Computer Architecture

Keivan Navi

Cal Poly Pomona University

knavi@cpp.edu

Office hours: Tu/Th 5:25 Pm to 6:55 Pm

Office: 8-49

Computer Architecture

Some Sample CPU Architectures

Different Instruction Format will result in different CPU Architecture and Vice Versa

- Operation Operand Format:
  As an example:
- Add 100 -> AC<= AC +(100)
- It seems we can add the accumulator with any memory cell in the main memory.
- But in reality the realization of this, may result in a huge delay in some cases, even if it is doable.
- Imagine that we need the same memory cell for the next instruction. Then we are obligated to spend extra time to access the main memory for the same information.
- It is better to use an intermediate register to save the possible required data.

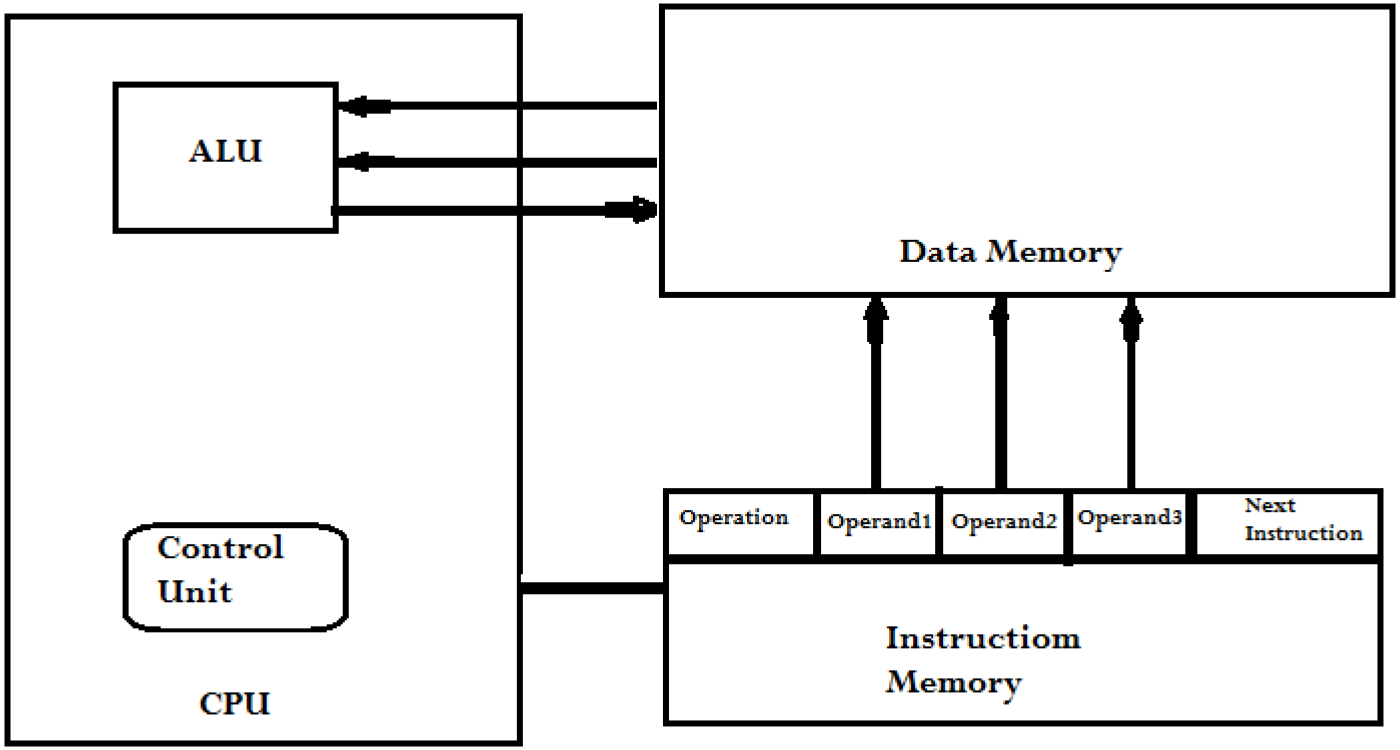
Memory-Memory Architecture (Separate Memory/ Harvard Architecture)

Operation Operand1, Operand2, Operand3, Next Instruction

Add 100,101,104

100 <= (101) + (104)

- The programmer must take care of the next address. There is no program counter.



