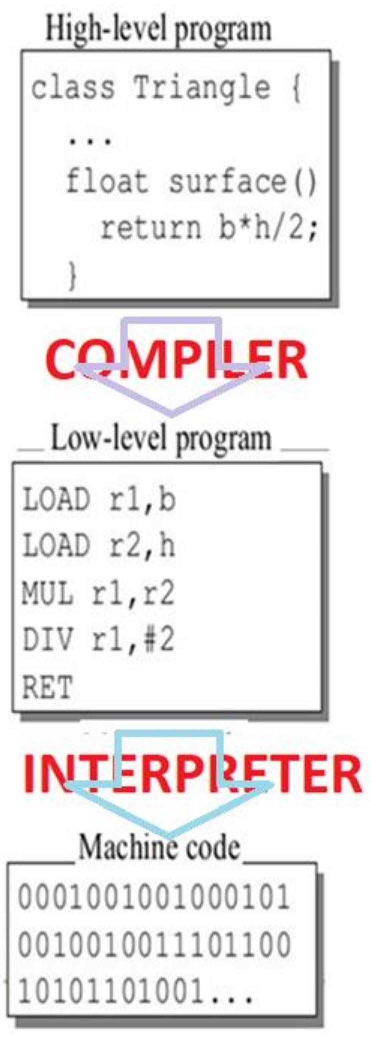
Sample questions: Lecture-4

What is the first step to run a program written in high level language?

Programs in high level language contain English words, arithmetic/mathematical notations, symbols etc. Data are represented in decimal, texts etc.
Functional units of a digital computer (CPU, Memory, storage, Input/output devices) are designed to process, store and transfer information in binary forms. It means, Programs and data are to be converted to binary for the electronic functional units of a computer.
Statements in high level programs are converted into a sequential list of instructions for basic operations. Instructions are translated/converted/interpreted in binary bit strings of 1's and 0's called machine codes. Data, both numbers and texts are also converted into binary bit strings of 1's and 0's. The process is explained with the following diagram.



```
class Triangle {
  ...
  float surface()
    return b*h/2;
  }
```

```
LOAD r1,b
LOAD r2,h
MUL r1,r2
DIV r1,#2
RET
```

```
0001001001000101
0010010011101100
10101101001...
```

It is to be noted that 8 bits or multiple of 8-bits are used to form machine codes. Similarly 8-bits or multiple of 8-bits are used to represent data.

Why programs and data are required to convert into binary?

The CPU is a semiconductor device, designed to work with binary coded instructions, called machine codes and perform arithmetic and logical operations on binary coded data.

How a program is loaded to RAM?
A program, written in high level language or in assembly language, is converted into a sequential list of machine codes. The list of machine codes as generated by the interpreter are loaded into consecutive locations main memory locations in exactly the same sequence.
The address of first instruction to be assigned for a program is usually set by the operating system to make the process easy for programmers.
Example: A byte addressable RAM and each instruction is of 1 Byte

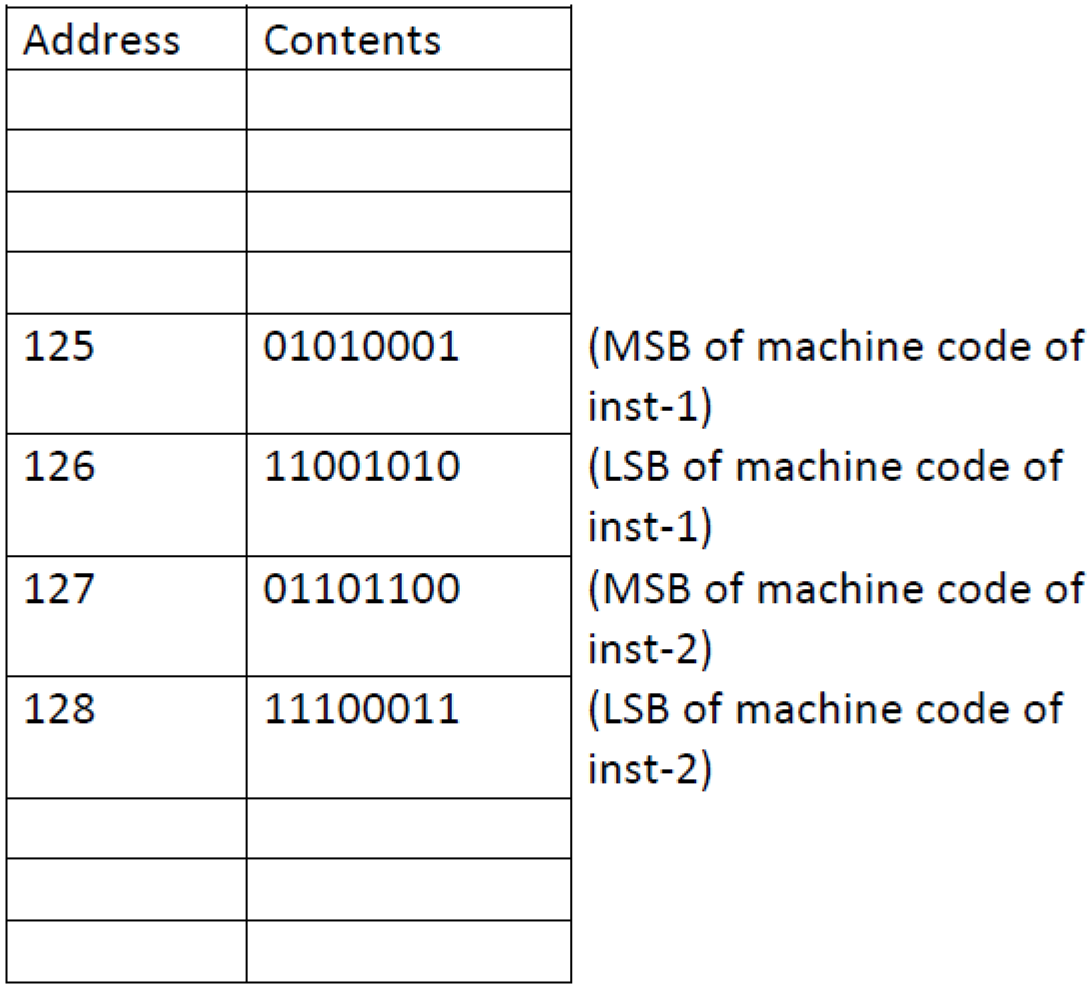
Address	Contents	
125	11001010	(machine code of inst-1)
126	01010001	(machine code of inst-2)
127	11100011	(machine code of inst-3)
128	01101100	(machine code of inst-4)

