BITS Pilani

Microprocessors & Interfacing

INSTRUCTION SET

Dr. Gargi Prabhu
Department of CS & IS

Type	Instruction	Source	Address Generation	Destination
Register	MOV AX,BX	Register BX		Register AX
Immediate	MOV CH,3AH	Data 3AH		Register CH
Direct	MOV [1234H],AX	Register AX	DS × 10H + DISP 10000H + 1234H	Memory address 11234H
Register indirect	MOV [BX],CL	Register CL	DS × 10H + BX 10000H + 0300H	Memory address 10300H
Base-plus-index	MOV [BX+SI],BP	Register SP	DS × 10H + BX + SI 10000H + 0300H + 0200H	Memory address 10500H
Register relative	MOV CL,[BX+4]	Memory address 10304H	DS × 10H + BX + 4 10000H + 0300H + 4	Register CL
Base relative-plus-index	MOV ARRAY[BX+SI],DX	Register DX	DS × 10H + ARRAY + BX + SI 10000H + 1000H + 0300H + 0200H	Memory address 11500H
Scaled index	MOV [EBX+2 × ESI],AX	Register AX	DS × 10H + EBX + 2 × ESI 10000H + 00000300H + 00000400H	Memory address 10700H

Notes: EBX = 00000300H, ESI = 00000200H, ARRAY = 1000H, and DS = 1000H

Register Indirect Addressing

Assembly Language	*Size*	*Operation*
MOV CX,[BX]	16 bits	Copies the word contents of the data segment memory location addressed by BX into CX
MOV [BP],DL*	8 bits	Copies DL into the stack segment memory location addressed by BP
MOV [DI],BH	8 bits	Copies BH into the data segment memory location addressed by DI
MOV [DI],[BX]	—	Memory-to-memory transfers are not allowed except with string instructions
MOV AL,[EDX]	8 bits	Copies the byte contents of the data segment memory location addressed by EDX into AL
MOV ECX,[EBX]	32 bits	Copies the doubleword contents of the data segment memory location addressed by EBX into ECX
MOV RAX,[RDX]	64 bits	Copies the quadword contents of the memory location address by the linear address located in RDX into RAX (64-bit mode)

*Note: Data addressed by BP or EBP are by default in the stack segment, while other indirect addressed instructions use the data segment by default.

History of Assemblers

- The assembler for the Intel 8086 processor, like many other assemblers, was typically written in a low-level programming language, often in assembly language itself or in a mix of assembly language and a higher-level language like C.
- Example: **GNU Assembler (released in 1986)**, assembler developed by the GNU Project is written in C. It is the default back-end of GCC.

