Module 1: 8051 Microcontroller Basics

Inside the computer

Computer: 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. It is a programmable device made up of both hardware and software. The various components of the computer are called hardware. A set of instructions written for the computer to solve a specific task is called program and collection of programs is called software.

One of the most important feature of a computer is its memory. Common terms used to describe amount of memory used are bit, byte, Nibble etc.

Bit: Its a binary digit that can have only 0 or 1.

Nibble: Half byte: 4 bits   0000

Byte: 8 bits   0000 0000

word: 2 byte---16 bits -- 0000 0000 0000 0000

above all are composed of combination of 0 and 1. Terms used to describe amount of memory used in IBM PC's and compatibles is given below.

Kilobyte(K)---2^{10} bytes ---1024 bytes

Megabyte(M)--- 2^{20} bytes

Gigabytes(G) -- 2^{30} bytes

Terabytes(T) 2^{40} bytes

For example if computer has 16 megabytes of memory i.e 16x 2^{20} =2^{24} i.e it is having 24 addressable lines with 16 megabytes of memory. Each location can have a maximum of 1 byte of data.

Internal Organization of Computers

A computer that is designed using a microprocessor as its CPU, is known as a microcomputer. The computer hardware consists of four main components. The central processing unit which acts as computer's brain. 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.

It is having two types of memory. 1) RAM 2) ROM

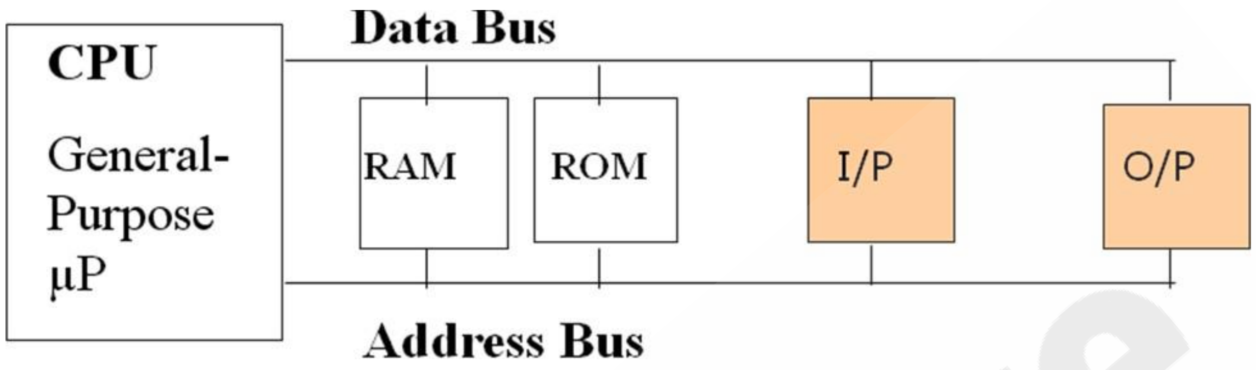


Fig 1. Block diagram of a microcomputer

CPU and its relation to RAM and ROM

CPU to process information data must be stored in RAM or ROM. RAM and ROM are called primary memory and disks are called secondary memory.

Inside the CPU

Program stored in memory provides instructions to CPU to perform. The function is to fetch instructions from memory and execute them. CPU registers stores information temporarily. Registers inside the CPU may be 8 bit, 16 bit,32 bit, 64 bit etc. depending on CPU. If bigger registers are used then it is good for CPU but cost will increase. CPU has ALU which is responsible for performing arithmetic operations like add, subtract and logical functions like AND, OR and NOT. Program counter always points to address of next instruction to be executed. After the instruction execution program counter will be automatically incremented to point next instruction. Instruction decoder interprets the instruction fetched into CPU.

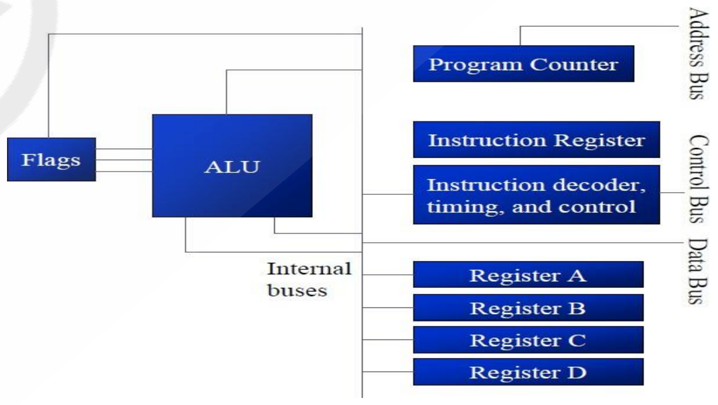


Fig 2: Internal block diagram of CPU

Microprocessors

A microprocessor is a general purpose digital computer central processing unit (CPU). Also known as a 'Computer on Chip'. 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 figure.

To make a computer, microcontroller 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. 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.

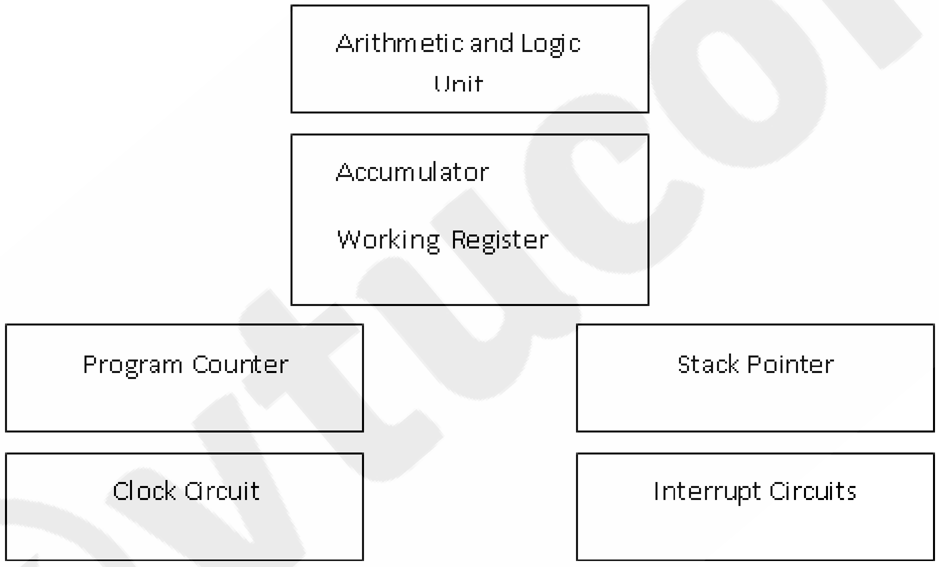


Fig.3.Block Diagram of a Microprocessor

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. 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 programs 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.

Microcontrollers and Embedded Processors

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. 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. Like the microprocessor, a microcontroller is a general purpose device, but one that is meant to read data, perform limited calculations on that data.

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.

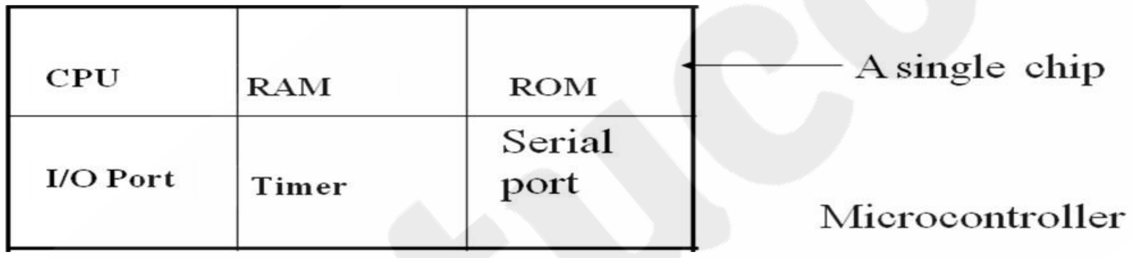


Fig4. Block diagram of a single chip computer

Comparison between Microprocessor and Microcontroller

Microprocessor	Microcontroller
1) It contains ALU, General purpose registers, stack pointer, program counter, interrupt circuits etc.	1) It contains microprocessors and in addition it has built in ROM, RAM, I/O devices, timers and counters.
2) Requires more hardware, increases in PCB	2) Requires less hardware, reduced PCB size.
3) Access time for memory and I/O devices are more	3) Access time is less due to in built memory and I/O port
4) Many instructions to move data between memory and CPU	4) One or more instructions to move data between memory and CPU
5) More flexible from design point of view	5) Less flexible
6) Single memory map for data and code	6) Separate memory map for data and code.
7) Few pins are multifunctional	7) More pins are multifunctional
8) One or more bit handling instructions are	8) More number of bit handling instructions are

available	available.

Criteria for choosing a Microcontroller

There are wide varieties of Microcontrollers available in the market. Program written for one Microcontroller will not run others. Choice of the microcontroller is based on three parameters.

1. It must perform the required task efficiently and effectively.

   a) Speed, Amount of RAM and ROM on chip, Power consumption

   b) Number of I/O pins and timer on chip

   c) Cost per unit, Ease of upgrading

   d) Packaging-No of Pins and Packaging formats

2. Availability of software development tools such as compilers and assemblers, debuggers.
3. Availability of reliable source for the microcontroller.

