VTU eNotes On Modern Control Theory

Electrical And Electronics Engineering

Addressing Modes and Operations: Introduction, Addressing modes, External data Moves, Code Memory, Read Only Data Moves / Indexed Addressing mode, PUSH and POP Opcodes, Data exchanges, Example Programs; Byte level logical Operations, Bit level Logical Operations, Rotate and Swap Operations, Example Programs. Arithmetic Operations: Flags, Incrementing and Decrementing, Addition, Subtraction, Multiplication and Division, Decimal Arithmetic, Example Programs.

Instruction set of 8051

1. **Data transfer instructions**

a. MOV <dest-byte>,<src-byte>-

Function: Move byte variable

Description: The byte variable indicated by the second operand is copied into the location specified by the first operand. The source byte is not affected. No other register or flag is affected.

1. mov direct , A
2. mov A, @R_i
3. mov A, R_n
4. mov direct, direct
5. mov A, #data

EX: MOV 30h, A

MOV A,@R0 ; moves the content of memory pointed to by Ro into A

MOV A, R_1; ;moves the content of Register R_1 to Accumulator A

MOV 20h,30h;moves the content of memory location 30h to 20h

MOV A,#45h;moves 45h to Accumulator A

b.MOV <dest-bit>,<src-bit>

Function: Move bit data

Description: MOV <dest-bit>,<src-bit> copies the Boolean variable indicated by the second operand into the location specified by the first operand. One of the operands must be the carry flag; the other may be any directly addressable bit. No other register or flag is affected.

Example: MOV P1.3,C; moves the carry bit to 3rd bit of port1

C. MOV DPTR,#data16
Function: Load Data Pointer with a 16-bit constant
Description: MOV DPTR,#data16 loads the Data Pointer with the 16-bit constant indicated. The 16-bit constant is loaded into the second and third bytes of the instruction. The second byte (DPH) is the high-order byte, while the third byte
(DPL) holds the lower-order byte. No flags are affected.
This is the only instruction which moves 16 bits of data at once.
Example: The instruction,
MOV DPTR, # 4567H
loads the value 4567H into the Data Pointer. DPH holds 45H, and DPL holds 67H.

d. MOVC A,@A+ <base-reg>
Function: Move Code byte
Description: The MOVC instructions load the Accumulator with a code byte or constant from program memory. The address of the byte fetched is the sum of the original unsigned 8-bit Accumulator contents and the contents of a 16-bit base register, which may be either the Data Pointer or the PC. In the latter case, the PC is incremented to the address of the following instruction before being added with the Accumulator; otherwise the base register is not altered. Sixteen-bit addition is performed so a carry-out from the low-order eight bits may propagate through higher-order bits. No flags are affected.

e. MOVC A,@A+PC

(PC) ← (PC) + 1
(A) ← ((A) + (PC))

f. MOVX <dest-byte>,<src-byte>

Function: Move External

Description: The MOVX instructions transfer data between the Accumulator and a byte of external data memory, which is why "X" is appended to MOV. There are two types of instructions, differing in whether they provide an 8-bit or 16-bit indirect address to the external data RAM.
In the first type, the contents of R0 or R1 in the current register bank provide an 8-bit address multiplexed with data on P0. Eight bits are sufficient for external I/O expansion decoding or for a relatively small RAM array. For somewhat larger arrays, any output port pins can be used to output higher-order address bits. These pins are
controlled by an output instruction preceding the MOVX.
In the second type of MOVX instruction, the Data Pointer generates a 16-bit address. P2 outputs the high-order eight address bits (the contents of DPH), while P0 multiplexes the low-order eight

bits (DPL) with data. The P2 Special Function Register retains its previous contents, while the P2 output buffers emit the contents of DPH.
This form of MOVX is faster and more efficient when accessing very large data arrays (up to 64K bytes), since no additional instructions are needed to set up the output ports.
It is possible to use both MOVX types in some situations. A large RAM array with its high-order address lines driven by P2 can be addressed via the Data Pointer, or with code to output high-order address bits to P2, followed by a MOVX instruction using R0 or R1.
Example: An external 256 byte RAM using multiplexed address/data lines is connected to the 8051 Port 0. Port 3 provides
control lines for the external RAM. Ports 1 and 2 are used for normal I/O. Registers 0 and 1 contain 12H and
34H. Location 34H of the external RAM holds the value 56H. The instruction sequence,
MOVX A,@R1
MOVX @R0,A
copies the value 56H into both the Accumulator and external RAM location 12H.