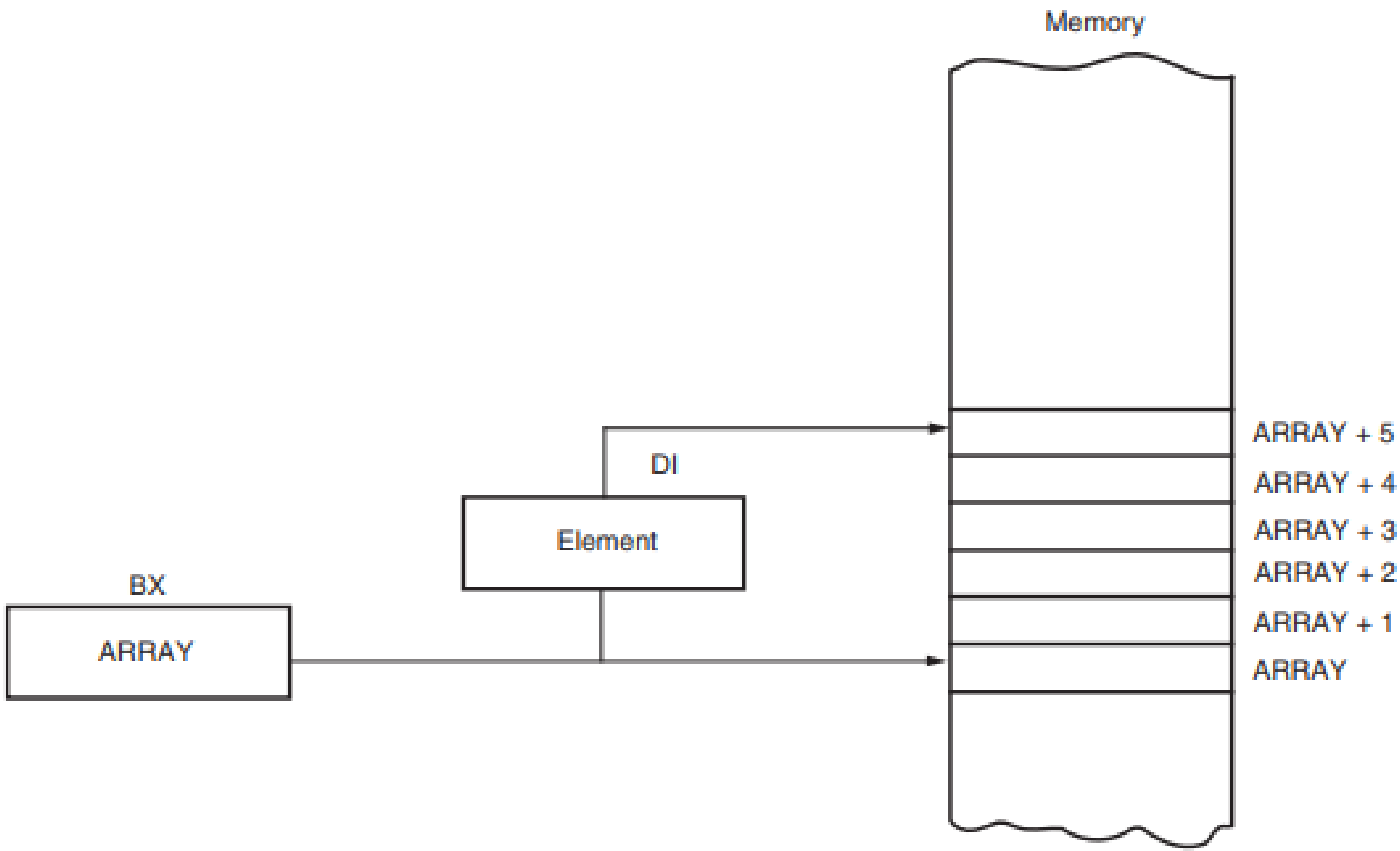
Base-Plus-Index Addressing

- Uses one Base Register (BP or BX) and one index register(DI or SI)
- Base register holds the relative position of an element in the array
- Index register holds the relative position of an element in an array

Where to use Base-Plus-Index Addressing?

- A major use of the base-plus-index addressing mode is to address elements in a memory array.
- Elements in an array located in the data segment at memory location ARRAY must be accessed.
- To accomplish this, load the BX register (base) with the beginning address of the array and the DI register (index) with the element number to be accessed.

Example



Memory
DI
Element
BX
ARRAY
ARRAY + 5
ARRAY + 4
ARRAY + 3
ARRAY + 2
ARRAY + 1
ARRAY

