Microcontroller (EEC421)

Lecture 5

Dr. Islam Mohamed

Electrical Engineering Department
Shoubra Faculty of Engineering, Benha University
Islam.ahmed@fen.bu.edu.eg

- **Lecture 5: Moving Data**

- ✓ INTRODUCTION
- ✓ MOVING DATA
- ✓ Addressing Modes
  - I. Immediate Addressing Mode
  - II. Register Addressing Mode
  - III. Direct Addressing Mode
  - IV. Indirect Addressing Mode
- ✓ External Data Moves
- ✓ Code Memory Read-Only Data Moves 51
- ✓ PUSH and POP Opcodes
- ✓ Data Exchanges

INTRODUCTION

- 8051 Instruction set indicates that:

✓ The ALU in combination with the registers can be controlled by binary operational codes to perform arithmetic operations.

✓ Can transfer data inside the CPU.

✓ Can perform Logical and arithmetic manipulations.

✓ Can make decisions based on the manipulation results.

✓ Can move data into and out of the MCU.

INTRODUCTION

- Programming Steps

✓ Step (1): Define the problem to be solved.

✓ Step (2): Solution Plan.

✓ Step (3): Draw Flowchart.

✓ Step (4): Program code.

✓ Step (5): Check the results.

INTRODUCTION

- Programming Steps

✓ Step (1): Define the problem to be solved.

(i) Where are the inputs to the program?

- ❖ Direct data is available.
- ❖ Data is stored in registers R1-R7 of the selected register bank.
- ❖ Data is stored in memory location.
- ❖ Data is stored in memory location whose address is pointed by registers R1-R7 of selected register bank.

(ii) What is the operation to be perform? A+B , A*B, A-B, …….

(iii) Where you want the output to be display or store?

INTRODUCTION

- Programming Steps

✓ Step (2): Solution Plan.

(i) How are you taking input data?

- ❖ Sensors
- ❖ Bluetooth
- ❖ Keyboard, ……

(ii) Which method you are using to solve the problem?

(iii) What are steps in this method?

(iii) How will you output the results?

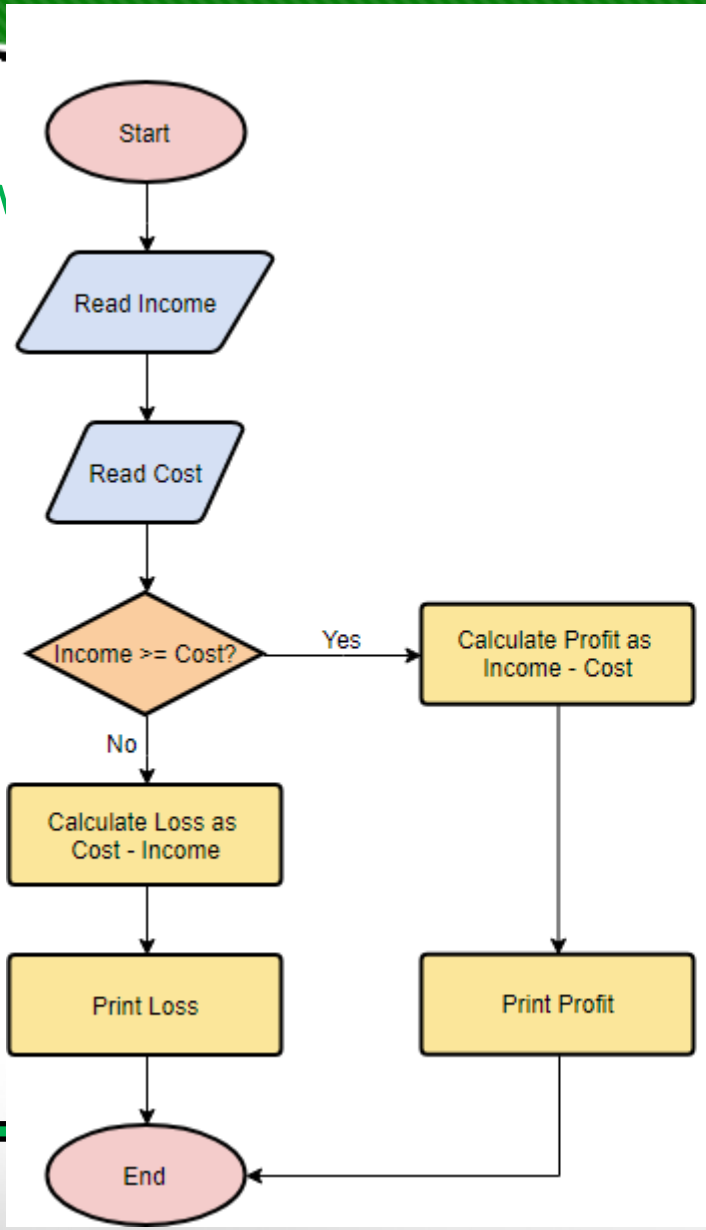
INTRODUCTION

- Programming Steps
  - ✓ Step (3): Draw Flow

Process

Start/End

- Example:



Find the profit/loss when income = 1,000, cost = 800

Start

Income = 1,000

Cost = 800

1,000 > 800? (Yes) Profit = 1,000 - 800

Profit = 200

End

INTRODUCTION

- Programming Steps

✓ Step (4): Program.

