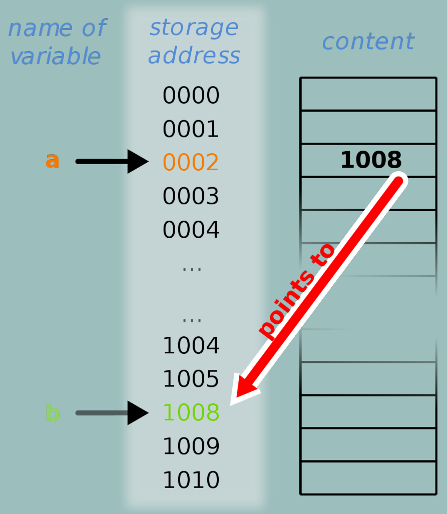


CHAPTER 6 —PART 2

ADDRESSING MODES 8051

SMJE 3183

ASSEMBLY LANGUAGE

Computer language between machine language and high-level language.

General format.

```
[label:]    mnemonic [operand] [, operand] [, …] [; comment]
START:      MOV         A,#0FF        ;'Start" Is a Label.
```

8051 DATA ADDRESSING MODES

Assembly language instructions

- Consist of an operation mnemonic and between zero and three operands separated by commas.
- Where two operand instructions occur, the destination is specified first, followed by the source operand.

Opcode dest, operand source, operand

ADDRESSING MODES

How you are addressing a given memory location

Example

Immediate Addressing	MOV A,#20H
Direct Addressing	MOV A,30H
Indirect Addressing	MOV A,@R0
External Direct	MOVX A,@DPTR
Code Indirect/Indexed	MOVC A,@A+DPTR

The first three modes provide access to the internal RAM and hardware register space.

- Thus can occur in both the source and destination operand fields.

IMMEDIATE ADDRESSING

Data in the operand field.

Data are preceded with # prefix.

- The immediate value is a maximum of 8-bits.
- All bits in the high-byte must be the same (00H or FFH)
  - MOV A, #0FEH

- One exception, when dealing with the DPTR register it can be 16-bits.

  - MOV DPTR, #2000H ; Load the value 2000H into the DPTR register

IMMEDIATE ADDRESSING

MOV A,#20H

- The value to be stored in memory immediately follows the operation code in memory
- The instruction itself dictates what value will be stored in memory.

DIRECT ADDRESSING

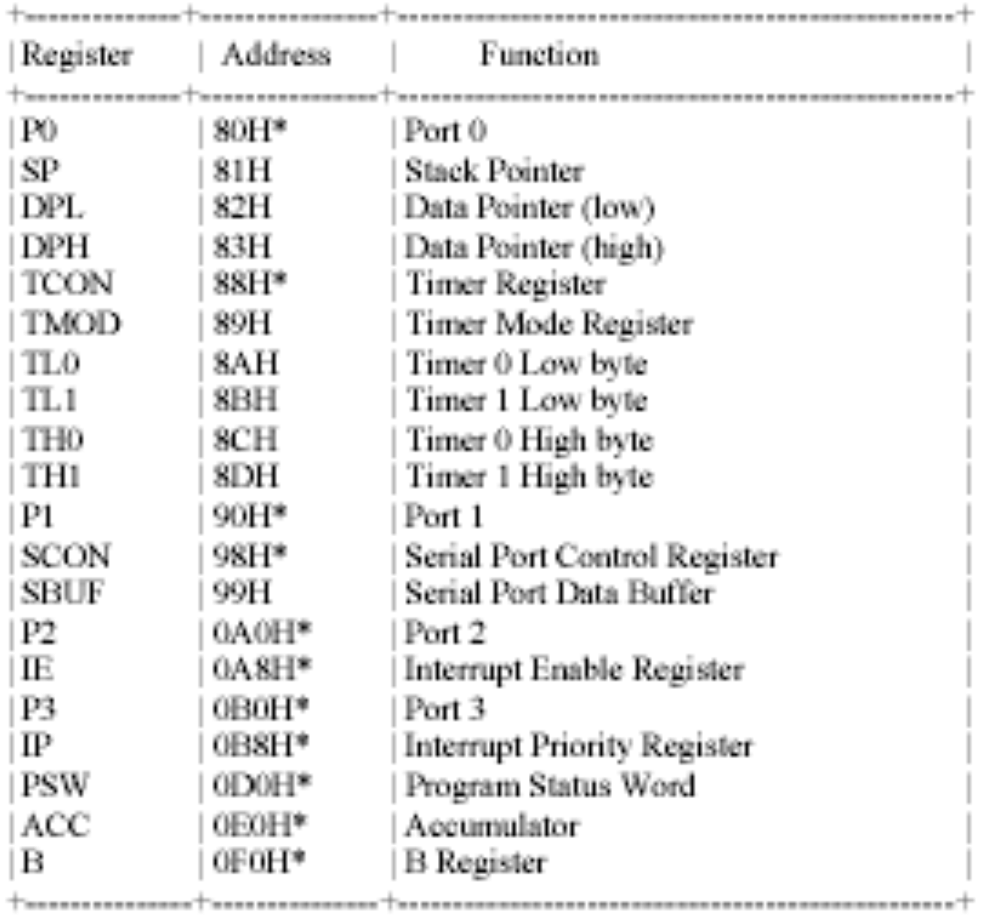
Can access any on-chip variable or hardware register.

