2	It uses single memory space for both instructions and data.	It has separate program memory and data memory
3	It is not possible to fetch instruction code and data	Instruction code and data can be fetched simultaneously
4	Execution of instruction takes more machine cycle	Execution of instruction takes less machine cycle
5	Uses CISC architecture	Uses RISC architecture
6	Instruction pre-fetching is a main feature	Instruction parallelism is a main feature
7	Also known as control flow or control driven computers	Also known as data flow or data driven computers
8	Simplifies the chip design because of single memory space	Chip design is complex due to separate memory space
9	Eg. 8085, 8086, MC6800	Eg.General purpose microcontrollers, special DSP chips etc.

1.2.3 Classification of Mircontrollers

1. 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.
2. Development of some popular microcontrollers is given in Table 4

Table 4: Comparison between HARVARD & VON- NEUMANN CPU Architectures

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)	.

3. 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 and control.
4. Application of microcontrollers are numerous. Starting from domestic applications such as in washing machines, TVs, air conditioners, microcontrollers are used in automobiles, process control industries, cell phones, electrical drives, robotics and in space applications.

1.3 Block diagram of 8051

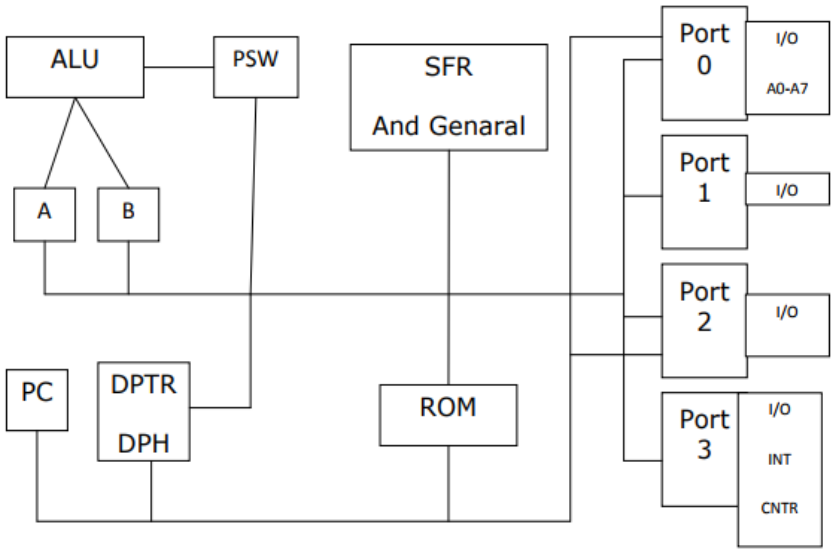


Fig. 1.3: Block diagram of 8051

The programming model of 8051 shows the 8051 as the collection of 8 and 16 bit registers and 8 bit memory locations. These registers and memory locations can be made to operate using software instructions that are incorporated as part of the program instructions.

Salient Features

1. **Eight bit CPU** with registers A (Accumulator) and B
2. **Sixteen bit Program counter (PC) and a data pointer (DPTR)**
3. 8 Bit Program Status Word (PSW)
4. 8 Bit Stack Pointer
5. **4K Code Memory(ROM)**
6. **Internal Data Memory of 128 Bytes(RAM)**
7. 32 I/O Pins arranged as 4 , 8 Bit ports
8. **Two 16 Bit Timer/Counter :T0, T1**
9. **Full Duplex serial data receiver/transmitter**
10. Control Registers :TCON,TMOD,SCON,PCON,IP and IE
11. Two External and Internal Interrupt sources

Microcontroller Chips:

1. Embedded (Self -Contained) 8 - bit Microcontroller
2. 16 to 32 Microcontrollers
3. Digital Signal Processors

1.4 PSW and Flag Bits

PSW (Program Status Word).

This is an 8 bit register which **contains the arithmetic status of ALU** and the **bank select bits of register banks.**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CY	**AC**	**F0**	**RS1**	**RS0**	**OV**	**-**	**P**

Note: CY- carry flag AC - auxiliary carry flag

1. F0 - available to the user for general purpose
2. RS1, RS0 - register bank select bits
3. OV - overflow P - parity
4. Stack Pointer (SP) – it contains the address of the data item on the top of the stack. Stack may reside anywhere on the internal RAM. On reset, SP is initialized to 07 so that the default stack will start from address 08 onwards.
5. Data Pointer (DPTR) – DPH (Data pointer higher byte), DPL (Data pointer lower byte).
6. This is a 16 bit register which is used to furnish address information for internal and external program memory and for external data memory. Program Counter (PC) – 16 bit PC contains the address of next instruction to be executed.
7. On reset PC will set to 0000. After fetching every instruction PC will increment by one.

1.5 8051 Register Banks and Stack

- **Register Banks:** 00h to 1Fh.
  1. The 8051 uses 8 general-purpose registers R0 through R7 (R0, R1, R2, R3, R4, R5, R6, and R7).
  2. There are four such register banks. Selection of register bank can be done through RS1, RS0 bits of PSW. On reset, the default Register Bank 0 will be selected.
- **Bit Addressable RAM**: 20h to 2Fh.
  1. The 8051 supports a special feature which allows access to bit variables. This is where individual memory bits in Internal RAM can be set or cleared.
  2. In all there are 128 bits numbered 00h to 7Fh. Being bit variables any one variable can have a value 0 or 1.
  3. A bit variable can be set with a command such as SETB and cleared with a command such as CLR.

Example instructions are:
SETB 25h; sets the bit 25h (becomes 1)
CLR 25h; clears bit 25h (becomes 0)

Note: bit 25h is actually bit 5 of Internal RAM location 24h. The Bit Addressable area of the RAM is just 16 bytes of Internal RAM located between 20h and 2Fh.

- **General Purpose RAM**: 30h to 7Fh.