Other members of 8051 family

There are two other members in 8051 family of Microcontroller. They are 8052 and 8031. 8052 has all standard features of 8051 as well as an extra 128 bytes of RAM and an extra timer. 8052 has 256 bytes of RAM and 3 timers. It also has 8K bytes of on chip program instead of 4K bytes. 8031 Microcontroller is often referred as a ROM less 8051 since it has 0K bytes of on chip ROM. To use this chip we must add external ROM to it. Two ports will be lost in the process of adding external ROM.

Comparison of 8051 family members

Feature	**8051**	**8052**	**8031**
ROM [on chip program space in bytes]	4K	8K	0K
RAM [bytes]	128	256	128
Timers	2	3	2
I/O pins	32	32	32
Serial port	1	1	1
Interrupt sources	6	8	6

Block diagram of 8051

8051 is the original member of 8051 family. Features of 8051 microcontroller are as shown below.

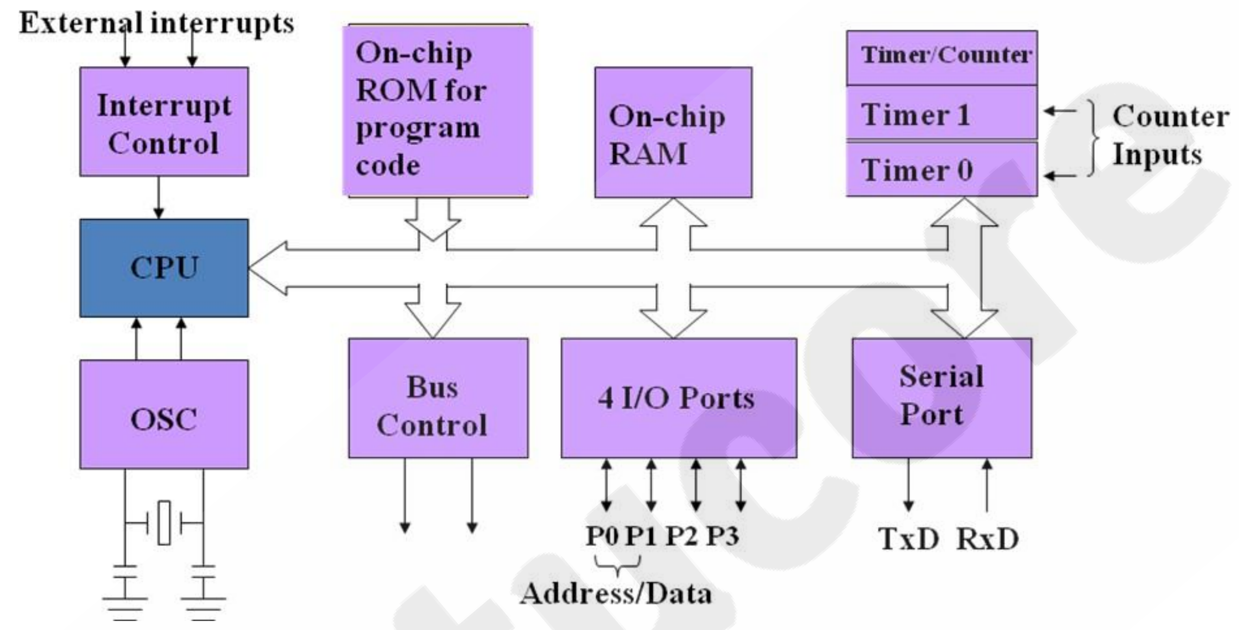


Fig 5. Architecture of 8051 Microcontroller

Salient Features

•Eight bit CPU with registers A (Accumulator) and B

•Sixteen bit Program counter (PC) and a data pointer (DPTR)

•8 Bit Program Status Word (PSW), 8 Bit Stack Pointer, 4K Code Memory

•Internal Memory of 128 Bytes, 32 I/O Pins arranged as 4 , 8 Bit ports

•Two 16 Bit Timer/Counter :T0, T1

•Full Duplex serial data receiver/transmitter

•Control Registers : TCON,TMOD,SCON,PCON,IP and IE

•Two External and Internal Interrupt sources

•Oscillator and clock circuits.

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.

Program Counter: It addresses program instruction bytes which are to be fetched from memory locations.

Data pointer: It contains two registers DPG and DPL. It is used to access internal or external data. It is under the control of program instructions and can be specified by 16 bit name.

8051 Clock and Instruction Cycle:

The heart of 8051 is the circuitry that generates the clock pulses by which all internal operations are synchronized. Pins XTAL1 and XTAL2 are provided for connecting resonator to form an oscillator. The crystal frequency is the basic internal frequency of the microcontroller. 8051 is designed to operate between 1MHz to 16MHz and generally operates with a crystal frequency 11.04962 MHz.

The oscillator formed by the crystal, capacitor and an on-chip inverter generates a pulse train at the frequency of the crystal. The clock frequency establishes the smallest interval to accomplish any simple instruction. The time taken to complete any instruction is called as machine cycle or instruction cycle. In 8051 one instruction cycle consists of 6 states or 12 clock cycles, instruction cycle is also referred as Machine cycle.

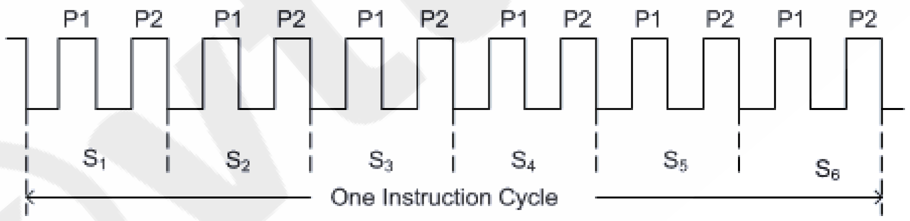


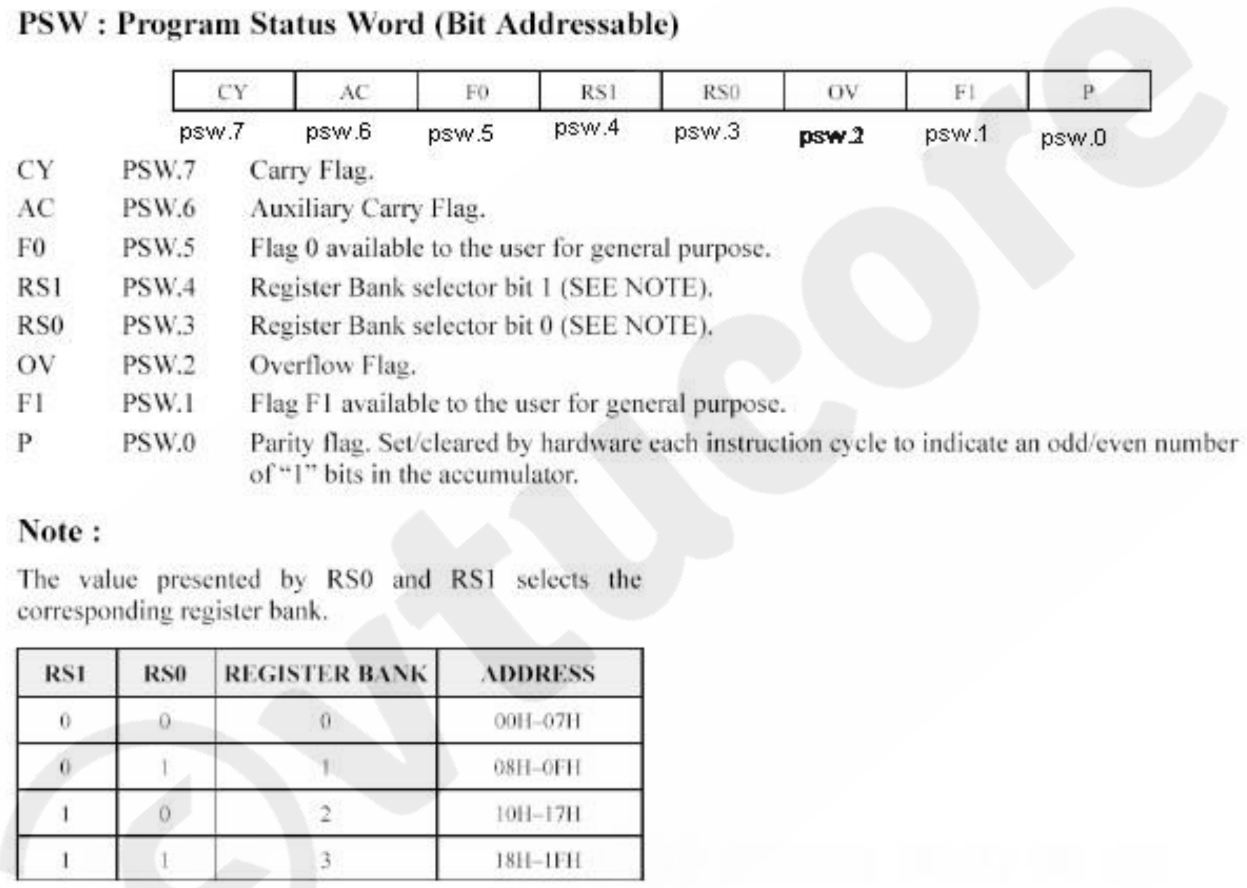
Fig. 6 Instruction cycle of 8051(Instruction cycle has six states (S_1 - S_6). Each state has two pulses (P1 and P2))

PSW and flag bits

Program Status word

It is an 8 bit register. Only 6 bits of it are used by 8051. Bits of PSW are shown below.