- Memory address (00H to 7FH)
- SFR address (80H to 0FFH)
- Predefined symbol may be used for SFR address

```
MOV   A,30H
MOV   A, SBUF        ;same as MOV  A,99H
```

- the value to be stored in memory is obtained by directly retrieving it from another memory location

ALL I/O PORTS AND SPECIAL FUNCTION REGISTERS ARE ASSIGNED ADDRESSES BETWEEN 80H AND 0FFH.



Register	Address	Function
P0	80H*	Port 0
SP	81H	Stack Pointer
DPL	82H	Data Pointer (low)
DPH	83H	Data Pointer (high)
TCON	88H*	Timer Register
TMOD	89H	Timer Mode Register
TL0	8AH	Timer 0 Low byte
TL1	8BH	Timer 1 Low byte
TH0	8CH	Timer 0 High byte
TH1	8DH	Timer 1 High byte
P1	90H*	Port 1
SCON	98H*	Serial Port Control Register
SBUF	99H	Serial Port Data Buffer
P2	0A0H*	Port 2
IE	0A8H*	Interrupt Enable Register
P3	0B0H*	Port 3
IP	0B8H*	Interrupt Priority Register
PSW	0D0H*	Program Status Word
ACC	0E0H*	Accumulator
B	0F0H*	B Register

* denotes bit addressable

INDIRECT ADDRESSING

Addressing mode uses the prefix **@**.

Used with R0, R1, the DPTR, or the PC

MOV A, @R0

- The only way to access the upper 128 bytes of Internal RAM which is found at the address indicated by R0.
- Indirect addressing always refers to Internal RAM
- Only **R0 and R1** may be used as pointer registers

EXTERNAL DIRECT

```
MOVX    A, @DPTR        ;read
MOVX    @DPTR, A        ;write
```

- It is used to access external memory
- Only two commands, utilize DPTR.
- DPTR is loaded with the address of external memory that you wish to read or write.

EXTERNAL INDIRECT

MOVX @R0, A

- Used in small projects with very small amount of external RAM
- Value of @R0 can only be 00H through FFH, thus limited to 256 bytes of External RAM.

INDEXED ADDRESSING

Use a register for storing a pointer to memory and another register for storing an offset.

The effective address (EA) is the sum of the two:

- EA = [@Pointer] + [Offset]

```
MOVC    A, @A+DPTR   ;Move byte from memory located at DPTR +A to A
```

BIT ADDRESS

Accessing bit-addressable location

- Data memory (00H to 7FH)
- SFR's (80H to 0FFH)

Ways to specify

- Explicitly giving the address

```
SETB    0E7H    ;set 1 to the content of address E7H
```

- Using the dot operator between byte address and bit position

```
SETB    ACC.7   ; set 1 to the address content of E7H
```

- Using predefined assembler symbol.

```
JNB     SBUF, $
```

CODE ADDRESS

Give in the form of a label

Used in the operand field for the jump instructions.

```
HERE:    *
         *
         SJMP    HERE
```

NUMBER BASES

```
MOV  A,#15          ; decimal
MOV  A,#23D         ; decimal
MOV  A,#1011B       ; binary
MOV  A,#0FH         ; hexadecimal
MOV  A,#17Q         ; octal
```

Digit must be the first character for hexadecimal constant to differentiate them from labels (ie "0A5H" not "A5H")

CHARACTER STRINGS

Character constants are enclosed in single quotes (‘)

The assembles will convert it to binary equivalent.

```
CJNE    A, # ’Q’, AGAIN
MOV     DPTR, # ’AB’
```

ARITHMETIC OPERATORS

+, -, *, /, MOD

Two of the same instructions:

```
MOV  A,#10 + 10H    = MOV  A,#1AH
MOV  A,#25 MOD 7    = MOV  A,#4
```

LOGICAL OPERATOR

OR	logical OR
AND	logical AND
XOR	logical Exclusive OR
NOT	logical NOT (complement)

SPECIAL OPERATOR

SHR	shift right
SHL	shift left
HIGH	high-byte
LOW	low-byte
( )	evaluate first

Two of the same instructions

```
MOV   A,#8 SHL 1         =     MOV   A,#10H
MOV   A,#HIGH 1234H      =     MOV   A,#12H
```

RATIONAL OPERATOR

EQ	=	equal
NE	<>	not equals
LT	<	less than
LE	<=	less than or equal to
GT	>	greater than
GE	>=	greater than or equal to

When used between two operands, the result is always false(0000H) or true (FFFFH)

OPERATOR PRECEDENCE

( )

HIGH LOW

*** / MOD SHL SHR**

+ -

EQ NE LT LE GT GE = <> < <= > >=

NOT

AND

OR XOR

Evaluated left to right