1. Even if 80 bytes of Internal RAM memory are available for general-purpose data storage, user should take care while using the memory location from 00 -2Fh.
2. Since these locations are also the default register space, stack space, and bit addressable space. It is a good practice to use general purpose memory from 30 – 7Fh.
3. The general purpose RAM can be accessed using direct or indirect addressing modes.

- **Fig. 1.4 shows Internal RAM organisation of 8051**

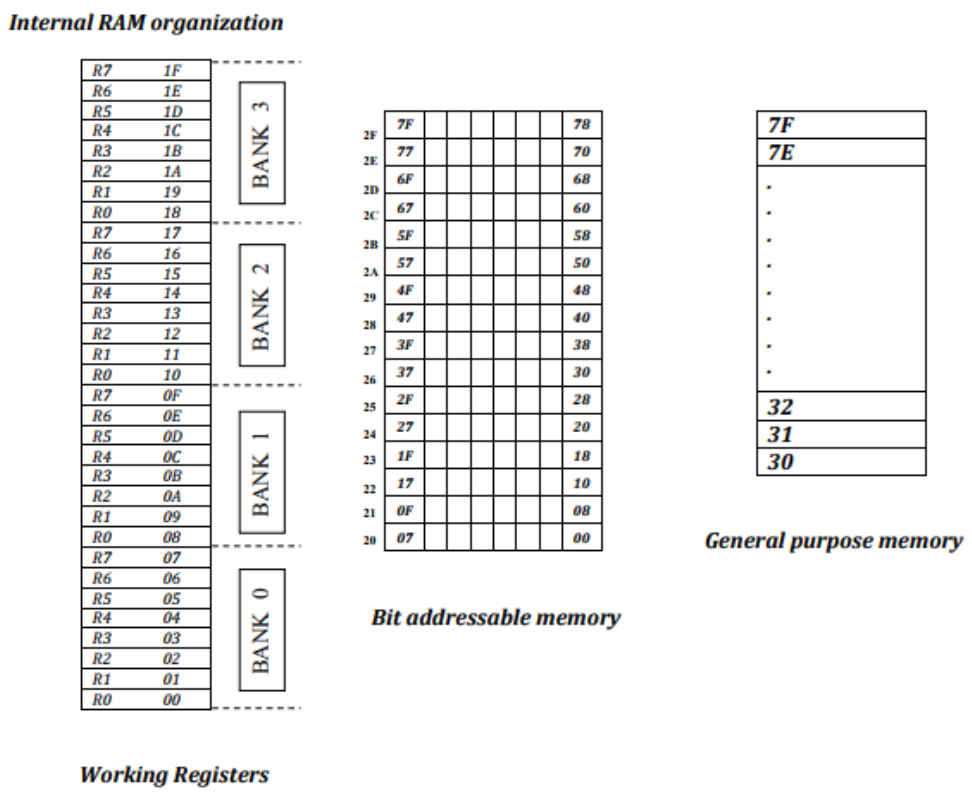


Fig. 1.4: Internal RAM organisation

1.5.1 Stack

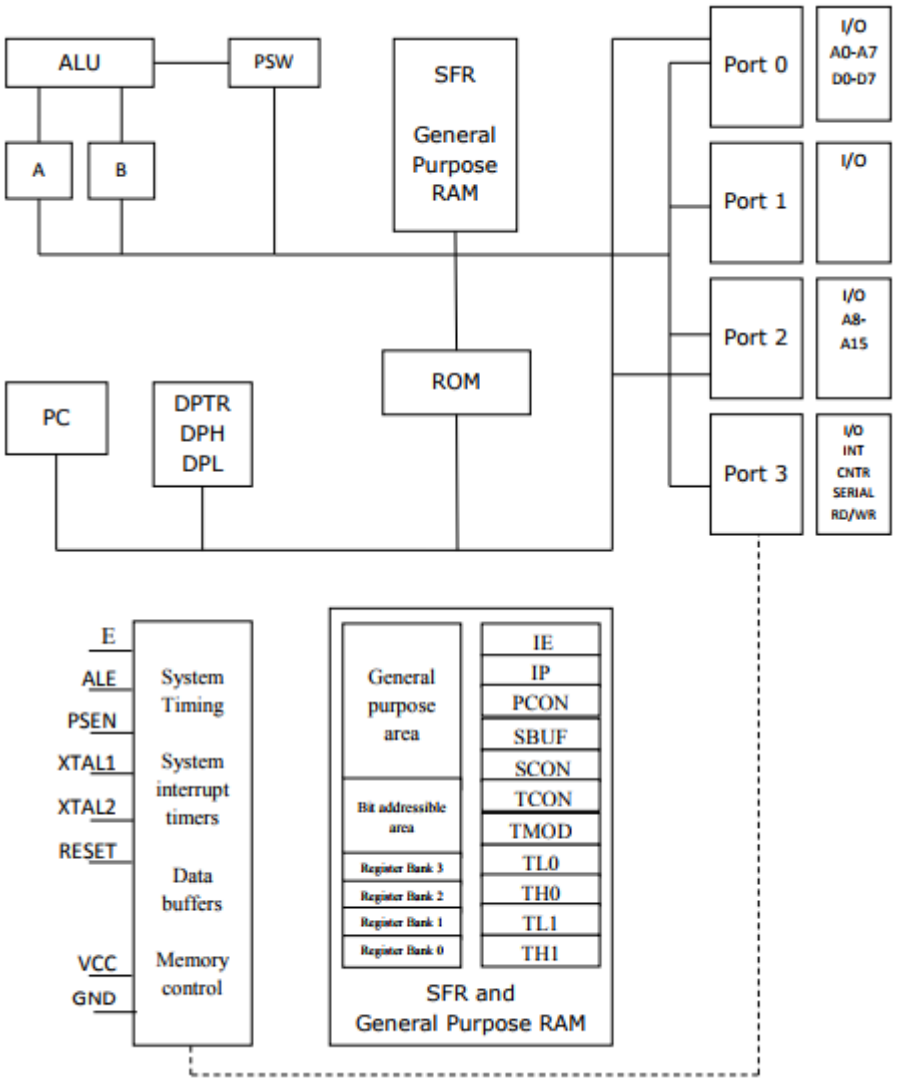
1. **A stack** is a **last in first out memory**.
2. In 8051 **internal RAM space can be used as stack**.
3. The **address of the stack** is contained in a register called **stack pointer.**
4. Instructions **PUSH and POP** are used for stack operations.
5. When a data is to be placed on the stack, the stack pointer **increments before storing the data on the stack** so that the stack grows up as data is stored (pre-increment).