Example

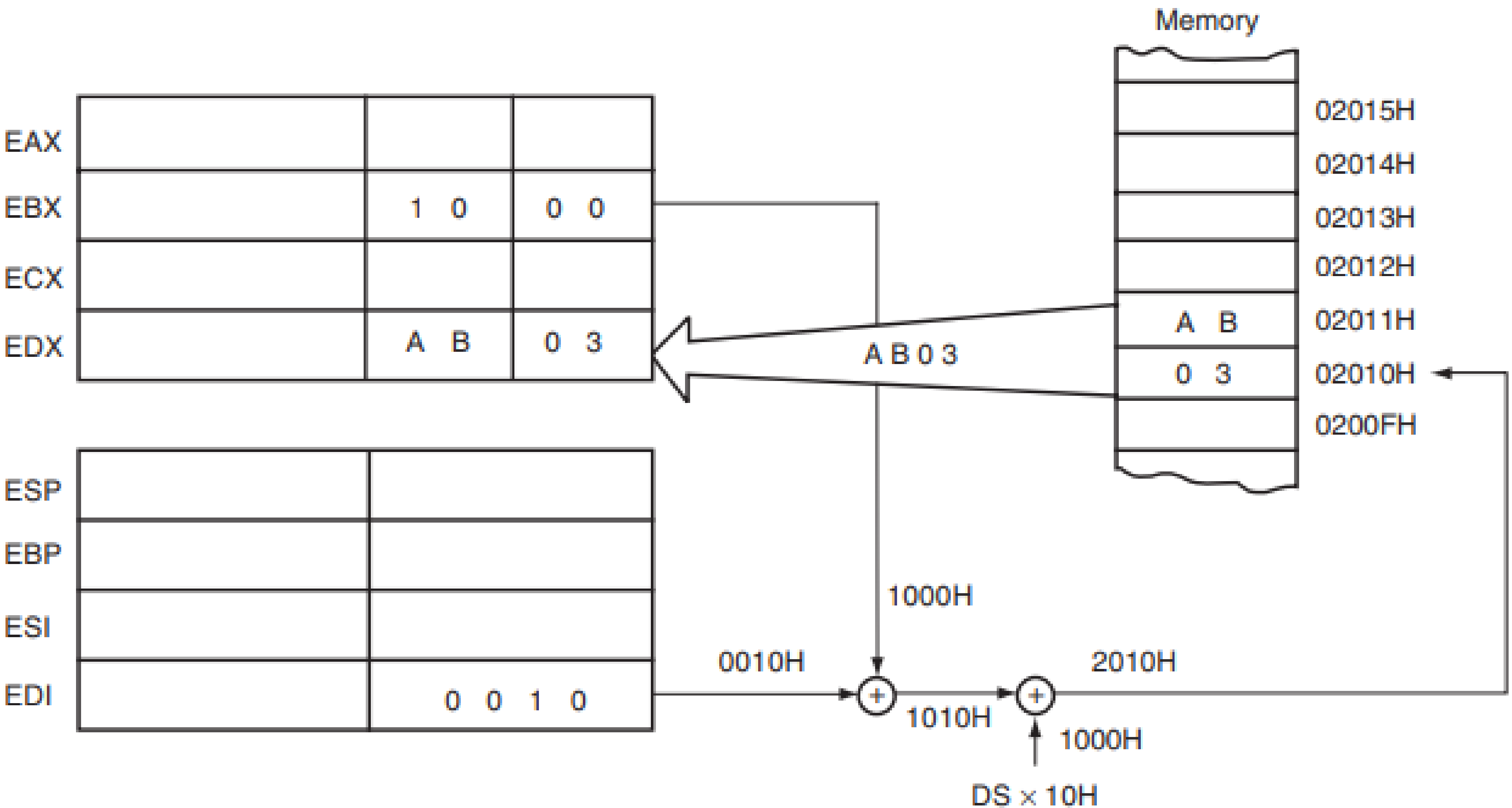
```
section .data
myArray DW 10, 20, 30, 40, 50  ; An array in memory

section .code
mov BX, OFFSET myArray ; address of the array
mov SI, 2     ; Index, pointing to the third element (30)
mov AX, [BX + SI] ; Move the value at (BX + SI) into AX
```

-> **After the execution of the MOV instruction, register AX would contain the value 30 because myArray[2] is 30**

Base-Plus-Index Addressing
MOV DX,[BX+DI]

BX=1000H DI=0010H DS=0100H



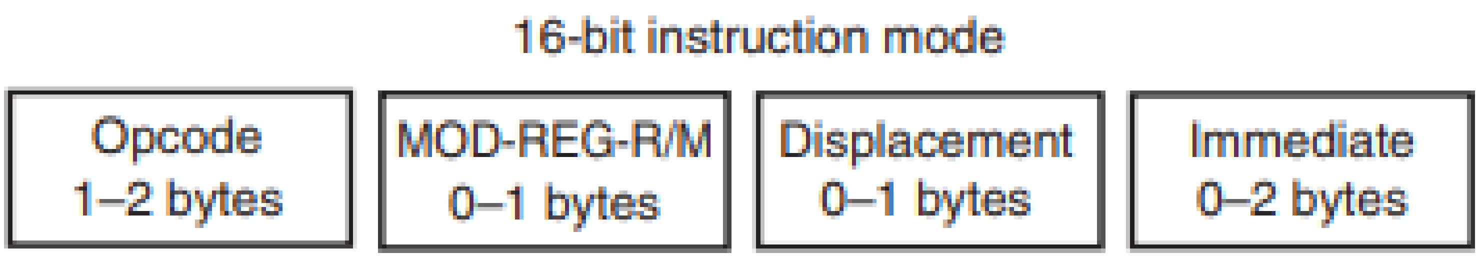
Instruction Set of Assembly Language

- Define the set of instructions that a processor can execute.
- Each type of processor has its own instruction set architecture (ISA), and assembly language provides a human-readable representation of the machine code instructions for that architecture.
- Instruction sets serve as a crucial interface between software and hardware, allowing for effective communication and coordination within a computer system.
- ISA leads to the standardization, compatibility, and performance of computing devices, fostering innovation and enabling a wide range of applications.