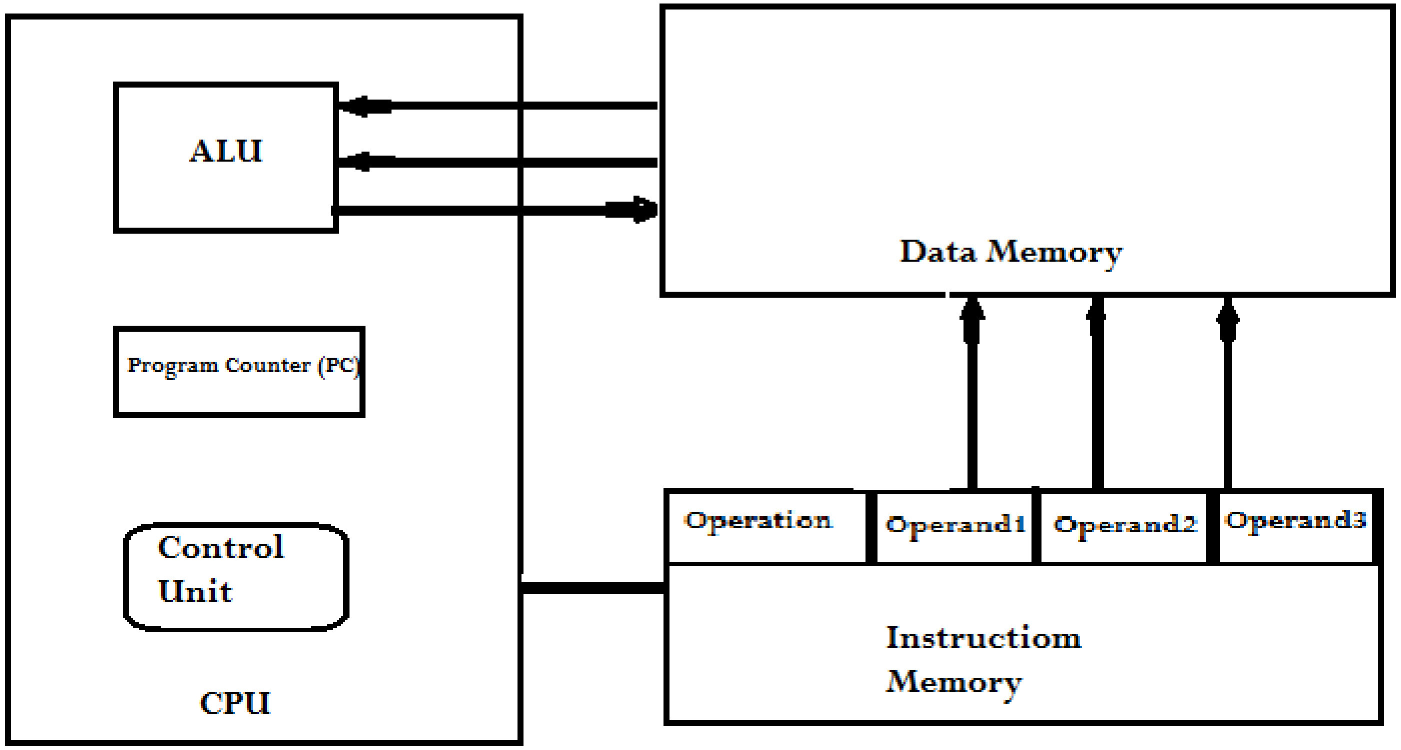
Memory-Memory Architecture (Separate Memory/ Harvard Architecture)

Operation Operand1, Operand2, Operand3

Sub 100, 105,102

100 <= (105) – (102)

- The Program Counter (PC) is the responsible of defining the next instruction, in other words the programmer won’t take care of it:



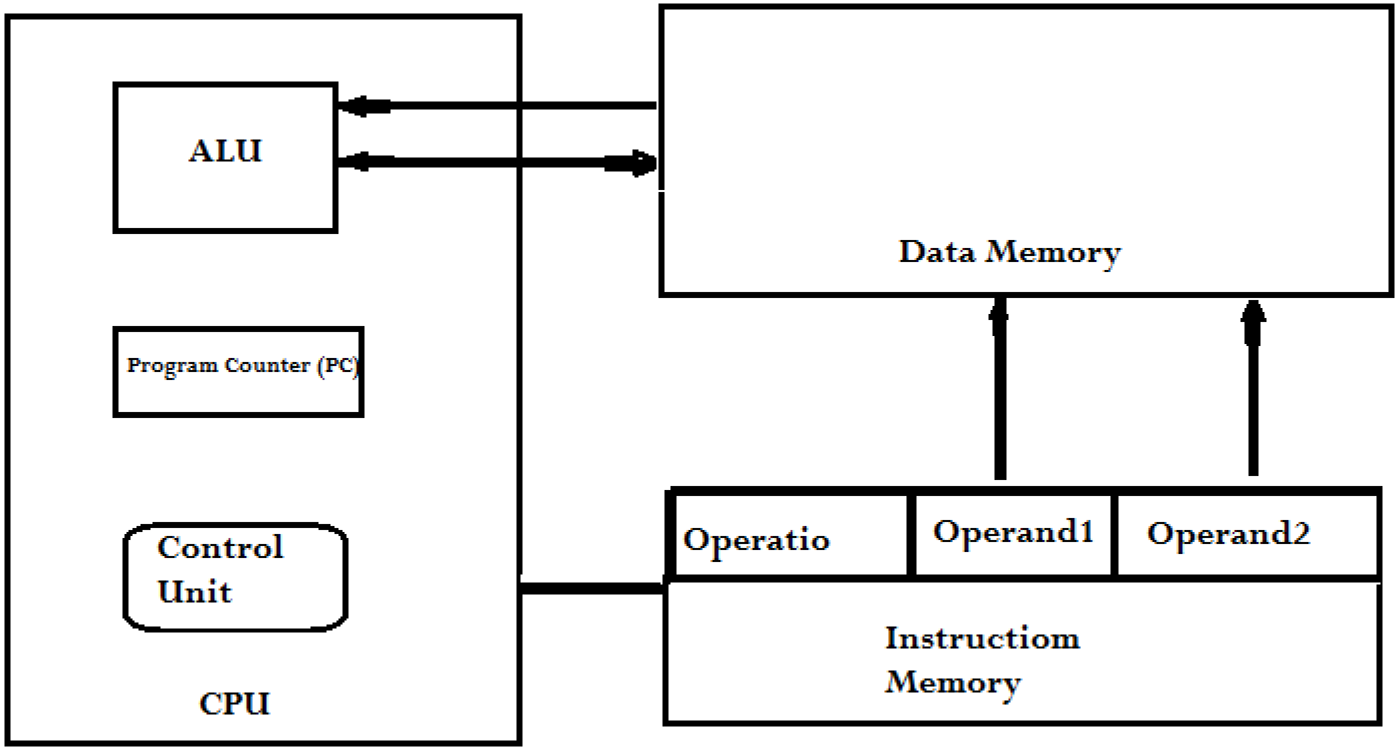
Memory-Memory Architecture (Separate Memory/ Harvard Architecture)