6. As the data is retrieved from the stack the byte is read from the stack, and then **SP decrements to point the next available byte** of stored data (post decrement). The stack pointer is set to 07 when the 8051 resets.
7. So that default stack memory starts from address location 08 onwards (to avoid overwriting the default register bank i.e., bank 0).

Eg; Show the stack and SP for the following.

```
[SP]=07 //CONTENT OF SP IS 07 (DEFAULT VALUE)
MOV R6, #25H ;          [R6] =25H //CONTENT OF R6 IS 25H
MOV R1, #12H ;          [R1] =12H //CONTENT OF R1 IS 12H
MOV R4, #0F3H ;         [R4] =F3H //CONTENT OF R4 IS F3H
PUSH 6 ;                [SP] =08 [08] = [06] =25H //CONTENT OF 08 IS 25H
PUSH 1 ;                [SP] =09 [09] = [01] =12H //CONTENT OF 09 IS 12H
PUSH 4 ;                [SP] =0A [0A] = [04] =F3H //CONTENT OF 0A IS F3H
POP 6 ;                 [06] = [0A] =F3H [SP] =09 //CONTENT OF 06 IS F3H
POP 1 ;                 [01] = [09] =12H [SP] =08 //CONTENT OF 01 IS 12H
POP 4 ;                 [04] = [08] =25H [SP] =07 //CONTENT OF 04 IS 25H
```

1.6 Internal Memory Organization of 8051



Salient features of 8051 microcontroller are given below.

1. Eight bit CPU
2. On chip clock oscillator
3. 4Kbytes of internal program memory (code memory) [ROM]
4. 128 bytes of internal data memory [RAM]
5. 64 Kbytes of external program memory address space.
6. 64 Kbytes of external data memory address space.
7. 32 bi directional I/O lines (can be used as four 8 bit ports or 32 individually addressable I/O lines) Two 16 Bit Timer/Counter :T0, T1
8. Full Duplex serial data receiver/transmitter
9. Four Register banks with 8 registers in each bank.
10. Sixteen bit Program counter (PC) and a data pointer (DPTR)
11. 8 Bit Program Status Word (PSW)
12. 8 Bit Stack Pointer
13. Five vector interrupt structure (RESET not considered as an interrupt.)
14. 8051 CPU consists of 8 bit ALU with associated registers like accumulator 'A' , B register,

15. PSW, SP, 16 bit program counter, stack pointer. ALU can perform arithmetic and logic functions on 8 bit variables.
16. 8051 has 128 bytes of internal RAM which is divided into
17. Working registers [00 – 1F] o Bit addressable memory area [20 – 2F] o General purpose memory area (Scratch pad memory) [30-7F]
18. 8051 has 4 K Bytes of internal ROM. The address space is from 0000 to 0FFFh. If the program size is more than 4 K Bytes 8051 will fetch the code automatically from external memory.
19. Accumulator is an 8 bit register widely used for all arithmetic and logical operations. Accumulator is also used to transfer data between external memory. B register is used along with Accumulator for multiplication and division. A and B registers together is also called MATH registers.

1.7 IO Port Usage in 8051

1. I/O Port pins, Ports and Circuits: One major feature of a microcontroller is versatility built into the I/O circuits that connect the 8051 to the outside world.
2. Out of 40 pins 24 pins may each be used for one of two entirely different functions yielding a total pin configuration of 64.
3. But the port pins have been multiplexed to perform different functions to make 8051 as 40 Pin IC
   The port pin circuitry is as shown below

Port-0

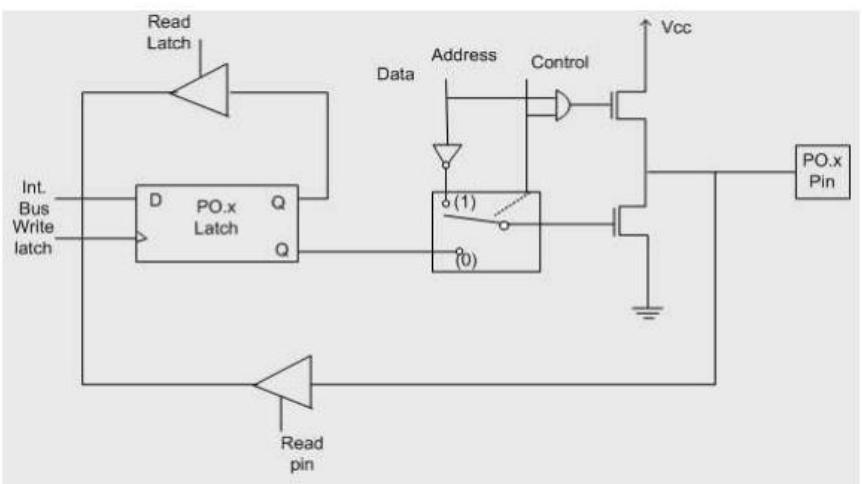


Fig. 1.5: Port -0

1. Port -0 has 8 pins (P0.0-P0.7).The structure of a Port-0 pin is shown in Fig.1.5..Port-0 can be configured as a normal bidirectional I/O port or it can be used for address/data interfacing for accessing external memory.
2. When control is '1', the port is used for address/data interfacing. When the control is '0', the port can be used as a normal bidirectional I/O port. Let us assume that control is '0'.

