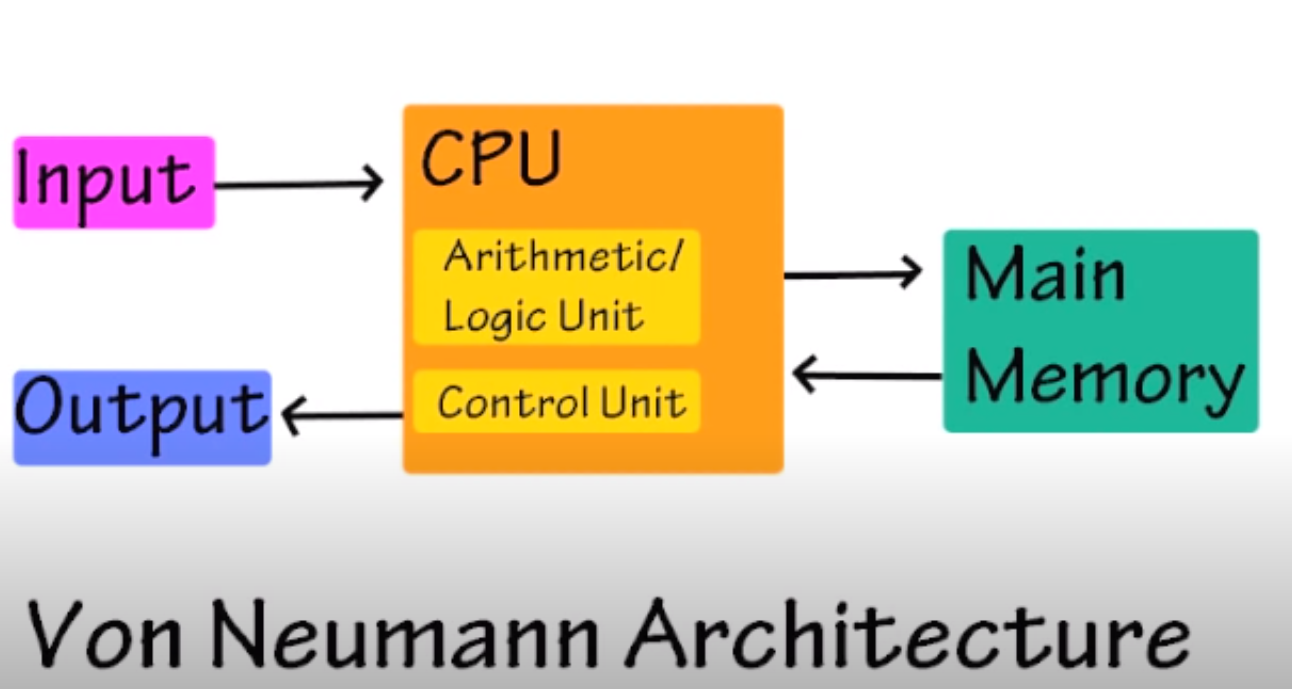
**Part 1 Tasks:**   
**In a written document, answer the following questions in relationship to the topics covered by this unit:**

* What is the classical concept of a Von Neumann machine architecture?

As discussed in the learning materials and shown in a few graphs Von Neumann architecture is the design on which most computers are based



The image from the materials is a good basic representation, input goes in, the CPU uses its ALU and CU, also feeding data in and out of main memory during the fetch/decode/execute cycles and then output is created.

* Describe the components of a CPU and what each one does.

As per the components as shown in the Von Neumann Architecture image the CPU consists of the arithmetic logic unit, which does the heavy lifting and performs logical operations and the control unit does all the decoding and executing. The CPU interacts with its various components throughout the fetch/decode/execute cycle and now by today’s standards can perform billions of calculations in seconds. The text also has a pretty cool picture of an Intel Xeon E7 processor that contains ten cores to demonstrate that CPU microprocessor technology has been advancing in line with Moore’s law...

Furthermore, the CU also controls the input and output, runs all the necessary computing power to maintain the functioning of the I/O of the computer. The control units fetches instructions further from memory, and memory stores only bits. The memory also stores both encoded instructions and data to operate on and is organized into cells. This is the key to the cyclical computing scheme that is the Von Neumann architecture. It gained popularity after the 1930s as engineers were trying to automate calculations in a more efficient way, and it just became so dominant since it was robust and easy to program.

* Describe 3 special purpose registers found in the Von Neumann machine architecture.

There are multiple special purpose register found in the Von Neumann architecture.

Three would include the Current Instruction Register, the Accumulator, and the Memory Address Register.

The MAR holds the addresses during all the calculations so it can designate the address of whatever the current instruction is- or where it has to go. The CIR as per its name is holding that calculation, or instruction rather that is actually currently be worked on, decoded and then executed. The Accumulator then holds all that data being processing and returns results.

The rest of the special registers in the Von Neumann architecture include the program counter that holds memory addresses for what’s next in queue, and the memory data register or MDR (as opposed to MAR) which holds all that data at the address the MAR provided and where it waits to be received by the primary memory.

So as per the architecture, it performs this cyclical route in calculations to operate and that architecture, framework, structure- the science behind it is the Von Neumann architecture.

**Part 2 Tasks:**

**The code:**

section .text ; This must be here to tell the compiler

; where Program’s instructions start

global \_start ; Must be declared for using gcc

\_start: ; Tell linker entry point

mov ebx, 1 ; File descriptor (stdout)

mov edx, len ; Message length

mov ecx, msg ; Message to write

mov ebx, 1 ; File descriptor (stdout)

mov eax, 4 ; System call number (sys\_write)

int 0x80 ; Call the kernel

mov ebx, 1 ; File descriptor (stdout)

mov eax, 4 ; System call number (sys\_write)

mov edx, len2 ; Message length

mov ecx, msg2 ; Message to write

int 0x80 ; Call the kernel

mov ebx, 1 ; File descriptor (stdout)

mov eax, 4 ; System call number (sys\_write)

mov edx, len3 ; Message length

mov ecx, msg3 ; Message to write

int 0x80 ; Call the kernel

mov ebx, 1 ; File descriptor (stdout)

mov eax, 4 ; System call number (sys\_write)

mov edx, len4 ; Message length

mov ecx, msg4 ; Message to write

int 0x80 ; Call the kernel

mov eax, 1 ; System call number (sys\_exit)

int 0x80 ; Call the kernel

section .data ; This must be here to tell the compiler

; where Program’s data starts

msg db 'Hello, world!',0xa ; Line 1

len equ $ - msg ; Length of our dear string

msg2 db 'CE 242 is an awesome and fun class!',0xa ; Line 2

len2 equ $ - msg2 ; Length of 2nd string

msg3 db 'CTU is a great University!',0xa ; Line 3

len3 equ $ - msg3 ; Length of 3rd string

msg4 db 'I love the USA!',0xa ; Line 4

len4 equ $ - msg4 ; Length of 4th string

* Screenshot:

