**Chapter 10: ADDRESSING MODES**

**Topic – 1: Operands**

**Note!**

**🡪 The operands we pass by name, are actually their memory addresses.**

**Basic Modes Of Addressing**

* Register addressing
* Immediate addressing
* Memory addressing

**Register Addressing**

* These are instructions which involve **a** **register** & **a** **variable** or **two registers**.

**Immediate Addressing**

* Involves moving an **immediate constant** to a **register** or a **reserved variable**.

***add num, 96***

***mov rcx, 78h***

**Direct Memory Addressing**

* In **direct memory addressing** mode, the named **variable** is searched all over the main memory.
* This address after reaching the **offset** of that variable, is stored in **DS** register.
* **Effective address:** Offset
* As we mention more variables in the program, their **offset** values are stored in a **symbol table**.

***mov [0x800], rax***

***mov ax, [0x5000]***

**Direct-Offset Addressing**

* Uses **arithmetic operators** to modify addresses.

***my\_table dw 4,5,6,7,8 ; Our table (same as array in C)***

***mov cl, my\_table[2] ; Moves third element to CL***

***mov cl, my\_table + 2 ; Same as above***

**Indirect Memory Addressing**

* We mostly use **base registers** of **BX**, **BP** series & **index registers** of **DI**, **SI** series for this purpose.
* This way of addressing is used in structures containing **multiple elements**.
* Like **arrays/tables** etc.
* We use **[ ]** around register to refer to the **element** that is in address in register.
* And **without** it, we are referring to the **memory address** it is storing.

***mov ebx, [my\_array] ; Offset of array moved to RBX***

***mov [ebx], 100 ; array[0] = 100***

***mov ebx, 2 ; rbx = rbx + 2 (next element)***

***mov [ebx], 200 ; array[1] = 200***

* The **3rd line** moves by just **one element** because size of a **WORD** is **2-bytes**.

**Topic – 2: MOV Instruction Ambiguity**

**Type Specification**

* When using the **MOV** instruction, the **size** of **both operands** must be **same**.
* But it can cause some ambiguity like we **don’t** specify the **type** of value we are moving into the register.

***mov [ebx], 100 ; Unspecified type***

***mov WORD[ebx], 100 ; Specified and safe***

**Constant Types Table**

|  |  |  |
| --- | --- | --- |
| **Type Specifier** | **Bytes Addressed** | **Definition Keyword** |
| **BYTE** | **1** | **DB** |
| **WORD** | **2** | **DW** |
| **DWORD** | **4** | **DD** |
| **QWORD** | **8** | **DQ** |
| **TBYTE** | **10** | **DT** |

**Variable Types Table**

|  |  |  |
| --- | --- | --- |
| **Type Specifier** | **Bytes Addressed** | **Definition Keyword** |
| **BYTE (reserve)** | **1** | **RESB** |
| **WORD (reserve)** | **2** | **RESW** |
| **DWORD (reserve)** | **4** | **RESD** |
| **QWORD (reserve)** | **8** | **RESQ** |
| **TBYTE (reserve)** | **10** | **REST** |