3. When the port is used as an input port, '1' is written to the latch. In this situation both the output MOSFETs are 'off'. Hence the output pin floats. This high impedance pin can be pulled up or low by an external source.
4. When the port is used as an output port, a '1' written to the latch again turns 'off' both the output MOSFETs and causes the output pin to float. An external pull-up is required to output a '1'.
5. But when '0' is written to the latch, the pin is pulled down by the lower MOSFET. Hence the output becomes zero. When the control is '1', address/data bus controls the output driver MOSFETs. If the address/data bus (internal) is '0', the upper MOSFET is 'off' and the lower MOSFET is 'on'.
6. The output becomes '0'. If the address/data bus is '1', the upper transistor is 'on' and the lower transistor is 'off'.
7. Hence the output is '1'. Hence for normal address/data interfacing (for external memory access) no pull-up resistors are required. Port-0 latch is written to with 1's when used for external memory access

Port-1

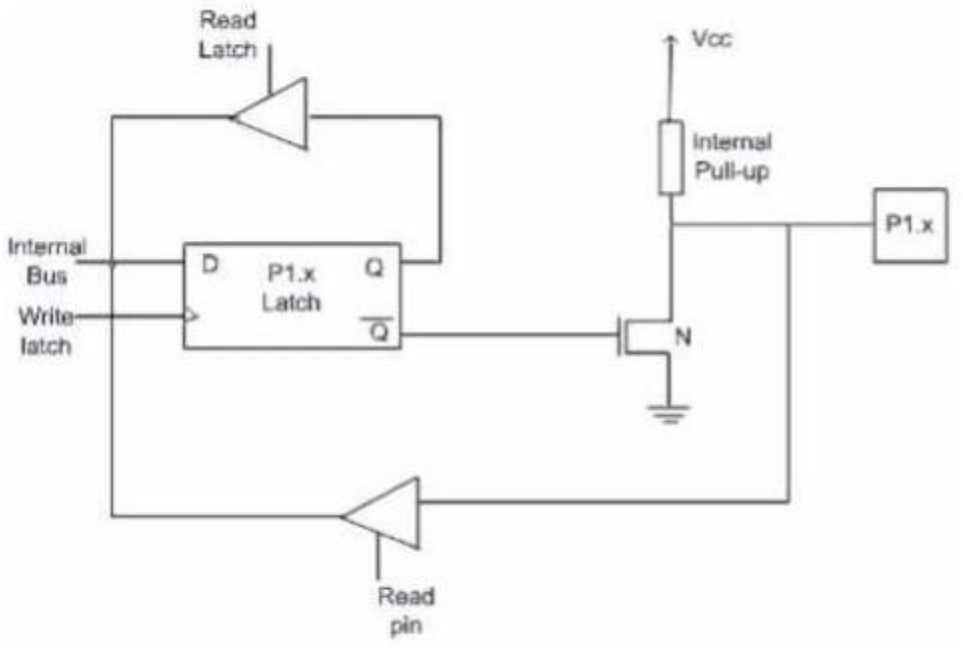


Fig 1.6: Port 1 Structure

1. Port-1 has 8 pins (P1.1-P1.7) .The structure of a port-1 pin is shown in fig Fig.1.6
2. Port-1 does not have any alternate function i.e. it is dedicated solely for I/O interfacing. When used as output port, the pin is pulled up or down through internal pull-up.
3. To use port- 1 as input port, '1' has to be written to the latch. In this input mode when '1' is written to the pin by the external device then it reads fine.
4. But when '0' is written to the pin by the external device then the external source must sink current due to internal pull-up. If the external device is not able to sink the current the pin voltage may rise, leading to a possible wrong reading.

Port-2

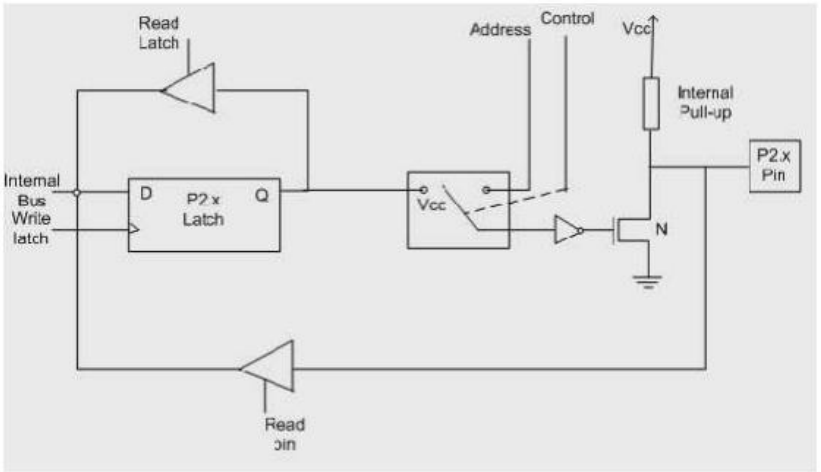


Fig 1.7: Port 2 Structure

1. Port-2 has 8-pins (P2.0-P2.7) . The structure of a port-2 pin is shown in Fig 1.7
2. Port-2 is used for higher external address byte or a normal input/output port. The I/O operation is similar to Port-1.
3. Port-2 latch remains stable when Port-2 pin are used for external memory access. Here again due to internal pull-up there is limited current driving capability.