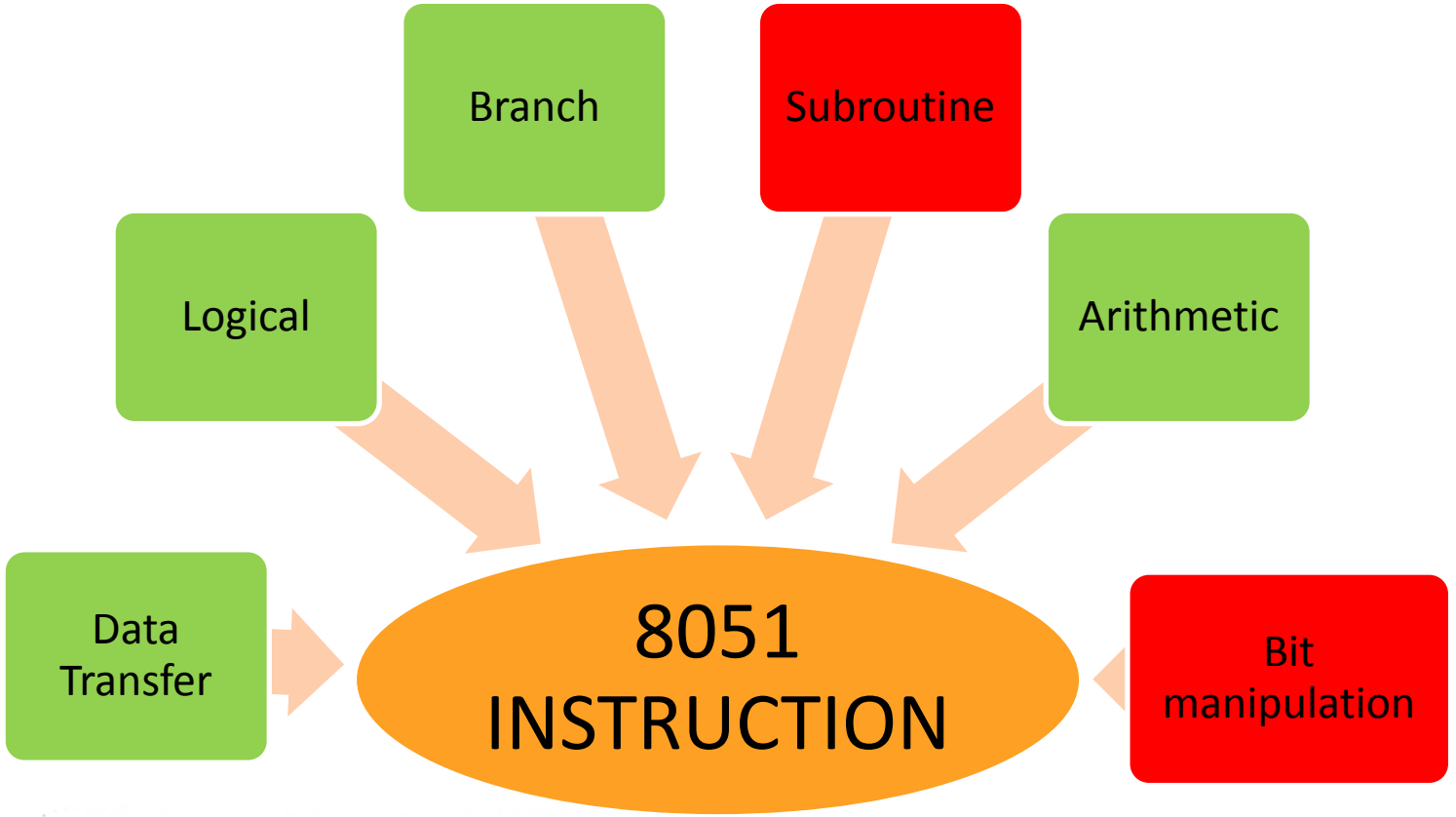
❖ Go on putting the instructions instead of flowchart blocks

✓ Step (5): Check the results. .

