Topic-III

80X86 Addressing Modes

T1. Barry B Brey, The Intel Microprocessors .Pearson, Eight Ed. 2009. Chapter 3, 4

Jan 30th --Feb 6th 2021

Types of Instructions

- Data Transfer Instructions
- Arithmetic Instructions
- Logical Instructions
- Branch and Program control Instructions

Addressing Modes

Instruction = Opcode, Operand

Opcode/ Operation Field - the type of operation which is to be performed by processor

Operand – the data on which the operation is going to be performed

The way in which an operand is specified in an instruction is called as *Addressing Mode*.

Addressing Modes

- The processor executes an instruction-
- it performs the specified function on data.
- Data called operands
- May be a part of the instruction
- May reside in one of the internal registers of the microprocessors
- May be stored at an address in memory

Addressing Modes

- Register Addressing
- Immediate Addressing
- Direct Addressing
- Register Indirect Addressing
- Base-plus-index Addressing
- Register Relative Addressing
- Base relative -plus-indexed Addressing
- Scaled Indexed Addressing

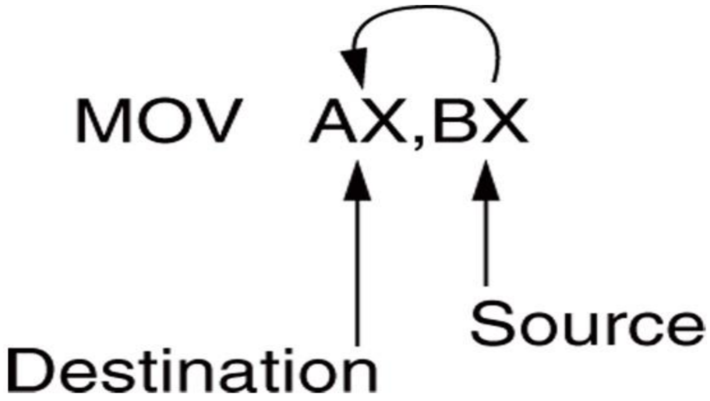
These data-addressing modes are found with all versions of the Intel microprocessor. except for the scaled-index-addressing mode, found only in 80386 through Core2

MOV Instruction

- MOV destination, source

DATA ADDRESSING MODES

- MOV instruction is a common and flexible instruction.
  - provides a basis for explanation of data-addressing modes
- **Source** is to the right and **destination** the left, next to the opcode MOV.
  - an **opcode**, or operation code, tells the microprocessor which operation to perform



Data Transfer Instructions

- **MOV DST, SRC**

➢ Copies the content of source to destination
➢ No Flags Affected
➢ Size of source and destination must be the same
➢ Source can be register, memory, or immediate data
➢ Destination can be register or memory location

Different MOV options

R ← M

M ← R

R ← R

M ← I

R ← I

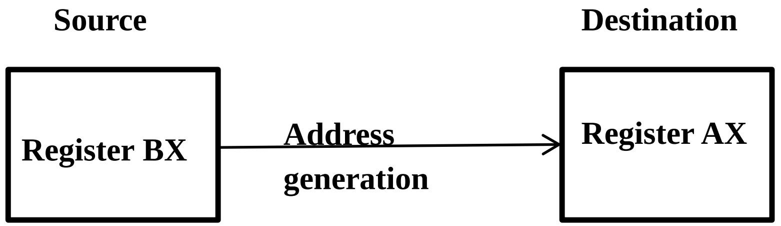
Addressing Modes

- Register Addressing
- Immediate Addressing
- Direct Addressing
- Register Indirect Addressing
- Base-plus-index Addressing
- Register Relative Addressing
- Base relative -plus-indexed Addressing
- Scaled Indexed Addressing

Addressing Modes

- **Register Addressing**

➢ MOV AX, BX



Source

Destination

Register BX

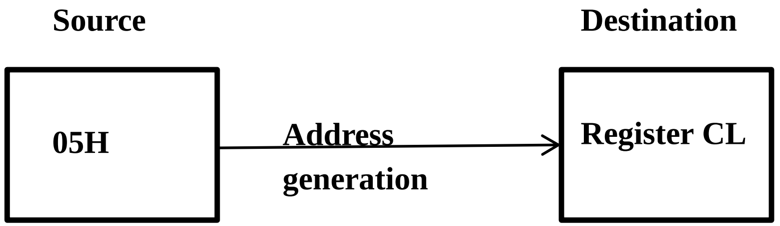
Address generation

Register AX

- **Immediate Addressing**

➢ MOV AX, 1420_H

➢ MOV CL, 05_H



Source

Destination

05H

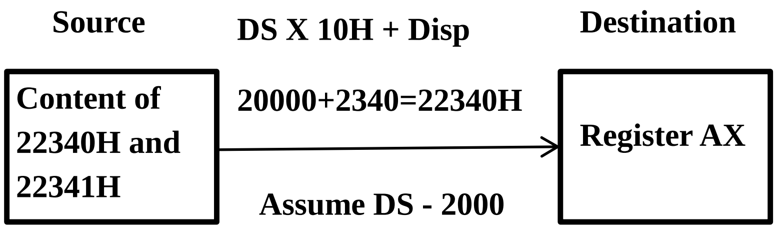
Address generation

Register CL

Addressing Modes

- **Direct Addressing**

➢ MOV AX, [2340_H]



Source

DS X 10H + Disp

Destination

Content of 22340H and 22341H

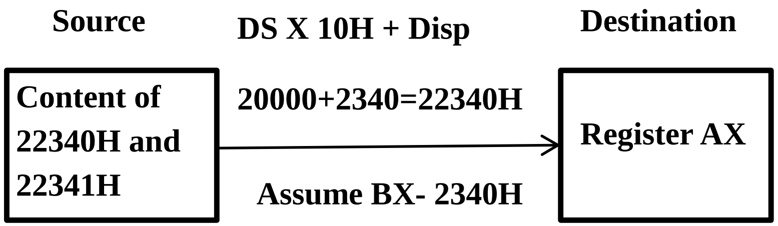
20000+2340=22340H

Register AX

Assume DS - 2000

- **Register Indirect Addressing**

➢ MOV AX, [BX]



Source

DS X 10H + Disp

Destination

Content of 22340H and 22341H

20000+2340=22340H

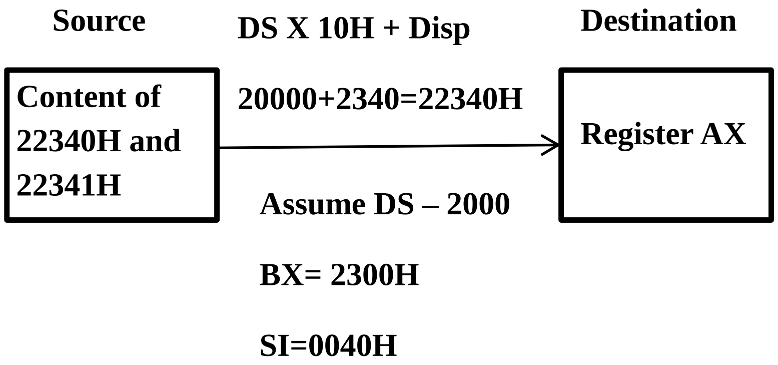
Register AX

Assume BX- 2340H

Addressing Modes

- **Base-plus-index Addressing**

➢ MOV AX, [BX+SI]



