**Chapter 3 Exercise**

**1. What do the following MOV instructions accomplish?**

**(a)** MOV AX, BX → Copies contents of **BX into AX**.  
**(b)** MOV BX, AX → Copies contents of **AX into BX**.  
**(c)** MOV BL, CH → Copies contents of **CH into BL**.  
**(d)** MOV ESP, EBP → Copies contents of **EBP into ESP**.  
**(e)** MOV RAX, RCX → Copies contents of **RCX into RAX**.

**2. List the 8-bit registers that are used for register addressing.**

**Answer:** AL, BL, CL, DL, AH, BH, CH, DH.

**3. List the 16-bit registers that are used for register addressing.**

**Answer:** AX, BX, CX, DX, SI, DI, BP, SP.

**4. List the 32-bit registers that are used for register addressing in the 80386 through the Core2 microprocessors.**

**Answer:** EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP.

**5. List the 64-bit registers available to the 64-bit mode of the Pentium 4 and Core2.**

**Answer:** RAX, RBX, RCX, RDX, RSI, RDI, RBP, RSP, R8–R15.

**6. List the 16-bit segment registers used with register addressing by MOV, PUSH, and POP.**

**Answer:** CS (Code Segment), DS (Data Segment), ES (Extra Segment), SS (Stack Segment), FS, GS.

**7. What is wrong with the MOV BL, CX instruction?(EXAM)**

**Answer:** Operand sizes don’t match — **BL is 8-bit**, **CX is 16-bit**.

**8. What is wrong with the MOV DS, SS instruction?(EXAM)**

**Answer:** You **cannot directly move** data between **segment registers**. You must use a **general-purpose register** as an intermediate.

**9. Select an instruction for each of the following tasks:**

**(a)** Copy EBX into EDX → MOV EDX, EBX  
**(b)** Copy BL into CL → MOV CL, BL  
**(c)** Copy SI into BX → MOV BX, SI  
**(d)** Copy DS into AX → MOV AX, DS  
**(e)** Copy AL into AH → MOV AH, AL  
**(f)** Copy R8 into R10 → MOV R10, R8

**10. Select an instruction for each of the following tasks:**

**(a)** Move 12H into AL → MOV AL, 12H  
**(b)** Move 123AH into AX → MOV AX, 123AH  
**(c)** Move 0CDH into CL → MOV CL, 0CDH  
**(d)** Move 1000H into RAX → MOV RAX, 1000H  
**(e)** Move 1200A2H into EBX → MOV EBX, 1200A2H

**11. What special symbol is sometimes used to denote immediate data?**

**Answer:** The **# (hash)** symbol (e.g., MOV AL, #25H).

**12. What is the purpose of the .MODEL TINY statement?**

**Answer:** Defines a **memory model** where **code and data share the same segment** (single segment model).

**13. What assembly language directive indicates the start of the CODE segment?(EXAM)**

**Answer:** .CODE

**14. What is a label?**

**Answer:** A **name** used to identify a **memory location** or **instruction address**, e.g.,  
START: MOV AX, BX.

**15. The MOV instruction is placed in what field of a statement?**

**Answer:** In the **opcode (mnemonic) field** of the assembly statement.

### ****16. A label may begin with what characters?****

**Answer:**  
A label may begin with a **letter (A–Z or a–z)** or **underscore (\_)**.  
It **cannot begin with a digit**.

### ****17. What is the purpose of the**** .EXIT ****directive?****

**Answer:**  
The .EXIT directive tells the assembler to **return control to the operating system** — it marks the **end of the program** execution.

### ****18. Does the**** .MODEL TINY ****statement cause a program to assemble as an**** .EXE ****program?****

**Answer:**  
**No.** The .MODEL TINY directive assembles a **.COM (command)** type program,  
**not** an .EXE file.  
(COM programs have code and data in one segment.)

### ****19. What tasks does the**** .STARTUP ****directive accomplish in the small memory model?****

**Answer:**  
The .STARTUP directive:

* Initializes the **DS (Data Segment)** register.
* Sets up the **stack segment** automatically.
* Jumps to the program’s **starting point** for execution.

