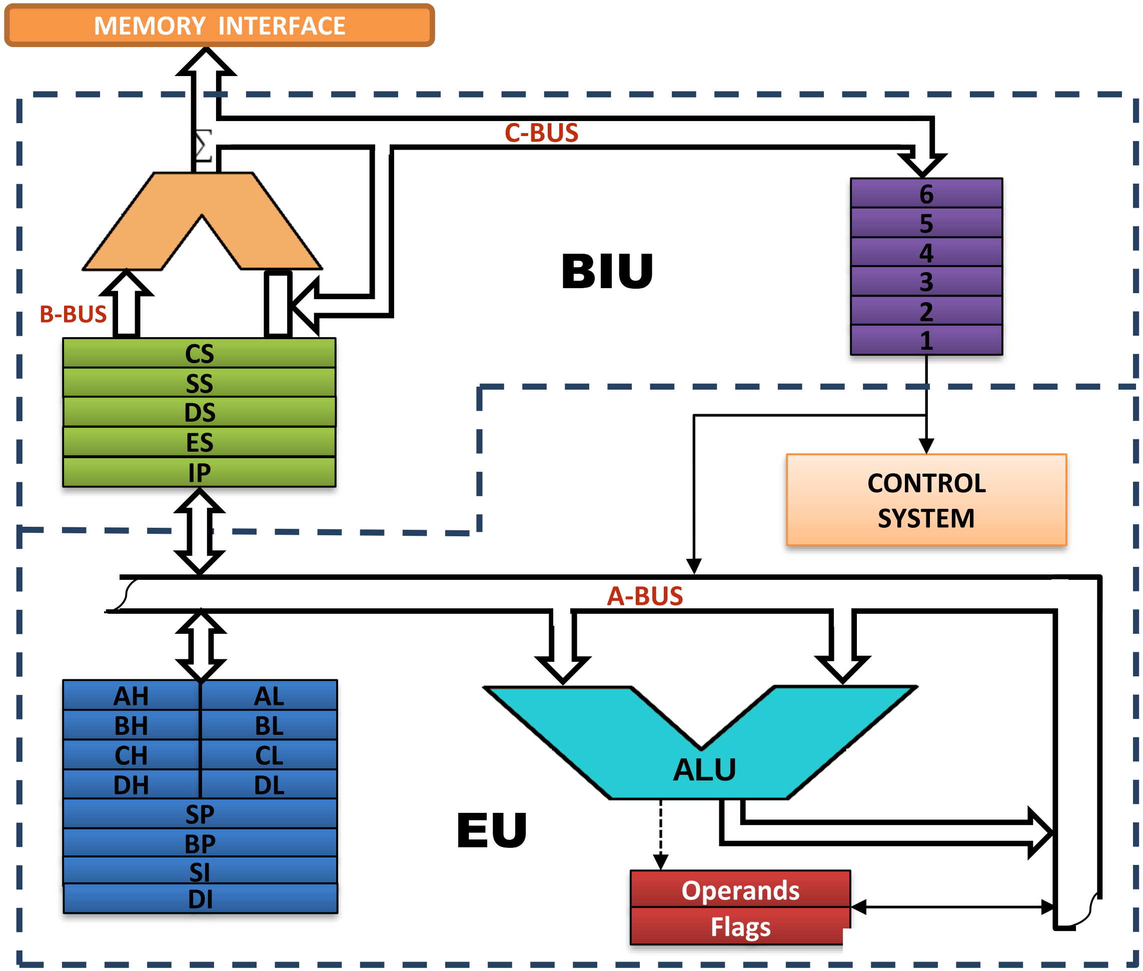
BITS Pilani

Microprocessors & Interfacing

Addressing Modes

Dr. Gargi Prabhu
Department of CS & IS

8086 Architecture



Machine Language

- A sequence of binary codes for the instructions you want the microprocessor to execute
- What form a machine can understand?
- Why is it difficult to write machine code?
  - Memorizing thousands of binary instruction codes
  - An error can occur when working with long series of 1's and 0's
  - Can hexadecimal representation help?
  - Thousands of instructions code to cope with