Source

DS X 10H + Disp

Destination

Content of 22340H and 22341H

20000+2340=22340H

Register AX

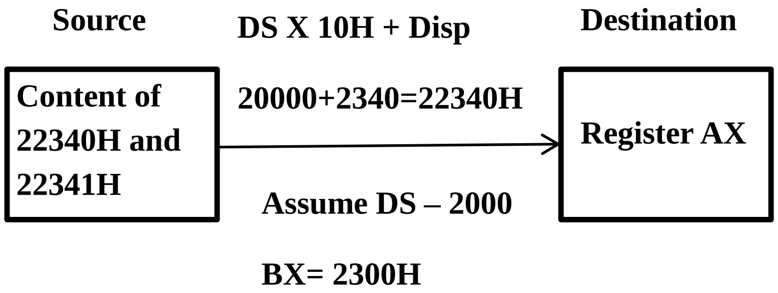
Assume DS – 2000

BX= 2300H

SI=0040H

- **Register Relative Addressing**

➢ MOV AX, [BX+40]



Source

DS X 10H + Disp

Destination

Content of 22340H and 22341H

20000+2340=22340H

Register AX

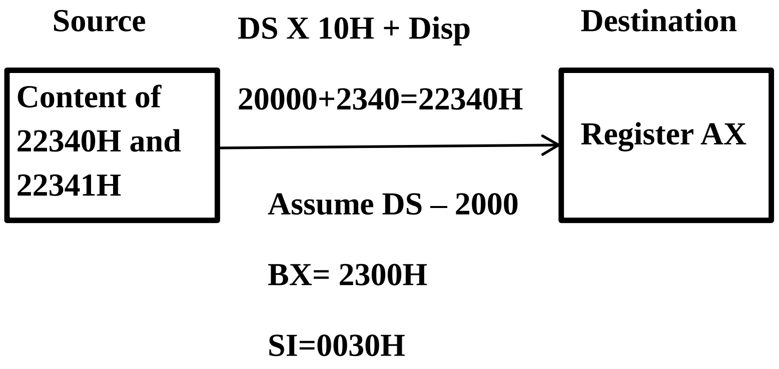
Assume DS – 2000

BX= 2300H

Addressing Modes

- **Base relative-plus-indexed Addressing**

➢ MOV AX, [BX+SI+10]



Source

DS X 10H + Disp

Destination

Content of 22340H and 22341H

20000+2340=22340H

Register AX

Assume DS – 2000

BX= 2300H

SI=0030H

- Scaled Indexed Addressing

Addressing Modes

- The microprocessor contains these 8-bit register names used with register addressing: AH, AL, BH, BL, CH, CL, DH, and DL.

- 16-bit register names: AX, BX, CX, DX, SP, BP, SI, and DI.

- In 80386 & above, extended 32-bit register names are: EAX, EBX, ECX, EDX, ESP, EBP, EDI, and ESI.

- 64-bit mode register names are: RAX, RBX, RCX, RDX, RSP, RBP, RDI, RSI, and R8 through R15.

Important for instructions

- To use registers that are the same size.
  - *never* mix an 8-bit \with a 16-bit register, an 8- or a 16-bit register with a 32-bit register
  - this is not allowed by the microprocessor and results in an error when assembled
- If hexadecimal data begin with a letter, the assembler requires the data start with a **0**.
  - to represent a hexadecimal F2, 0F2H is used in assembly language

- The source register’s contents do not change.

- the destination register’s contents do change

- The contents of the destination register or destination memory location change for all instructions except the CMP and TEST instructions.

- The MOV BL, CL instruction does not affect the leftmost 8 bits of register BX.

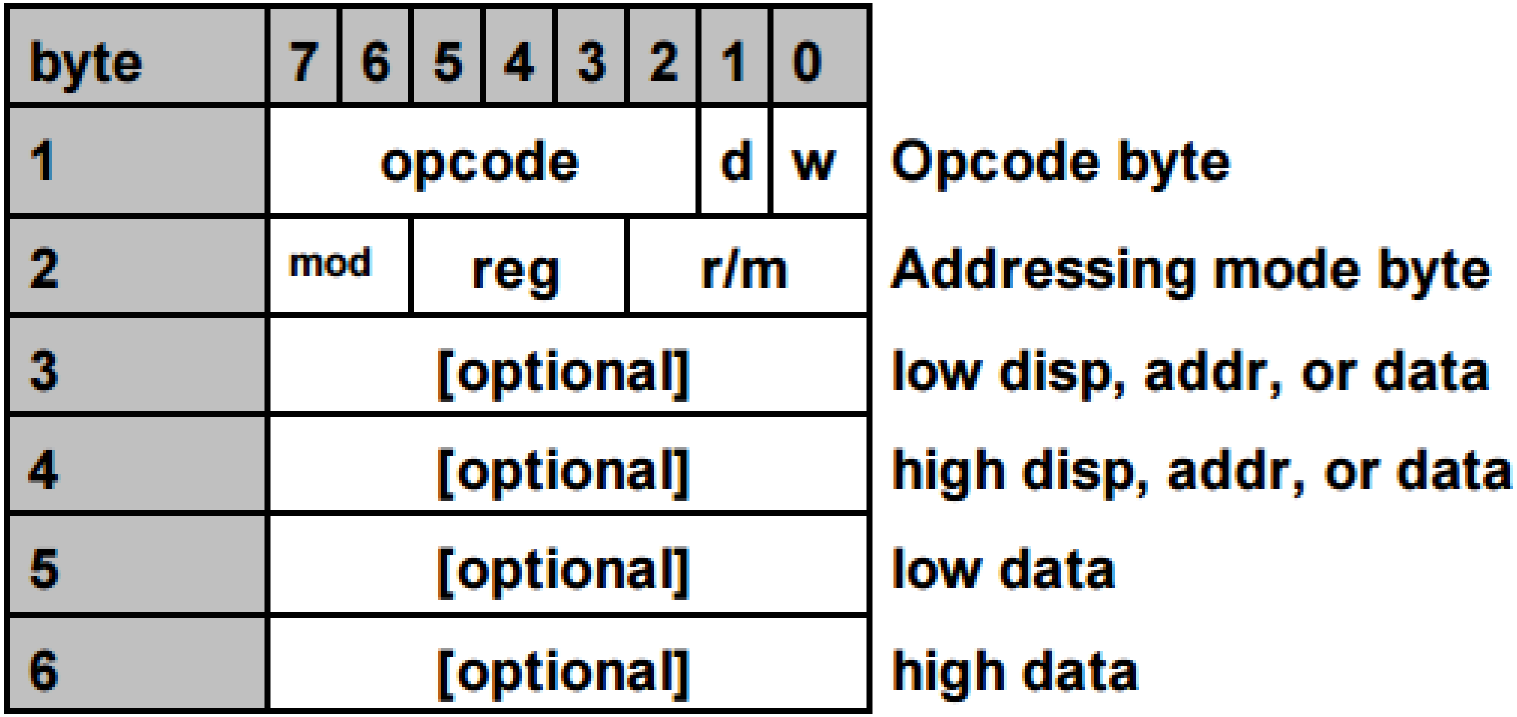
- In some cases, indirect addressing requires specifying the size of the data by the **special assembler directive** BYTE PTR, WORD PTR, DWORD PTR, or QWORD PTR.
  - these directives indicate the size of the memory data addressed by the memory **pointer** (PTR)