Instruction Format

Opcode

- Selects the operation (add, sub, mov or so on)
- Either 1 or 2 byte long



Instruction Format

Opcode

- First 6 bits are binary opcode
- 1 bit is Direction
- D=1, data flow from R/M field to the REG filed
- D=0, data flow from REG filed to R/M field

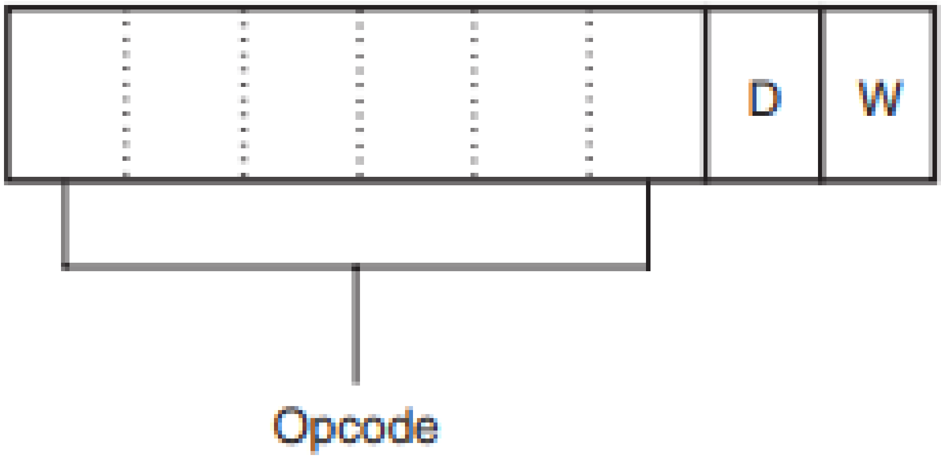


Fig. Byte 1 of Opcode

Instruction Format

Opcode

- First 6 bits are binary opcode
- 1 bit is Word
- W=1 , data size is word or doubleword
- W=0, data size is always a byte

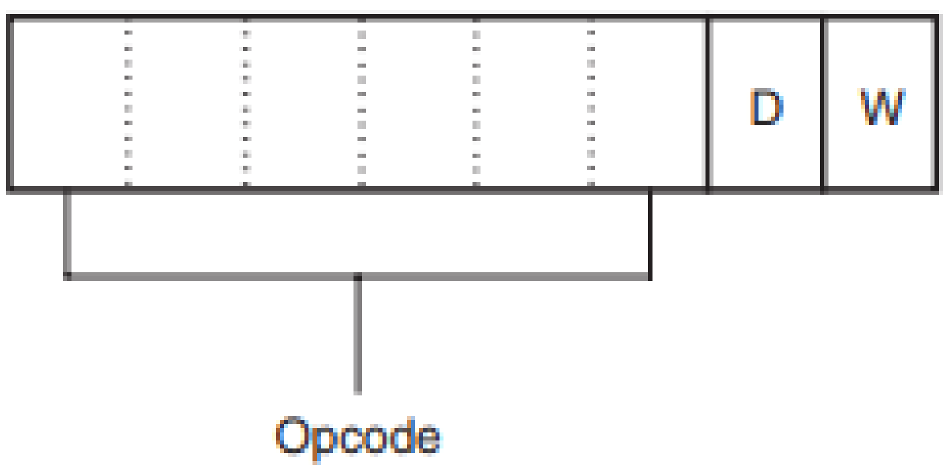


Fig. Byte 1 of Opcode

Instruction Format

Byte 2 of instruction

- MOD – specifies addressing mode for the selected instruction
- 11- Register addressing mode
- 00,01,10 – Data memory addressing modes