MOVX A,@DPTR
(A) ⟵ ((DPTR))

PUSH direct
Function: Push onto stack
Description: The Stack Pointer is incremented by one. The contents of the indicated variable is then copied into the internal RAM location addressed by the Stack Pointer. No flags are affected.
Example: On entering an interrupt routine, the Stack Pointer contains 09H. The Data Pointer holds the value 0123H. The
following instruction sequence,
PUSH DPL
PUSH DPH
leaves the Stack Pointer set to 0BH and stores 23H and 01H in internal RAM locations 0AH and 0BH,
respectively.

POP direct

Function: Pop from stack.
Description: The contents of the internal RAM location addressed by the Stack Pointer is read, and the Stack Pointer is decremented by one. The value read is then transferred to the directly addressed byte indicated. No flags are affected.
Example: The Stack Pointer originally contains the value 32H, and internal RAM locations 30H through 32H contain the values 20H, 23H, and 01H, respectively. The following instruction sequence,

POP DPH
POP DPL
leaves the Stack Pointer equal to the value 30H and sets the Data Pointer to 0123H.

2. Arithmetic Group of Instructions

a. ADD A,<src-byte>

Function: Add

Description: ADD adds the byte variable indicated to the Accumulator, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.
OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not bit 6; otherwise, OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands.
Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C3H (1100001lB), and register 0 holds 0AAH (10101010B). The following instruction,

ADD A,R0

leaves 6DH (01101101B) in the Accumulator with the AC flag cleared and both the carry flag and OV set to 1.

ADD A, direct
(A) ◄— (A) + (direct)

ADD A, @Ri
(A) ◄— (A) + data

ADDC A, <src-byte>
Function: Add with Carry
Description: ADDC simultaneously adds the byte variable indicated, the carry flag and the Accumulator contents, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not out of bit 6; otherwise OV
is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive
operands or a positive sum from two negative operands.
Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.
Example: The Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) with the carry flag set. The following instruction,
ADDC A,R0
leaves 6EH (01101110B) in the Accumulator with AC cleared and both the Carry flag and OV set to 1.

ADDC A,Rn
Operation: ADDC
(A) ← (A) + (C) + (Rn)

ADDC A, direct

Operation: ADDC
(A) ←(A) + (C) + (direct)

ADDC A,@Ri
Operation: ADDC
(A) ← (A) + (C) + ((Ri))

ADDC A, #data
Operation: ADDC
(A) ← (A) + (C) + #data

SUBB A,<src-byte>
Function: Subtract with borrow
Description: SUBB subtracts the indicated variable and the carry flag together from the Accumulator, leaving the result in the Accumulator. SUBB sets the carry (borrow) flag if a borrow is needed for bit 7 and clears C otherwise. (If C was set *before* executing a SUBB instruction, this indicates that a borrow was needed for the previous step in a
multiple-precision subtraction, so the carry is subtracted from the Accumulator along with the source operand.)
AC is set if a borrow is needed for bit 3 and cleared otherwise. OV is set if a borrow is needed into bit 6, but not into bit 7, or into bit 7, but not bit 6.

When subtracting signed integers, OV indicates a negative number produced when a negative value is subtracted from a positive value, or a positive result when a positive number is subtracted from a negative number.
The source operand allows four addressing modes: register, direct, register-indirect, or immediate.
Example: The Accumulator holds 0C9H (11001001B), register 2 holds 54H (01010100B), and the carry flag is set. The instruction,
SUBB A,R2
will leave the value 74H (01110100B) in the accumulator, with the carry flag and AC cleared but OV set.

Instructions	**OpCode**	**Bytes**	**Flags**
SUBB A,*#data*	0x94	2	C, AC, OV
SUBB A,*iram addr*	0x95	2	C, AC, OV
SUBB A,@R0	0x96	1	C, AC, OV
SUBB A,@R1	0x97	1	C, AC, OV
SUBB A,R0	0x98	1	C, AC, OV
SUBB A,R1	0x99	1	C, AC, OV
SUBB A,R2	0x9A	1	C, AC, OV
SUBB A,R3	0x9B	1	C, AC, OV
SUBB A,R4	0x9C	1	C, AC, OV
SUBB A,R5	0x9D	1	C, AC, OV
SUBB A,R6	0x9E	1	C, AC, OV
SUBB A,R7	0x9F	1	C, AC, OV

SUBB A,Rn
Operation: SUBB
(A) ← (A)(C) - (Rn)

SUBB A, direct
Operation: SUBB

(A) ← (A) - (direct)

SUBB A,@Ri
Operation: SUBB
(A) ← (A) - ((Ri))

SWAP A
Function: Swap nibbles within the Accumulator
Description: SWAP A interchanges the low- and high-order nibbles (four-bit fields) of the Accumulator (bits 3 through 0 and bits 7 through 4). The operation can also be thought of as a 4-bit rotate instruction. No flags are affected.
Example: The Accumulator holds the value 0C5H (11000101B). The instruction,
SWAP A
leaves the Accumulator holding the value 5CH (01011100B

Operation: SWAP
(A3-0) D (A7-4)

XCH A,<byte>

Function: Exchange Accumulator with byte variable
Description: XCH loads the Accumulator with the contents of the indicated variable, at the same time writing the original Accumulator contents to the indicated variable. The source/destination operand can use register, direct, or
register-indirect addressing.
Example: R0 contains the address 20H. The Accumulator holds the value 3FH (0011111lB). Internal RAM location 20H holds the value 75H (01110101B).
The following instruction,
XCH A,@R0
leaves RAM location 20H holding the values 3FH (00111111B) and 75H (01110101B) in the accumulator.

