**Chapter 3: ASSEMBLY LANGUAGE FUNDAMENTALS**

**Topic – 1: Running NASM On Arch Linux**

**Check Processor Info**

***uname -a***

* The **processor** must be **x86-64**.

**Install NASM Assembler**

***sudo pacman -S nasm***

**Make Object File**

***nasm -f elf64 -o hello.o hello.asm***

**Make Executable File**

***ld hello.o -o hello***

**Run The Program**

***./hello***

**Topic – 2: Basic Assembly Language Program**

**First MASM Program**

* Let’s create a program that **adds two numbers** & **stores** the sum in a variable.

***.data ; this is data area***

***sum DWORD 0 ; variable 'sum' of type double word***

***.code ; this is code area***

***main PROC ; main procedure starts***

***mov eax, 5***

***add eax, 6***

***mov sum, eax***

***INVOKE ExitProcess, 0 ; ends the program by returning 0***

***main ENDP ; main program’s ending marker***

* ***DWORD*** assigns **32-bits** of memory.
* Notice that we can put **any data type** variable with that assigned space.
* ***.data*** & ***.code*** are segments, we will see ***.stack*** segment later.
* **Literals:** Constants

**Equivalent NASM Code**

***section .data***

***sum dd 0***

***section .text ; code section***

***global \_start ; program starts***

***\_start: ; main procedure starts***

***mov eax, 5***

***add eax, 6***

***mov [sum], eax ; [] is strictly needed***

***; Exiting the program***

***mov eax, 1 ; sys\_exit system call***

***xor ebx, ebx ; returns 0***

***int 0x80 ; interrupt to call sys\_exit***

* ***int*** above stands for **interrupt**.

**Integer Literals**

* It can have an optional **leading symbol** and/or **tailing radix** character.
* A number **without** any radix is a **decimal integer literal**.
* **Hexadecimal** **26** will be **26h**.

|  |  |
| --- | --- |
| **Tailing Radix Character** | **Meaning** |
| **h** | **Hexadecimal** |
| **q/o** | **Octal** |
| **d/t** | **Decimal** |
| **b/y** | **Binary** |
| **r** | **Encoded real** |