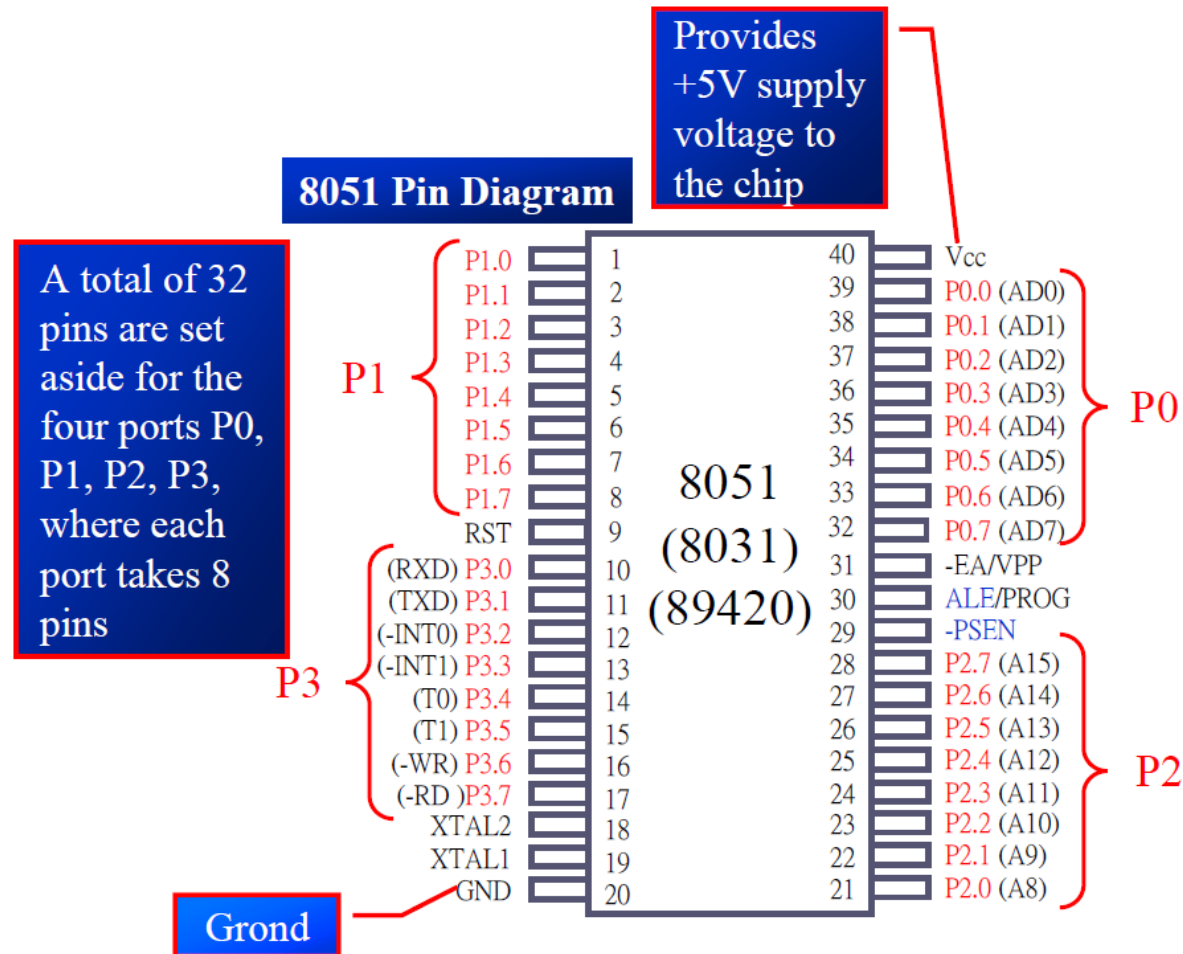


Embedded System (Multidisciplinary Minor)