Port-3

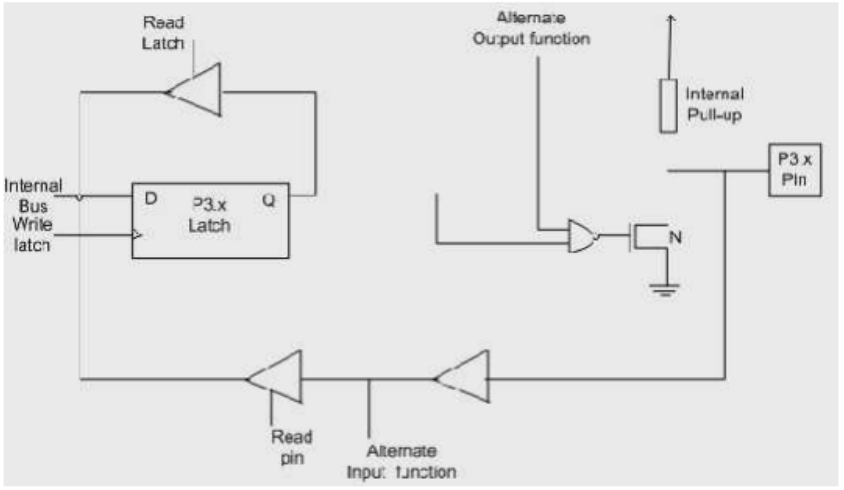


Fig 1.7: Port 3 Structure

1. Each pin of Port-3 can be individually programmed for I/O operation or for alternate function. The alternate function can be activated only if the corresponding latch has been written to '1'.
2. To use the port as input port, '1' should be written to the latch. This port also has internal pull-up and limited current driving capability.
3. Alternate functions of Port-3 pins –

P3.0	RxD
P3.1	TxD
P3.2	INT0
P3.3	INT1
P3.4	T0
P3.5	T1
P3.6	WR
P3.7	RD

Note:

1. Port 1, 2, 3 each can drive 4 LS TTL inputs.
2. Port-0 can drive 8 LS TTL inputs in address /data mode. For digital output port, it needs external pull-up resistors.
3. Ports-1,2and 3 pins can also be driven by open-collector or open-drain outputs.

- Each Port 3 bit can be configured either as a normal I/O or as a special function bit. Reading a port (port-pins) versus reading a latch.
- There is a subtle difference between reading a latch and reading the output port pin. The status of the output port pin is sometimes dependant on the connected load.
- For instance if a port is configured as an output port and a '1' is written to the latch, the output pin should also show '1'.
- If the output is used to drive the base of a transistor, the transistor turns 'on'. If the port pin is read, the value will be '0' which is corresponding to the base-emitter voltage of the transistor.
- Reading a latch: Usually the instructions that read the latch, read a value, possibly change it, and then rewrite it to the latch. These are called "read-modify-write" instructions.

1.8 Types of Special Function Registers and their uses in 8051

1. The **8051 operations that do not use the internal RAM addresses from 00h to 7fh** are done by a group of specific internal registers each called a **specific function register** (SFR) which may be addressed much like internal RAM using addresses from 80h to FFh.
2. Some SFRs are also bit addressable as is the case for the bit area of RAM. This feature allows the programmer the programmer to change only what needs to be altered leaving the remaining bits in that SFR unchanged. Not all of the addresses from 80h to FFh are used for SFRs .
3. Only the addressed ones can be used in programming SFRs and equivalent internal RAM addresses are shown in Fig.1.8.

4. SFR Map: The set of Special Function Registers (SFRs) contain important registers such as Accumulator, Register B, I/O Port latch registers, Stack pointer, Data Pointer, Processor Status Word (PSW) and various control registers.
5. The detailed map of various registers is shown in the following Fig.1.8. The PC is not part of the SFR 0e0h or 8ch. and has no internal RAM address.

F8H								
F0H	B*							
E8H								
E0H	ACC*							
D8H								
D0H	PSW*							
C8H	(T2CON)*		(RCAP2L)	(RCAP2H)	(TL2)	(TH2)		
C0H								
B8H	IP*							
B0H	P3*							
A8H	IE*							
A0H	P2*							
98H	SCON*	SBUF						
90H	P1*							
88H	TCON*	TMOD	TL0	TL1	TH0	TH1		
80H	P0*	SP	DPL	DPH				PCON