Assembly Language

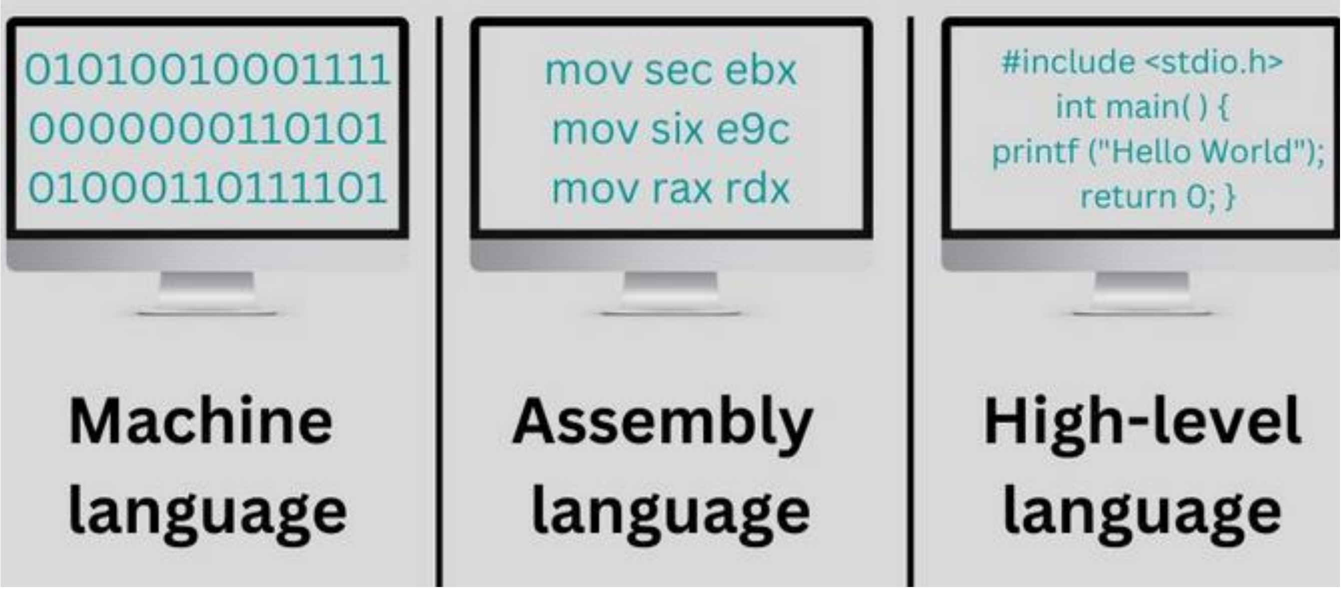
- Assembly language program is translated to machine language, loaded in memory and run.
- Use two-,three- or four-letter mnemonics to represent each instruction type

e.g. ADD, SUB, OR, XOR

- Assembly language instructions are written with four fields

LABEL FIELD	OP CODE FIELD	OPERAND FIELD	COMMENT FIELD
NEXT:	ADD	AL,07H	;ADD Correction factor

Machine <- High Level Language



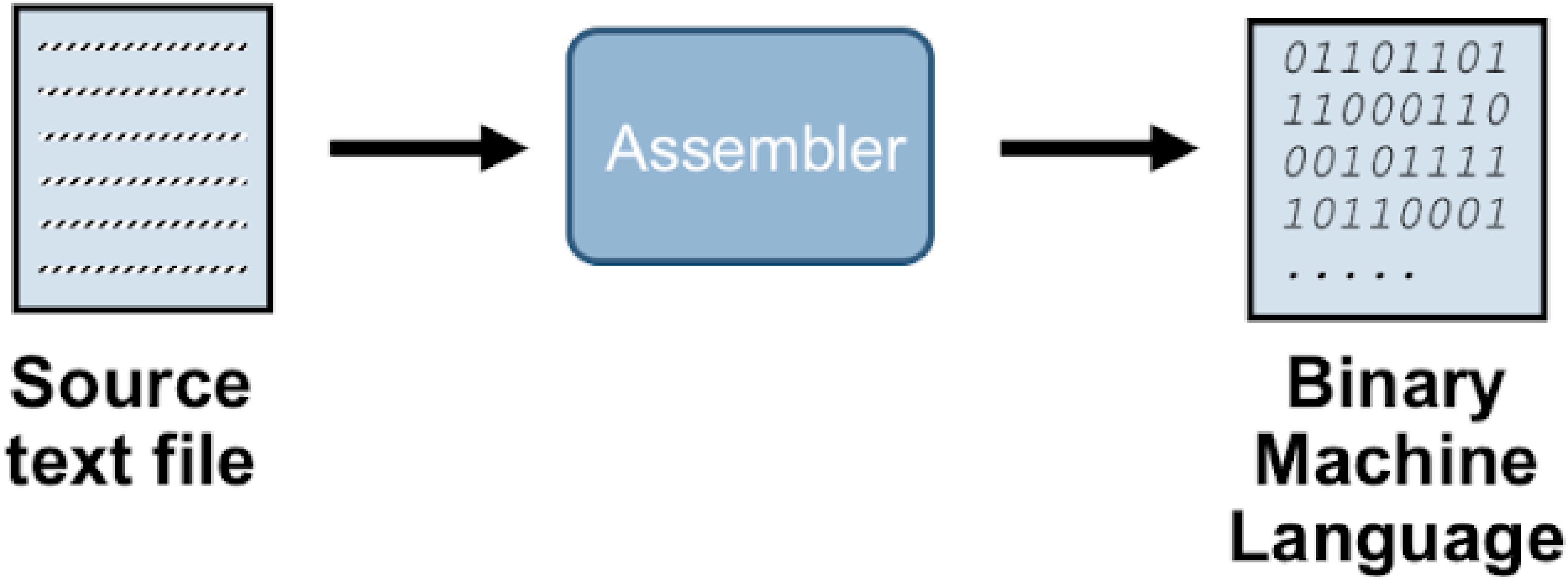
```
010100100011111
0000000110101
01000110111101
```

```
mov sec ebx
mov six e9c
mov rax rdx
```

```
#include <stdio.h>
int main( ) {
printf ("Hello World");
return 0; }
```