XCHD A,@Ri
Function: Exchange Digit
Description: XCHD exchanges the low-order nibble of the Accumulator (bits 3 through 0), generally representing a hexadecimal or BCD digit, with that of the internal RAM location indirectly addressed by the specified register.
The high-order nibbles (bits 7-4) of each register are not affected. No flags are affected.
Example: R0 contains the address 20H. The Accumulator holds the value 36H (00110110B). Internal RAM location 20H holds the value 75H (01110101B).
The following instruction,
XCHD A,@R0

leaves RAM location 20H holding the value 76H (01110110B) and 35H (00110101B) in the Accumulator.

CPL A

Function: Complement Accumulator
Description: CPLA logically complements each bit of the Accumulator (one's complement). Bits which previously contained a 1 are changed to a 0 and vice-versa. No flags are affected.

Example: The Accumulator contains 5CH (01011100B).
The following instruction,
CPL A
leaves the Accumulator set to 0A3H (10100011B).

CPL bit

Function: Complement bit
Description: CPL bit complements the bit variable specified. A bit that had been a 1 is changed to 0 and vice-versa. No other flags are affected. CLR can operate on the carry or any directly addressable bit.

Example: Port 1 has previously been written with 5BH (01011101B). The following instruction sequence, CPL P1.1CPL
P1.2 leaves the port set to 5BH (01011011B).

DA A

Function: Decimal-adjust Accumulator for Addition
Description: DA A adjusts the eight-bit value in the Accumulator resulting from the earlier addition of two variables (each in packed-BCD format), producing two four-bit digits. Any ADD or ADDC instruction may have been used to perform the addition.
If Accumulator bits 3 through 0 are greater than nine or if the AC flag is one, six is added to the Accumulator producing the proper BCD digit in the low-order nibble. This internal addition sets the carry flag if a carry-out of the low-order four-bit field propagates through all high-order bits, but it does not clear the carry flag otherwise.
If the carry flag is now set, or if the four high-order bits now exceed nine, these high-order bits are incremented by six, producing the proper BCD digit in the high-order nibble. Again, this sets the carry flag if there is a carry-out of the high-order bits, but does not clear the carry. The carry flag thus indicates if the sum of the original two BCD variables is greater than 100, allowing multiple precision decimal addition. OV is not affected.

DEC byte
Function: Decrement

Description: DEC byte decrements the variable indicated by 1. An original value of 00H underflows to 0FFH. No flags are affected.
Example: Register 0 contains 7FH (01111111B). Internal RAM locations 7EH and 7FH contain 00H and 40H, respectively.
The following instruction sequence,
DEC @R0
DEC R0
DEC @R0
leaves register 0 set to 7EH and internal RAM locations 7EH and 7FH set to 0FFH and 3FH.

DEC A
DEC Rn
DEC direct
DEC @Ri

DIV AB
Function: Divide
Description: DIV AB divides the unsigned eight-bit integer in the Accumulator by the unsigned eight-bit integer in register B.
The Accumulator receives the integer part of the quotient; register B receives the integer remainder. The carry and OV flags are cleared.
Exception: if B had originally contained 00H, the values returned in the Accumulator and B-register are undefined and the overflow flag are set. The carry flag is cleared in any case.
Example: The Accumulator contains 251 (0FBH or 11111011B) and B contains 18 (12H or 00010010B). The following instruction,
DIV AB
leaves 13 in the Accumulator (0DH or 00001101B) and the value 17 (11H or 00010001B) in B, since
251 = (13 x 18) + 17. Carry and OV are both cleared.

INC <byte>
Function: Increment
Description: INC increments the indicated variable by 1. An original value of 0FFH overflows to 00H. No flags are affected.
Example: Register 0 contains 7EH (011111110B). Internal RAM locations 7EH and 7FH contain 0FFH and 40H,
respectively. The following instruction sequence,
INC @R0
INC R0
INC @R0

VTU eNotes On Modern Control Theory (Electrical And Electronics Engineering)

Publisher : VTU eLearning

Author : Panel Of Experts