Operation Operand1, Operand2

And 100, 105

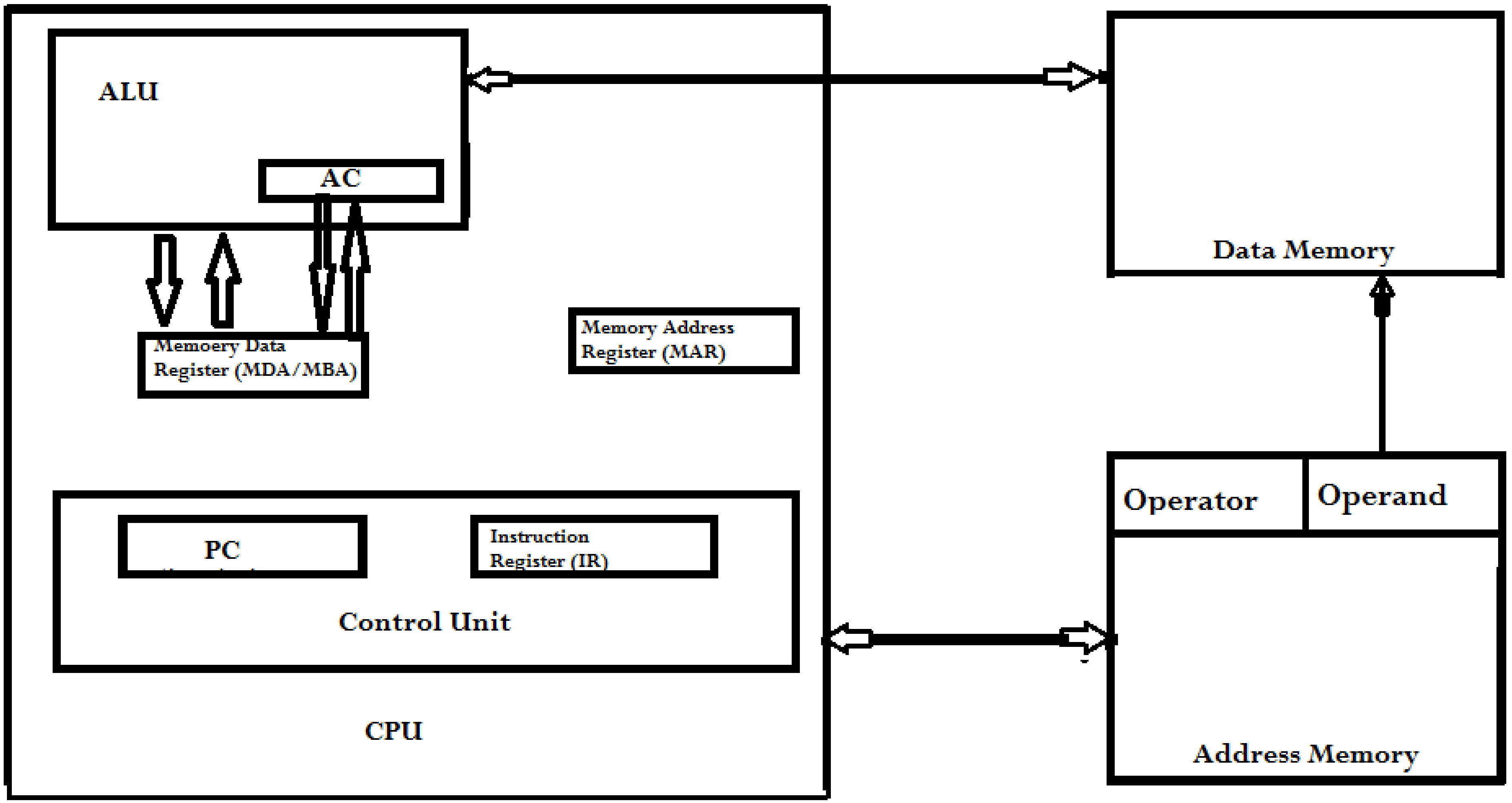
100 <= (100) . (105)

- Operand1 can act as “Source” as well as “Destination”.



Register-Memory Architecture (Separate Memory/ Harvard Architecture)
Operation Operand
Add 70
ac <= (ac) + (70)

- Accumulator is “Source” and “Destination” and one operand reside in the memory.


ALU
AC
Memoery Data Register (MDA/MBA)
Memory Address Register (MAR)
PC
Instruction Register (IR)
Control Unit
CPU
Data Memory
Operator
Operand
Address Memory