Assembler

- Reads the file of assembly language program and generates the correct binary code for each instruction



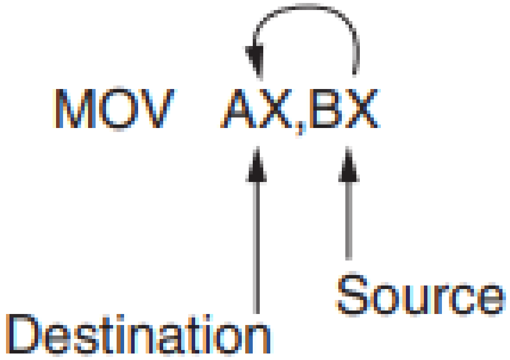
Addressing Modes

- Different ways in which a processor can access data are referred to as its addressing modes.

- Why we need addressing modes?
  - Choosing the appropriate addressing mode can lead to more compact and faster code execution, optimizing program efficiency and performance.
  - Addressing modes determine how operands are located, whether in registers or memory. Efficient use of registers is crucial for performance, as accessing data from registers is faster than accessing data from memory.
  - By allowing operations on variables or constants located at different memory addresses or registers, addressing modes support the creation of versatile and powerful instructions.
  - Addressing modes facilitate the manipulation of arrays and data structures.

MOV Instruction

- MOV instruction is a very common and flexible instruction
- Provides a basis for the explanation of the data-addressing modes.



- Transfers the word contents of the source register (BX) into the destination register (AX).
- The source and destination are often called operands.

Register Addressing

- Transfers a copy of a byte or word from the source register or contents of a memory location to the destination register or memory location

  e.g. MOV CX, DX

- copies the word-sized contents of register DX into register CX
- CX= 2A84H DX=4971H
- After instruction, DX= 4971H CX=4971H

Immediate Addressing

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

e.g. MOV AL, 22H

- copies a byte-sized 22H into register AL.