❖ Find the codes for instructions feed in 8051 MCU then

execute the program (get the results)

INTRODUCTION



Data Transfer Instructions

- Perform the operation of transferring the data from one location to another location.
- The location from which the data is transferred is called “the source” location.
- The location to which place the data is placed that is called “the destination” location.
- These instructions are just copy the data from the source to destination but the data aren’t deleted from the source.
- Example: MOV A,R2 ➔ Copy the data from R2 to the accumulator (A)

Data Transfer Instructions

- There are 28 distinct mnemonics that copy data from a source to a destination, they may be divided into the following three main types:

1) MOV destination, source
2) PUSH source or POP destination
3) XCH destination, source

- Four addressing modes are used to access data:

1) Immediate addressing mode
2) Register addressing mode
3) Direct addressing mode
4) Indirect addressing mode

Data Transfer Instructions

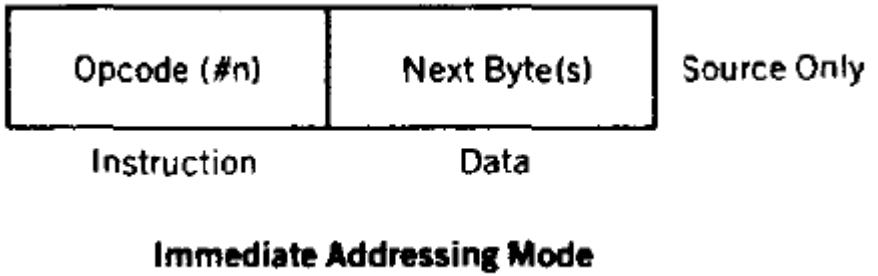
- The MOV opcodes involve data transfers within the 8051 memory. This memory is divided into the following four distinct physical parts:
    1) Internal RAM
    2) Internal special-function registers
    3) External RAM
    4) Internal and external ROM
- following five types of opcodes are used to move data:

    1: MOV, 2: MOVX, 3: MOVC

    4: PUSH and POP

    5: XCH

Addressing Modes



Immediate Addressing Mode

1) Immediate Addressing Mode

- The simplest way to get data to a destination is to make the source of the data part of the opcode. The data source is then immediately available as part of the instruction itself.
- When the 8051 executes an immediate data move, the program counter is automatically incremented to point to the byte(s) following the opcode byte in the program memory. Whatever data is found there is copied to the destination address.

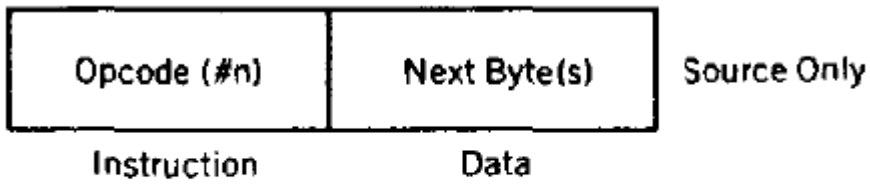
Addressing Modes

1) Immediate Addressing Mode

- The mnemonic for immediate data is the pound sign (#).

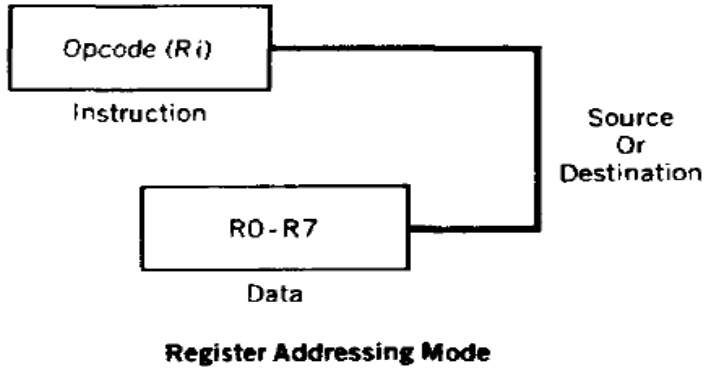
  Occasionally,
- If one forgets to use the # for immediate data. The resulting opcode is often a legal command that is assembled with no objections by the assembler.

- Example: MOV A, #45h➔ Copy the data 45h to A.



Immediate Addressing Mode

Addressing Modes



Register Addressing Mode

2) Register Addressing Mode

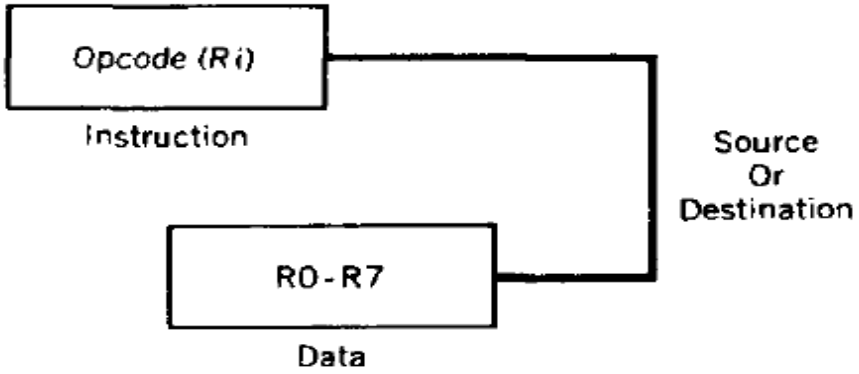
- Certain register names may be used as part of the opcode mnemonic as sources or destinations of data. Registers A, DPTR, and R0 to R7 may be named as part of the opcode mnemonic. Other registers in the 8051 may be addressed using the direct addressing mode.
- Remember that the registers used in the opcode as RO to R7 are the ones that are currently chosen by the bank-select bits, RS0 and RSI in the PSW.