Fig 1.8: Special Function Registers and the addresses

1.9 Pins Of 8051

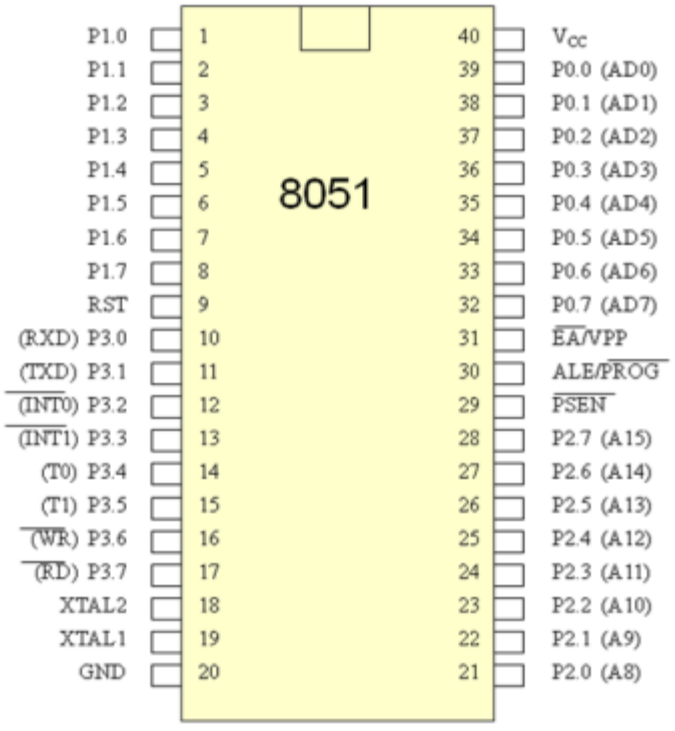


Fig 1.9: Pin diagram of 8051

Table 4: Pinout Description of 8051

Pins 1-8	PORT 1. Each of these pins can be configured as an input or an output.
Pin 9	RESET. A logic one on this pin disables the microcontroller and clears the contents of most registers. In other words, the positive voltage on this pin resets the microcontroller. By applying logic zero to this pin, the program starts execution from the beginning.
Pins10-17	PORT 3. Similar to port 1, each of these pins can serve as general input or output. Besides, all of them have alternative functions
Pin 10	RXD. Serial asynchronous communication input or Serial synchronous communication output.
Pin 11	TXD. Serial asynchronous communication output or Serial synchronous communication clock output.
Pin 12	INT0.External Interrupt 0 input
Pin 13	INT1. External Interrupt 1 input
Pin 14	T0. Counter 0 clock input
Pin 15	T1. Counter 1 clock input
Pin 16	WR. Write to external (additional) RAM
Pin 17	RD. Read from external RAM
Pin18, 19	XTAL2, XTAL1. Internal oscillator input and output. A quartz crystal which specifies operating frequency is usually connected to these pins.
Pin 20	GND. Ground.
Pin 21-28	Port 2. If there is no intention to use external memory then these port pins are configured as general inputs/outputs. In case external memory is used, the higher address byte, i.e. addresses A8-A15 will appear on this port. Even though memory with capacity of 64Kb is not used, which means that not all eight port bits are used for its addressing, the rest of them are not available as inputs/outputs.
Pin 29	PSEN. If external ROM is used for storing program then a logic zero (0) appears on it every time the microcontroller reads a byte from memory.
Pin 30	ALE. Prior to reading from external memory, the microcontroller puts the lower address byte (A0-A7) on P0 and activates the ALE output. After receiving signal from the ALE pin, the external latch latches the state of P0 and uses it as a memory chip address. Immediately after that, the ALE pin is returned its previous logic state and P0 is now used as a Data Bus.
Pin 3	EA. By applying logic zero to this pin, P2 and P3 are used for data and address transmission with no regard to whether there is internal memory or not. It means that even there is a program written to the microcontroller, it will not be executed. Instead, the program written to external ROM will be executed. By applying logic one to the EA pin, the microcontroller will use both memories, first internal then external (if exists).
Pin 32-3	PORT 0. Similar to P2, if external memory is not used, these pins can be used as general inputs/outputs. Otherwise, P0 is configured as address output (A0-A7) when the ALE pin is driven high (1) or as data output (Data Bus) when the ALE pin is driven low (0).
Pin 40	VCC. +5V power supply

1.10 Memory Address Decoding, 8031/51 Interfacing With External ROM and RAM

Connecting External Memory: The following figure shows the connection between an 8051 and external memory

Interfacing External Memory:

1. The system designer is not limited by the amount of internal ROM and RAM available on chip.
2. Two separate external memory spaces are made available by the 16 bit Program Counter PC and DPTR and by different control pins for enabling the external ROM and RAM chips.
3. Internal control entry accesses the correct physical memory , depending on the machine cycle state and opcode being executed .
4. There are several reasons for adding external memory, particularly Program Memory, when applying the 8051 in a system. When project is in the prototype stage, having a masked internal ROM for each program "try" is prohibitive.
5. To help the programmer the manufacturers make available an EPROM version, the 8751, which has 4K of on-chip EPROM that may be programmed and erased as needed as the program is developed If external program/data memory are to be interfaced, they are interfaced in the following way

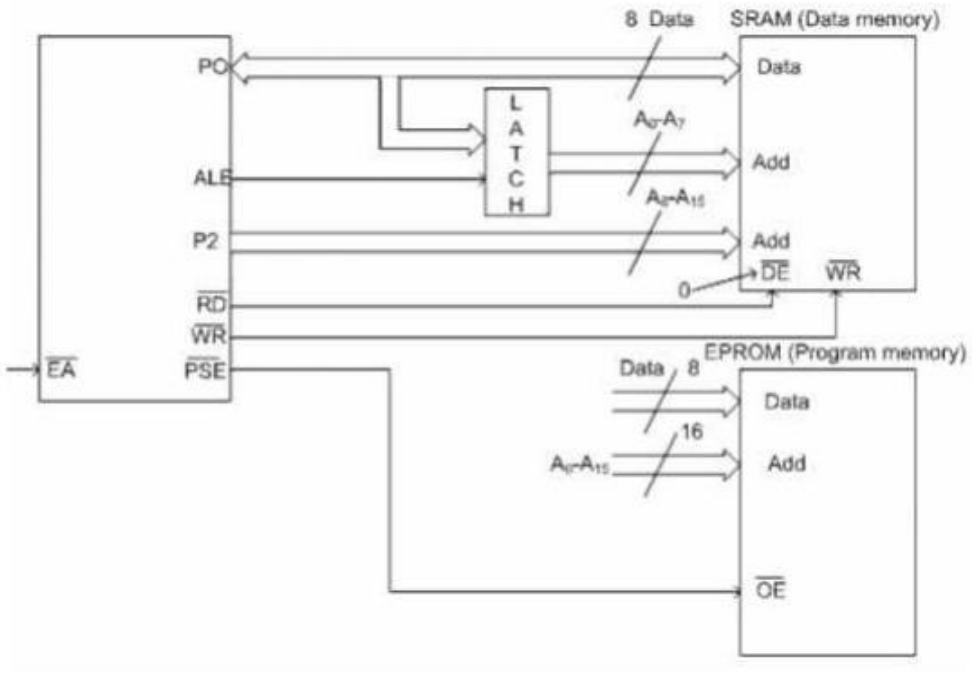


Fig 1.10: Diagram for Interfacing of External Memory

Interfacing of 16 K Byte of RAM and 32 K Byte of EPROM to 8051 Number of address lines required for 16 Kbyte memory is 14 lines and that of 32Kbytes of memory is 15 lines.

The connections of external memory is shown below.