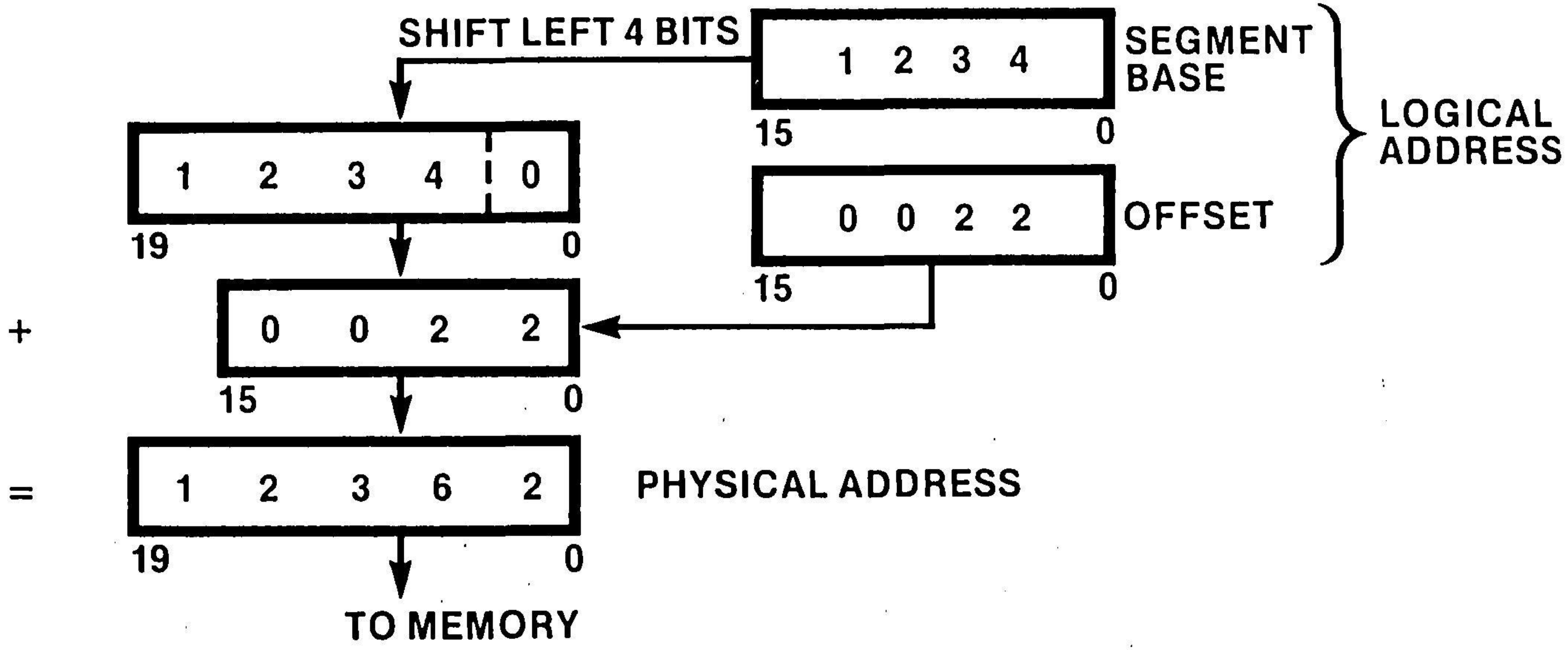
e.g. MOV CX, 437BH

- Copied in CH and CL memory location

e.g. MOV EBX, 12345678H [In 80386]

- copies a doubleword-sized l2345678H into the 32-bit-wide EBX register

Accessing Data in Memory



Direct Addressing

- Moves a byte or word between a memory location addressing and a register.
- The instruction set does not support a memory-to memory transfer, except with the MOVS instruction.

e.g. MOV CX, LIST

- copies the word-sized contents of memory location LIST into register CX

e.g. MOV BL, [437AH]

BIU calculates physical address

e.g. DS=2000

Direct Addressing

e.g. MOV BX, [437AH]

- Each memory address in 8086 represents a byte in storage

- Word must come from two memory locations

BL= Data at 437AH

BH= Data at 437BH

Low Byte - low address   **High Byte – high address**

Direct Addressing

e.g. MOV BX, 437AH

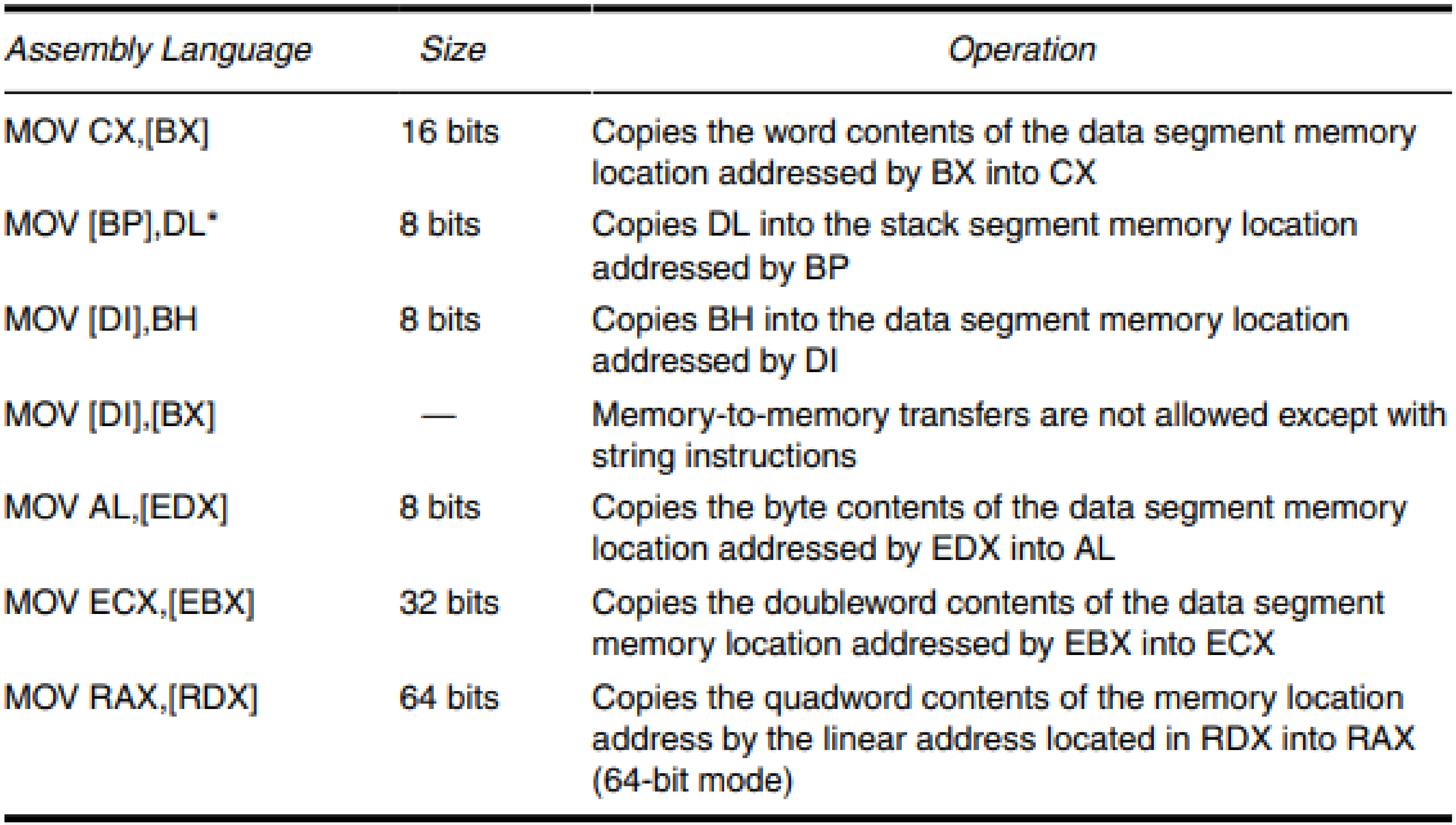
Register Indirect Addressing

- Transfers a byte or word between a register and a memory location addressed by an index or base register.
- The index and base registers are BP, BX, DI.