Addressing Modes

2) Register Addressing Mode

- The following table shows all possible MOV opcodes using immediate and register addressing modes:

Mnemonic	Operation
MOV A,#n	Copy the immediate data byte n to the A register
MOV A,Rr	Copy data from register Rr to register A
MOV Rr,A	Copy data from register A to register Rr
MOV Rr,#n	Copy the immediate data byte n to register Rr
MOV DPTR,#nn	Copy the immediate 16-bit number nn to the DPTR register



Register Addressing Mode

Addressing Modes

2) Register Addressing Mode

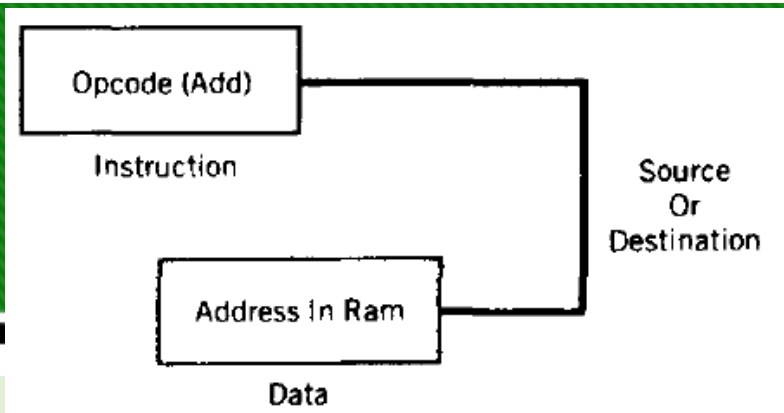
- Examples

Mnemonic	Operation
MOV A,#0F1h	Move the immediate data byte F1h to the A register
MOV A,R0	Copy the data in register R0 to register A
MOV DPTR,#0ABCDh	Move the immediate data bytes ABCDh to the DPTR
MOV R5,A	Copy the data in register A to register R5
MOV R3,#1Ch	Move the immediate data byte 1Ch to register R3

- Important Notes

- It is impossible to have immediate data as a destination.
- All numbers must start with a decimal number (0-9), or the assembler assumes the number is a label.
- Register-to-register moves using the register addressing mode occur between registers A and RO to R7.

Addressing Modes



3) Direct Addressing Mode

- All 128 bytes of internal RAM and the SFRs may be addressed directly using the single byte address assigned to each RAM location and each special-function register.
- Internal RAM uses addresses from 00 to 7Fh to address each byte. The SFR addresses exist from 80h to FFh at the following locations:

SFR	ADDRESS (HEX)
A	0E0
B	0F0
DPL	82
DPH	83
IE	0A8
IP	0B8
P0	80
P1	90
P2	0A0
P3	0B0

PCON	87
PSW	0D0
SBUF	99
SCON	98
SP	81
TCON	88
TMOD	89
TH0	8C
TL0	8A
TH1	8D
TL1	8B

Addressing Modes

3) Direct Addressing Mode

- Note the use of a leading O for all numbers that begin with an alphabetic (alpha) character.
- RAM addresses 00 to 1Fh are also the locations assigned to the four banks of eight working registers, RO to R7. This assignment means that R2 of register bank O can be addressed in the register mode as R2 or in the direct mode as 02h. The direct addresses of the working registers are as follows:

Addressing Modes

3) Direct Addressing Mode

BANK REGISTER		ADDRESS (HEX)	BANK REGISTER		ADDRESS (HEX)
0	R0	00	2	R0	10
0	R1	01	2	R1	11
0	R2	02	2	R2	12
0	R3	03	2	R3	13
0	R4	04	2	R4	14
0	R5	05	2	R5	15
0	R6	06	2	R6	16
0	R7	07	2	R7	17
1	R0	08	3	R0	18
1	R1	09	3	R1	19
1	R2	0A	3	R2	1A
1	R3	0B	3	R3	1B
1	R4	0C	3	R4	1C
1	R5	0D	3	R5	1D
1	R6	0E	3	R6	1E
1	R7	0F	3	R7	1F

Addressing Modes

3) Direct Addressing Mode

The moves made possible using direct, immediate, and register addressing modes are as follows:

Mnemonic	Operation
MOV A,add	Copy data from direct address add to register A
MOV add,A	Copy data from register A to direct address add
MOV Rr,add	Copy data from direct address add to register Rr
MOV add,Rr	Copy data from register Rr to direct address add
MOV add,#n	Copy immediate data byte n to direct address add
MOV add1,add2	Copy data from direct address add2 to direct address add1

Examples:

Mnemonic	Operation
MOV A,80h	Copy data from the port 0 pins to register A
MOV 80h,A	Copy data from register A to the port 0 latch
MOV 3Ah,#3Ah	Copy immediate data byte 3Ah to RAM location 3Ah
MOV R0,12h	Copy data from RAM location 12h to register R0
MOV 8Ch,R7	Copy data from register R7 to timer 0 high byte
MOV 5Ch,A	Copy data from register A to RAM location 5Ch
MOV 0A8h,77h	Copy data from RAM location 77h to IE register

Addressing Modes

3) Direct Addressing Mode

- Important Notes

- MOV instructions that refer to direct addresses above 7Fh that are not SFRs will result in errors. The SFRs are physically on the chip; all other addresses above 7Fh do not physically exist.
- Moving data to a port changes the port latch; moving data from a port gets data from the port pins.
- Moving data from a direct address to itself is not predictable and could lead to errors.

Addressing Modes

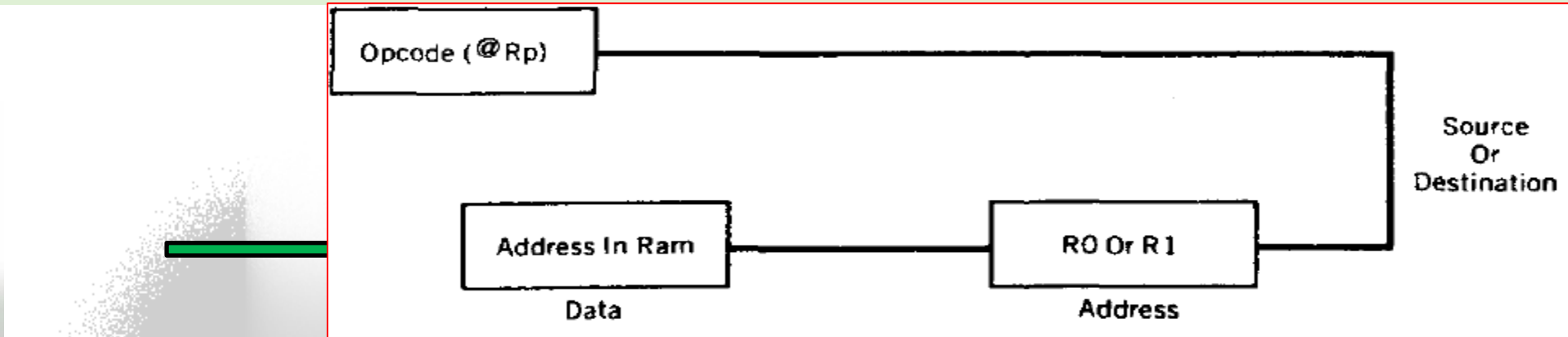
4) Indirect Addressing Mode

- Remember: For all the addressing modes covered to this point, the source or destination of the data is an absolute number or a name.
- Inspection of the opcode reveals exactly what are the addresses of the destination and source. For example, the opcode MOV A,R7 says that the A register will get a copy of whatever data is in register R7; MOV 33h,#32h moves the hex number 32 to hex RAM address 33.

Addressing Modes

4) Indirect Addressing Mode

- The indirect addressing mode uses a register to hold the actual address that will finally be used in the data move; the register itself is not the address, but rather the number in the register.
- Indirect addressing for MOV opcodes uses register RO or RI, often called "data pointers," to hold the address of one of the data locations, which could be a RAM or an SFR address.
- The number that is in the pointing register (Rp) cannot be known un less the history of the register is known. The mnemonic symbol used for indirect addressing is the "at" sign, which is printed as@.



Addressing Modes

4) Indirect Addressing Mode

- The moves made possible are as follows:

Mnemonic	Operation
MOV @Rp,#n	Copy the immediate byte n to the address in Rp
MOV @Rp,add	Copy the contents of add to the address in Rp
MOV @Rp,A	Copy the data in A to the address in Rp
MOV add,@Rp	Copy the contents of the address in Rp to add
MOV A,@Rp	Copy the contents of the address in Rp to A

- Examples:

Mnemonic	Operation
MOV A,@R0	Copy the contents of the address in R0 to the A register
MOV @R1,#35h	Copy the number 35h to the address in R1
MOV add,@R0	Copy the contents of the address in R0 to add
MOV @R1,A	Copy the contents of A to the address in R1
MOV @R0,80h	Copy the contents of the port 0 pins to the address in R0

The number in register Rp must be a RAM or an SFR address.