Fig 7: Program Status word



PSW : Program Status Word (Bit Addressable)

CY	AC	F0	RS1	RS0	OV	F1	P
psw.7	psw.6	psw.5	psw.4	psw.3	psw.2	psw.1	psw.0

CY	PSW.7	Carry Flag.
AC	PSW.6	Auxiliary Carry Flag.
F0	PSW.5	Flag 0 available to the user for general purpose.
RS1	PSW.4	Register Bank selector bit 1 (SEE NOTE).
RS0	PSW.3	Register Bank selector bit 0 (SEE NOTE).
OV	PSW.2	Overflow Flag.
F1	PSW.1	Flag F1 available to the user for general purpose.
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of "1" bits in the accumulator.

Note :

The value presented by RS0 and RS1 selects the corresponding register bank.

RS1	RS0	REGISTER BANK	ADDRESS
0	0	0	00H–07H
0	1	1	08H–0FH
1	0	2	10H–17H
1	1	3	18H–1FH

Carry Flag(C): This flag sets when there is a carry out from D7 bit. This flag bit is affected after 8 bit addition or subtraction. It can also be set to 0 or 1 directly by instruction such as SETB C and CLR C.

Auxiliary Carry Flag(AC): If there is a carry out from D3 to D4 during an ADD or SUB operation then this bit sets. Otherwise it is

8051 Register banks and Stacks

The collection of general purpose registers (R0-R7) is called as register banks, which accept one byte of data. The register bank is a part of the RAM memory in the microcontrollers, and it is used to store the program instructions.

A total of 32 bytes of RAM are set aside for the register banks and stack. These 32 bytes are divided into 4 banks of registers in which each bank has 8 registers, RO – R7. RAM locations from 0 to 7 are set aside for bank 0 of RO – R7 where RO is RAM location 0, Rl is RAM location 1, R2 is location 2, and so on, until memory location 7, which belongs to R7 of bank 0. The second bank of registers RO – R7 starts at RAM location 08H and goes upto location OFH. The third bank of RO – R7 starts at memory location 10H and goes upto location 17H. Finally, RAM locations 18H to 1FH are set aside for the fourth bank of RO – R7. The following figure shows how the 32 bytes are allocated into 4 banks:

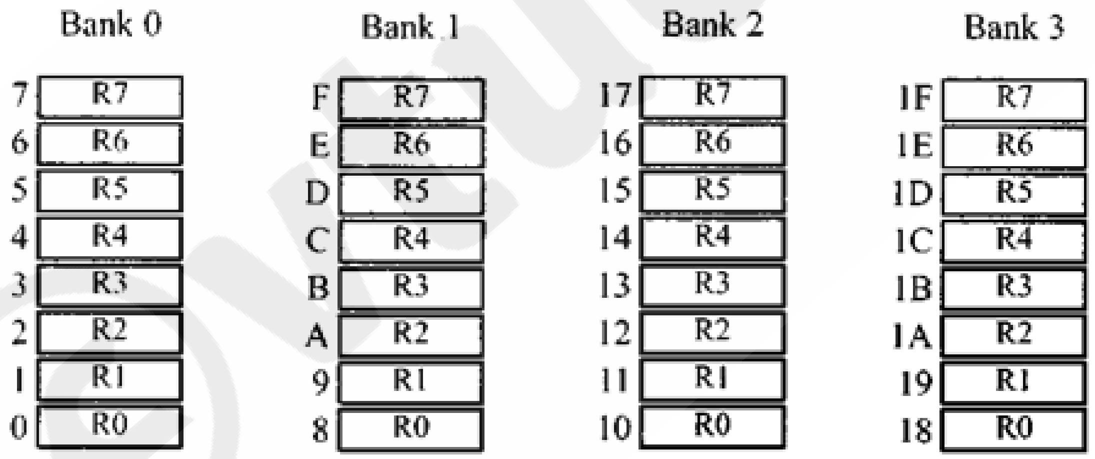


Fig 8. 8051 Register Banks and their RAM Addresses

As shown in above Figure, bank 1 uses the same RAM space as the stack. This is a major problem in programming the 8051. We must either not use register bank 1, or allocate another area of RAM for the stack.

Ex: MOV R0, #99H; /* Load R0 with 99H, RAM location 0H has the value 99H*/

Default register bank 0 is accessed when programming 8051. We can switch to other banks by the use of PSW (program status word) register. Bits D4 and D3 of the PSW are used to select the desired register bank.

The Stack and Stack pointer:

The stack refers to an area of internal RAM that is used in conjunction with certain opcodes to store and retrieve data quickly. The 8 bit Stack Pointer (SP) register is used by the 8051 to hold internal RAM address that is called the top of the stack.

When data is to be placed on the stack , the SP increments before storing data on the stack so that the stack grows up as data is stored. Whenever data is retrieved from the stack, the byte is read from the stack and then the SP decrements to point to the next available byte of stored data.

Operation of the Stack and Stack Pointer:

Operation of the stack is as shown below. The SP is set to 07 when the 8051 is reset and can be changed to any internal RAM address by the programmer. The stack is limited in height to the size of internal RAM. The stack can overwrite valuable data in register banks, bit addressable RAM and scratched pad RAM areas. It is programmer's responsibility to make it sure that the stack does not grow beyond predefined bounds. The stack is normally placed high in the internal RAM by an appropriate choice of the number placed in SP register, to avoid conflict with registers or RAM.

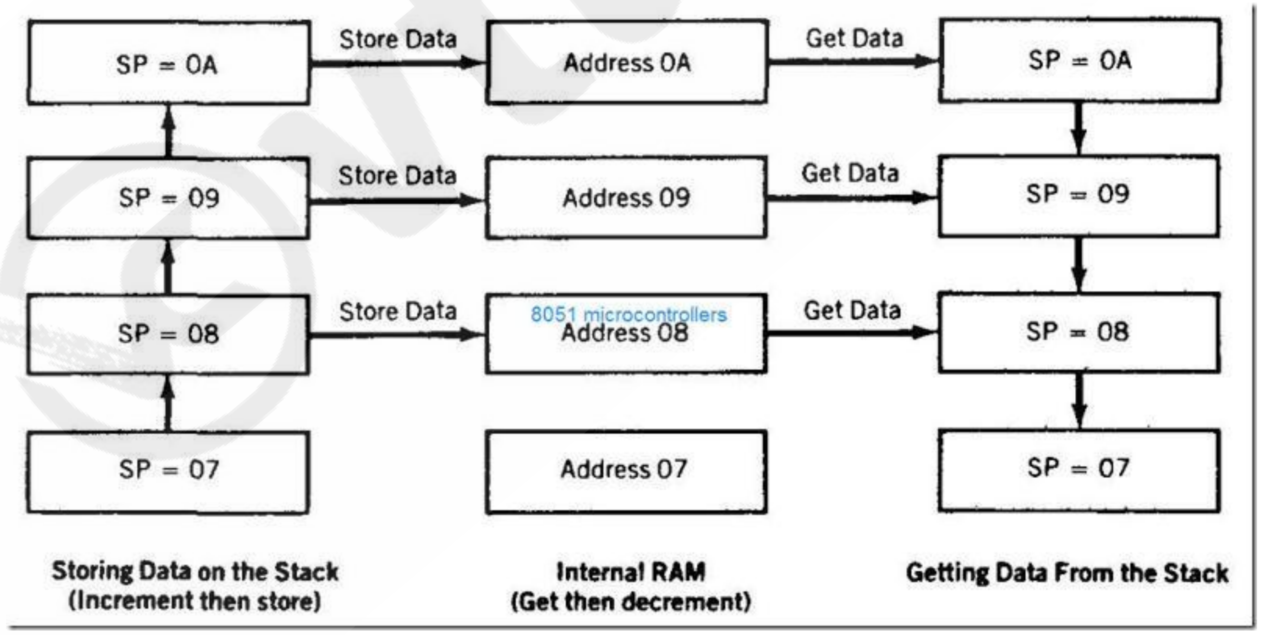


Fig 9: Stack operation

Internal Memory organization of 8051

A functioning computer memory for program code bytes, commonly in ROM, and RAM memory for variable data that can be altered as the program runs. Additional memory can be added externally using suitable circuits.

The 8051 has Harvard architecture which uses the same address in different memories for code and data The internal circuitry accesses the current memory based on the nature of operation in the program.

Internal RAM: The 128 bytes internal RAM is organized into 3 distinct areas.