e.g. MOV AX, [BX]

- copies the word-sized data from the data segment offset address indexed by BX into register AX.

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)

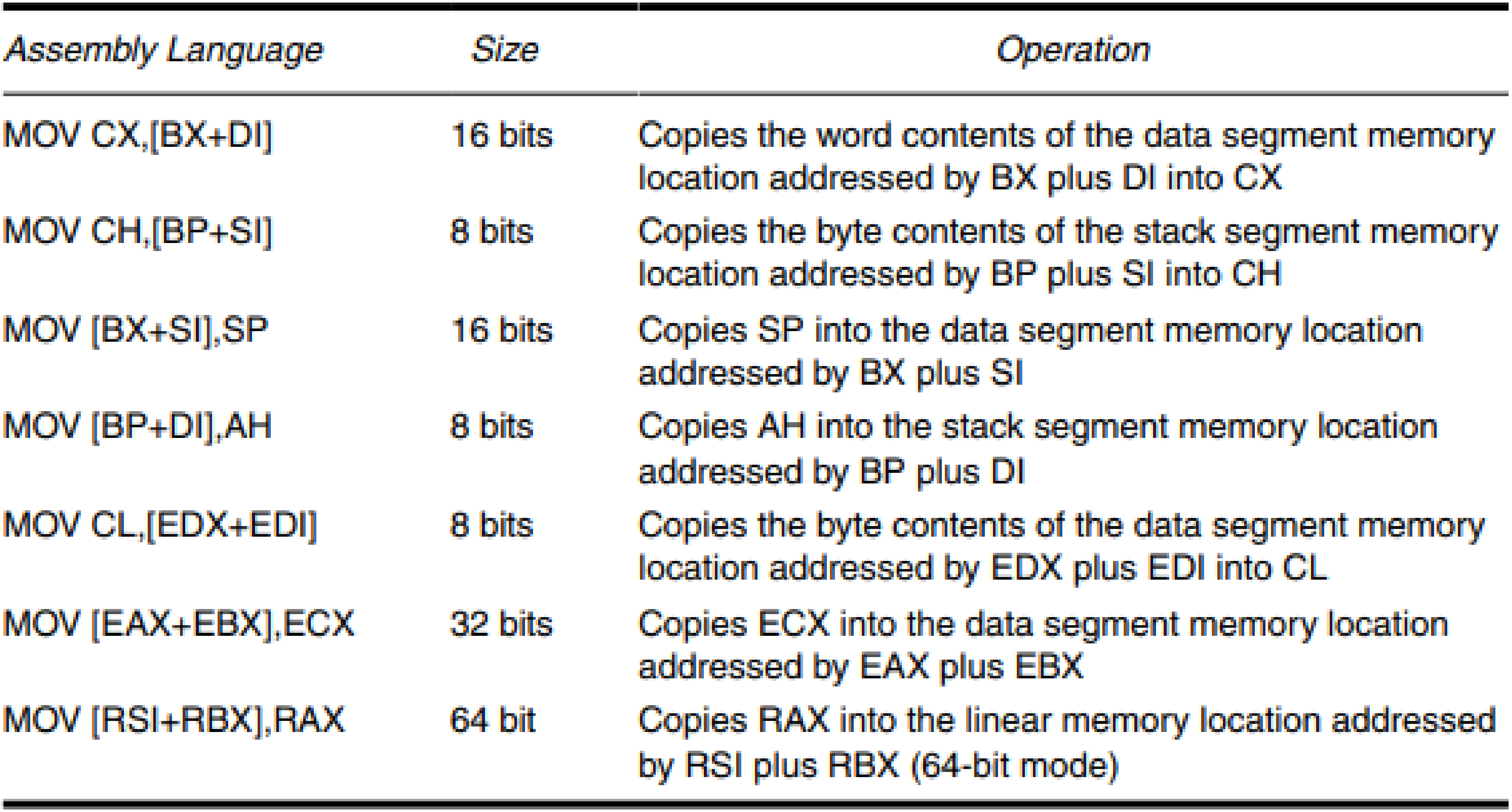
Base-plus-index Addressing

- Transfers a byte or word between a register and the memory location addressed by a base register (BP or BX) plus an index register (DI or SI).

e.g. MOV [BX+DI], CL

- copies the byte-sized contents of register CL into the data segment memory location addressed by BX plus DI.)