MOV Instruction

Assembly language	Size	Operation
MOV AL,BL	8-bits	Copies BL into AL
MOV AX,CX	16-bits	Copies CX into AX
MOV SP,BP	16-bits	Copies BP into SP
MOV AL,BX	16-bits	Not allowed
MOV CS,SS	-----	Not allowed
MOV CS,AX	-----	Not allowed

Encoding of 8086 Instructions

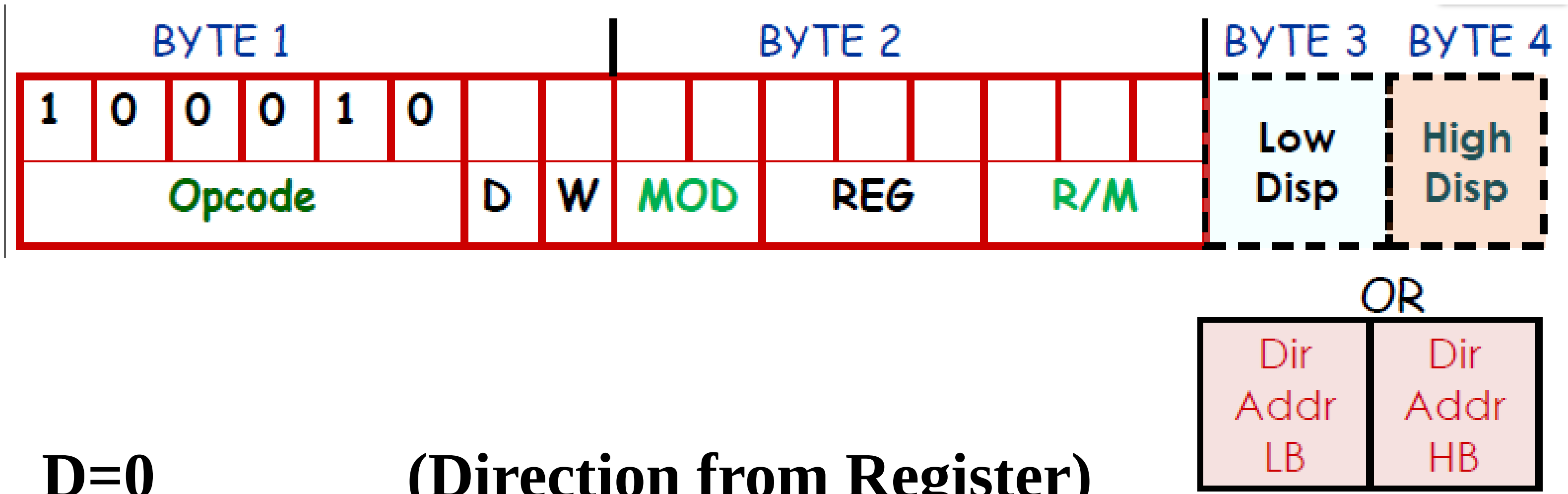
- ➢ 8086 Instructions are represented as binary numbers.
- ➢ In 8086, Instructions size can be in between 1 to 6 bytes



byte	7	6	5	4	3	2	1	0	
1	opcode						d	w	Opcode byte
2	mod		reg			r/m			Addressing mode byte
3	[optional]								low disp, addr, or data
4	[optional]								high disp, addr, or data
5	[optional]								low data
6	[optional]								high data

This is the general instruction format used by the majority of 2-operand instructions

Addressing Modes in 8086



BYTE 1 | BYTE 2 | BYTE 3 | BYTE 4

1	0	0	0	1	0											Low Disp	High Disp
Opcode						D	W	MOD		REG			R/M				

OR

Dir Addr LB	Dir Addr HB

D=0 (Direction from Register)

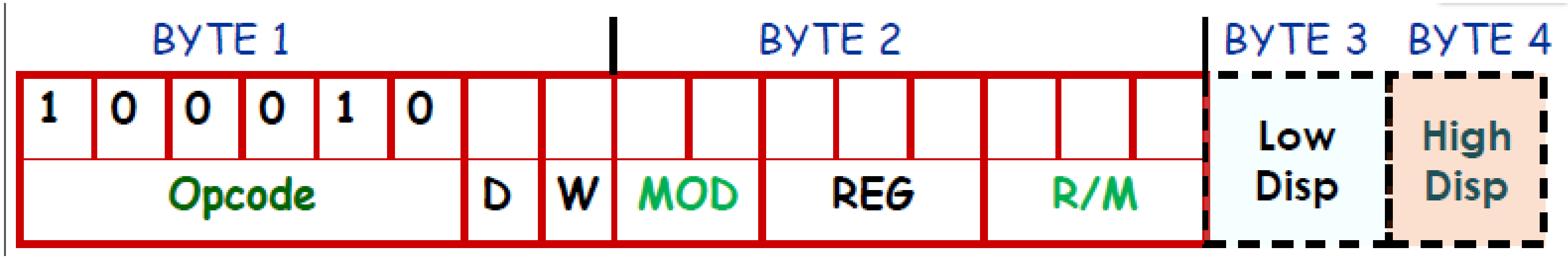
D=1 (Direction to Register)

W=0 (Data-byte)

W=1 (Data-word)

MOD+ R/M - Addressing Modes

Addressing Mode Byte 1



BYTE 1								BYTE 2								BYTE 3	BYTE 4
1	0	0	0	1	0											Low Disp	High Disp
Opcode						D	W	MOD		REG			R/M				

D (direction) field

D (direction) field specifies the direction of data movement:

D = 1 data moves from operand specified by R/M field to operand specified by REG field

D = 0 data moves from operand specified by REG field to operand specified by R/M field

Addressing Mode **Byte 1**

W (word/byte)

W (word/byte) specifies operand size

W = 1 data is word.

W = 0 data is byte.

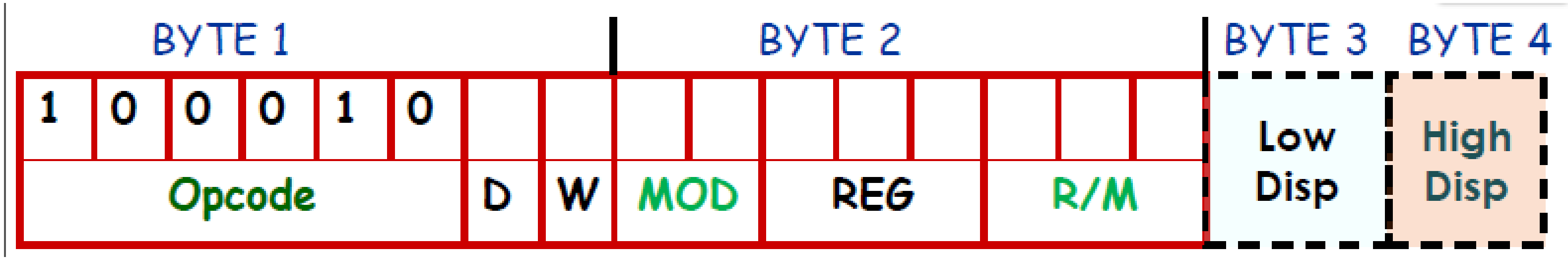
W (word/byte) in 80386 (32 bit processor)

W (word/byte) specifies operand size

W = 1 data is 32 bit.

W = 0 data is byte.