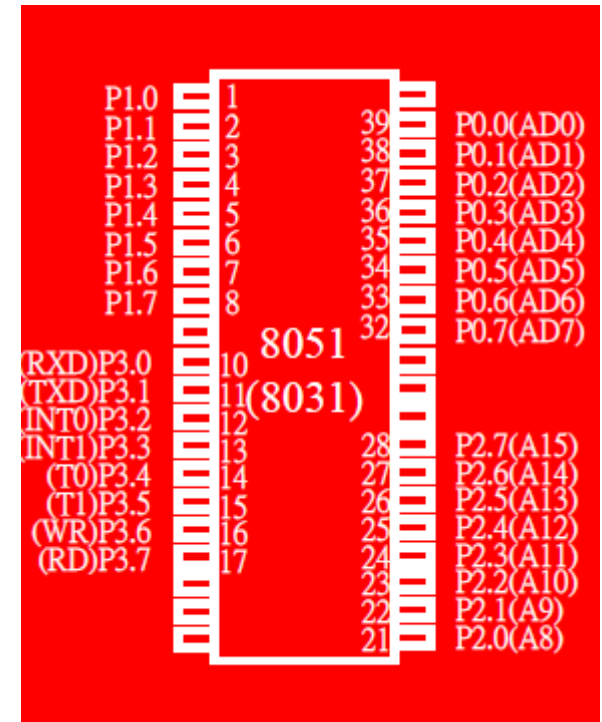
Module 4: 8051 Addressing Modes and Programming

8051 Microcontroller-Pin Diagram



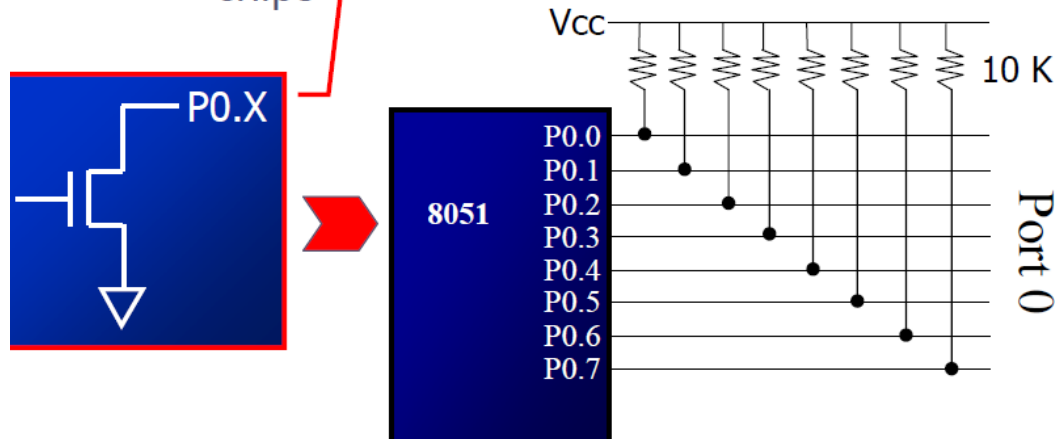
8051 Microcontroller-I/O Ports

- ❑ 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
 - To reconfigure it as an input, a 1 must be sent to the port
 - To use any of these ports as an input port, it must be programmed