Base-plus-index Addressing



Assembly Language	*Size*	*Operation*
MOV CX,[BX+DI]	16 bits	Copies the word contents of the data segment memory location addressed by BX plus DI into CX
MOV CH,[BP+SI]	8 bits	Copies the byte contents of the stack segment memory location addressed by BP plus SI into CH
MOV [BX+SI],SP	16 bits	Copies SP into the data segment memory location addressed by BX plus SI
MOV [BP+DI],AH	8 bits	Copies AH into the stack segment memory location addressed by BP plus DI
MOV CL,[EDX+EDI]	8 bits	Copies the byte contents of the data segment memory location addressed by EDX plus EDI into CL
MOV [EAX+EBX],ECX	32 bits	Copies ECX into the data segment memory location addressed by EAX plus EBX
MOV [RSI+RBX],RAX	64 bit	Copies RAX into the linear memory location addressed by RSI plus RBX (64-bit mode)

Register relative Addressing

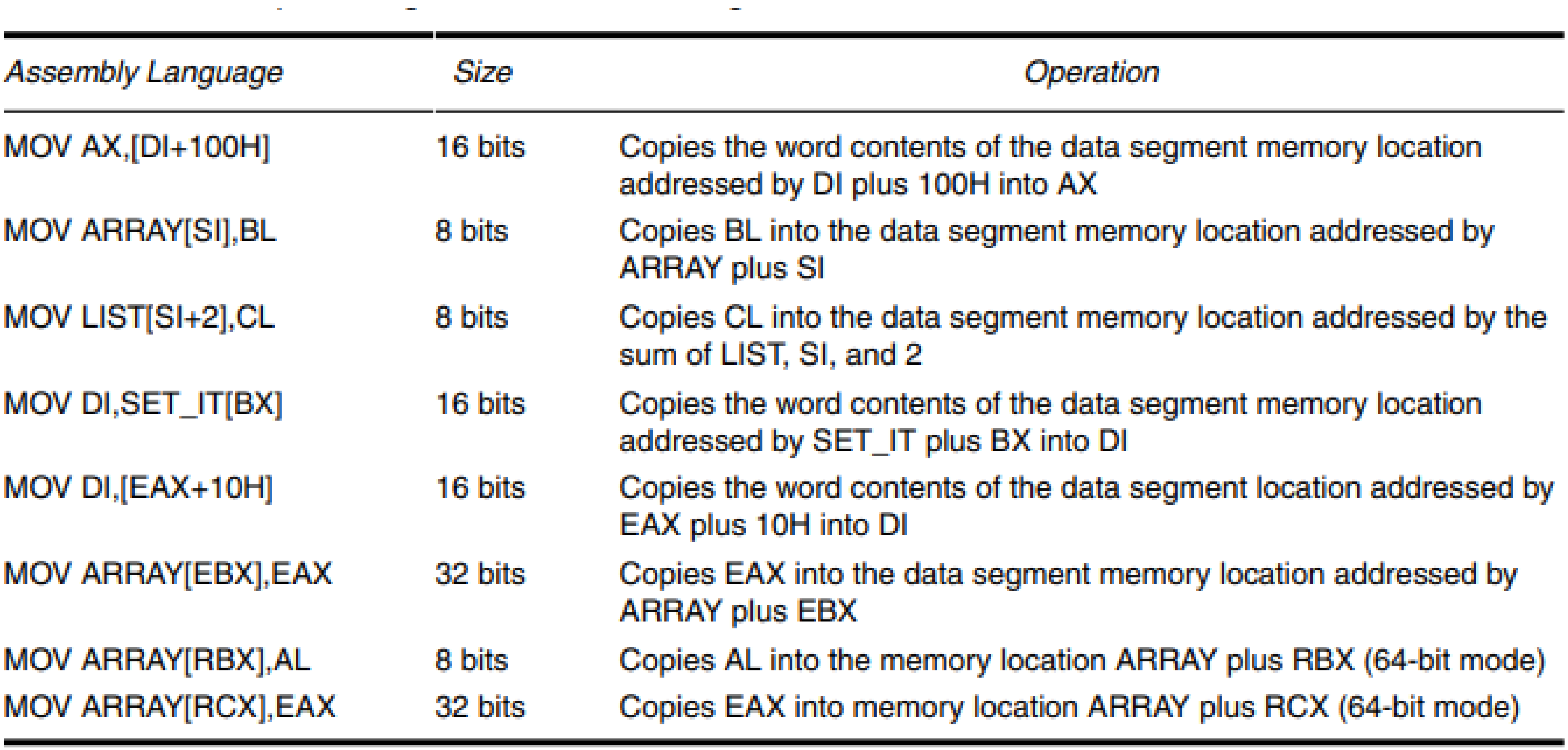
- Moves a byte or word between a register and the memory location addressed by an index or base register plus a displacement.

e.g. MOV AX,[BX+4] or MOV AX,ARRAY[BX].