How instructions are loaded into RAM if a Byte addressable memory is used and each machine code is of 2 bytes.
Machine codes are loaded in consecutive locations of RAM. In case of a byte addressable RAM and each machine code of 2 bytes, two consecutive locations are required for each machine code. The machine codes are loaded into RAM in two standards: Little endian and Big endian.
In little endian, lower byte of machine code (LSB) is loaded into lower address of RAM and higher byte of machine code (MSB) is loaded into higher address of RAM.

Address	Contents	
125	11001010	(LSB of machine code of inst-1)
126	01010001	(MSB of machine code of inst-1)
127	11100011	(LSB of machine code of inst-2)
128	01101100	(MSB of machine code of inst-2)

(2 Bytes Instructions stored in Little Endian form)

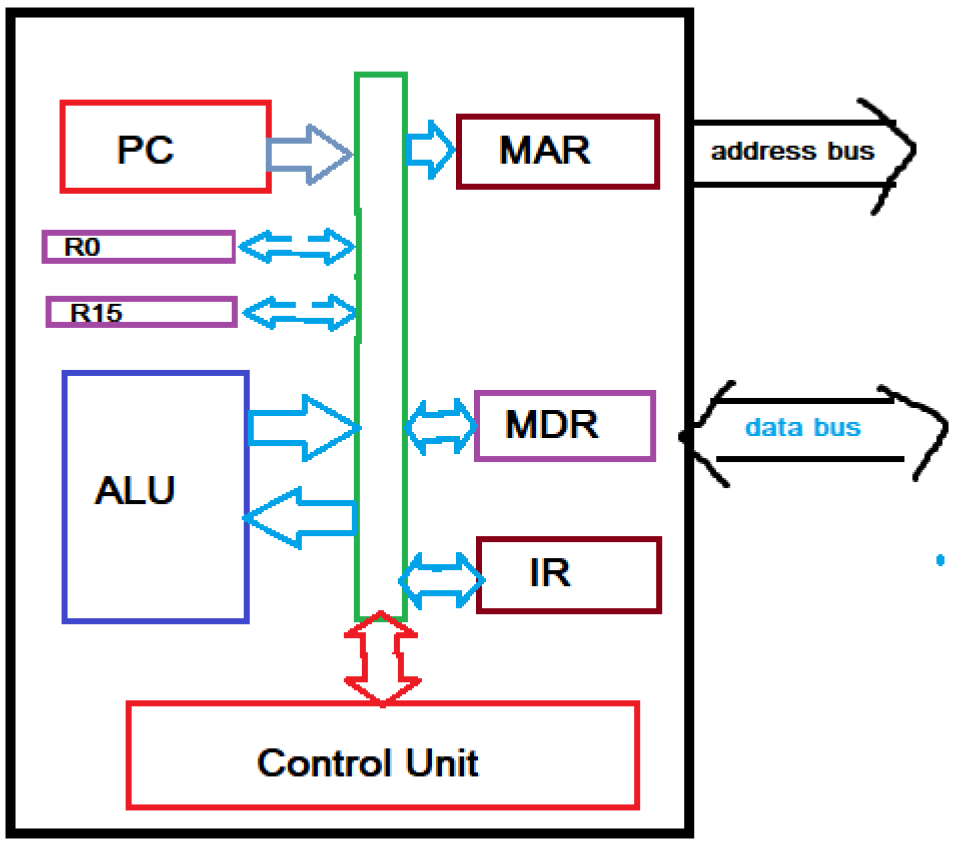
In Big endian, higher byte of machine code (MSB) is loaded into lower address of RAM and lower byte of machine code (LSB) is loaded into higher address of RAM.



Address	Contents	
125	01010001	(MSB of machine code of inst-1)
126	11001010	(LSB of machine code of inst-1)
127	01101100	(MSB of machine code of inst-2)
128	11100011	(LSB of machine code of inst-2)

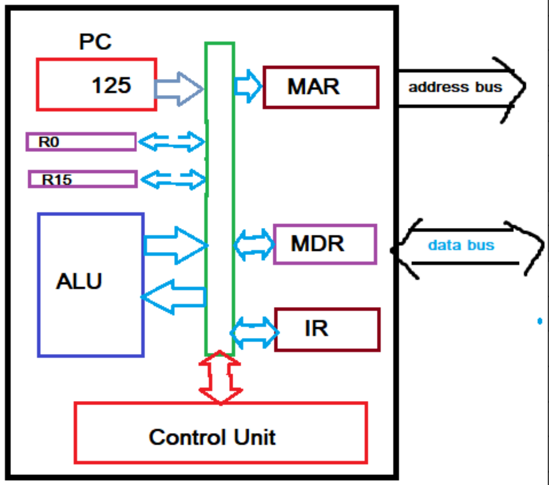
(2 Bytes Instructions stored in Big Endian form)

Show important registers used by the CPU to run a program?
CPU uses following register to run any program
PC: Program Counter
MAR: Memory Address Register
MDR: Memory Data Register
IR: Instruction Register



In most processors, similar register are found but in different names assigned by manufacturers.