I/O Ports

- ❑ 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



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
```

Key Consideration

- **Port 0 (P0):**

- Unlike P1, P2, and P3, Port 0 lacks internal pull-up resistors. Therefore, when using Port 0 as an output and needing to output a logic '1' (high), external pull-up resistors must be connected to its pins. When used as an input, its pins will float if no external device drives them high.

- **Ports 1, 2, and 3 (P1, P2, P3):**

- These ports have internal pull-up resistors, meaning they can source current and output a logic '1' without external pull-ups.

In the 8051 microcontroller, the input/output (I/O) ports are initialized primarily by writing specific values to their corresponding registers. **All four ports (P0, P1, P2, P3) are configured as inputs by default upon a reset.**

To configure a port or a specific pin as an output:

- Write a logic '0' (low) to the corresponding bit in the port's latch. This turns on the output driver for that pin, allowing it to sink current and output a low voltage (0V).

To configure a port or a specific pin as an input:

- Write a logic '1' (high) to the corresponding bit in the port's latch. This effectively turns off the output driver, allowing the external device to control the pin's voltage level, which can then be read by the microcontroller.

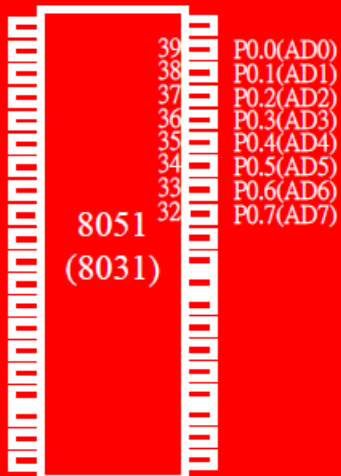
- ❑ In order to make port 0 an input, the port must be programmed by writing 1 to all the bits

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           ;keep doing it
```


I/O PROGRAMMING

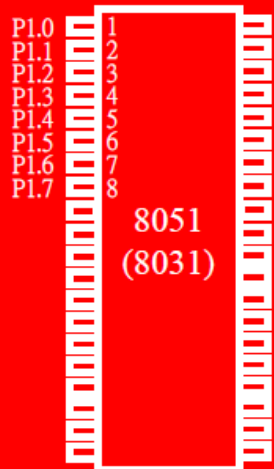
Dual Role of Port 0



- ❑ 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

I/O PROGRAMMING

Port 1



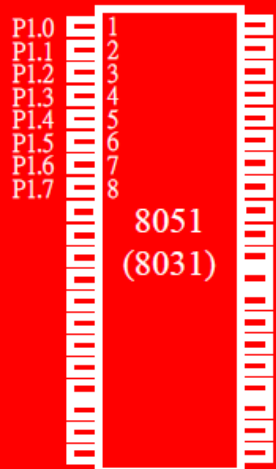
- ❑ Port 1 can be used as input or output
 - In contrast to port 0, this port does not need any pull-up resistors since it already has pull-up resistors internally
 - Upon reset, port 1 is configured as an input port

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
```

I/O PROGRAMMING

Port 1 as Input



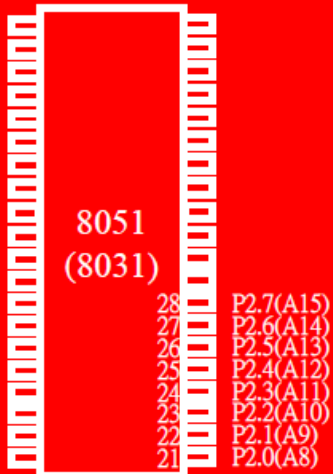
- ❑ To make port 1 an input port, it must be programmed as such by writing 1 to all its bits

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
```

I/O PROGRAMMING

Port 2



- ❑ Port 2 can be used as input or output
 - Just like P1, port 2 does not need any pull-up 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 system, P2 is used as simple I/O

-
- ❑ 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, this is not the way it is most commonly used

Port 3 has the additional function of providing some extremely important signals

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



I/O PROGRAMMING

Different ways of Accessing Entire 8 Bits

The entire 8 bits of Port 1 are accessed

```
BACK:  MOV    A, #55H
        MOV    P1, A
        ACALL  DELAY
        MOV    A, #0AAH
        MOV    P1, A
        ACALL  DELAY
        SJMP   BACK
```

Rewrite the code in a more efficient manner by accessing the port directly without going through the accumulator

```
BACK:  MOV    P1, #55H
        ACALL  DELAY
        MOV    P1, #0AAH
        ACALL  DELAY
        SJMP   BACK
```

Another way of doing the same thing

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

I/O BIT MANIPULATION PROGRAMMING

I/O Ports and Bit Addressability

- Sometimes we need to access only 1 or 2 bits of the port

```
BACK:  CPL    P1.2           ;complement P1.2
        ACALL  DELAY
        SJMP   BACK
```