1. 32 bytes from address 00h to 1fh that make up 32 working registers organized as 4 memory banks of 8 registers each. The 4 register banks are numbered 0 to 3 and are made up of 8 registers named R0 to R7. Each register can be addressed by name or by its RAM addresses. Thus R0 of bank3 is R0 (if bank3 is selected) or address 18h (where bank3 is selected) . Bits RS0 and RS1 in the PSW determine which bank of registers is currently in use at any time when program is running. Register banks not selected can be used as general purpose RAM. Bank0 is selected by default on reset..
2. A bit addressable area of 16 bytes occupies RAM byte addresses 20h to 2fh, forming total of 128 bits. An addressable bit may be specified by its bit address of 00h to 7fh or 8 bits may form any byte address from 20h to 2fh.For example bit address 4fh is also bit 7 of byte address 29h. Addressable bits are useful when the program need only remember a binary event.
3. A general purpose RAM area above the bit area from 30h to 7f h, addressable as byte.

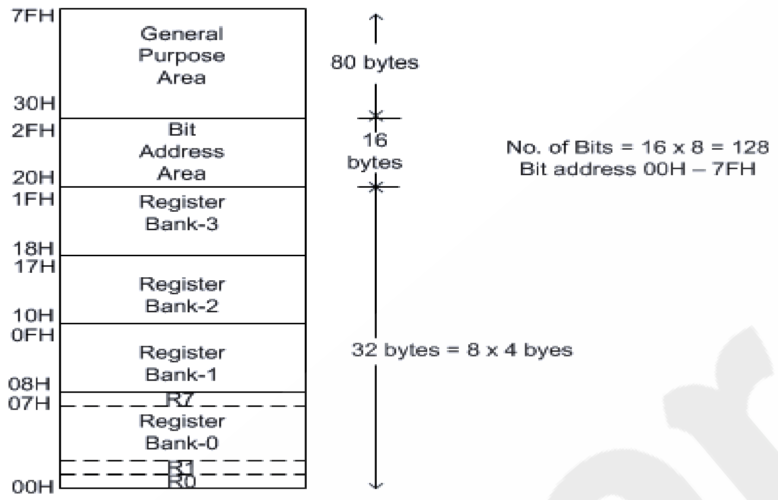


Fig.10. Internal RAM structure

Internal ROM

8051 is organized so that data memory and program code memory can be two entirely different physical memory entities. Each has the same address ranges. The internal program ROM occupies code address space 000h to 0fffh. The PC is normally used to address program code bytes from address 0000h to ffffh. Program addresses higher than offfh which exceed the internal ROM capacity will cause the 8051 to automatically fetch code bytes from external memory, addresses 00h to ffffh by connecting the external access pin (EA) to ground. 8051 microcontroller supports 4K bytes internal ROM. Program addresses higher than 0FFFH which excedes internal ROM capacity will cause 8051 to automatically fetch code bytes from external program memory.

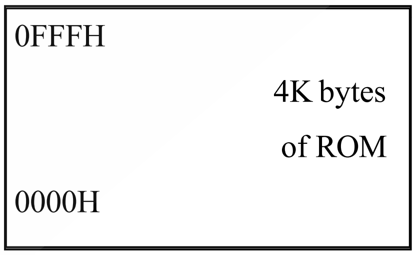


Fig 11: Internal ROM

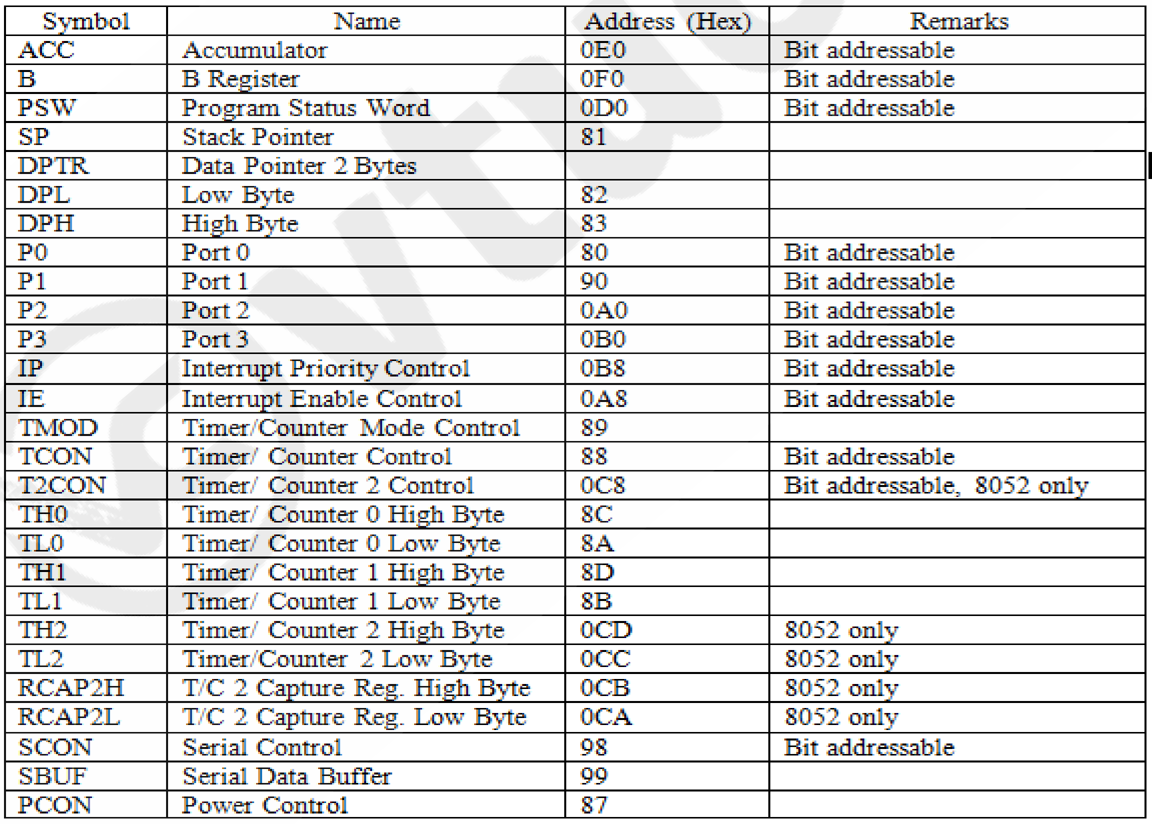
External Memory

8051 supports external memory of 64K bytes of data memory and 64K bytes of program memory. External memory is interfaced and accessed by using 16 address lines available on port 0 and port 2. Address and data lines are multiplexed on port 0.

Types of Special Function Registers (SFRs)

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. Some SFRs are also bit addressable as is the case for the bit area of RAM.

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. Some of these registers are bit addressable. The detailed map of various registers is shown in the following figure.



Symbol	Name	Address (Hex)	Remarks
ACC	Accumulator	0E0	Bit addressable
B	B Register	0F0	Bit addressable
PSW	Program Status Word	0D0	Bit addressable
SP	Stack Pointer	81	
DPTR	Data Pointer 2 Bytes		
DPL	Low Byte	82	
DPH	High Byte	83	
P0	Port 0	80	Bit addressable
P1	Port 1	90	Bit addressable
P2	Port 2	0A0	Bit addressable
P3	Port 3	0B0	Bit addressable
IP	Interrupt Priority Control	0B8	Bit addressable
IE	Interrupt Enable Control	0A8	Bit addressable
TMOD	Timer/Counter Mode Control	89	
TCON	Timer/ Counter Control	88	Bit addressable
T2CON	Timer/ Counter 2 Control	0C8	Bit addressable, 8052 only
TH0	Timer/ Counter 0 High Byte	8C	
TL0	Timer/ Counter 0 Low Byte	8A	
TH1	Timer/ Counter 1 High Byte	8D	
TL1	Timer/ Counter 1 Low Byte	8B	
TH2	Timer/ Counter 2 High Byte	0CD	8052 only
TL2	Timer/Counter 2 Low Byte	0CC	8052 only
RCAP2H	T/C 2 Capture Reg. High Byte	0CB	8052 only
RCAP2L	T/C 2 Capture Reg. Low Byte	0CA	8052 only
SCON	Serial Control	98	Bit addressable
SBUF	Serial Data Buffer	99	
PCON	Power Control	87	