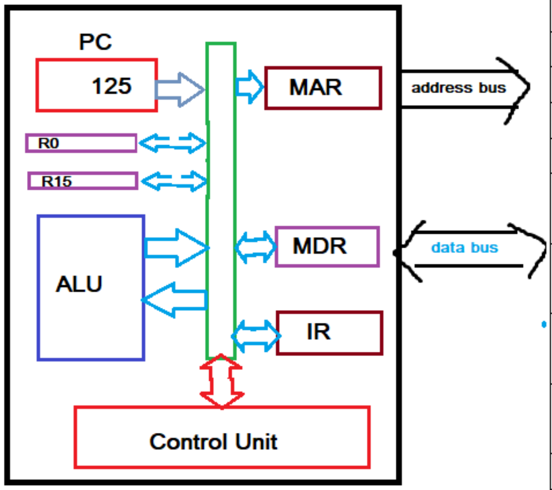
What is Program Counter?
Program Counter is a special register in CPU.
It holds the address of Instruction (machine code) to be read from RAM.
To run a program, the machine codes and data are loaded into RAM and then operating system loads program counter with the address of 1st instruction of the program, shown below.



Address	Contents	
125	11001010	(machine code of inst-1)
126	01010001	(machine code of inst-2)
127	11100011	(machine code of inst-3)
128	01101100	(machine code of inst-4)

Moreover, as CPU reads the machine code, means, the machine code is copied from RAM to a register within CPU, the program counter is designed to increment so that it points the next instruction to be read.
In above case, the PC will be incremented by 1 using an adder close to it so that PC can hold address of next instruction to be read from RAM. Once CPU reads machine code of 1st instruction, PC will be incremented to 126 to point 2nd instruction.

Will PC always be incremented by 1?
How the PC will be incremented, depends on memory word size and size of machine code.
If a byte addressable RAM is used and size of each machine code is 1 byte, then the PC will be incremented by 1.
If a byte addressable RAM is used and size of each machine code is 2 bytes (16-bits), then PC will be incremented by 2, shown below



Address	Contents	
125	11001010	(LSB of machine code of inst-1)
126	01010001	(MSB of machine code of inst-1)
127	11100011	(LSB of machine code of inst-2)
128	01101100	(MSB of machine code of inst-2)

If a byte addressable RAM is used and size of each machine code is 4 bytes (32-bits), then PC will be incremented by 4.

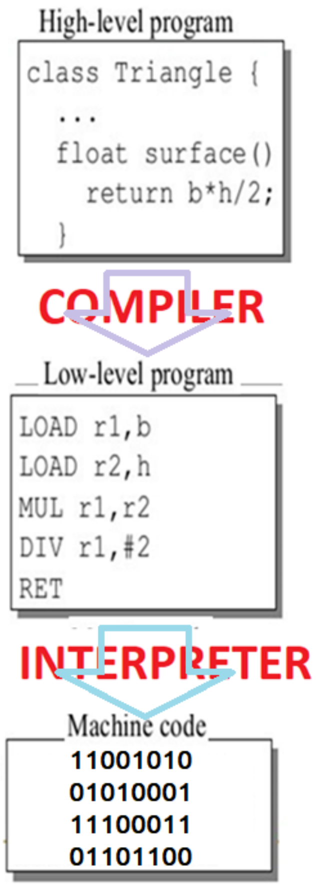
What is IR?
Instruction register. This register always holds machine code of instruction. CPU reads machine code of instruction and stores in IR.

How does CPU get the address of Data stored in RAM?
Machine code of instructions hold memory address for data to be read from RAM and used in operation. Here a typical machine code is shown

Operation code	RAM address for data-1	RAM address for data-2	Result field
110000	11100011	00110011	1010

CPU first reads machine code of instruction. When the machine code is decoded at the control unit, CPU will get the memory addresses for data to be read from RAM.
CPU sends the address (11100011) of 1st data to address bus and reads data-1, saves to a register. Then CPU sends the address (00110011) of 2nd data to address bus and reads data-1, saves to another register.

How does CPU run a program?



```
class Triangle {
  ...
  float surface()
    return b*h/2;
  }
```

```
LOAD r1,b
LOAD r2,h
MUL r1,r2
DIV r1,#2
RET
```

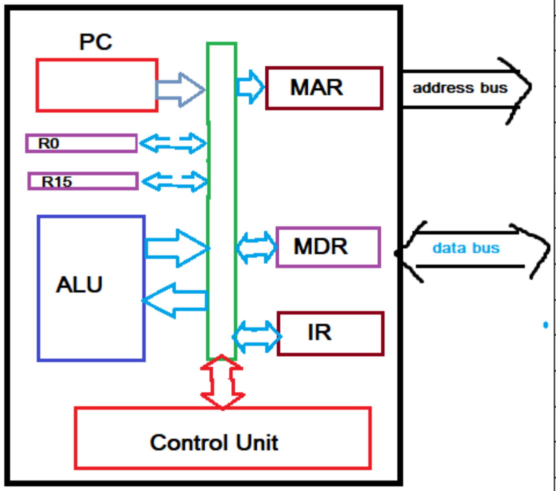
```
11001010
01010001
11100011
01101100
```

STEP-1:
Program is converted into a list of instructions to be processed in a sequence as generated by the compiler. Instructions are represented into binary strings of 1's and 0's called machine codes. Data are also converted into binary bit strings of 1's and 0's.

