**Chapter 8: Registers Revision**

**Topic – 1: Memory Segments**

**Introduction**

* We can replace the ***section*** keyword with ***segment***.
* As per the program we write, system memory is divided into many **memory segments** to store different types of data.
* One segment stores **instructions**, one stores **data**, while another one stores the **program stack**.

**Comment About Segments**

* Data segment is represented by ***.data*** & ***.bss*** sections.
* Memory space occupied by ***.data*** is **fixed** from starting, while it may change for variables in ***bss*** section.
* We can change the data present at data section.
* The difference between ***data*** & ***bss*** section is that variables at data comes initialized while variables in ***bss*** is initialized in runtime.
* ***text*** section is also a **fixed area** because we **can’t** modify the program when it is running.
* **Stack segment:** Stores the values of data passed to **functions & procedures** throughout the program.
* Stores and removes data in style of a **stack**.

**Topic – 2: Registers**

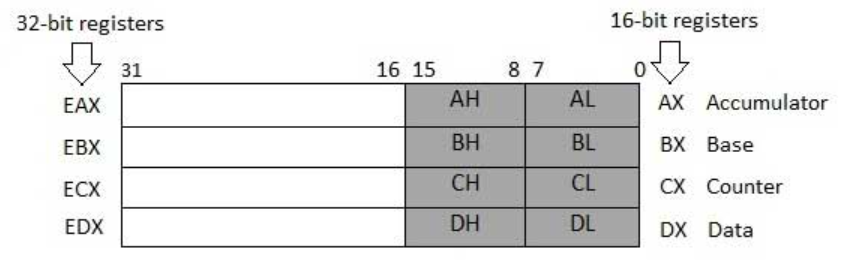
**Introduction**

* Data can be stored in memory too **during** execution of a program.
* But this process is **slow**, as it includes sending **signal** for **data request** on the **control bus** & waiting for the data to be transferred from data bus.
* **Registers:** Fast access memory locations in CPU.

**Types Of Registers**

* **General registers –** Data registers, pointer registers & index registers.
* **Control registers**
* **Segment registers**

**Data Registers**



* For the **64-bit** registers, same registers are extended by **32-bits**.
* **Primary accumulator (AX):** Used in I/O & arithmetic operations.
* **Base register (BX):** Used for addressing through **indexes**.
* **Count register (CX):** Generally used to store **loop count**.
* **Data register (DX):** Used in I/O & arithmetic operations with **AX**.

**Pointer Registers**

* Includes **RIP**, **RSP** & **RBP**.
* **Instruction pointer (IP):** Points to the address of **next instruction** to be executed. Gives **address** of **current instruction** when combined with bits of **CS** register.
* **Stack pointer (SP):** Points to the **offset** of **program stack**. Gives **position** of **current instruction** when combined with **SS** register.
* **Base pointer (BP):** Helps in referencing to **parameter variables** passed to a sub-routine. Combined with **SS** register to get address of a particular variable.
* **BP** is often combined with **DI** & **SI** for **special addressing** purposes.

**Index Registers**

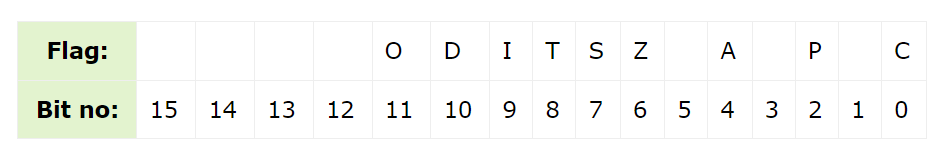
* These are **RSI** & **RDI**.
* Are primarily used in **addressing through indexes** & sometimes for **addition** or **subtraction**.
* **Source index (SI) –** Used as **source index** in string operations.
* **Destination index (DI) –** Used as **destination index** in string operations.

**Control Registers**

* 32-bit **IP** register & 32-bit **flag register** are together known as ***control registers***.
* Some instructions that **compare** or do **mathematical calculations** may result in changing status of certain **flags**.

**Flag Bits**

* **Overflow flag (OF)**
* **Direction flag (DF):** Tells the **direction** of string’s character comparison. **0** means **left to right**.
* **Interrupt flag (IF):** Disables **external interrupts** like keyboard at **0**.
* **Trap flag (TF):** It is enabled when an **instruction is getting executed** & **no** other program can interfere at that time.
* **Sign flag (SF)**
* **Zero flag (ZF)**
* **Auxiliary carry flag (AX):** Enabled when a **carry** is generated for **4th bit** from **3rd bit**.
* **Parity flag (PF):** Set to **0** when the number of **1s** in a result is even.
* **Carry flag (CF)**
* **CF** also stores **last bit** of a **shift** & **rotate** operation.



**Segment Registers**

* **CS** register stores the starting address of the **code segment**.
* **DS** register stores the starting address of the **data segment**.
* **SS** register stores the starting address of the **stack segment**.
* Other segment registers – **ES** (extra segment), **FS** & **GS**.

**Topic – 3: Memory Address**

* Addresses are stored in **hexadecimal** form.
* And **segment addresses** (offsets) are divisible by **decimal 16** (**hex 10**).
* So, the last digit **isn’t** stored in **segment register**, being very obvious.
* To reference any memory location in a segment, the processor combines **segment address** with **offset** of location.

**Topic – 4: Repeated Star Program**

**Tip!**

**🡪 In new programs, focus on comment line for code & understanding what’s new.**

***section .data***

***msg times 9 db '\*' ; A string with 9 repeated \****

***section .text***

***global \_start***

***\_start:***

***mov edx, 9 ; Number of bytes to write***

***mov ecx, msg ; msg to write***

***mov eax, 1***

***xor ebx, ebx***

***int 0x80***

* Notice that no **newline** (***0xa***) is used for **constant** ***msg***.

***OUTPUT:***

***\*\*\*\*\*\*\*\*\****