(Used in small memory model programs to handle segment setup.)

### ****20. What is a displacement? How does it determine the memory address in a**** MOV DS:[2000H], AL ****instruction?(EXAM)****

**Answer:**

* A **displacement** is the **offset value** (the distance from the start of a segment).
* In MOV DS:[2000H], AL,  
  the **effective address (EA)** = 2000H.
* The **physical address** = (DS × 10H) + 2000H.

If DS = 1000H, → Physical address = 10000H + 2000H = **12000H**.

### ****21. What do the symbols [ ] indicate?****

**Answer:**  
Brackets **[ ]** indicate **memory addressing** —  
that the operand refers to a **memory location**, not a register or immediate data.  
Example:  
MOV AL, [BX] → Move the byte from **memory at address in BX** into AL.

### ****22. Suppose that DS = 0200H, BX = 0300H, and DI = 0400H. Determine the memory address accessed by each instruction (real mode).****

#### (a) MOV AL, [1234H]

Effective address = 1234H  
Physical address = (DS × 10H) + 1234H = (0200 × 10H) + 1234H = **3234H**

#### (b) MOV EAX, [BX]

Effective address = BX = 0300H  
Physical address = (DS × 10H) + 0300H = 02000H + 0300H = **02300H**

#### (c) MOV [DI], AL

Effective address = DI = 0400H  
Physical address = (DS × 10H) + 0400H = 02000H + 0400H = **02400H**

### ****23. What is wrong with a**** MOV [BX], [DI] ****instruction?****

**Answer:**  
Both operands are **memory operands**, but the **MOV** instruction does **not allow memory-to-memory transfer** directly.  
→ You must use a register as an intermediate.  
Example:

MOV AL, [BX]

MOV [DI], AL

### ****24. Choose an instruction that requires**** BYTE PTR****.****

**Answer:**  
MOV BYTE PTR [SI], AL  
(Specifies that the operand is a byte in memory.)

### ****25. Choose an instruction that requires**** WORD PTR****.****

**Answer:**  
MOV WORD PTR [BX], AX  
(Specifies that the operand is a 16-bit word in memory.)

### ****26. Choose an instruction that requires**** DWORD PTR****.****

**Answer:**  
MOV DWORD PTR [DI], EAX  
(Specifies that the operand is a 32-bit double word in memory.)

### ****27. Select an instruction that requires**** QWORD PTR****.****

**Answer:**  
MOVQ QWORD PTR [RAX], XMM0  
(Used in 64-bit mode for 8-byte memory operations.)

### ****28. Explain the difference between the**** MOV BX, DATA ****instruction and the**** MOV BX, OFFSET DATA ****instruction.****

| **Instruction** | **Meaning** | **Result** |
| --- | --- | --- |
| MOV BX, DATA | Moves the **contents (value)** stored at label DATA | BX ← [DATA] |
| MOV BX, OFFSET DATA | Moves the **address (offset)** of label DATA | BX ← Address of DATA |

### ****29. Suppose that DS = 1000H, SS = 2000H, BP = 1000H, and DI = 0100H. Determine the memory address accessed by each instruction (real mode).****

#### (a) MOV AL, [BP+DI]

Uses **SS** by default (because BP is used).  
EA = BP + DI = 1000H + 0100H = 1100H  
Physical = (SS × 10H) + 1100H = (2000H × 10H) + 1100H = **21100H**

#### (b) MOV CX, [DI]

Uses **DS** by default.  
EA = DI = 0100H  
Physical = (DS × 10H) + 0100H = (1000H × 10H) + 0100H = **10100H**

#### (c) MOV EDX, [BP]

Uses **SS** by default.  
EA = BP = 1000H  
Physical = (SS × 10H) + 1000H = (2000H × 10H) + 1000H = **21000H**

### ****30. What, if anything, is wrong with a**** MOV AL, [BX][SI] ****instruction?****

**Answer:**  
✅ Nothing is wrong — it is **valid**.  
[BX][SI] means **BX + SI** addressing mode.  
Equivalent to:  
MOV AL, [BX + SI]