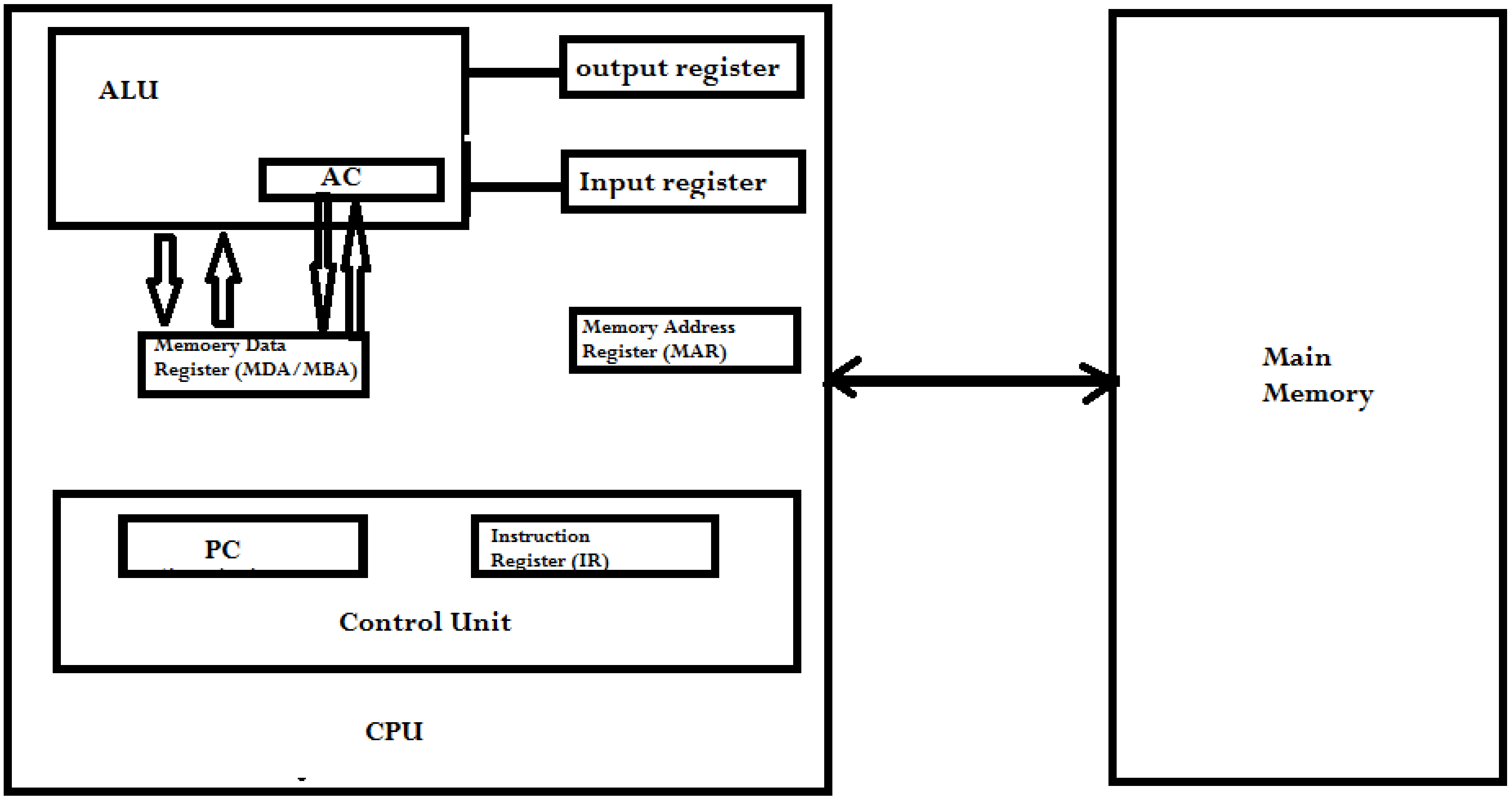
Register-Memory Architecture (Unified Memory / Von Neumann Architecture)

Operation Operand:

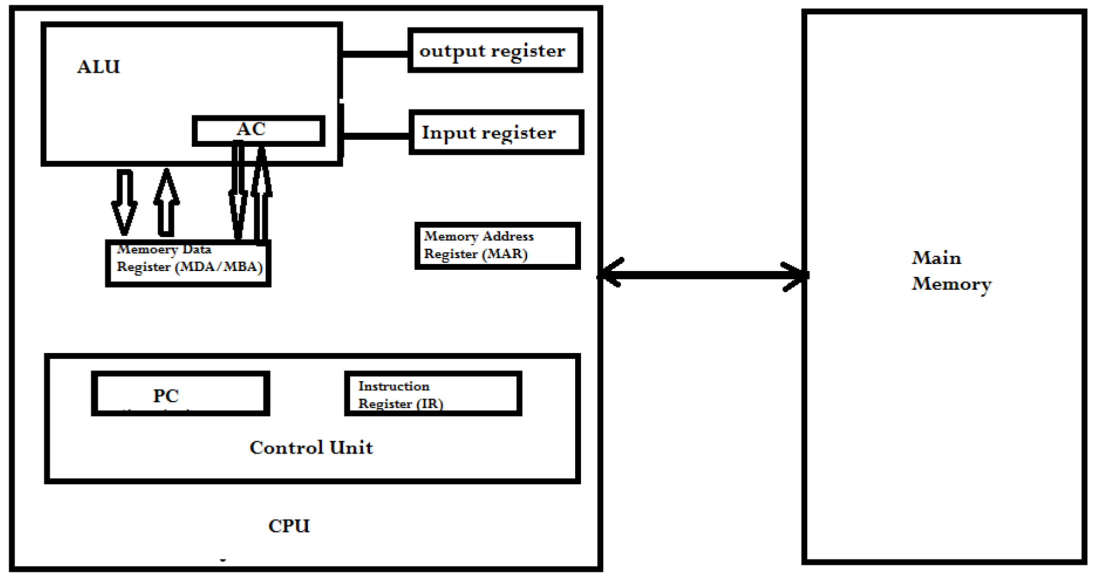
Add Mem1   (Add 100)

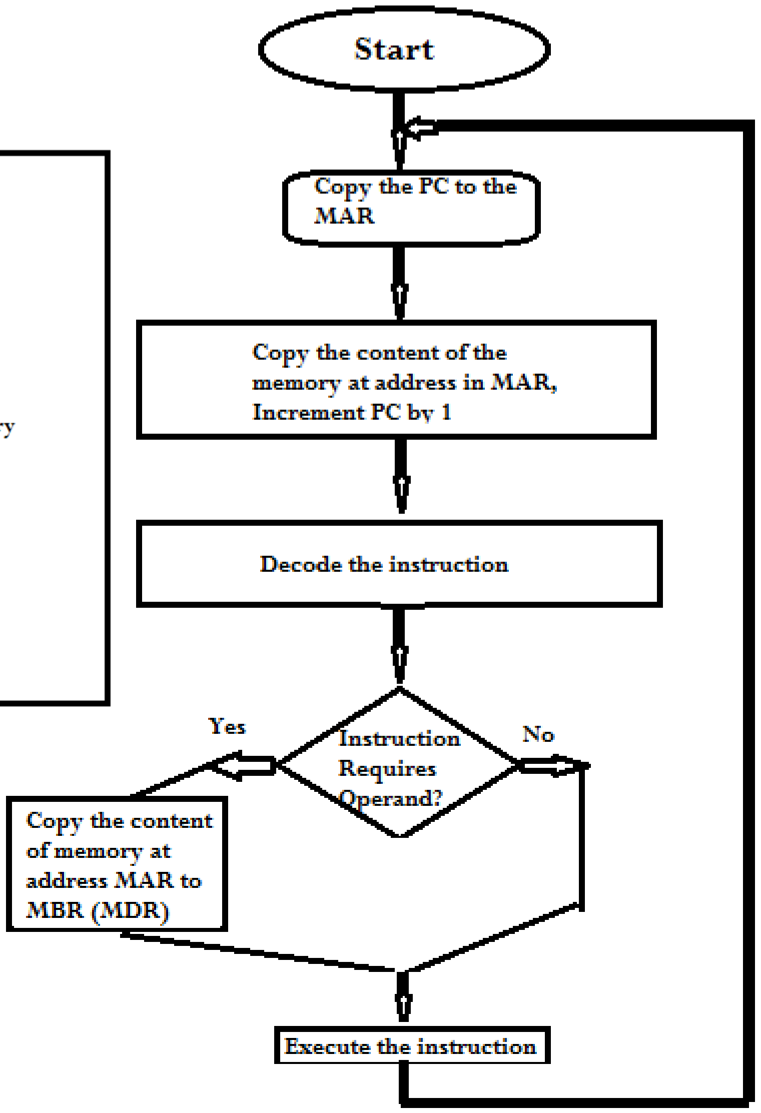
AC<= AC +(100)