...

Step-2:

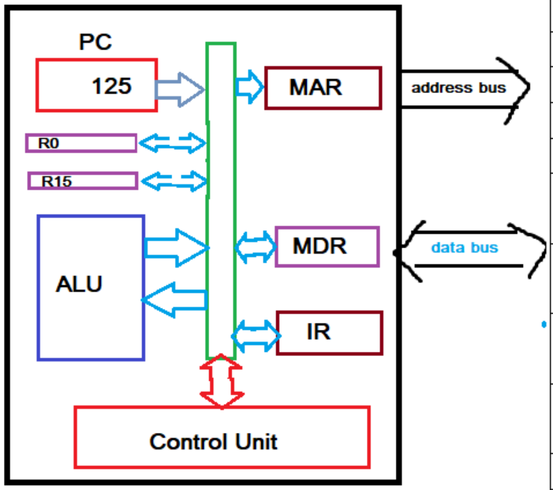
Machine codes are loaded to RAM in consecutive locations. The starting address is assigned by the operating system. In following figure, it is 125.



Address	Contents	
$\boxed{125}$	11001010	(machine code of inst-1)
126	01010001	(machine code of inst-2)
127	11100011	(machine code of inst-3)
128	01101100	(machine code of inst-4)

Step-3:

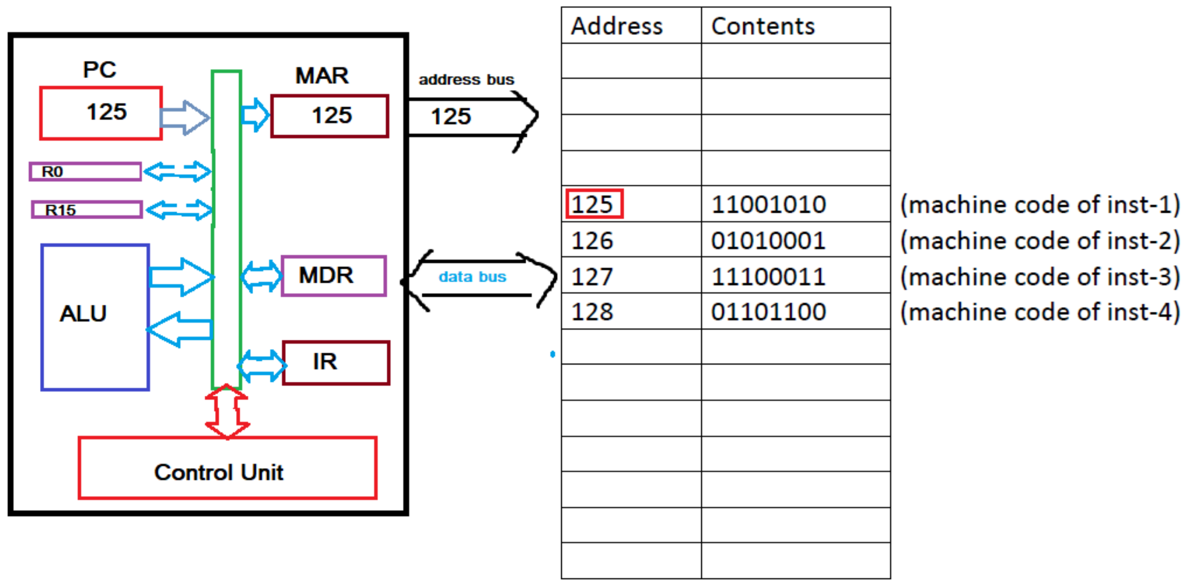
Program counter is also loaded with address of 1st instruction, 125 by the operating system



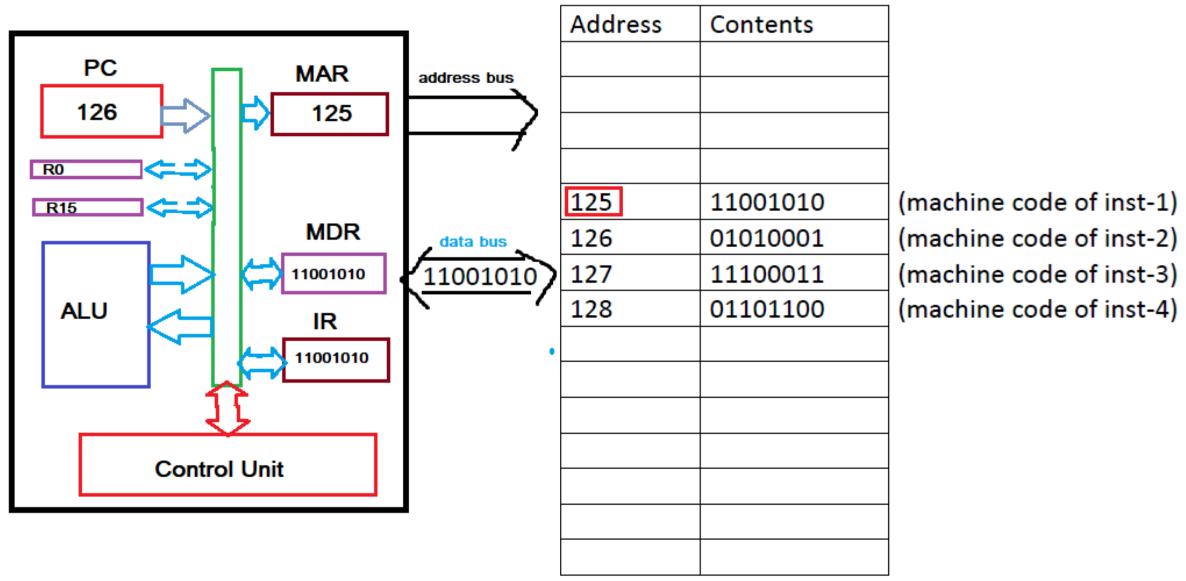
Address	Contents	
125	11001010	(LSB of machine code of inst-1)
126	01010001	(MSB of machine code of inst-1)
127	11100011	(LSB of machine code of inst-2)
128	01101100	(MSB of machine code of inst-2)