Only registers R0 or R1 may be used for indirect addressing.

External Addressing using MOVX and MOVC

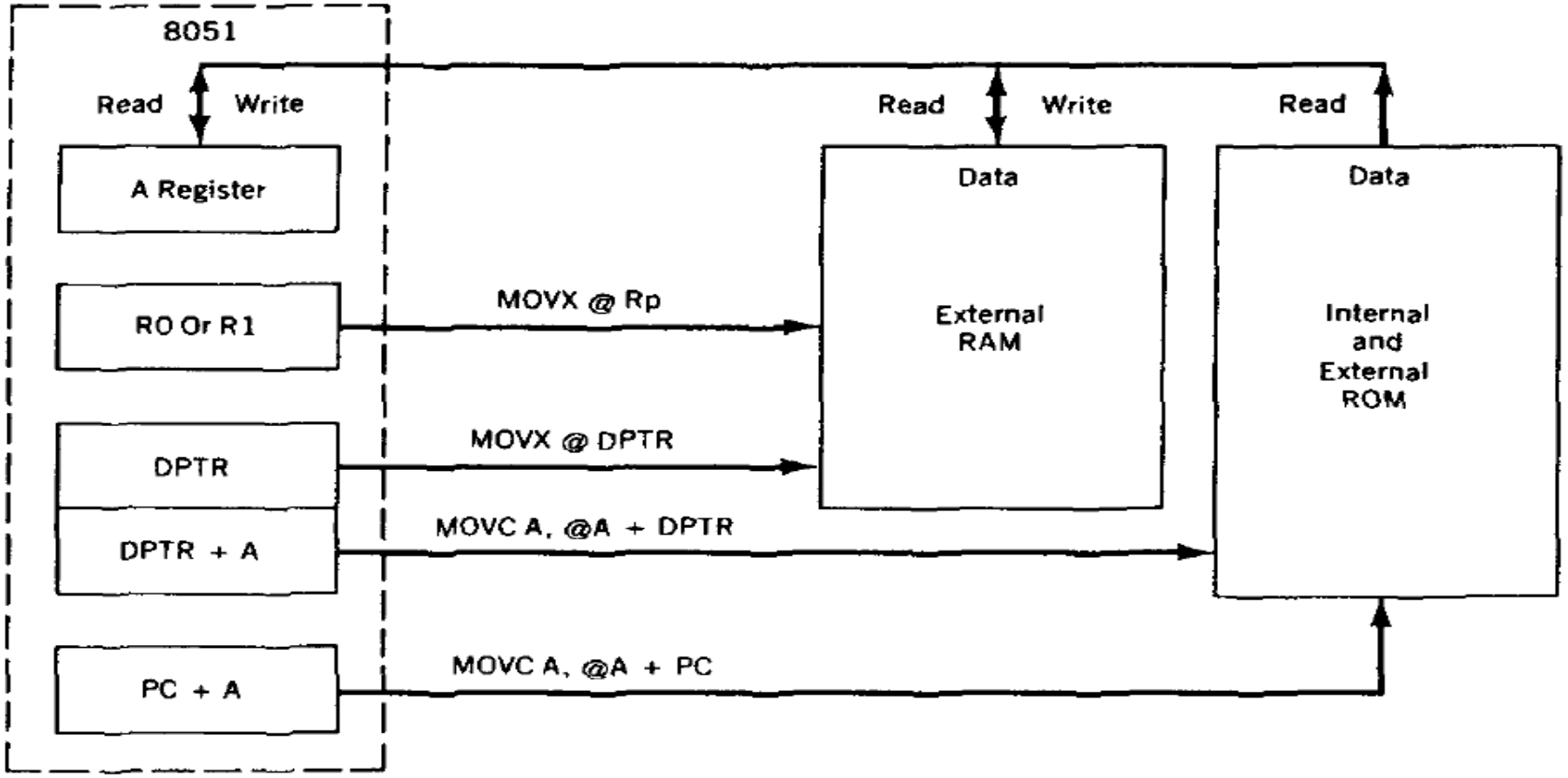
- External Data Moves

- As discussed in Chapter 2, it is possible to expand RAM and ROM memory space by adding external memory chips to the 8051 microcontroller.
- The external memory can be as large as 64K bytes for each of the RAM and ROM memory areas.
- Opcodes that access this external memory always use indirect addressing to specify the external memory.
- An X is added to the MOV mnemonics to serve as a reminder that the data move is external to the 8051.

External Addressing using MOVX and MOVC

- External Data Moves

Figure shows that registers RO, R1, and the DPTR can be used to hold the address of the data byte in external RAM. RO and RI are limited to external RAM address ranges of OOh to 0FFh, while the DPTR register can address the maximum RAM space of 0000h to 0FFFFh.