Fig 12: SFR Map

Pin configuration of 8051

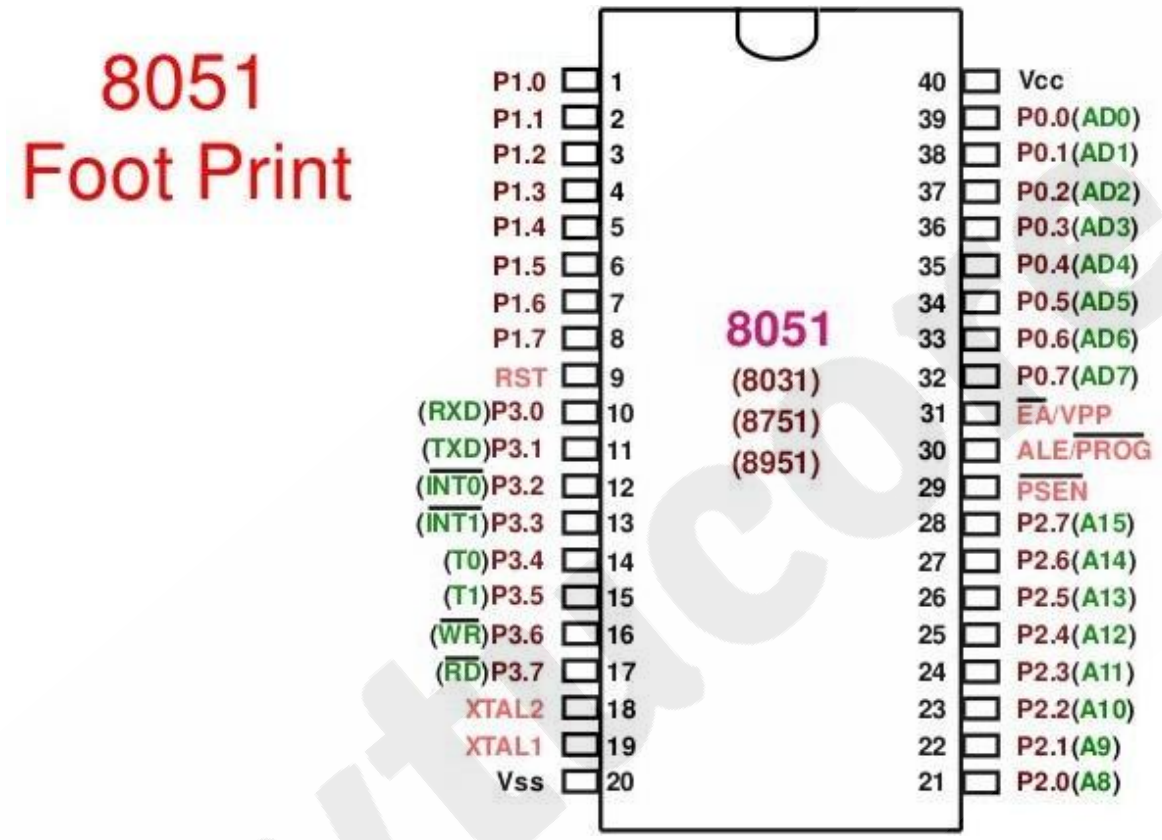


Fig 13. Pin configuration of 8051

Pin Description

- **Pins 1 to 8** − These pins are known as Port 1. It is a general purpose port. This port doesn't serve any other functions. It is internally pulled up, bi-directional I/O port.
- **Pin 9** − It is a RESET(RST) pin, which is used to reset the microcontroller to its initial values.
- **Pins 10 to 17** − These pins are known as Port 3. This port serves some functions like interrupts, timer input, control signals, serial communication signals RxD and TxD, etc.
- **Pins 18 & 19** − These pins are used for interfacing an external crystal to get the system clock.
- **Pin 20** − This pin provides the power supply to the circuit.

- **Pins 21 to 28** − These pins are known as Port 2. It serves as I/O port. Higher order address bus signals are also multiplexed using this port.
- **Pin 29** − This is PSEN pin which stands for Program Store Enable. It is used to read a signal from the external program memory.
- **Pin 30** − This is EA pin which stands for External Access input. It is used to enable/disable the external memory interfacing.
- **Pin 31** − This is ALE pin which stands for Address Latch Enable. It is used to demultiplex the address-data signal of port.
- **Pins 32 to 39** − These pins are known as Port 0. It serves as I/O port. Lower order address and data bus signals are multiplexed using this port.
- **Pin 40 −** This pin is used to provide power supply to the circuit.

IO Port Usage in 8051

The four 8-bit I/O ports P0, P1, P2 and P3 each uses 8 pins. All the ports upon RESET are configured as input, ready to be used as input ports. When the first 0 is written to a port, it becomes an output port. To reconfigure it as an input, 1 must be sent to the port. To use any of these ports as an input port, it must be programmed. It can be used for input or output, each pin must be connected externally to a 10K ohm pull-up resistor. This is due to the fact that P0 is an open drain, unlike P1, P2, and P3. Open drain is a term used for MOS chips in the same way that open collector is used for TTL chips.

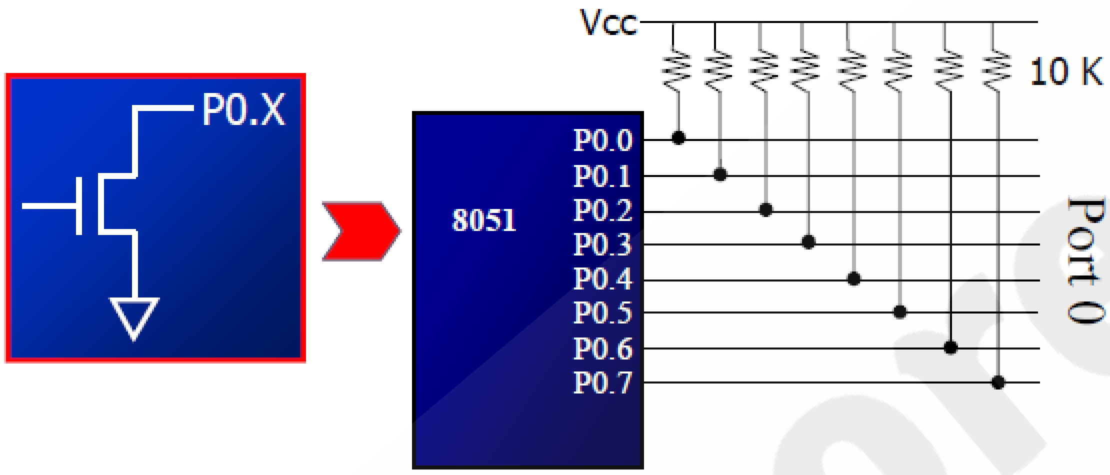


Fig 14: Port 0 with Pull up registers

The following code will continuously send out to port 0 the alternating value 55H and AAH

```
BACK: MOV A,#55H
          MOV P0,A
          ACALL DELAY
          MOV A,#0AAH
          MOV P0,A
          ACALL DELAY
          SJMP BACK
```

Port 0 as input

In order to make port 0 an input, the port must be programmed by writing 1 to all the bits. Port 0 is configured first as an input port by writing 1s to it, and then data is received from that port and sent to P1

```
MOV A,#0FFH         ;A=FF hex
MOV P0,A            ;make P0 an i/p port ;by writing it all 1s
BACK: MOV A,P0      ;get data from P0
MOV P1,A            ;send it to port 1
SJMP BACK
```

Dual role of Port 0

Port 0 is also designated as AD0-AD7, allowing it to be used for both address and data. When connecting an 8051/31 to an external memory, port 0 provides both address and data.

Port 1 can be used as input or output

In contrast to port 0, this port does not need any pull-up resistors since it already has pull-up resistors internally. Upon reset, port 1 is configured as an input port.

The following code will continuously send out to port 0 the alternating value 55H and AAH

```
        MOV A,#55H
BACK: MOV P1,A
        ACALL DELAY
        CPL A
        SJMP BACK
```

To make port 1 an input port, it must be programmed as such by writing 1to all its bits.

Port 1 is configured first as an input port by writing 1s to it, then data is received from that port and saved in R7 and R5

```
MOV A,#0FFH         ;A=FF hex
MOV P1,A            ;make P1 an input port ;by writing it all 1s
MOV A,P1            ;get data from P1
MOV R7,A            ;save it to in reg R7
ACALL DELAY         ;wait
MOV A,P1            ;another data from P1
MOV R5,A            ;save it to in reg R5
```