Step-4:

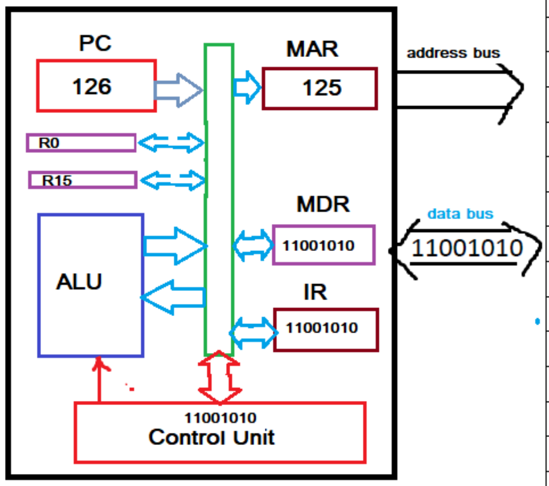
The contents of PC is transferred/loaded to MAR and sent to RAM through address bus.



Step-5:
CPU reads machine code of instruction from RAM through data bus and saves it to MDR and then loads the machine code to IR.
PC is incremented to 126 to point next instruction to be read from RAM



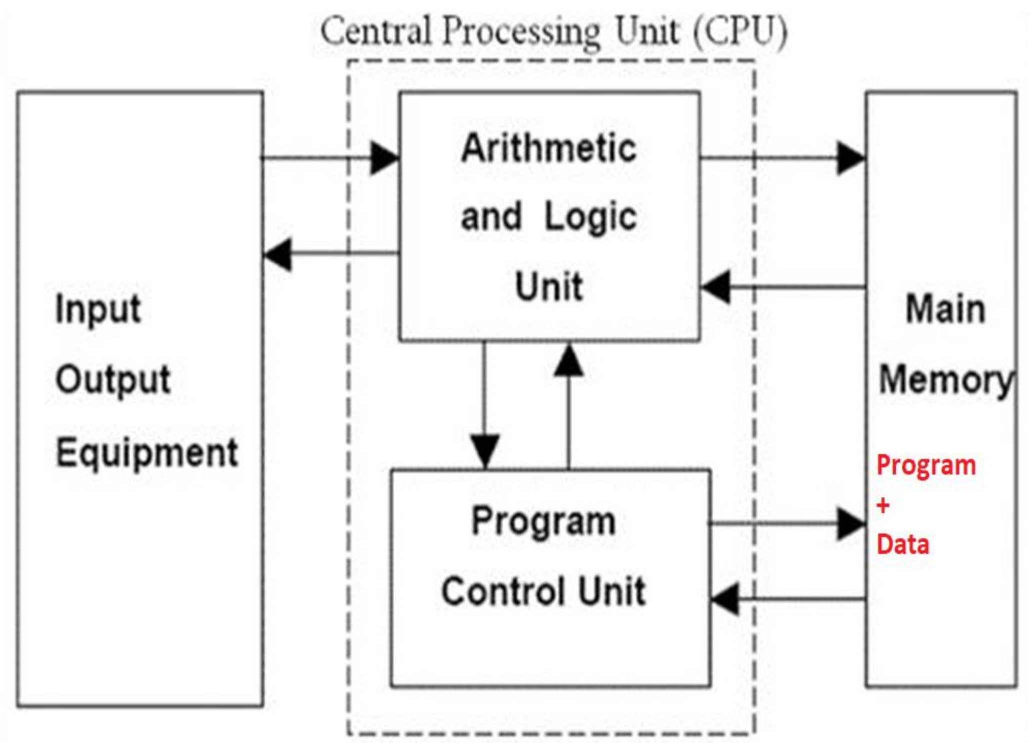
Step-6:
The machine code is decoded at the control unit and instruction is executed electronically.



Address	Contents	
125	11001010	(machine code of inst-1)
126	01010001	(machine code of inst-2)
127	11100011	(machine code of inst-3)
128	01101100	(machine code of inst-4)