External Addressing using MOVX and MOVC

- External Data Moves

Mnemonic	Operation
MOVX A,@Rp	Copy the contents of the external address in Rp to A
MOVX A,@DPTR	Copy the contents of the external address in DPTR to A
MOVX @Rp,A	Copy data from A to the external address in Rp
MOVX @DPTR,A	Copy data from A to the external address in DPTR

- Examples

Mnemonic	Operation
MOVX @DPTR,A	Copy data from A to the 16-bit address in DPTR
MOVX @R0,A	Copy data from A to the 8-bit address in R0
MOVX A,@R1	Copy data from the 8-bit address in R1 to A
MOVX A,@DPTR	Copy data from the 16-bit address in DPTR to A

- Important Notes

- An external data moves must involve the A register.
- Rp can address 256 bytes; DPTR can address 64K bytes.
- MOVX is normally used with external RAM or 1/0 addresses.
- There are two sets of RAM addresses between 00 and 0FFh: one internal and one external to the 8051.

External Addressing using MOVX and MOVC

- Code Memory Read-Only Data Moves

- Data moves between RAM locations and 8051 registers are made by using MOV and MOYX opcodes.
- The data is usually of a temporary or "scratch pad" nature and disappears when the system is powered down.
- There are times when access to a preprogrammed mass of data is needed, such as when using tables of predefined bytes. This data must be permanent to be of repeated use and is stored in the program ROM using assembler directives that store programmed data anywhere in ROM that the programmer wishes.

External Addressing using MOVX and MOVC

- Code Memory Read-Only Data Moves

- Access to this data is made possible by using indirect addressing and the A register in conjunction with either the PC or the DPTR, as shown in pervious Figure.
- In both cases, the number in register A is added to the pointing register to form the address in ROM where the desired data is to be found.
- The data is then fetched from the ROM address so formed and placed in the A register. The original data in A is lost, and the addressed data takes its place.

Mnemonic	Operation
MOVC A,@A+DPTR	Copy the code byte, found at the ROM address formed by adding A and the DPTR, to A
MOVC A,@A+PC	Copy the code byte, found at the ROM address formed by adding A and the PC, to A

Note that the DPTR and the PC are not changed; the A register contains the ROM byte found at the address formed.

External Addressing using MOVX and MOVC

- Examples

Mnemonic	Operation
MOV DPTR,#1234h	Copy the immediate number 1234h to the DPTR
MOV A,#56h	Copy the immediate number 56h to A
MOVC A,@A+DPTR	Copy the contents of address 128Ah to A
MOVC A,@A+PC	Copies the contents of address 4059h to A if the PC contained 4000h and A contained 58h when the opcode is executed.

- Important Notes

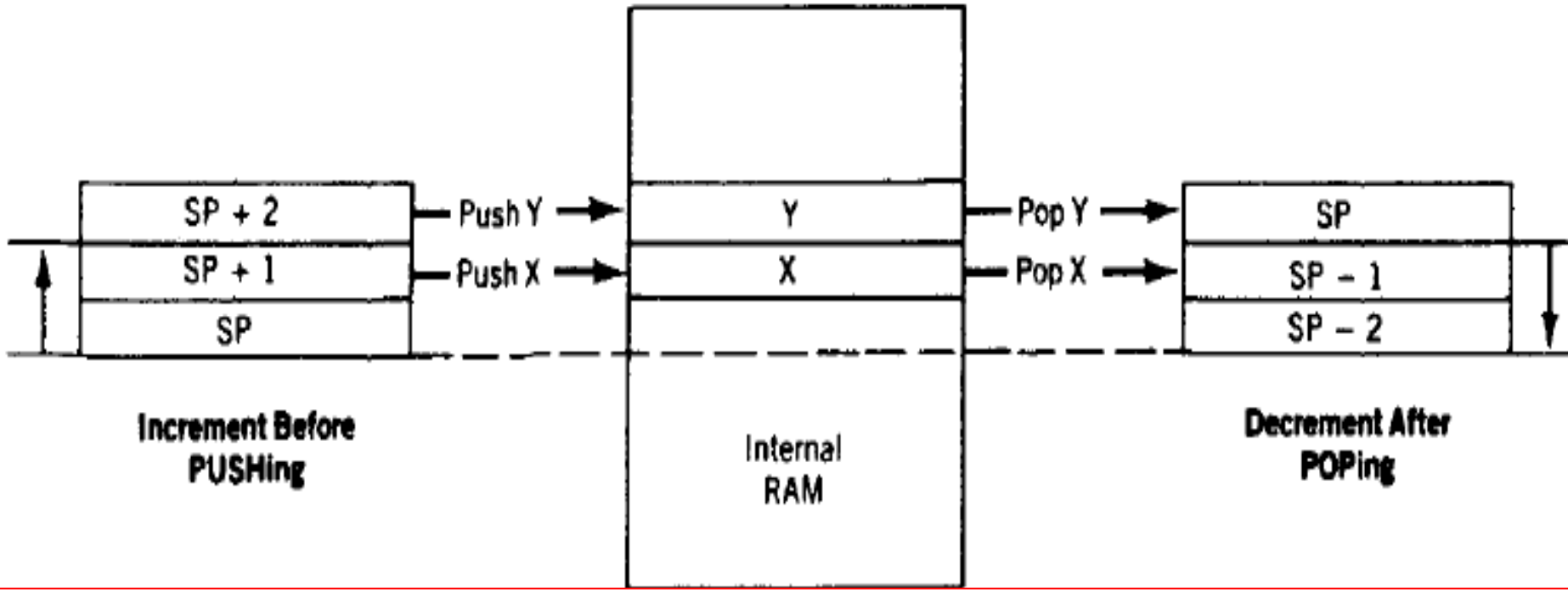
- The PC is incremented by one (to point to the next instruction) before it is added to A to form the final address of the code byte.
- All data is moved from the code memory to the A register.
- MOVC is normally used with internal or external ROM and can address 4K of internal or 64K bytes of external code.