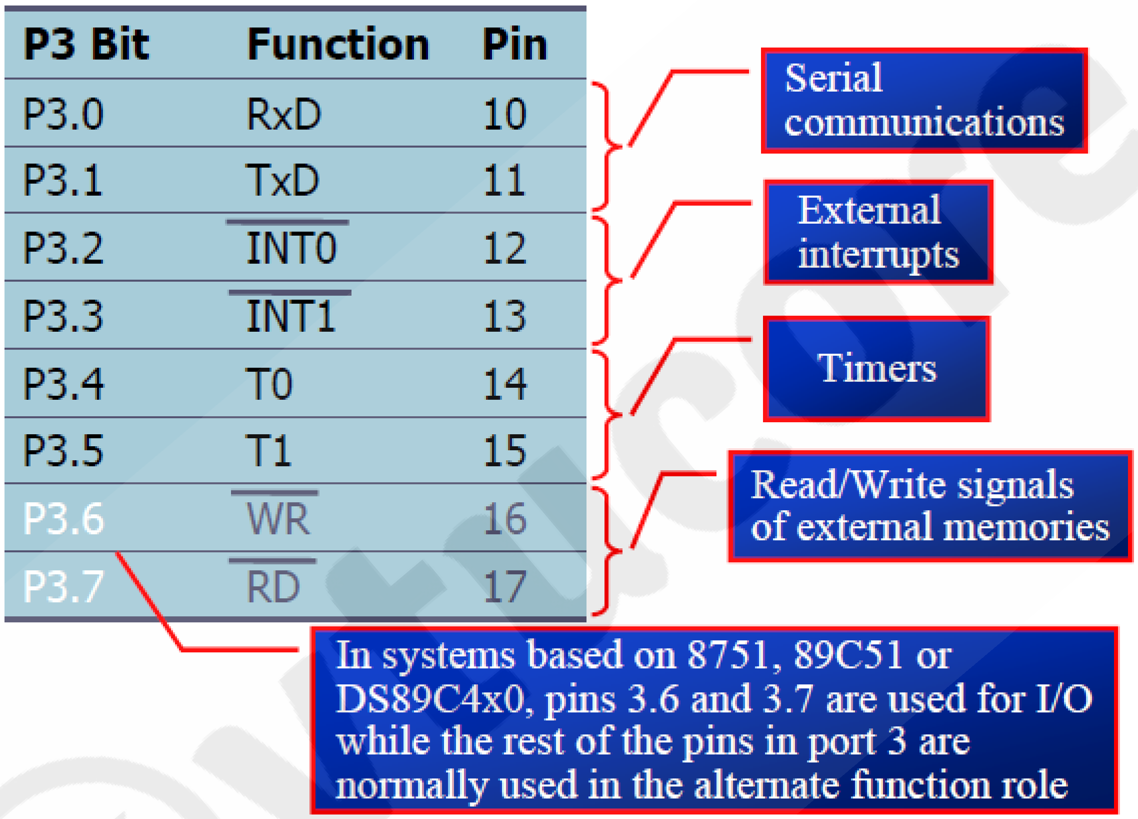
Port 2 can be used as input or output

Just like P1, port 2 does not need any pullup resistors since it already has pull-up resistors internally. Upon reset, port 2 is configured as an input port. To make port 2 an input port, it must be programmed as such by writing 1 to all its bits. In many 8051-based systems, P2 is used as simple I/O. In 8031-based systems, port 2 must be used along with P0 to provide the 16-bit address for the external memory. Port 2 is also designated as A8 – A15, indicating its dual function. Port 0 provides the lower 8 bits via A0 – A7.

Port 3 can be used as input or output

Port 3 does not need any pull-up resistors. Port 3 is configured as an input port upon reset.Port 3 has the additional function of providing some extremely important signals.

Fig 15: Port 3 alternate functions



P3 Bit	Function	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2	$\overline{\text{INT0}}$	12
P3.3	$\overline{\text{INT1}}$	13
P3.4	T0	14
P3.5	T1	15
P3.6	$\overline{\text{WR}}$	16
P3.7	$\overline{\text{RD}}$	17

Serial communications
External interrupts
Timers
Read/Write signals of external memories

In systems based on 8751, 89C51 or DS89C4x0, pins 3.6 and 3.7 are used for I/O while the rest of the pins in port 3 are normally used in the alternate function role

Memory Address Decoding

The CPU provides the address of the data desired, but it is the job of the decoding circuitry to locate the selected memory block . Memory chips have one or more pins called CS (chip select), which must be activated for the memory contents to be accessed. Sometimes the chip select is also referred as chip enable (CE).

In connecting a memory chip to the CPU, note the following points

1. The data bus of the CPU is connected directly to the data pins of the memory chip

2. Control signals RD (read) and WR (memory write) from the CPU are connected to the OE (output enable) and WE (write enable) pins of the memory chip.

3. In the case of the address buses, while the lower bits of the address from the CPU go directly to the memory chip address pins, the upper ones are used to activate the CS pin of the memory chip.

Normally memories are divided into blocks and the output of the decoder selects a given memory block

1. Using simple logic gates
2. Using the 74LS138
3. Using programmable logics.

1. Simple Logic gate address decoder

The simplest way of decoding circuitry is the use of NAND or other gates. The fact that the output of a NAND gate is active low, and that the CS pin is also active low makes them a perfect match.

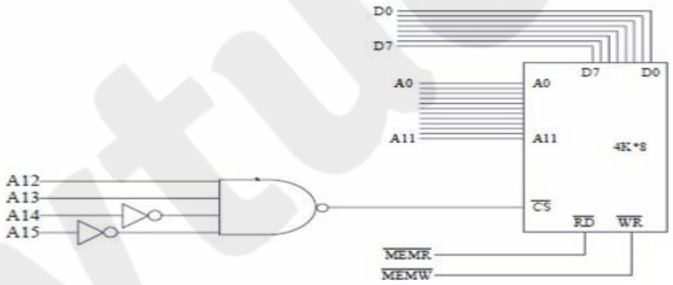
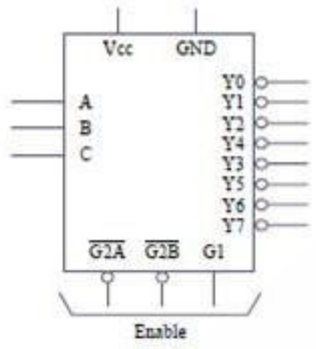


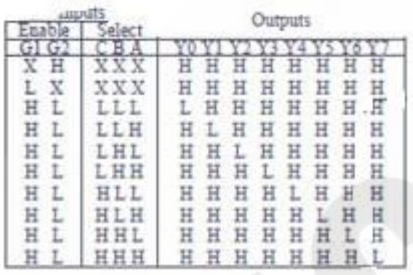
Fig 16: Logic gate as decoder

2. Using the 74LS138 3-8 decoder

This is one of the most widely used address decoder. The 3 inputs A, B, and C generate 8 active low outputs Y0 – Y7. Each Y output is connected to CS of a memory chip, allowing control of 8 memory blocks by a single 74LS138. In the 74LS138, where A, B, and C select which output is activated, there are three additional inputs, G2A, G2B, and G1. G2A and G2B are both active low, and G1 is active high. If any one of the inputs G1, G2A, or G2B is not connected to an address signal, they must be activated permanently either by Vcc or ground, depending on the activation level.



Function Table



Inputs			Outputs							
Enable		Select								
G1	G2	C B A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X X X	H	H	H	H	H	H	H	H
L	X	X X X	H	H	H	H	H	H	H	H
H	L	L L L	L	H	H	H	H	H	H	H
H	L	L L H	H	L	H	H	H	H	H	H
H	L	L H L	H	H	L	H	H	H	H	H
H	L	L H H	H	H	H	L	H	H	H	H
H	L	H L L	H	H	H	H	L	H	H	H
H	L	H L H	H	H	H	H	H	L	H	H
H	L	H H L	H	H	H	H	H	H	L	H
H	L	H H H	H	H	H	H	H	H	H	L

Figure: 74LS138

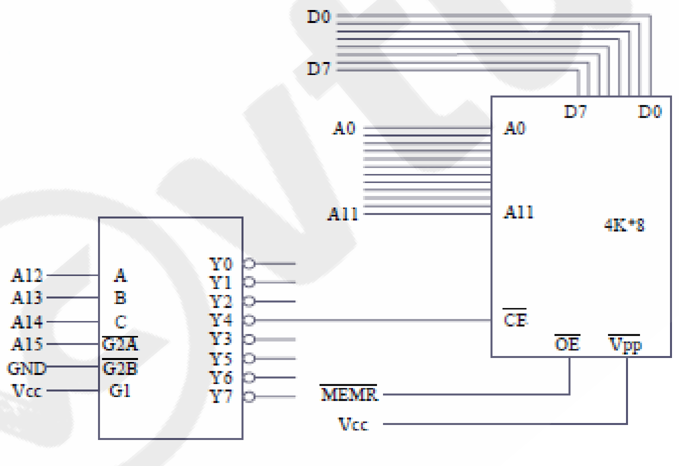


Fig 17: 74LS138 as decoder

Find the address range for the Following. (a) Y4 (b) Y2 and (c) Y7.

(a) The address range for Y4 is calculated as follows.

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1

The above shows that the range for Y4 is 4000H to 4FFFH. We notice that A15 must be 0 for the decoder to be activated. Y4 will be selected when A14 A13 A12 = 100 (4 in binary). The remaining A11-A0 will be 0 for the lowest address and 1 for the highest address.

(b) The address range for Y2 is 2000H to 2FFFH.

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1

(c) The address range for Y7 is 7000H to 7FFFH.

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6		A5	A4	A3	A2	A1	A0
0	1	1	1	0	0	0	0	0	0		0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1

4. Using programmable logic as an address decoder

Other widely used decoders are programmable logic chips such as PAL and GAL chips. One disadvantage of these chips is that one must have access to a PAL/GAL software and burner, whereas the 74LS138 needs neither of these. The advantage of these chips is that they are much more versatile since they can be programmed for any combination of address ranges.

8031/51 interfacing with external ROM and RAM

8031/51 interfacing with external ROM

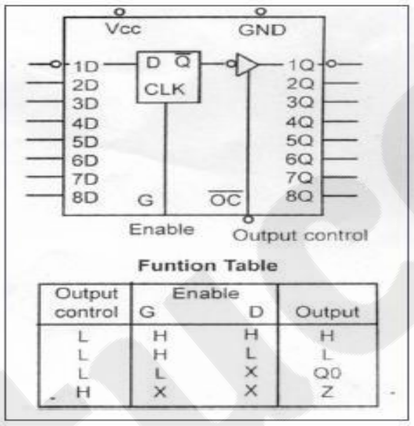
The 8031 chip is a ROM less version of the 8051. It is exactly like any member of the 8051 family as far as executing the instructions and features are concerned, but it has no on-chip ROM. To make the 8031 execute 8051 code, it must be connected to external ROM memory containing the program code. 8031 is ideal for many systems where the on-chip ROM of 8051 is not sufficient, since it allows the program size to be as large as 64K bytes.

For 8751/89C51/DS5000-based system, we connected the EA pin to Vcc to indicate that the program code is stored in the microcontroller’s on-chip ROM. To indicate that the program code is stored in external ROM, this pin must be connected to GND.