;another variation of the above program

```
AGAIN: SETB   P1.2           ;set only P1.2
        ACALL  DELAY
        CLR    P1.2          ;clear only P1.2
        ACALL  DELAY
        SJMP   AGAIN
```

P0	P1	P2	P3	Port Bit
P0.0	P1.0	P2.0	P3.0	D0
P0.1	P1.1	P2.1	P3.1	D1
P0.2	P1.2	P2.2	P3.2	D2
P0.3	P1.3	P2.3	P3.3	D3
P0.4	P1.4	P2.4	P3.4	D4
P0.5	P1.5	P2.5	P3.5	D5
P0.6	P1.6	P2.6	P3.6	D6
P0.7	P1.7	P2.7	P3.7	D7

Example 4-2

Write the following programs.

Create a square wave of 50% duty cycle on bit 0 of port 1.

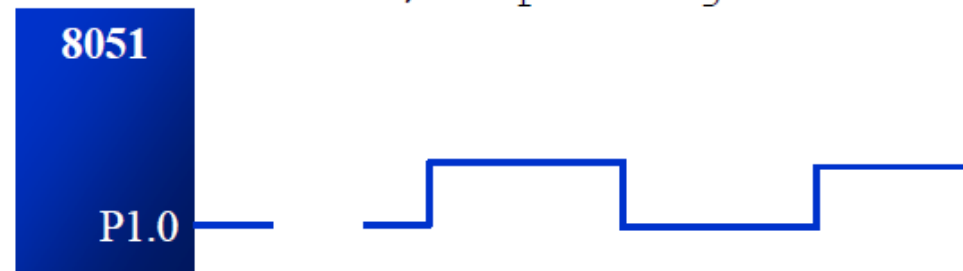
Solution:

The 50% duty cycle means that the “on” and “off” state (or the high and low portion of the pulse) have the same length. Therefore, we toggle P1.0 with a time delay in between each state.

```
HERE:  SETB    P1.0    ;set to high bit 0 of port 1
        LCALL   DELAY  ;call the delay subroutine
        CLR     P1.0    ;P1.0=0
        LCALL   DELAY
        SJMP    HERE   ;keep doing it
```

Another way to write the above program is:

```
HERE:  CPL     P1.0    ;set to high bit 0 of port 1
        LCALL   DELAY  ;call the delay subroutine
        SJMP    HERE   ;keep doing it
```



I/O BIT MANIPULATION PROGRAMMING

I/O Ports and Bit Addressability (cont')

- ❑ Instructions that are used for signal-bit operations are as following

Single-Bit Instructions

Instruction	Function
SETB bit	Set the bit (bit = 1)
CLR bit	Clear the bit (bit = 0)
CPL bit	Complement the bit (bit = NOT bit)
JB bit, target	Jump to target if bit = 1 (jump if bit)
JNB bit, target	Jump to target if bit = 0 (jump if no bit)
JBC bit, target	Jump to target if bit = 1, clear bit (jump if bit, then clear)

I/O BIT MANIPULATION PROGRAMMING

Reading Single Bit into Carry Flag (cont')

Example 4-7

A switch is connected to pin P1.0 and an LED to pin P2.7. Write a program to get the status of the switch and send it to the LED

Solution:

```
        SETB  P1.7           ;make P1.7 an input
AGAIN:  MOV   C,P1.0         ;read SW status into CF
        MOV   P2.7,C         ;send SW status to LED
        SJMP  AGAIN         ;keep repeating
```

However 'MOV
P2, P1' is a valid
instruction

The instruction
'MOV
P2.7, P1.0' is
wrong, since such
an instruction does
not exist



I/O BIT MANIPULATION PROGRAMMING

Reading Input Pins vs. Port Latch

- ❑ In reading a port
 - Some instructions read the status of port pins
 - Others read the status of an internal port latch
- ❑ Therefore, when reading ports there are two possibilities:
 - Read the status of the input pin
 - Read the internal latch of the output port
- ❑ Confusion between them is a major source of errors in 8051 programming
 - Especially where external hardware is concerned