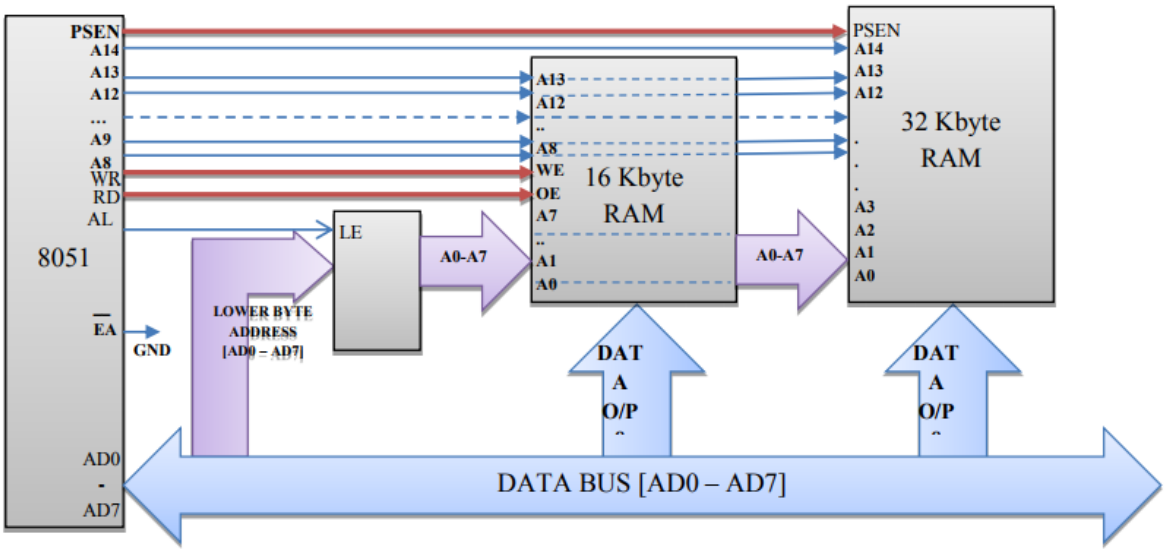


Fig 1.11: Pin diagram of 8051

1. The lower order address and data bus are multiplexed. De-multiplexing is done by the latch. Initially the address will appear in the bus and this latched at the output of latch using ALE signal.
2. The output of the latch is directly connected to the lower byte address lines of the memory. Later data will be available in this bus. Still the latch output is address it self.
3. The higher byte of address bus is directly connected to the memory. The number of lines connected depends on the memory size.
4. The RD and WR (both active low) signals are connected to RAM for reading and writing the data.
5. PSEN of microcontroller is connected to the output enable of the ROM to read the data from the memory.
6. EA (active low) pin is always grounded if we use only external memory. Otherwise, once the program size exceeds internal memory the microcontroller will automatically switch to external memory.

Accessing external memory:

1. For external program memory, always 16 bit address is used.
2. For example –Access to external data memory can be either 8-bit address or 16-bit address - 8-bit address-
   MOVX A, @Rp where Rp is either R0 or R1
   MOVX @Rp, A 16 bit address- MOVX A,@DPTR
   MOV X @DPTR, A.

The external memory access in 8051 can be shown by a schematic diagram as given in Fig 1.12

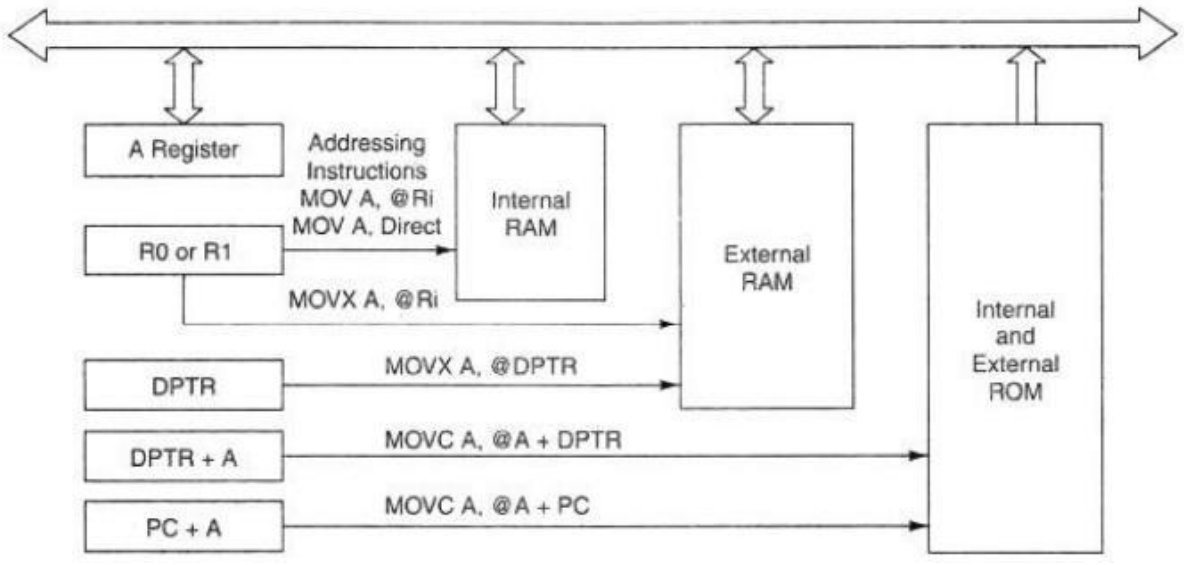


Fig 1.12. Schematic diagram of external memory access

1.11 8051 Addressing Modes

Various methods of accessing the data are called addressing modes.
8051 addressing modes are classified as follows.
1. Immediate addressing.
2. Register addressing.
3. Direct addressing.
4. Indirect addressing.
5. Relative addressing.
6. Absolute addressing.
7. Long addressing.
8. Indexed addressing.
9. Bit inherent addressing.
10. Bit direct addressing.