PUSH and POP Opcodes

- The PUSH and POP opcodes specify the direct address of the data.
- The data moves between an area of internal RAM, known as the stack, and the specified direct address.
- The stack pointer special-function register (SP) contains the address in RAM where data from the source address will be PUSHed, or where data to be POPed to the destination address is found.
- The SP register actually is used in the indirect addressing mode but is not named in the mnemonic. It is implied that the SP holds the indirect address when ever PUSHing or POPing.

PUSH and POP Opcodes

Figure shows the operation of the stack pointer as data is PUSHed or POPed to the stack area in internal RAM.



The SP register is set to 07h when the 8051 is reset, which is the same direct address in internal RAM as register R7 in bank 0. The first PUSH opcode would write data to R0 of bank 1. The SP should be initialized by the programmer to point to an internal RAM address above the highest address likely to be used by the program.

PUSH and POP Opcodes

- A PUSH opcode copies data from the source address to the stack.
- SP is incremented by one before the data is copied to the internal RAM location contained in SP so that the data is stored from low addresses to high addresses in the internal RAM.
- The stack grows up in memory as it is PUSHed. Excessive PUSHing can make the stack exceed 7Fh (the top of internal RAM), after which point data is lost.
- A POP opcode copies data from the stack to the destination address.
- SP is decremented by one after data is copied from the stack RAM address to the direct destination to ensure that data placed on the stack is retrieved in the same order as it was stored.

PUSH and POP Opcodes

Mnemonic	Operation
PUSH add	Increment SP; copy the data in add to the internal RAM address contained in SP
POP add	Copy the data from the internal RAM address contained in SP to add; decrement the SP

- Examples

Mnemonic	Operation
MOV 81h,#30h	Copy the immediate data 30h to the SP
MOV R0, #0ACh	Copy the immediate data ACh to R0
PUSH 00h	SP = 31h; address 31h contains the number ACh
PUSH 00h	SP = 32h; address 32h contains the number ACh
POP 01h	SP = 31h; register R1 now contains the number ACh
POP 80h	SP = 30h; port 0 latch now contains the number ACh

PUSH and POP Opcodes

- Important Notes

- When the SP reaches FFh it "**rolls over**" to 00h (RO).
- RAM ends at address 7Fh; PUSHes above 7Fh result in errors.
- The SP is usually set at addresses above the register banks.
- The SP may be PUSHed and POPed to the stack.
- Direct addresses, not register names, must be used for most registers.
- The stack mnemonics have no way of knowing which bank is in use.

Data Exchanges

- MOV, PUSH, and POP opcodes all involve copying the data found in the source address to the destination address; the original data in the source is not changed.
- Exchange instructions actually move data in two directions: from source to destination and from destination to source.
- All addressing modes except immediate may be used in the XCH (exchange) opcodes:

Data Exchanges

Mnemonic	Operation
XCH A,Rr	Exchange data bytes between register Rr and A
XCH A,add	Exchange data bytes between add and A
XCH A,@Rp	Exchange data bytes between A and address in Rp
XCHD A,@Rp	Exchange lower nibble between A and address in Rp

- Examples

Mnemonic	Operation
XCH A,R7	Exchange bytes between register A and register R7
XCH A,0F0h	Exchange bytes between register A and register B
XCH A,@R1	Exchange bytes between register A and address in R1
XCHD A,@R1	Exchange lower nibble in A and the address in R1

Data Exchanges

- Important Notes

- All exchanges are internal to the 8051.
- All exchanges use register A.
- When using XCHD, the upper nibble of A and the upper nibble of the address location in Rp do not change.

- This section concludes the listing of the various data moving instructions; the remaining sections will concentrate on using these opcodes to write short programs.