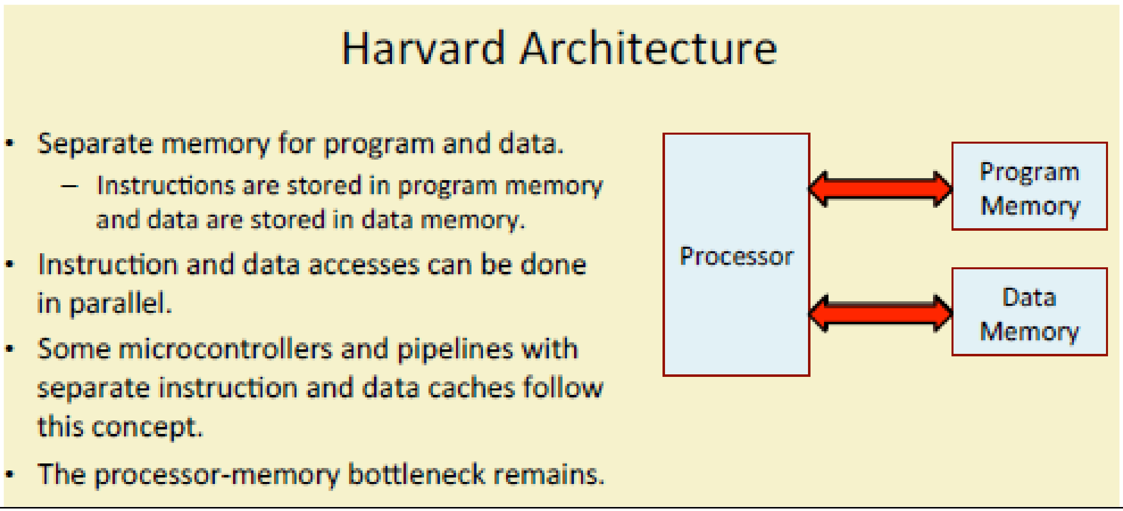
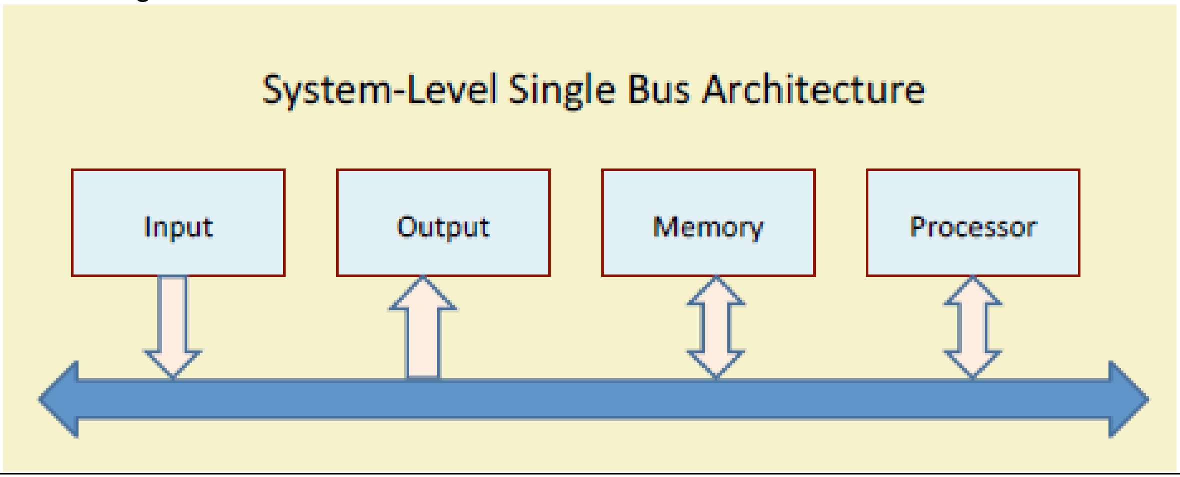
Steps 4 to 6 are repeated for all instructions. As a results the user program is run.
It is to be noted that the CPU is designed to follow a number of steps/tasks to process any instruction in general and the steps or tasks are repeated for each instruction. The CPU is designed to process instruction, basic operation, at a time.
…

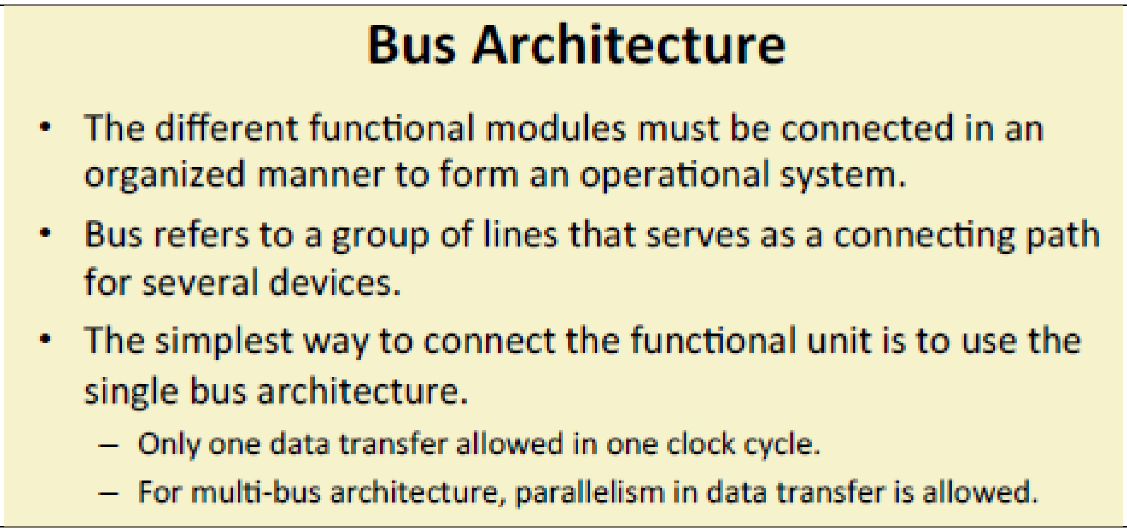
Show the Von Neumann architecture?



List the main characteristics of Von Neumann architecture.

- Data and instructions are both stored in a single primary storage
- Instructions are read from memory one at a time and in order (serially as it is stored/as it appears in a program).

