Addressing modes in 8086 microprocessor

Difficulty Level : Easy • Last Updated : 23 Feb, 2022

Prerequisite – Addressing modes, Addressing modes in 8085 microprocessor
The way of specifying data to be operated by an instruction is known as **addressing modes**. This specifies that the given data is an immediate data or an address. It also specifies whether the given operand is register or register pair.

Types of addressing modes:

1. **Register mode –** In this type of addressing mode both the operands are registers.
   Example:

```
MOV AX, BX
XOR AX, DX
ADD AL, BL
```

2. **Immediate mode –** In this type of addressing mode the source operand is a 8 bit or 16 bit data. Destination operand can never be immediate data.
   Example:

```
MOV AX, 2000
MOV CL, 0A
ADD AL, 45
AND AX, 0000
```

   Note that to initialize the value of segment register an register is required.

```
MOV AX, 2000
MOV CS, AX
```

3. **Displacement or direct mode –** In this type of addressing mode the effective address is directly given in the instruction as displacement.
   Example:

```
MOV AX, [DISP]
MOV AX, [0500]
```

4. **Register indirect mode –** In this addressing mode the effective address is in SI, DI or BX.
   Example: Physical Address = Segment Address + Effective Address

```
MOV AX, [DI]
ADD AL, [BX]
MOV AX, [SI]
```

5. **Based indexed mode –** In this the effective address is sum of base register and index register.

```
Base register: BX, BP
Index register: SI, DI
```

The physical memory address is calculated according to the base register.
Example:

```
MOV AL, [BP+SI]
MOV AX, [BX+DI]
```

6. **Indexed mode –** In this type of addressing mode the effective address is sum of index register and displacement.
Example:

```
MOV AX, [SI+2000]
MOV AL, [DI+3000]
```

7. **Based mode –** In this the effective address is the sum of base register and displacement.
Example:

```
MOV AL, [BP+ 0100]
```

8. **Based indexed displacement mode –** In this type of addressing mode the effective address is the sum of index register, base register and displacement.
Example:

```
MOV AL, [SI+BP+2000]
```

9. **String mode –** This addressing mode is related to string instructions. In this the value of SI and DI are auto incremented and decremented depending upon the value of directional flag.
Example:

```
MOVS B
MOVS W
```

10. **Input/Output mode –** This addressing mode is related with input output operations.
Example:

```
IN A, 45
OUT A, 50
```

11. **Relative mode –**
In this the effective address is calculated with reference to instruction pointer.
Example:

```
JNZ 8 bit address
IP=IP+8 bit address
```

Data transfer instructions in 8086 microprocessor

Difficulty Level : Basic • Last Updated : 30 May, 2022

Data transfer instructions are the instructions which transfers data in the microprocessor. They are also called copy instructions.

Following is the table showing the list of data transfer instructions:

OPCODE	OPERAND	EXPLANATION	EXAMPLE
MOV	D, S	D = S	MOV AX, [SI]
PUSH	D	pushes D to the stack	PUSH DX
POP	D	pops the stack to D	POP AS
PUSHA	none	put all the registers into the stack	PUSHA
POPA	none	gets words from the stack to all registers	POPA
XCHG	D, S	exchanges contents of D and S	XCHG [2050], AX
IN	D, S	copies a byte or word from S to D	IN AX, DX
OUT	D, S	copies a byte or word from D to S	OUT 05, AL
XLAT	none	translates a byte in AL using a table in the memory	XLAT
LAHF	none	loads AH with the lower byte of the flag register	LAHF
SAHF	none	stores AH register to lower byte of the flag register	SAHF
PUSHF	none	copies the flag register at the top of the stack	PUSHF
POPF	none	copies a word at the top of the stack to the flag register	POPF

Here D stands for destination and S stands for source.
D and S can either be register, data or memory address.

Logical instructions in 8086 microprocessor

Last Updated : 10 May, 2022

Logical instructions are the instructions which perform basic logical operations such as AND, OR, etc. In 8086 microprocessor, the destination operand need not be the accumulator. Following is the table showing the list of logical instructions:

OPCODE	OPERAND	DESTINATION	EXAMPLE
AND	D, S	D = D AND S	AND AX, 0010
OR	D, S	D = D OR S	OR AX, BX
NOT	D	D = NOT of D	NOT AL
XOR	D, S	D = D XOR S	XOR AL, BL
TEST	D, S	performs bit-wise AND operation and affects the flag register	TEST [0250], 06
SHR	D, C	shifts each bit in D to the right C times and 0 is stored at MSB position	SHR AL, 04
SHL	D, C	shifts each bit in D to the left C times and 0 is stored at LSB position	SHL AX, BL
ROR	D, C	rotates all bits in D to the right C times	ROR BL, CL
ROL	R, C	rotates all bits in D to the left C times	ROL BX, 06
RCR	D, C	rotates all bits in D to the right along with carry flag C times	RCR BL, CL
RCL	R, C	rotates all bits in D to the left along with carry flag C times	RCL BX, 06

Here D stands for destination, S stands for source and C stands for count. They can either be register, data or memory address.

Next

Data transfer instructions in 8086 microprocessor

Related Articles

String manipulation instructions in 8086 microprocessor

Last Updated : 04 Mar, 2022

String is a series of data byte or word available in memory at consecutive locations. It is either referred as byte string or word string. Their memory is always allocated in a sequential order. Instructions used to manipulate strings are called string manipulation instructions.

Following is the table showing the list of string manipulation instructions:

OPCODE	OPERAND	EXPLANATION	EXAMPLE
REP	instruction	repeat the given instruction till CX != 0	REP MOVSB
REPE	instruction	repeat the given instruction while CX = 0	REPE
REPZ	instruction	repeat the given instruction while ZF = 1	REPZ
REPNE	instruction	repeat the given instruction while CX != 0	REPNE
REPNZ	instruction	repeat the given instruction while ZF = 0	REPNZ
MOVSB	none	moves contents of byte given by DS:SI into ES:DI	MOVSB
MOVSW	none	moves contents of word given by DS:SI into ES:DI	MOVSW
MOVD	none	moves contents of double word given by DS:SI into ES:DI	MOVD
LODSB	none	moves the byte at address DS:SI into AL; SI is incr/decr by 1	LODSB
LODSW	none	moves the word at address DS: SI into AX; SI is incr/decr by 2	LODSW
LODSD	none	moves the double word at address DS:SI into EAX; SI is incr/decr by 4	LODSD
STOSB	none	moves contents of AL to byte address given by ES:DI; DI is incr/dec by 1	STOSB
STOSW	none	moves the contents of AX to the word address given by ES:DI; DI is incr/decr by 2	STOSW
STOSD	none	moves contents of EAX to the DOUBLE WORD address given by ES:DI; DI is incr/decr by 4	STOSD
SCASB	none	compares byte at ES:DI with AL and sets flags according to result	SCASB
SCASW	none	compares word at ES:DI with AX and sets flags	SCASW
SCASD	none	compares double word at ES:DI with EAX and sets flags	SCASD
CMPSB	none	compares byte at ES:DI with byte at DS:SI and sets flags	CMPSB
CMPSW	none	compares word at ES:DI with word at DS:SI and sets flags	CMPSW