E.g. MOV AL, [DI] – No displacement

MOV AL,[DI+1000H]- 16 bit displacement

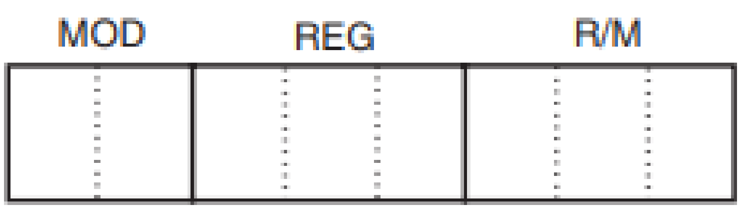


Fig. Byte 1 of Instruction

MOD	*Function*
00	No displacement
01	8-bit sign-extended displacement
10	16-bit signed displacement
11	R/M is a register

Sign Extension

- Sign bit of a number is used to extend the bit-width of the number while preserving its sign.
- Necessary when working with data of different sizes or when performing operations that involve different-sized operands.
- Sign extension ensures that the sign of the original value is maintained when the value is extended to a larger bit-width.

8 bit displacement are sign extended to 16 bit displacements

E.g. 00H-7FH

Extended to 0000H -007FH

E.g. 80H- FFH

Extended to FF80H-FFFH

Instruction Format

Fig. Byte 1 of Instruction

Byte 2 of instruction

- REG – registers assigned for the instruction

Code	W = 0 *(Byte)*	W = 1 *(Word)*	W = 1 *(Doubleword)*
000	AL	AX	EAX
001	CL	CX	ECX
010	DL	DX	EDX
011	BL	BX	EBX
100	AH	SP	ESP
101	CH	BP	EBP
110	DH	SI	ESI
111	BH	DI	EDI

Instruction Format

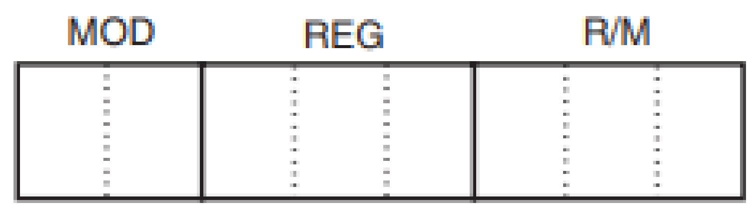


Fig. Byte 1 of Instruction

Byte 2 of instruction

- R/M – If the MOD field contains 00, 01, or 10 for 16 bit instructions

R/M Code	*Addressing Mode*
000	DS:[BX+SI]
001	DS:[BX+DI]
010	SS:[BP+SI]
011	SS:[BP+DI]
100	DS:[SI]
101	DS:[DI]
110	SS:[BP]*
111	DS:[BX]

Instruction Format

R/M \ MOD	00	01	10	11 W = 0	11 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	[BP] + [DI] + d16	BL	BX
100	[SI]	[SI] + d8	[SI] + d16	AH	SP
101	[DI]	[DI] + d8	[DI] + d16	CH	BP
110	d16 (direct address)	[BP] + d8	[BP] + d16	DH	SI
111	[BX]	[BX] + d8	[BX] + d16	BH	DI
	MEMORY MODE	MEMORY MODE	MEMORY MODE	REGISTER MODE	REGISTER MODE

d8 = 8-bit displacement   d16 = 16-bit displacement

Instruction Format

E.g.

2 Byte Instruction – 8BECH

Binary – 1000 1011 1110 1100

Opcode – 100010 (MOV Instruction)

D and W = 1 -> word moves into destination register specified in the REG field

REG filed = 101 -> register BP

MOD=11 -> R/M is register

R/M = 100 (SP)

-> Instruction moves data from SP into BP -> MOV BP, SP

Instruction Format

Opcode						D	W
1	0	0	0	1	0	1	1

MOD		REG			R/M		
1	1	1	0	1	1	0	0

Opcode = MOV
D = Transfer to register (REG)
W = Word
MOD = R/M is a register
REG = BP
R/M = SP

Example

What is the machine code for

MOV AX, BX

Example

What is the machine code for

MOV AX, BX

Opcode- 100010

D=1

W= 1

REG= 000

MOD=11

R/M= 011

Instruction = 1001011 00011011

Example

What is the machine code for

ADD [BX][DI]+1234H, AX

Opcode for ADD is 000000.

innovate achieve lead

BITS Pilani
Pilani Campus

Thank You