Since the PC (program counter) of the 8031/51 is 16-bit, it is capable of accessing up to 64K bytes of program code. In the 8031/51, port 0 and port 2 provide the 16-bit address to access external memory. P0 provides the lower 8 bit address A0 – A7, and P2 provides the upper 8 bit address A8 – A15. P0 is also used to provide the 8-bit data bus D0 – D7. P0.0 – P0.7 are used for both the address and data paths address/data multiplexing.

ALE (address latch enable) pin is an output pin for 8031/51. ALE = 0, P0 is used for data path, ALE = 1, P0 is used for address path

To extract the address from the P0 pins we connect P0 to a 74LS373 and use the ALE pin to latch the address.



Output control	Enable G	D	Output
L	H	H	H
L	H	L	L
L	L	X	Q0
H	X	X	Z

Fig 18: 74LS373 D Latch

Normally ALE = 0, and P0 is used as a data bus, sending data out or bringing data in. Whenever the 8031/51 wants to use P0 as an address bus, it puts the addresses A0 – A7 on the P0 pins and activates ALE = 1.

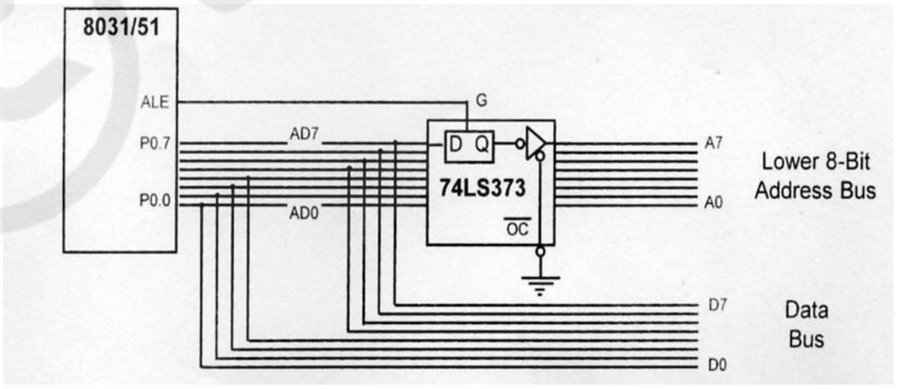


Fig 19: Address/Data multiplexing

PSEN (program store enable) signal is an output signal for the 8031/51microcontroller and must be connected to the OE pin of a ROM containing the program code. It is important to emphasize

the role of EA and PSEN when connecting the 8031/51 to external ROM. When the EA pin is connected to GND, the 8031/51 fetches opcode from external ROM by using PSEN.
The connection of the PSEN pin to the OE pin of ROM. In systems based on the 8751/89C51/ DS5000 where EA is connected to Vcc, these chips do not activate the PSEN pin. This indicates that the on-chip ROM contains program code.

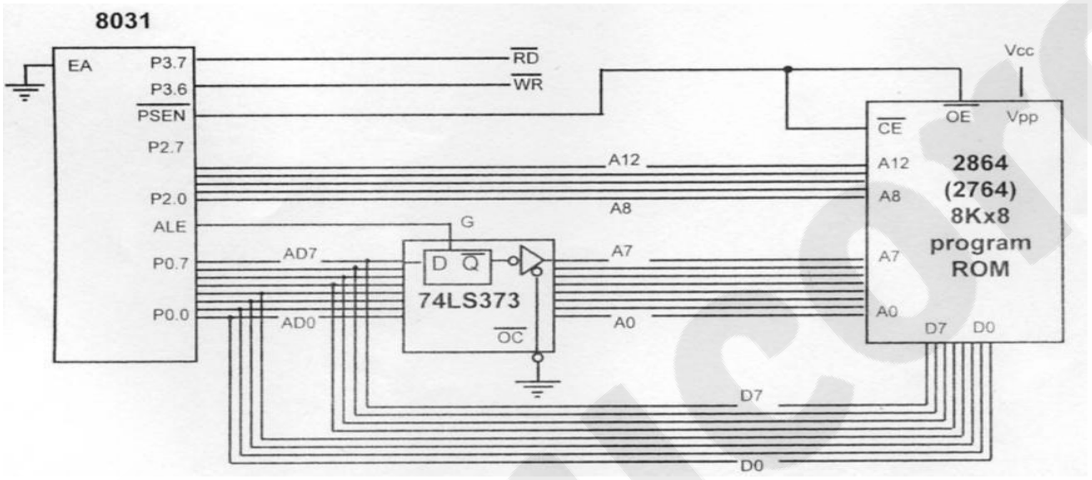


Fig 20: Data , address and control buses for the 8031/51

In an 8751 system we could use onchip ROM for boot code and an external ROM will contain the user’s program. We still have EA = Vcc, Upon reset 8051 executes the on-chip program first, then when it reaches the end of the on-chip ROM, it switches to external ROM for rest of program.

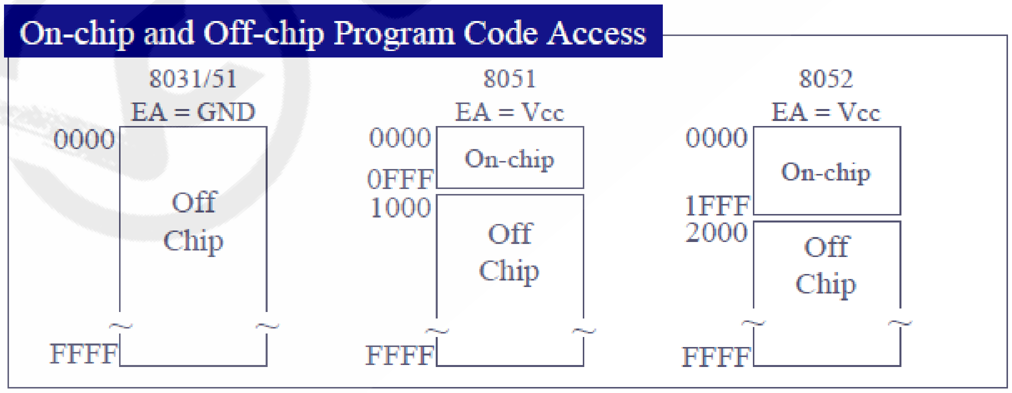


Fig 21: On chip and off chip program code access

8051 data memory space

The 8051 has 128K bytes of address space. 64K bytes are set aside for program code. Program space is accessed using the program counter (PC) to locate and fetch instructions. In some example we placed data in the code space and used the instruction MOVC A,@A+DPTR to get data, where C stands for code. The other 64K bytes are set aside for data. The data memory space is accessed using the DPTR register and an instruction called MOVX, where X stands for external. The data memory space must be implemented externally. We use RD to connect the 8031/51 to external ROM containing data. For the ROM containing the program code,
PSEN is used to fetch the code.

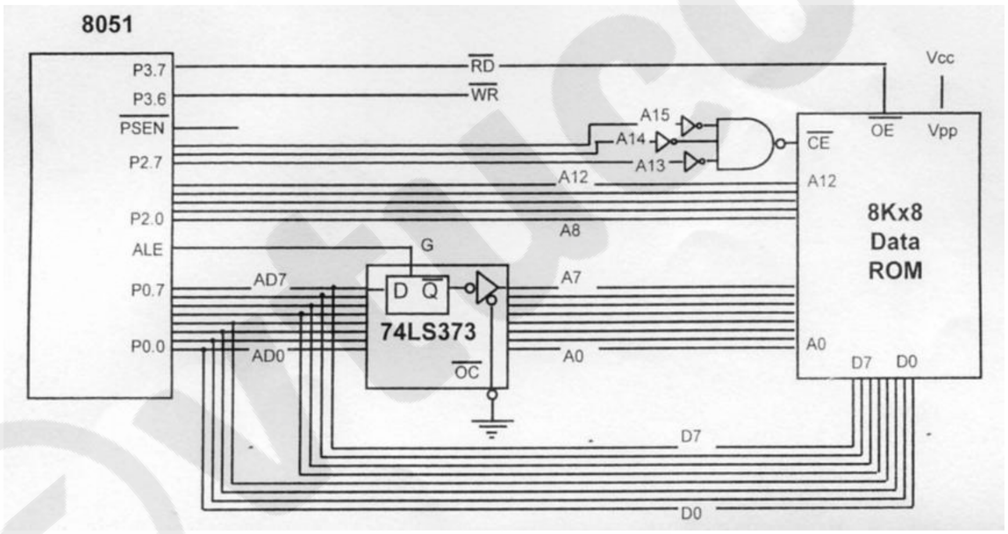


Fig 22: 8051 connection to external data ROM

MOVX is a widely used instruction allowing access to external data memory space. MOVX A,@DPTR instruction is used to bring externally stored data into the CPU.

8031 Connection to External Data ROM and External Program ROM

Design of an 8031-based system with 8K bytes of program ROM and 8K bytes of data ROM is as shown below. For program ROM, PSEN is used to activate both OE and CE. For data ROM, we use RD to active OE, while CE is activated by a Simple decoder.