- The first instruction loads AX from the data segment address formed by BX plus 4.

- The second instruction loads AX from the data segment memory location in ARRAY plus the contents of BX.

Register relative Addressing



Assembly Language	*Size*	*Operation*
MOV AX,[DI+100H]	16 bits	Copies the word contents of the data segment memory location addressed by DI plus 100H into AX
MOV ARRAY[SI],BL	8 bits	Copies BL into the data segment memory location addressed by ARRAY plus SI
MOV LIST[SI+2],CL	8 bits	Copies CL into the data segment memory location addressed by the sum of LIST, SI, and 2
MOV DI,SET_IT[BX]	16 bits	Copies the word contents of the data segment memory location addressed by SET_IT plus BX into DI
MOV DI,[EAX+10H]	16 bits	Copies the word contents of the data segment location addressed by EAX plus 10H into DI
MOV ARRAY[EBX],EAX	32 bits	Copies EAX into the data segment memory location addressed by ARRAY plus EBX
MOV ARRAY[RBX],AL	8 bits	Copies AL into the memory location ARRAY plus RBX (64-bit mode)
MOV ARRAY[RCX],EAX	32 bits	Copies EAX into memory location ARRAY plus RCX (64-bit mode)

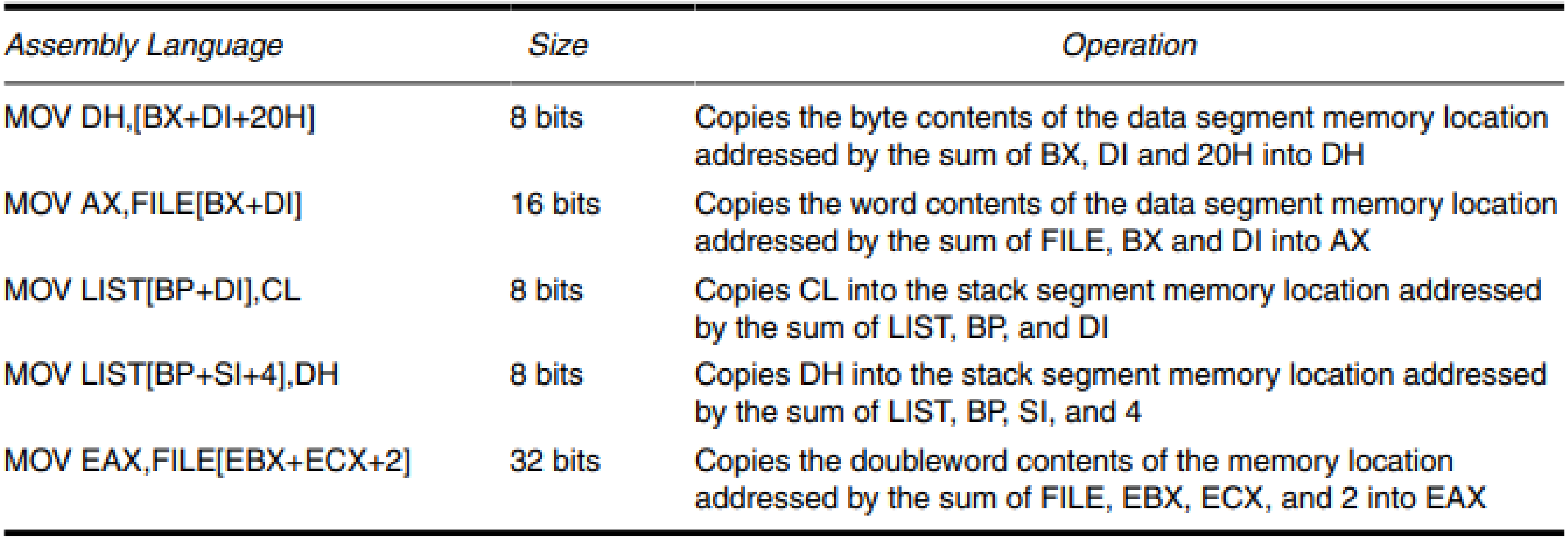
Base relative-plus-index Addressing

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

e.g. MOV AX, ARRAY[BX+DI] or MOV AX, [BX+DI+4]

- These instructions load AX from a data segment memory location.
- The first instruction uses an address formed by adding ARRAY, BX, and DI and the second by adding BX, DI, and 4.)

Base relative-plus-index Addressing



Assembly Language	*Size*	*Operation*
MOV DH,[BX+DI+20H]	8 bits	Copies the byte contents of the data segment memory location addressed by the sum of BX, DI and 20H into DH
MOV AX,FILE[BX+DI]	16 bits	Copies the word contents of the data segment memory location addressed by the sum of FILE, BX and DI into AX
MOV LIST[BP+DI],CL	8 bits	Copies CL into the stack segment memory location addressed by the sum of LIST, BP, and DI
MOV LIST[BP+SI+4],DH	8 bits	Copies DH into the stack segment memory location addressed by the sum of LIST, BP, SI, and 4
MOV EAX,FILE[EBX+ECX+2]	32 bits	Copies the doubleword contents of the memory location addressed by the sum of FILE, EBX, ECX, and 2 into EAX

Scaled-index Addressing

- Available only in the 80386 through the addressing Pentium 4 microprocessor.
- The second register of a pair of registers is modified by the scale factor of to generate the operand memory address.
- Second register is modified by a scale factor of 2×, 4×, 8× to generate operand memory address.

e.g. MOV EDX, [EAX+4*EBX]

- loads EDX from the data segment memory location addressed by EAX plus four times EBX.