Addressing Mode Byte 2



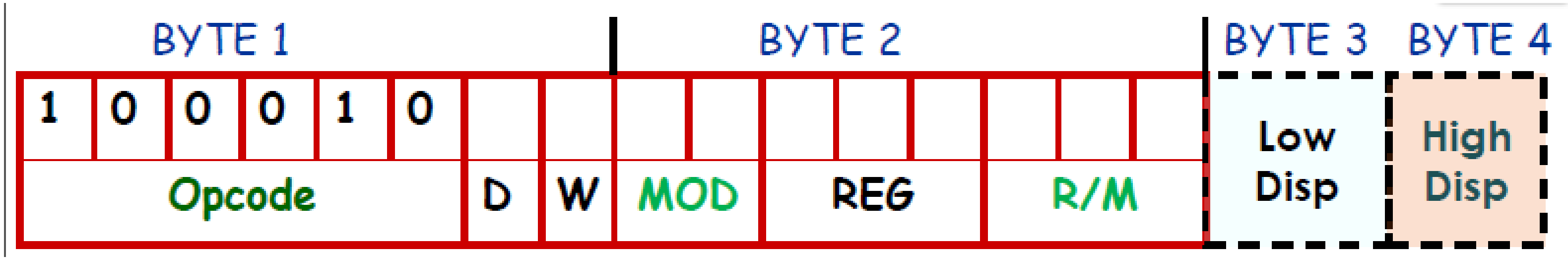
Contains three fields

Mod 2 Bits (mode; determines how R/M field is interpreted)

Reg 3 Bits (register) or SREG (Seg register)

R/M 3 Bits (register/memory)

Addressing Mode Byte 2



MOD (2 bits)

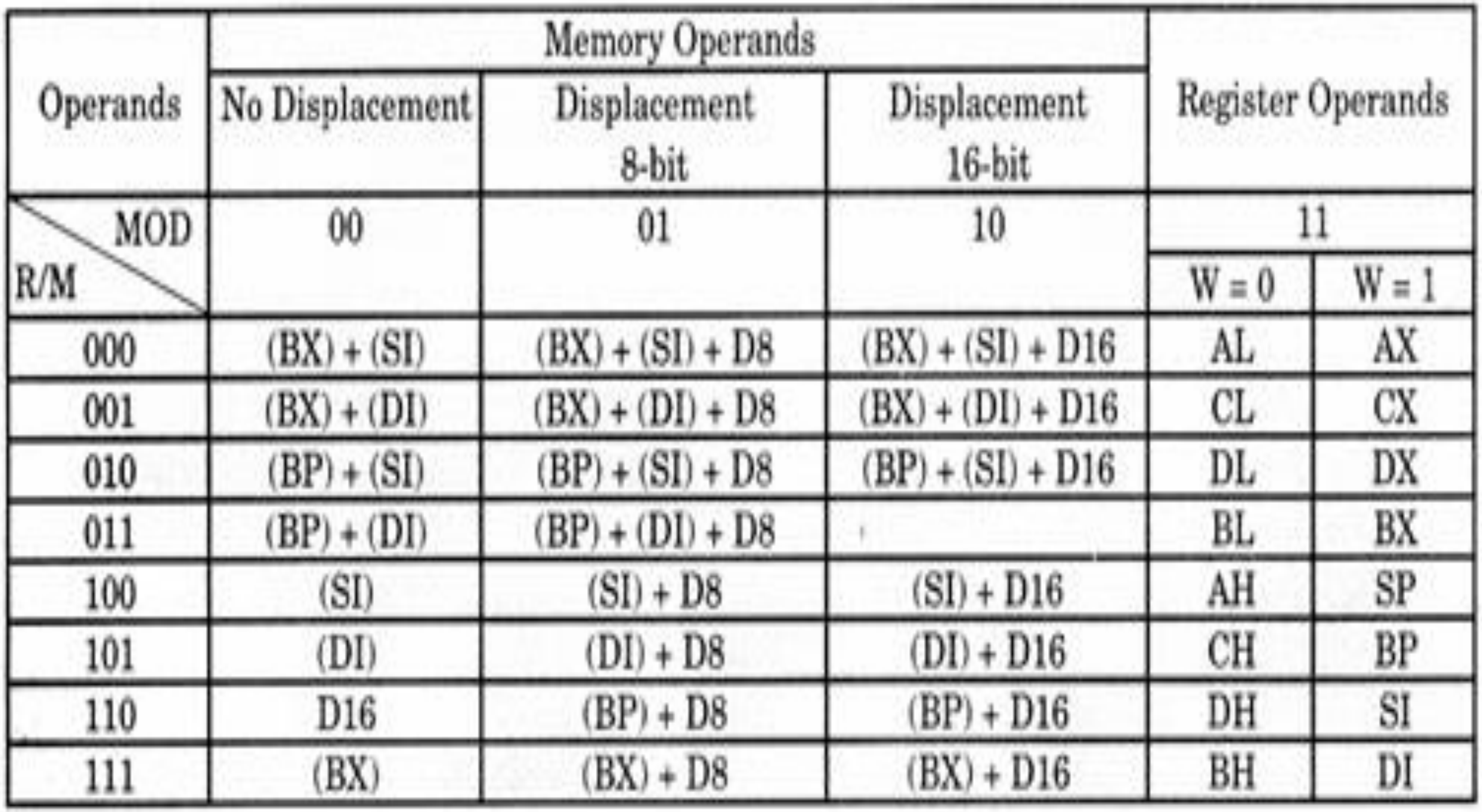
MOD = 00 Memory operand with no displacement

MOD = 01 Memory operand with 8 bit displacement

MOD = 10 Memory operand with 16 bit displacement

MOD = 11 Register operand

MOD with R/M (16 bits)



Operands	Memory Operands			Register Operands	
	No Displacement	Displacement 8-bit	Displacement 16-bit		
MOD R/M	00	01	10	11	
				W = 0	W = 1
000	(BX) + (SI)	(BX) + (SI) + D8	(BX) + (SI) + D16	AL	AX
001	(BX) + (DI)	(BX) + (DI) + D8	(BX) + (DI) + D16	CL	CX
010	(BP) + (SI)	(BP) + (SI) + D8	(BP) + (SI) + D16	DL	DX
011	(BP) + (DI)	(BP) + (DI) + D8		BL	BX
100	(SI)	(SI) + D8	(SI) + D16	AH	SP
101	(DI)	(DI) + D8	(DI) + D16	CH	BP
110	D16	(BP) + D8	(BP) + D16	DH	SI
111	(BX)	(BX) + D8	(BX) + D16	BH	DI

MOD with R/M (32 bits)

MOD (R/M)	00	01	10	11	
				W=0	W=1
000	EAX	EAX +D8	EAX +D32	AL	EAX
001	ECX	ECX +D8	ECX +D32	CL	ECX
010	EDX	EDX +D8	EDX +D32	DL	EDX
011	EBX	EBX +D8	EBX +D32	BL	EBX
100	SCALED INDEX	SCALED INDEX+D8	SCALED INDEX+D32	AH	ESP
101	D32	EBP+D8	EBP+D32	CH	EBP
110	ESI	ESI+D8	ESI+D32	DH	ESI
111	EDI	EDI+D8	EDI+D32	BH	EDI

Registers (REG)