- The next page flowchart explains the behavior of this CPU


ALU
AC
output register
Input register
Memoery Data Register (MDA/MBA)
Memory Address Register (MAR)
PC
Instruction Register (IR)
Control Unit
CPU
Main Memory

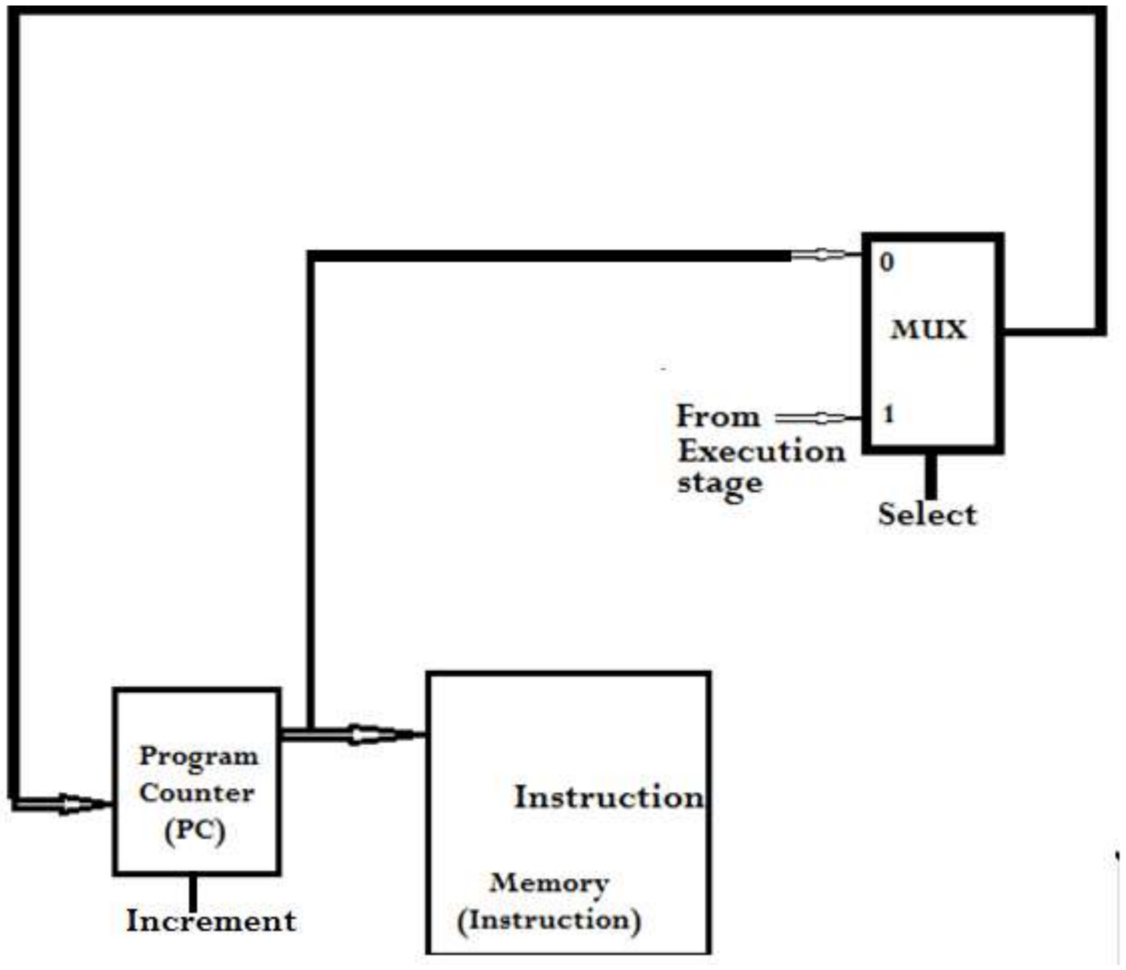
Register-Memory Architecture (Unified Memory / Von Neumann Architecture) continued


ALU
output register
AC
Input register
Memoery Data Register (MDA/MBA)
Memory Address Register (MAR)
PC
Instruction Register (IR)
Control Unit
CPU
Main Memory