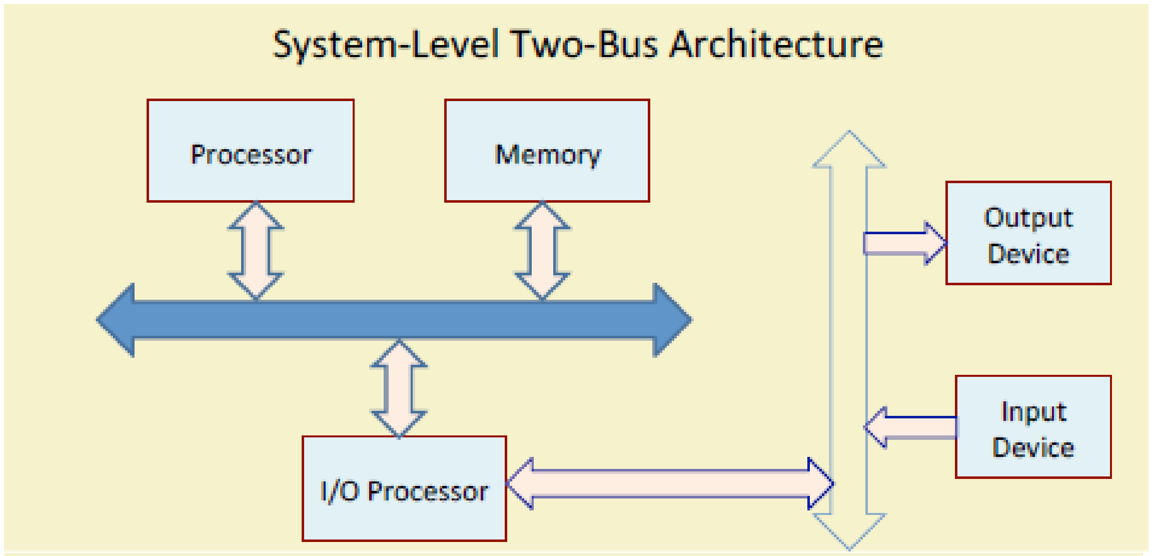
	• The processor decodes and executes one instruction at a time. • Result is stored • CPU reads next instruction from RAM • This process continues till the end of the program. • Parallel implementation of instructions is not allowed. • Instructions can only be carried out one at a time and sequentially. • CPU cannot read Instruction and data simultaneously due to a single bus system
	What is Von Neumann bottleneck? An instruction read/fetch and a data operation cannot occur at the same time because they share a common bus. CPU has to wait and remains idle for a certain amount of time while low speed memory is being accessed. Since a single bus is used, the data rate of bus will decide the overall performance of Von Neumann architecture.
	What is Harvard architecture?  Harvard Architecture • Separate memory for program and data. – Instructions are stored in program memory and data are stored in data memory. • Instruction and data accesses can be done in parallel. • Some microcontrollers and pipelines with separate instruction and data caches follow this concept. • The processor-memory bottleneck remains. Processor Program Memory Data Memory
	What is Single bus architecture?  System-Level Single Bus Architecture Input Output Memory Processor

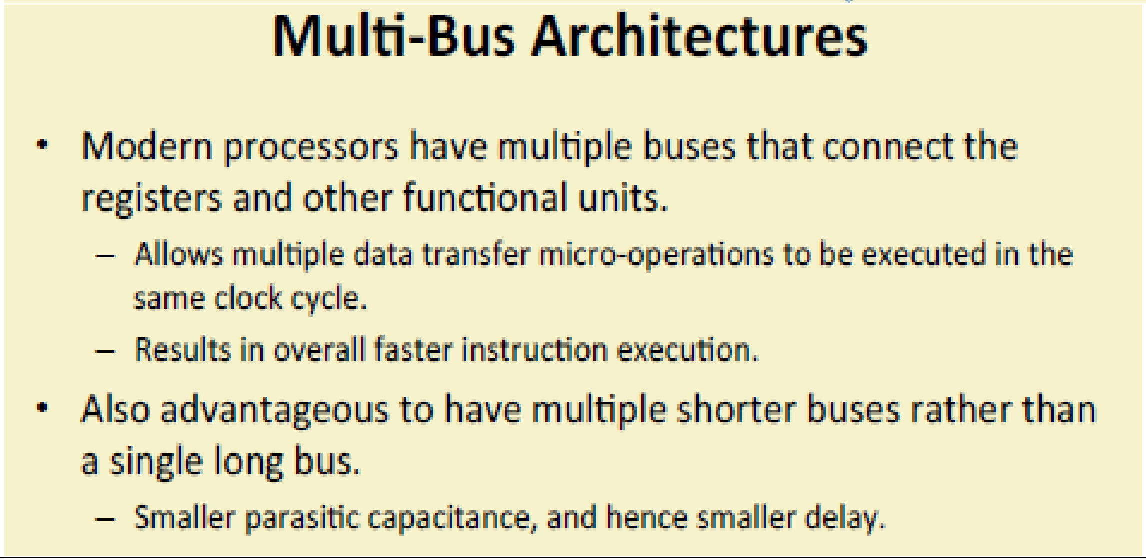


Bus Architecture

- The different functional modules must be connected in an organized manner to form an operational system.
- Bus refers to a group of lines that serves as a connecting path for several devices.
- The simplest way to connect the functional unit is to use the single bus architecture.
  - Only one data transfer allowed in one clock cycle.
  - For multi-bus architecture, parallelism in data transfer is allowed.

What is multi-bus architecture?