EAX/AX/AL	000
EBX/BX/BL	011
ECX/CX/CL	001
EDX/DX/DL	010
ESP/SP/AH	100
EBP/BP/CH	101
ESI/SI/DH	110
EDI/DI/BH	111

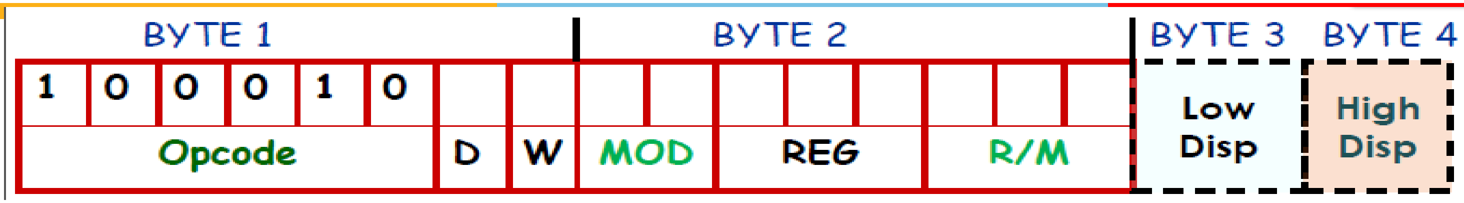
MOD = 00 Memory operand with no displacement

Examples

MOV AX,[2A45]

MOV AX,[DI]

MOD = 00 Memory operand with no displacement



BYTE 1			BYTE 2			BYTE 3	BYTE 4
1 0 0 0 1 0							
Opcode	D	W	MOD	REG	R/M	Low Disp	High Disp

MOV [BX],CL

- w = 0 because we are dealing with a byte
d = 0 because REG to R/M

- therefore first byte is (1000 1000) = 88H

- since no displacement,

- we can use MOD=00 REG=001 and R/M=111 = 0000 1111 = 0FH

result: 88 0F

MOD = 01 Memory operand with 8 bits displacement

Examples

MOV AX,[BP+2]

MOV DX,[BX+DI+4]

MOV [BX-4],AX

MOD = 01 Memory operand with 8 bits displacement

Examples

MOV [BX+10h],CL

- w = 0 because we are dealing with a byte
- d = 0 because REG to R/M
- therefore first byte is (1000 1000) = 88H

- since 10H can be encoded as an 8-bit displacement, we can use MOD=01 REG=001 and R/M=111 = 0100 1111 = 4FH

and the last byte is 10H

result: 88 4F 10

Note: MOV [BX+10H],CX = 89 4F 10

MOD = 10 Memory operand with 16 bits displacement

Examples

ADD AX,[BX+1000H]

MOV [BX+10h],CL with a 16-bit displacement,

(MOD 10)

88 8F 10 00

MOD = 11 Register operand

Examples

MOV AX, BX

- w = 1 because we are dealing with words
MOD = 11 because it is register-register

- if d = 0 then REG = source (BX) and R/M = dest (AX)
= 1000 1001 1101 1000 (89 D8)

- if d = 1 then REG = source (AX) and R/M = dest (BX)
= 1000 1011 1100 0011 (8B C3)

Instruction set using addressing modes

Addressing Modes

- Register Addressing
- Immediate Addressing
- Direct Addressing
- Register Indirect Addressing
- Base-plus-index Addressing
- Register Relative Addressing
- Base relative -plus-indexed Addressing
- Scaled Indexed Addressing

These data-addressing modes are found with all versions of the Intel microprocessor. except for the scaled-index-addressing mode, found only in 80386 through Core2

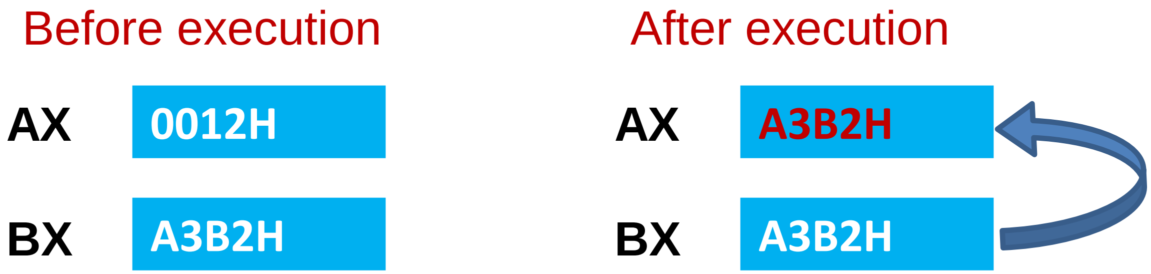
INSTRUCTION FORMAT

BYTE 1			BYTE 2			BYTE 3	BYTE 4
						LOW DISP.	HIGH DISP.
OPCODE	D	W	MOD	REG	R/M		

BYTE 1			BYTE 2	BYTE 3
	1 BIT	3 BITS	LOW DISP.	HIGH DISP.
OPCODE	W	REG		

Register Addressing

MOV AX,BX ; Copies contents of BX to AX register



Before execution

AX 0012H

BX A3B2H

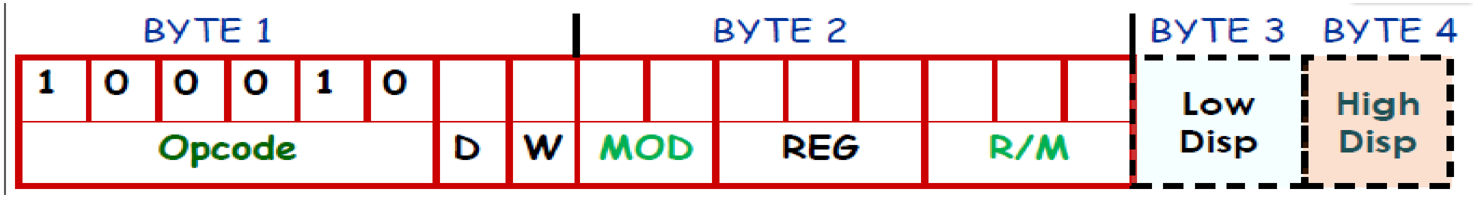
After execution

AX A3B2H

BX A3B2H

Register Addressing

MOV AX,BX



BYTE 1 BYTE 2 BYTE 3 BYTE 4

1 0 0 0 1 0

Opcode D W MOD REG R/M

Low Disp

High Disp

1	0	0	0	1	0	0	1	1	1	0	1	1	0	0	0

89D8

1	0	0	0	1	0	1	1	1	1	0	0	0	0	1	1

8BC3

Immediate Addressing

- Transfers the source-immediate byte or word of data into the destination register or memory location.

- Source operand is always immediate data that is a constant.

- Destination operand can be a register or a memory location.

- Immediate addressing operates on a byte or word of data.