1. Immediate addressing.

In this addressing mode the data is provided as a part of instruction itself. In other words data immediately follows the instruction.

Eg. MOV A,#30H
ADD A, #83H ; Symbol indicates the data is immediate

2. Register addressing.

In this addressing mode the register will hold the data. One of the eight general registers (R0 to R7) can be used and specified as the operand.

Eg. MOV A,R0 ADD A,R6 R0 – R7 will be selected from the current selection of register bank. The default register bank will be bank 0

3. Direct addressing

1. There are two ways to access the internal memory. Using direct address and indirect address. Using direct addressing mode we can not only address the internal memory but SFRs also.
2. In direct addressing, an 8 bit internal data memory address is specified as part of the instruction and hence, it can specify the address only in the range of 00H to FFH. In this addressing mode, data is obtained directly from the memory.
   Eg. MOV A,60h ADD A,30h

4. Indirect addressing

1. The indirect addressing mode uses a register to hold the actual address that will be used in data movement.
2. Registers R0 and R1 and DPTR are the only registers that can be used as data pointers. Indirect addressing cannot be used to refer to SFR registers.
3. Both R0 and R1 can hold 8 bit address and DPTR can hold 16 bit address.
   Eg. MOV A,@R0 ADD A,@R1 MOVX A,@DPTR

5. Indexed addressing.

1. In indexed addressing, either the program counter (PC), or the data pointer (DTPR)—is used to hold the base address, and the A is used to hold the offset address.
2. Adding the value of the base address to the value of the offset address forms the effective address. Indexed addressing is used with JMP or MOVC instructions.
3. Look up tables are easily implemented with the help of index addressing.
4. Eg. MOVC A, @A+DPTR // copies the contents of memory location pointed by the sum of the accumulator A and the DPTR into accumulator A.
   MOVC A, @A+PC // copies the contents of memory location pointed by the sum of the accumulator A and the program counter into accumulator A

6. Relative Addressing.

1. Relative addressing is used only with conditional jump instructions. The relative address, (offset), is an 8 bit signed number, which is automatically added to the PC to make the address of the next instruction.
2. The 8 bit signed offset value gives an address range of +127 to —128 locations. The jump destination is usually specified using a label and the assembler calculates the jump offset accordingly.
3. The advantage of relative addressing is that the program code is easy to relocate and the address is relative to position in the memory.
   Eg. SJMP LOOP1 JC BACK

7. Absolute addressing

1. Absolute addressing is used only by the AJMP (Absolute Jump) and ACALL (Absolute Call) instructions.
2. These are 2 bytes instructions. The absolute addressing mode specifies the lowest 11 bit of the memory address as part of the instruction.
3. The upper 5 bit of the destination address are the upper 5 bit of the current program counter.
4. Hence, absolute addressing allows branching only within the current 2 Kbyte page of the program memory.

   Eg. AJMP LOOP1 ACALL LOOP2

8. Long Addressing

1. The long addressing mode is used with the instructions LJMP and LCALL. These are 3 byte instructions.
2. The address specifies a full 16 bit destination address so that a jump or a call can be made to a location within a 64 Kbyte code memory space.

   Eg. LJMP FINISH LCALL DELAY

9. Bit Inherent Addressing

1. In this addressing, the address of the flag which contains the operand, is implied in the opcode of the instruction.

   Eg. CLR C ; Clears the carry flag to 0

10. Bit Direct Addressing

2. In this addressing mode the direct address of the bit is specified in the instruction. The RAM space 20H to 2FH and most of the special function registers are bit addressable. Bit address values are between 00H to 7FH.

   Eg. CLR 07h ; Clears the bit 7 of 20h RAM space SETB 07H ; Sets the bit 7 of 20H RAM space.

1.11.1 PROGRAMS

1. Write a program to store data FFH into RAM memory locations 50H to 58H using **direct addressing mode**

```
ORG 0000H; Set program counter 0000H
MOV A, #0FFH; Load FFH into A
MOV 50H, A ; Store contents of A in location 50H
MOV 51H, A ; Store contents of A in location 5IH
MOV 52H, A ; Store contents of A in location 52H
MOV 53H, A ; Store contents of A in location 53H
MOV 54H, A ; Store contents of A in location 54H
MOV 55H, A ; Store contents of A in location 55H
```

```
MOV 56H, A ; Store contents of A in location 56H
MOV 57H, A ; Store contents of A in location 57H
MOV 58H, A ; Store contents of A in location 58H
END
```

2. Write a program to store data FFH into RAM memory locations 50H to 58H using indirect addressing mode.

```
ORG 0000H; Set program counter 0000H
MOV A, #0FFH ; Load FFH into A
MOV RO, #50H ; Load pointer, R0-50H
MOV R5, #08H ; Load counter, R5-08H
 Start: MOV @RO, A ; Copy contents of A to RAM pointed by R0
INC RO ; Increment pointer
DJNZ R5, start ; Repeat until R5 is zero
END
```

Note : More Programs will be solved in classroom discussion

Outcomes

At the end of the module, students will be able to:

CO1: Interpret the architectural features of 8051 microcontroller and its peripherals, Memory organization, Memory interfacing and looping instructions. **[L4, MODULE 1, 2]**

CO2: Develop 8051 programs in assembly language to solve arithmetic and logical programs. **[L3 MODULE 1, 2]**

TEXT BOOKS:

1. The 8051 Microcontroller and Embedded Systems Using Assembly and C , Muhammad Ali Mazadi , Pearson 2nd Edition, 2008.

Reference Books

1. The 8051 Microcontroller, Kenneth Ayala, Cengage Learning , 3rd Edition, 2005
2. The 8051 Microcontroller and Embedded Systems ,Manish K Patel, McGraw Hill, 2014
Microcontrollers: Architecture, Programming, Interfacing and System Design, Raj Kamal , Pearson ,1st Edition, 2012