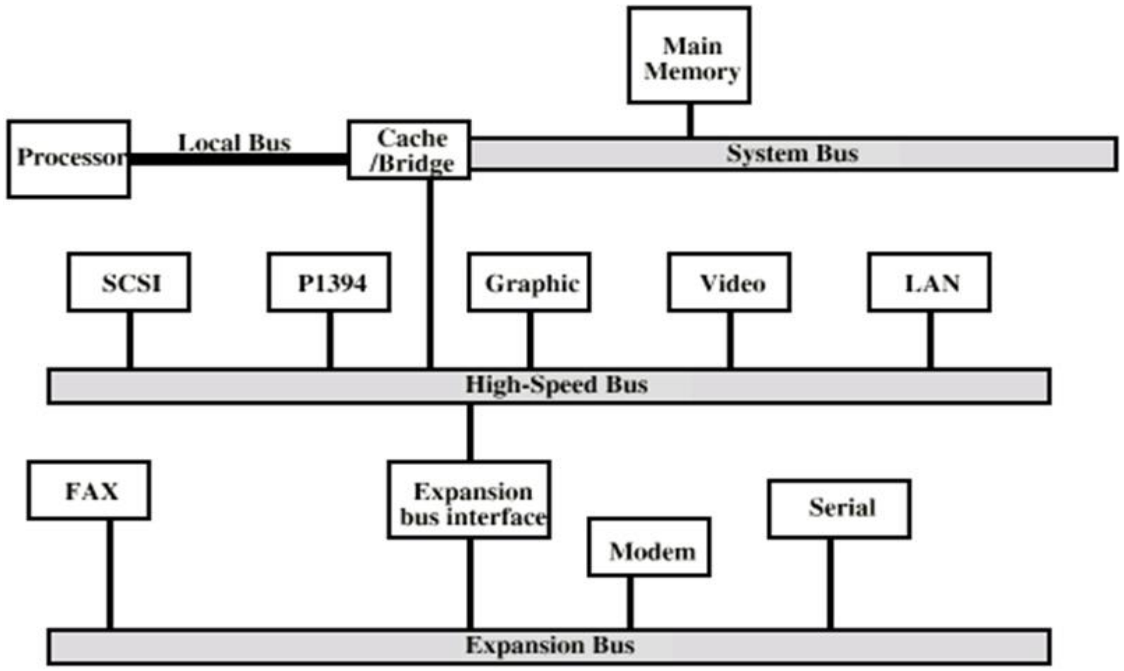




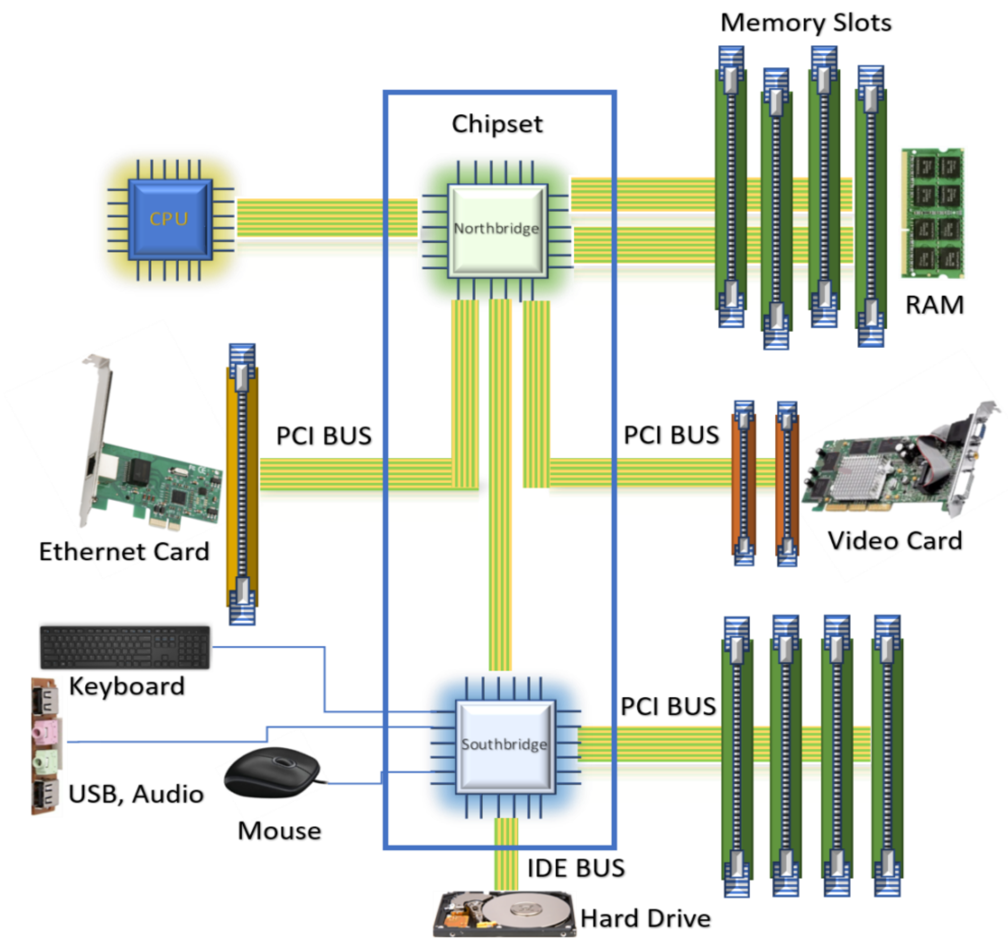
Multi-Bus Architectures

- Modern processors have multiple buses that connect the registers and other functional units.
  - Allows multiple data transfer micro-operations to be executed in the same clock cycle.
  - Results in overall faster instruction execution.
- Also advantageous to have multiple shorter buses rather than a single long bus.
  - Smaller parasitic capacitance, and hence smaller delay.

Show a typical commercial multi-bus architecture.



Main Memory
Processor
Local Bus
Cache /Bridge
System Bus
SCSI
P1394
Graphic
Video
LAN
High-Speed Bus
FAX
Expansion bus interface
Modem
Serial
Expansion Bus



Memory Slots
Chipset
CPU
Northbridge
RAM
PCI BUS
PCI BUS
Ethernet Card
Video Card
Keyboard
PCI BUS
Southbridge
USB, Audio
Mouse
IDE BUS
Hard Drive