READING INPUT PINS VS. PORT LATCH

Reading Latch for Output Port

- ❑ Some instructions read the contents of an internal port latch instead of reading the status of an external pin
 - For example, look at the `ANL P1, A` instruction and the sequence of actions is executed as follow
 1. It reads the internal latch of the port and brings that data into the CPU
 2. This data is ANDed with the contents of register A
 3. The result is rewritten back to the port latch
 4. The port pin data is changed and now has the same value as port latch

READING INPUT PINS VS. PORT LATCH

Reading Latch for Output Port (cont')

❑ *Read-Modify-Write*

- The instructions read the port latch normally read a value, perform an operation then rewrite it back to the port latch

Instructions Reading a latch (Read-Modify-Write)

Mnemonics	Example
ANL PX	ANL P1,A
ORL PX	ORL P2,A
XRL PX	XRL P0,A
JBC PX.Y,TARGET	JBC P1.1,TARGET
CPL PX.Y	CPL P1.2
INC PX	INC P1
DEC PX	DEC P2
DJNZ PX.Y,TARGET	DJNZ P1,TARGET
MOV PX.Y,C	MOV P1.2,C
CLR PX.Y	CLR P2.3
SETB PX.Y	SETB P2.3

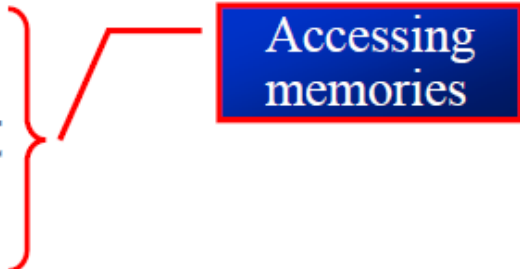
Note: x is 0, 1, 2,
or 3 for P0 – P3



DEEMED TO BE
UNIVERSITY
—RAMRAO ADIK—
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

- ❑ The ports in 8051 can be accessed by the Read-modify-write technique
 - This feature saves many lines of code by combining in a single instruction all three actions
 1. Reading the port
 2. Modifying it
 3. Writing to the port

ADDRESSING MODES

- ❑ The CPU can access data in various ways, which are called *addressing modes*
 - Immediate
 - Register
 - Direct
 - Register indirect
 - Indexed
- 
- A red bracket groups the last three items in the list: 'Direct', 'Register indirect', and 'Indexed'. A red line extends from the middle of this bracket to a blue rectangular box with a red border. Inside the box, the text 'Accessing memories' is written in white.

IMMEDIATE ADDRESSING MODE

- ❑ The source operand is a constant
 - The immediate data must be preceded by the pound sign, "#"
 - Can load information into any registers, including 16-bit DPTR register
 - DPTR can also be accessed as two 8-bit registers, the high byte DPH and low byte DPL

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

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

REGISTER ADDRESSING MODE

- ❑ Use registers to hold the data to be manipulated

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

- ❑ The source and destination registers must match in size

➤ MOV DPTR,A will give an error

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

- ❑ The movement of data between Rn registers is not allowed

➤ MOV R4,R7 is invalid



ACCESSING MEMORY

Direct Addressing Mode

- ❑ It is most often used the direct addressing mode to access RAM locations 30 – 7FH

- The entire 128 bytes of RAM can be accessed

Direct addressing mode

- The register bank locations are accessed by the register names

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

- ❑ Contrast this with immediate addressing mode

Register addressing mode

- There is no “#” sign in the operand

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

ACCESSING MEMORY

SFR Registers and Their Addresses

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

MOV 0E0H, #55H	;is the same as
MOV A, #55h	;load 55H into A
MOV 0F0H, R0	;is the same as
MOV B, R0	;copy R0 into B

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

ACCESSING MEMORY

SFR Registers and Their Addresses (cont')

Special Function Register (SFR) Addresses

Symbol	Name	Address
ACC*	Accumulator	0E0H
B*	B register	0F0H
PSW*	Program status word	0D0H
SP	Stack pointer	81H
DPTR	Data pointer 2 bytes	
DPL	Low byte	82H
DPH	High byte	83H
P0*	Port 0	80H
P1*	Port 1	90H
P2*	Port 2	0A0H
P3*	Port 3	0B0H
IP*	Interrupt priority control	0B8H
IE*	Interrupt enable control	0A8H
...