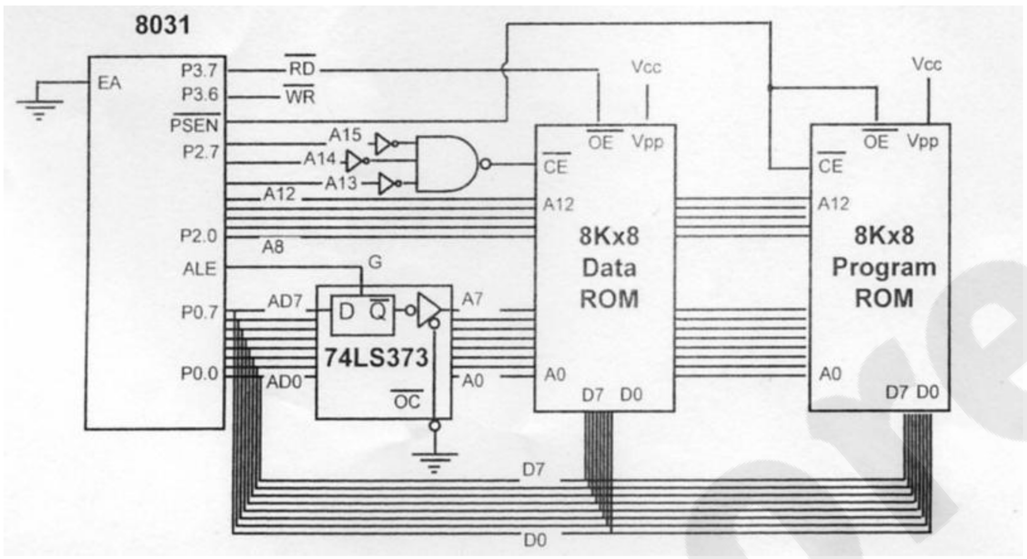


Fig 23: 8031 Connection to External Data ROM and External Program ROM

8051 Data memory space

To connect the 8051 to an external SRAM, we must use both RD (P3.7) and WR (P3.6) pins. In writing data to external data RAM, we use the instruction MOVX @DPTR, A. In some applications we need a large amount of memory to store data. The 8051 can support only 64K bytes of external data memory since DPTR is 16-bit.

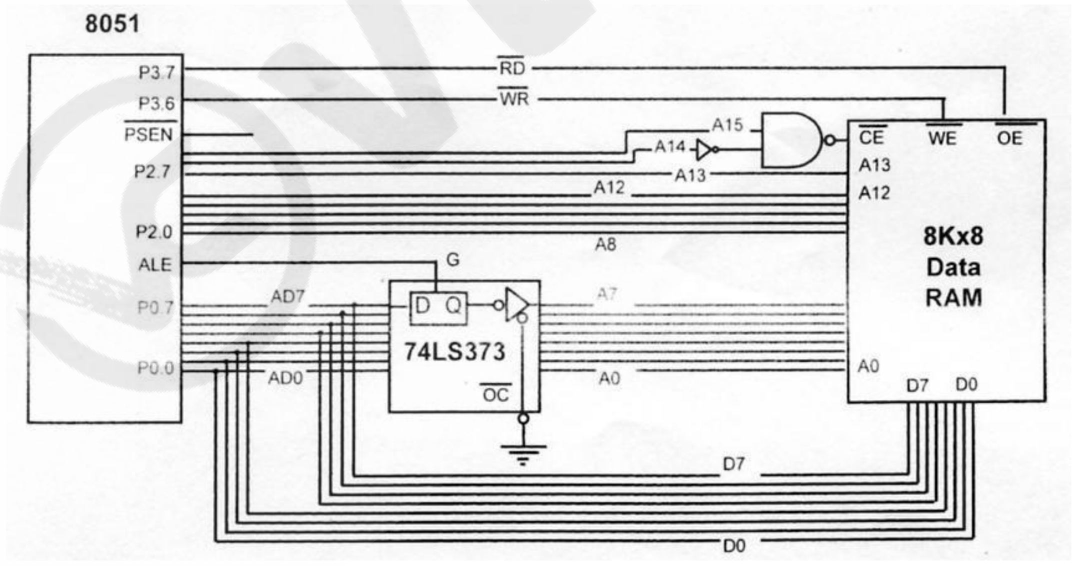


Fig 24: 8051 Connection to External Data RAM

External program memory is fetched if either of the following two conditions are satisfied.

1. EA (Enable Address) is low. The microcontroller by default starts searching for program from external program memory.
2. PC is higher than FFFH for 8051 or 1FFFH for 8052.
3. PSEN tells the outside world whether the external memory fetched is program memory or data memory. EA is user configurable. PSEN is processor controlled.

Some typical use of code/program memory access: External program memory can be not only used to store the code, but also for lookup table of various functions required for a particular application. Mathematical functions such as Sine, Square root, Exponential, etc. can be stored in the program memory (Internal or external) and these functions can be accessed using MOVC instruction.

Addressing Modes

8051 Addressing modes

Addressing mode is a way to address an operand. Operand means the data we are operating upon. It can be a direct address of memory, it can be register names, it can be any numerical data etc.

MOV A,#6AH

Here the data 6A is the operand, often known as source data. When this instruction is executed, the data 6AH is moved to accumulator A.

There are 5 different ways to execute this instruction and hence we say, we have got 5 addressing modes for 8051. They are **1.** Immediate addressing mode

2. Direct addressing mode
3. Register direct addressing mode
4. Register indirect addressing mode
5. indexed addressing mode.

1. Immediate Addressing Mode

In this mode data can be specified as a part of instruction to get data easily to a destination. Here source operand is constant. Here mnemonic for immediate data is pound sign(#).

This can copy immediate numbers form the opcode into registers(R0 to R7), A and DPTR.

Ex: MOV R0,#08H

MOV DPTR,#1234H

Let's begin with an example. **MOV A, #6AH**

In general we can write MOV A, #data

This addressing mode is named as "**immediate"** because it transfers an 8-bit data immediately to the accumulator (destination operand). The opcode for MOV A, # data is 74H. The opcode is saved in program memory at 0202 address. The data 6AH is saved in program memory 0203. This instruction is of two bytes and is executed in one cycle. So after the execution of this instruction, program counter will add 2 and move to 0204 of program memory.

Note: The '#' symbol before 6AH indicates that operand is a data (8 bit). If '#' is not present then the hexadecimal number would be taken as address.

2. Direct Addressing Mode

This is another way of addressing an operand. Here the address of the data (source data) is given as operand. Complete 128 bytes of internal RAM and SFR's may be addressed directly using single byte address assigned to each RAM location.

Ex: MOV A, 50H Load data stored in 50H into Accumulator

ADD A, 65H Adds the content of accumulator with content of 65H memory location.

3. Register Addressing Mode

In this addressing mode we use the register name directly (as source operand). **MOV A, R4.** At a time registers can take value from R0,R1…to R7. You may already know there are 32 such registers. There are 4 register banks named 0,1,2 and 3. Each bank has 8 registers named from R0 to R7. At a time only one register bank can be selected. Selection of register bank is made possible through a Special Function Register (SFR) named Processor Status Word (PSW). A data move does not alter the contents of data source address. Contents of destination address are replaced by source address contents.

Ex: MOV A,R4: Move the content of register R4 into accumulator

MOV R5,A: Move the content of accumulator into register R5

4. Indirect Addressing Mode

In this addressing mode, address of the data (source data to transfer) is given in the register operand. It uses a register to hold the actual address that will finally be used in data move. The register itself is not an address. R0 and R1 registers are used as pointer registers An example is shown below. To fetch of RAM location indirectly.

The instruction with indirect addresses uses @ sign.

Ex: MOV A, @R0: Load the content pointed by RAM location into Accumulator. Here the value inside

R0 is considered as an address, which holds the data to be transferred to accumulator. If R0 holds the value 20H, and we have a data 2F H stored at the address 20H, then the value 2FH will get transferred to accumulator after executing this instruction.

ADD A,@R1: Adds the content of accumulator with content pointed by RAM location.

Note: Only R0 and R1 are allowed to form a register indirect addressing instruction. In other words programmer can must make any instruction either using @R0 or @R1.

5. Indexed Addressing Mode

This mode uses a base register either as program counter or data pointer and an offset as accumulator in forming effective address for JMP or MOVC instruction. Jump table and lookup tables are easily created using indexed addressing.

MOVC A, @A+DPTR and MOVC A, @A+PC

where DPTR is data pointer and PC is program counter (both are 16 bit registers).

Lets take the first example.

MOVC A, @A+DPTR

The source operand is @A+DPTR and we know we will get the source data (to transfer) from this location. It is nothing but adding contents of DPTR with present content of accumulator. This addition will result a new data which is taken as the address of source data (to transfer). The data at this address is then transferred to accumulator.

The opcode for the instruction is 93H. DPTR holds the value 01FE, where 01 is located in DPH (higher 8 bits) and FE is located in DPL (lower 8 bits). Accumulator now has the value 02H. A 16 bit addition is performed and now 01FE H+02 H results in 0200 H. What ever data is in 0200 H will get transferred to accumulator. The previous value inside accumulator (02H) will get replaced with new data from 0200H.

This is a 1 byte instruction with 2 cycles needed for execution. .

The other example **MOVC A, @A+PC** works the same way as above example. The only difference is, instead of adding DPTR with accumulator, here data inside program counter (PC) is added with accumulator to obtain the target address.