Immediate Addressing

Assembly language	Size	Operation
MOV AL,44	8-bits	Copies 44 decimal into AL
MOV AX,44H	16-bits	Copies 0044 H into AX
MOV CL,11001110B	8-bits	Copies a 11001110 binary into CL

Immediate Addressing

MOV AH, $4C_H$

- **(AH) ← 0100 1100**

- **Before Execution AX = 9844_H**
- **After Execution AX = $4C44_H$**

Immediate Addressing

MOV AH, $4C_H$ - 1011 W REG

BYTE 1						BYTE 2	BYTE 3
				1 BIT	3 BITS	LOW DISP.	HIGH DISP.
OPCODE				W	REG		

1	0	1	1	0	1	0	0

B4 4C

Immediate Addressing

MOV CX, AD4C$_H$

(CX) ← 1010 1101 0100 1100

- **Before Execution CX = 9844$_H$**
- **After Execution CX = AD4C$_H$**

Immediate Addressing

MOV CX, AD4C$_H$ **-1011 W REG**

1	0	1	1	1	0	0	1

B94CAD

Little Endian

Little vs Big Endian

Little vs Big Endian (5627)

00000	27	00000	56
00001	56	00001	27

Little vs Big Endian (A0 49 56 27**)**

00000	27	00000	A0
00001	56	00001	49
00002	49	00002	56
00003	A0	00003	27

Direct Addressing:

MOV [1234H], AX

(M) ← AX

M = DS: 1234H ← (AL)

M = DS: 1235H ← (AH)

Direct Addressing:

- Moves a byte or word between a memory location and register.
  - Example: MOV AL,[1234H]

Direct Addressing

MOV AX, [1234$_H$]

(AX) ← DS:1234

- **DS = 0100$_H$**
- **ADDRESS = 01000+1234=02234**
- **02234$_H$** 54
- **02235$_H$** 82

- **AX = 8254$_H$**

Direct Addressing

MOV AX, $[1234_H]$

1	0	0	0	1	0	1	1	0	0	0	0	0	1	1	0

8B06 34 12

MOV $[1234_H]$, AL

1 0 0 0 1 0 0 0 00 000 110

88 06 34 12

Example 1:

MOV [1234H],AL ; Copies contents of AL into memory location DS:1234H

Program memory

02000	**88**
02001	06
02002	34
02003	12

CS = 0200, IP = 0000

Data memory

01000	**12**
01001	87
-------	------
02234	**26**

DS = 0100, Offset = 1234

Before execution

AL	**54H**

After execution

02234	**54H**
AL	**54H**

Example 2:

MOV CX,[1234H] ; Copies contents of memory location DS:1234H into CX

Program memory

02000	8B
02001	0E
02002	34
02003	12

CS = 0200, IP = 0000

Data memory

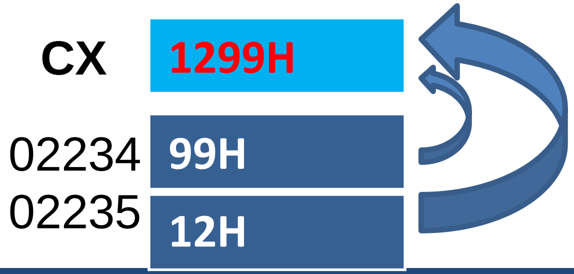
01000	12
01001	87
-------	------
02234	99
02235	12

DS = 0100, Offset = 1234

Before execution

CX	0045H

After execution



CX	1299H
02234	99H
02235	12H

Direct Addressing

MOV [D600$_H$], BX

[D600] ← (BX)

- **DS = 2000$_H$**
- **ADDRESS =** **20000+D600=2D600**
- **BX = 8A17$_H$**
- **2D600$_H$** 17
- **2D601$_H$** 8A

Direct Addressing

MOV [D600$_H$], BX

1	0	0	0	1	0	0	1	0	0	0	1	1	1	1	0

891E 00 D6

Direct Addressing

MOV [$\mathrm{D600_H}$], BH

1	0	0	0	1	0	0	0	0	0	1	1	1	1	1	0

883E 00 D6

Direct Addressing

MOV EAX, [1234_H]

(EAX) ← DS:1234

- **DS = 2000_H**
- **ADDRESS = 20000+1234=21234**

21234_H	74
21235_H	82
21236_H	A3
21237_H	45

- **EAX = 45 A3 82 74_H**

Register indirect Addressing:

MOV [BX], CL - Register Indirect addressing

- Transfers a byte or word between a register and a memory location addressed by an index or base register.
- Offset address can be in any of these registers BP,BX,DI and SI.
- Data segment is used by default with register indirect addressing.

Example: Register Indirect

MOV [BX],AL ; Copies contents AL to memory location DS:BX

Program memory

02000	**88**
02001	07
02002	
02003	

CS = 0200, IP = 0000

Data memory

01000	**12**
01001	87
-------	------
01012	**92**

DS = 0100, BX = 0012

Before execution

AL	**12H**
01012	**92H**

After execution

01012	**12H**
AL	**12H**

Lookup table with register indirect addressing

```
CS: 0200, DS: 0100
        MOV AL,0
        MOV CL,04H
        MOV BX,0000H
AGAIN:  MOV [BX],AL
        INC AL
        INC BX
        LOOP AGAIN
```

	H	L
AX		
BX		
CX		

Physical Address	Value

DS = 0100

Lookup table with register indirect addressing

CS: 0200, DS: 0100

```
        MOV AL,0
        MOV CL,04H
        MOV BX,0000H
AGAIN:   MOV [BX],AL
        INC AL
        INC BX
        LOOP AGAIN
```

	H	L
AX		
BX		
CX		

Physical Address	Value

DS = 0100

Lookup table with register indirect addressing

CS: 0200, DS: 0100

```
        MOV AL,0
        MOV CL,04H
        MOV BX,0000H
AGAIN:  MOV [BX],AL
        INC AL
        INC BX
        LOOP AGAIN
```

	H	L
AX	xx	00
BX		
CX	xx	04

Physical Address	Value

DS = 0100

Lookup table with register indirect addressing

CS: 0200, DS: 0100

```
        MOV AL,0
        MOV CL,04H
        MOV BX,0000H
AGAIN:  MOV [BX],AL
        INC AL ⬅
        INC BX
        LOOP AGAIN
```

	H	L
AX	xx	00
BX	00	00
CX	xx	04

DS*10H+BX	01000

Physical address

Physical Address	Value
01000	00

DS = 0100

Lookup table with register indirect addressing

CS: 0200, DS: 0100

```
        MOV AL,0
        MOV CL,04H
        MOV BX,0000H
AGAIN:   MOV [BX],AL
        INC AL
        INC BX ←
        LOOP AGAIN
```

	H	L
AX	xx	01
BX	00	00
CX	xx	04

DS*10H+BX	01000

Physical address

Physical Address	Value
01000	00

DS = 0100

Lookup table with register indirect addressing

CS: 0200, DS: 0100

```
        MOV AL,0
        MOV CL,04H
        MOV BX,0000H
AGAIN:  MOV [BX],AL
        INC AL
        INC BX
        LOOP AGAIN
```

	H	L
AX	xx	01
BX	00	01
CX	xx	04

DS*10H+BX	01000

Physical address

Physical Address	Value
01000	00

DS = 0100

Lookup table with register indirect addressing

```
CS: 0200, DS: 0100
        MOV AL,0
        MOV CL,04H
        MOV BX,0000H
AGAIN:  MOV [BX],AL ⬅
        INC AL
        INC BX
        LOOP AGAIN
```

	H	L
AX	xx	01
BX	00	01
CX	xx	03

DS*10H+BX	01000

Physical address

Physical Address	Value
01000	00

DS = 0100

Lookup table with register indirect addressing

```
CS: 0200, DS: 0100
        MOV AL,0
        MOV CL,04H
        MOV BX,0000H
AGAIN:    MOV [BX],AL
        INC AL ←
        INC BX
        LOOP AGAIN
```