ACCESSING MEMORY

SFR Registers and Their Addresses (cont')

Special Function Register (SFR) Addresses

Symbol	Name	Address
TMOD	Timer/counter mode control	89H
TCON*	Timer/counter control	88H
T2CON*	Timer/counter 2 control	0C8H
T2MOD	Timer/counter mode control	0C9H
TH0	Timer/counter 0 high byte	8CH
TL0	Timer/counter 0 low byte	8AH
TH1	Timer/counter 1 high byte	8DH
TL1	Timer/counter 1 low byte	8BH
TH2	Timer/counter 2 high byte	0CDH
TL2	Timer/counter 2 low byte	0CCH
RCAP2H	T/C 2 capture register high byte	0CBH
RCAP2L	T/C 2 capture register low byte	0CAH
SCON*	Serial control	98H
SBUF	Serial data buffer	99H
PCON	Power on/off control	87H

* Bit addressable

ACCESSING MEMORY

SFR Registers and Their Addresses (cont')

Example 5-1

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

Solution :

(a) MOV A, #55H ; A=55H
 MOV P1, A ; P1=55H
 MOV P2, A ; P2=55H

(b) From Table 5-1, P1 address=80H; P2 address=A0H
 MOV A, #55H ; A=55H
 MOV 80H, A ; P1=55H
 MOV 0A0H, A ; P2=55H

ACCESSING MEMORY

Stack and Direct Addressing Mode

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

Example 5-2

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

Solution:

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


ACCESSING MEMORY

Register Indirect Addressing Mode

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

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

ACCESSING MEMORY

Register Indirect Addressing Mode (cont')

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

ACCESSING MEMORY

Register Indirect Addressing Mode (cont')

Example 5-3

Write a program to copy the value 55H into RAM memory locations 40H to 41H using

(a) direct addressing mode, (b) register indirect addressing mode without a loop, and (c) with a loop

Solution:

(a)

```
MOV A, #55H    ;load A with value 55H
MOV 40H, A      ;copy A to RAM location 40H
MOV 41H, A      ;copy A to RAM location 41H
```

(b)

```
MOV A, #55H    ;load A with value 55H
MOV R0, #40H   ;load the pointer. R0=40H
MOV @R0, A      ;copy A to RAM R0 points to
INC R0         ;increment pointer. Now R0=41h
MOV @R0, A      ;copy A to RAM R0 points to
```

(c)

```
MOV A, #55H    ;A=55H
MOV R0, #40H   ;load pointer. R0=40H,
MOV R2, #02    ;load counter, R2=3
AGAIN: MOV @R0, A ;copy 55 to RAM R0 points to
INC R0         ;increment R0 pointer
DJNZ R2, AGAIN ;loop until counter = zero
```

ACCESSING MEMORY

Indexed Addressing Mode and On-chip ROM Access

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

```
MOVCA, @A+PC;  
MOVCA, @A+DPTR;
```

Example: Indexed Addressing Modes

; Lookup table stored in code memory

ORG 100H

TABLE: DB 10, 20, 30, 40, 50 ; Lookup table with 5 values

; Main program

ORG 0000H

MOV DPTR, #TABLE ; Load DPTR with the base address of the table

MOV A, #2 ; Load index 2 into Accumulator (to access the 3rd element, 30)

MOVC A, @A+DPTR ; Move the value from the table (TABLE + 2) into A

; A now contains 30 (the 3rd value in the table)

SJMP \$; Infinite loop to halt the program

END

Thank You