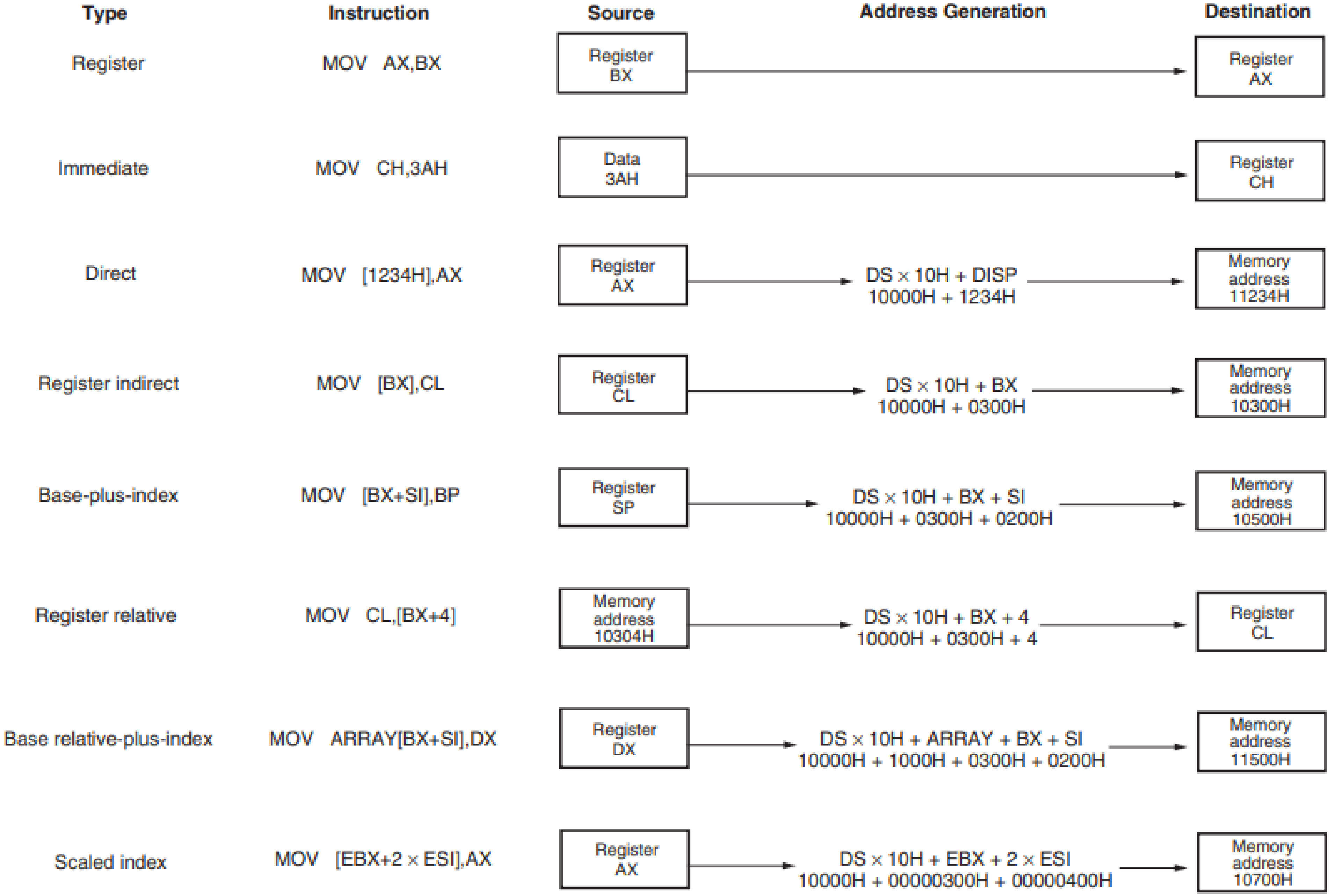
- Scaling allows access to word (2×), doubleword (4×), or quadword (8×) memory array data.

RIP Relative Addressing

- Only available to the 64-bit extensions on the addressing Pentium 4 or Core2.
- Allows access to any location in the memory system by adding a 32-bit displacement to the 64-bit contents of the 64-bit instruction pointer.

e.g. RIP = 1000000000H and a 32-bit displacement is 300H, the location accessed is 1000000300H.

- The displacement is signed so data located within +-2G from the instruction is accessible by this addressing mode.



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

Thank You