	H	L
AX	xx	01
BX	00	01
CX	xx	03

DS*10H+BX	01001

Physical address

Physical Address	Value
01000	00
01001	01

DS = 0100

Lookup table with register indirect addressing

```
CS: 0200, DS: 0100
          MOV AL,0
          MOV CL,04H
          MOV BX,0000H
AGAIN:    MOV [BX],AL
          INC AL
          INC BX ⬅
          LOOP AGAIN
```

	H	L
AX	xx	02
BX	00	01
CX	xx	03

DS*10H+BX	01001

Physical address

Physical Address	Value
01000	00
01001	01

DS = 0100

Lookup table with register indirect addressing

```
CS: 0200, DS: 0100
          MOV AL,0
          MOV CL,04H
          MOV BX,0000H
AGAIN:    MOV [BX],AL
          INC AL
          INC BX
          LOOP AGAIN ⬅
```

	H	L
AX	xx	02
BX	00	02
CX	xx	03

DS*10H+BX	01001

Physical address

Physical Address	Value
01000	00
01001	01

DS = 0100

Lookup table with register indirect addressing

CS: 0200, DS: 0100

```
            MOV AL,0
            MOV CL,04H
            MOV BX,0000H
AGAIN:      MOV [BX],AL
            INC AL
            INC BX
            LOOP AGAIN ⬅
```

	H	L
AX	xx	02
BX	00	02
CX	xx	03

DS*10H+BX	01001

Physical address

Physical Address	Value
01000	00
01001	01

DS = 0100

Lookup table with register indirect addressing

CS: 0200, DS: 0100

```
        MOV AL,0
        MOV CL,04H
        MOV BX,0000H
AGAIN:  MOV [BX],AL   ⬅
        INC AL
        INC BX
        LOOP AGAIN
```

	H	L
AX	xx	02
BX	00	02
CX	xx	02

DS*10H+BX	01001

Physical address

Physical Address	Value
01000	00
01001	01

DS = 0100

Lookup table with register indirect addressing

CS: 0200, DS: 0100

```
          MOV AL,0
          MOV CL,04H
          MOV BX,0000H
AGAIN:    MOV [BX],AL
          INC AL   ⬅
          INC BX
          LOOP AGAIN
```

	H	L
AX	xx	02
BX	00	02
CX	xx	01

DS*10H+BX	01002

Physical address

Physical Address	Value
01000	00
01001	01
01002	02

DS = 0100

Lookup table with register indirect addressing

CS: 0200, DS: 0100

```
         MOV AL,0
         MOV CL,04H
         MOV BX,0000H
AGAIN:   MOV [BX],AL
         INC AL
         INC BX  ⬅
         LOOP AGAIN
```

	H	L
AX	xx	03
BX	00	02
CX	xx	01

DS*10H+BX	01002

Physical address

Physical Address	Value
01000	00
01001	01
01002	02

DS = 0100

Lookup table with register indirect addressing

CS: 0200, DS: 0100

```
        MOV AL,0
        MOV CL,04H
        MOV BX,0000H
AGAIN:  MOV [BX],AL
        INC AL
        INC BX
        LOOP AGAIN ⬅
```

	H	L
AX	xx	03
BX	00	03
CX	xx	01

DS*10H+BX	01002

Physical address

Physical Address	Value
01000	00
01001	01
01002	02

DS = 0100

Lookup table with register indirect addressing

CS: 0200, DS: 0100

```
        MOV AL,0
        MOV CL,04H
        MOV BX,0000H
AGAIN:  MOV [BX],AL   ⬅
        INC AL
        INC BX
        LOOP AGAIN
```

	H	L
AX	xx	03
BX	00	03
CX	xx	00

DS*10H+BX	01002

Physical address

Physical Address	Value
01000	00
01001	01
01002	02

DS = 0100

Lookup table with register indirect addressing

CS: 0200, DS: 0100

```
        MOV AL,0
        MOV CL,04H
        MOV BX,0000H
AGAIN:  MOV [BX],AL
        INC AL  ⬅
        INC BX
        LOOP AGAIN
```

	H	L
AX	xx	03
BX	00	03
CX	xx	00

DS*10H+BX	01003

Physical address

Physical Address	Value
01000	00
01001	01
01002	02
01003	03

DS = 0100

Lookup table with register indirect addressing

CS: 0200, DS: 0100

```
            MOV AL,0
            MOV CL,04H
            MOV BX,0000H
AGAIN:      MOV [BX],AL
            INC AL
            INC BX  ⬅
            LOOP AGAIN
```

	H	L
AX	xx	04
BX	00	03
CX	xx	00

DS*10H+BX	01003

Physical address

Physical Address	Value
01000	00
01001	01
01002	02
01003	03

DS = 0100

Lookup table with register indirect addressing

CS: 0200, DS: 0100

```
         MOV AL,0
         MOV CL,04H
         MOV BX,0000H
AGAIN:   MOV [BX],AL
         INC AL
         INC BX
         LOOP AGAIN  ⬅
```

	H	L
AX	xx	04
BX	00	04
CX	xx	00

DS*10H+BX	01003

Physical address

Physical Address	Value
01000	00
01001	01
01002	02
01003	03

DS = 0100

Lookup table with register indirect addressing

CS: 0200, DS: 0100

```
        MOV AL,0
        MOV CL,04H
        MOV BX,0000H
AGAIN:  MOV [BX],AL
        INC AL
        INC BX
        LOOP AGAIN
```

Exit from the loop ⬅

	H	L
AX	xx	04
BX	00	04
CX	xx	00

DS*10H+BX	01003

Physical address

Physical Address	Value
01000	00
01001	01
01002	02
01003	03

DS = 0100

Register Indirect Addressing

MOV AX, [BX]

BX = 1234$_{H}$

(AX) ← DS:1234

DS = 2000$_{H}$

- **ADDRESS = 20000+1234=21234**
- **21234$_{H}$** 74
- **21235$_{H}$** 82

- **AX = 82 74$_{H}$**

Register Indirect Addressing

MOV AX, [BX]

1	0	0	0	1	0	1	1	0	0	0	0	0	1	1	1

8B07

Register Indirect Addressing

MOV [SI], BH

SI = $\mathrm{D600}_H$
DS = 2000_H

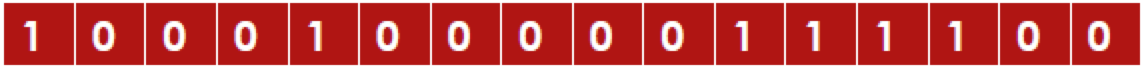
ADDRESS = 20000+D600=2D600

- **BX = 8A 17$_H$**

- **$\mathrm{2D600}_H$** ← 8A

Register Indirect Addressing

MOV [SI], BH



1	0	0	0	1	0	0	0	0	0	1	1	1	1	0	0

883C

Register Indirect Addressing

MOV EAX, [BX]

BX = 1234_H
(EAX) ← DS:1234_H

DS = 2000_H
ADDRESS = 20000+1234=21234

➢ **21234_H**	74
➢ **21235_H**	82
➢ **21236_H**	A3
➢ **21237_H**	45

➢ **EAX = 45 A3 82 74_H**

Register Indirect Addressing

MOV EAX, [ECX]

ECX = 0000 1234_H

(EAX) ← DS:1234_H

DS = 2000_H

ADDRESS = 20000+1234=21234

➢ **21234_H**	74
➢ **21235_H**	82
➢ **21236_H**	A3
➢ **21237_H**	45

- ➢ **EAX = 45 A3 82 74_H**

Base plus Indexed Addressing

- Transfers a byte or word between a register and the memory location addressed by a base register plus an index register.

- Registers involved are BP,BX,DI and SI.

- Base register often holds the beginning location of a memory array.
- Index register holds the relative position of an element in the array.
  - **Example:**

```
MOV DX,[BX+DI];
MOV CH,[BP+SI]
MOV [BX+SI], BP
```

Example: Locating data with base-plus-index addressing

MOV DL,[BX+DI]

Given CS: 0200   DS: 0100   BX: 1000H   DI: 0010H

1) **Calculate offset address**

   BX+DI = 1000H+0010H = 1010H

2) **Physical Address = DS*10H + 1010H**

   i.e. 01000+1010 = 02010H

Value at location 02010H is copied into DL register

Example: Locating data with base-plus-index addressing

CS: 0200 DS: 0100

Array starting address = BX= 0010H

Array Element No = DI

```
MOV BX,0010H
MOV DI,04H
MOV AL,[BX+DI]

MOV DI,09H
MOV [BX+DI],AL
```

Array of size 10

Element No	Value
0	12
1	32
2	52
3	65
4	21
5	35
6	48
7	99
8	44
9	95

Example: Locating data with base-plus-index addressing

CS: 0200 DS: 0100

Array starting address = BX= 0010H

Element No = DI

```
MOV BX,0010H
MOV DI,04H
MOV AL,[BX+DI]
```

After this instruction AL = 21

DS:[BX+DI] = 01014H

Array of size 10

Element No	Value
0	12
1	32
2	52
3	65
4	21
5	35
6	48
7	99
8	44
9	95

Example: Locating data with base-plus-index addressing

CS: 0200 DS: 0100

Array starting address = BX= 0010H

Element No = DI

```
MOV BX,0010H
MOV DI,04H
MOV AL,[BX+DI]
```

After this instruction AL = 21

```
MOV DI,09H
MOV [BX+DI],Al
```

DS:[BX+DI] = 01014H

Array of size 10

Element No	Value
0	12
1	32
2	52
3	65
4	21
5	35
6	48
7	99
8	44
9	95

Example: Locating data with base-plus-index addressing

CS: 0200 DS: 0100

Array starting address = BX= 0010H

Element No = DI

```
MOV BX,0010H
MOV DI,04H
MOV AL,[BX+DI]
```

After this instruction AL = 21

```
MOV DI,09H
MOV [BX+DI],AL
```

After this instruction 9th element in the array is replaced with A[9] = 21H

Array of size 10

Element No	Value
0	12
1	32
2	52
3	65
4	21
5	35
6	48
7	99
8	44
9	21

DS:[BX+DI]
= 01019H

Base plus Indexed Addressing

MOV AX, [BX+SI]

BX = 1200_H
SI = 0034_H

(AX) ← DS:1200+34

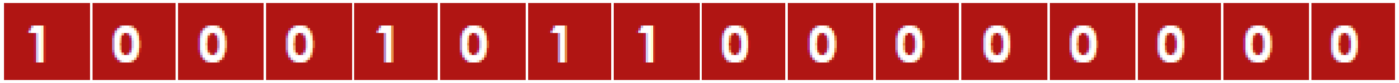
DS = 2000_H
ADDRESS = 20000+1200+34=21234

- **21234_H** 74
- **21235_H** 82

- **AX = 82 74_H**

Base plus Indexed Addressing

MOV AX, [BX+SI]



1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0

8B00

Base plus Indexed Addressing

MOV EAX, [BX+SI]

BX = 1200_H
SI = 0034_H
(EAX) ← DS:1200+34
DS = 2000_H
ADDRESS = 20000+1200+34=21234

➢ **21234_H**	74
➢ **21235_H**	82
➢ **21236_H**	A3
➢ **21237_H**	45

➢ **EAX = 45 A3 82 74_H**

Register Relative Addressing

- Similar to base-plus-index addressing and displacement addressing.

- Moves a byte or word between a register and the memory location addressed by an index or base register plus a displacement.

- Displacement is added to the contents of index or base register.
  - Example:

```
MOV AX,[BX+1000H]
MOV  AL,[DI+100H]
```

Example: Locating data using register relative addressing

MOV DL,[BX+1000H]

Given CS: 0200 DS: 0100 BX: 0100H Displacement = 1000H

1) Calculate offset address
   BX+Disp = 0100H+1000H = 1100H

2) Physical Address = DS*10H + 1100H
   i.e. 01000+1100 = 02100H

- ✓ **Value at location 02100H is copied into DL register**

Register Relative Addressing

MOV AX, [BX+34]

BX = 1200_H

(AX) ← DS:1200+34

DS = 2000_H

ADDRESS = 20000+1200+34=21234

➢ 21234_H	74
➢ 21235_H	82

➢ AX = 45 A3 82 74_H

Register Relative Addressing

MOV AX, [BX+34]

1	0	0	0	1	0	1	1	0	1	0	0	0	1	1	1

8B47 34

Register Relative Addressing

MOV [SI+600], BH

SI = D000$_H$
DS = 2000$_H$

ADDRESS = 20000+D000+600=2D600

- **BX 8A17$_H$**
- **2D600$_H$ 8A**

Register Relative Addressing

MOV [SI+600], BH

1	0	0	0	1	0	0	0	1	0	1	1	1	1	0	0

88BC 00 06

Register Relative Addressing

MOV EAX, [ECX+234_H]

ECX = 0000 1000_H

(EAX) ← DS:1234

DS = 2000_H

ADDRESS = 20000+1000+234=21234

➢ **21234_H**	74
➢ **21235_H**	82
➢ **21236_H**	A3
➢ **21237_H**	45

- **EAX = 45 A3 82 74_H**

Base relative-plus-index Addressing

- Similar to base-plus-index addressing mode.
- Transfers a byte or word between a register and the memory location addressed by a base and index register plus a displacement.
- Adds displacement to base and index register.
- Often used to address a two-dimensional array of memory data.
  - Example :

MOV AX,[BX+SI+100H]

MOV [BX+DI+20H],DH

Example : Addressing data with base relative-plus-index

MOV Al,[BX+SI+100H]

Given

CS: 0200 DS: 0100 BX: 0100H SI: 0020 Displacement = 100H

1) Calculate offset address

   BX+SI+Disp = 0100H+0020+100H = 0220H

2) Physical Address **= DS*10H + 0220H**

   i.e. 01000+0220 = 01220H

✓ **Value at location 01220H is copied into AX register**

Base Relative plus Indexed Addressing

MOV [BX+SI+600], AH

SI = 1000_H
BX = $C000_H$
DS:C000+1000+600 ← (AH)

DS = 2000_H
ADDRESS = 20000+C000+1000+600=2D600

- AX = $8A17_H$
- $2D600_H$ 8A

Base Relative plus Indexed Addressing

MOV [BX+SI+600], AH

1	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0

88B8 00 06

Default 16-bit segment and offset address combinations

SEGMENT	OFFSET	SPECIAL PURPOSE
CS	IP	INSTRUCTION ADDRESS
SS	SP (or) BP	STACK ADDRESS
DS	BX, DI, SI, an 8-bit number, 16 bit number	DATA ADDRESS
ES	DI for string Instructions	STRING DESTINATION ADDRESS