Start
Copy the PC to the MAR
Copy the content of the memory at address in MAR, Increment PC by 1
Decode the instruction
Instruction Requires Operand?
Yes
No
Copy the content of memory at address MAR to MBR (MDR)
Execute the instruction

Simple Fetch Architecture example

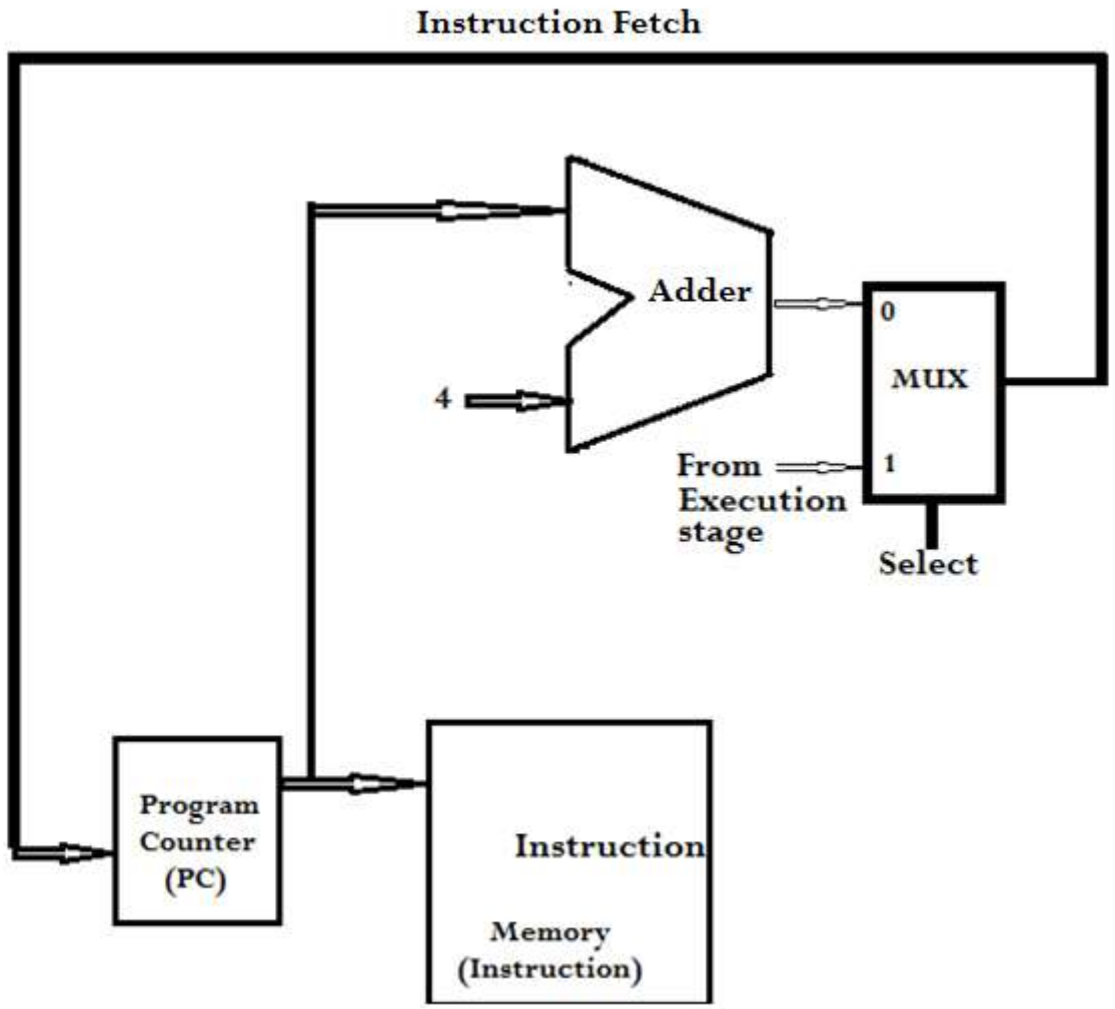
- Fetch: In the fetch cycle, the CPU retrieves the instruction from memory. The instruction is typically stored at the address specified by the program counter (PC). The PC is then incremented to point to the next instruction in memory.



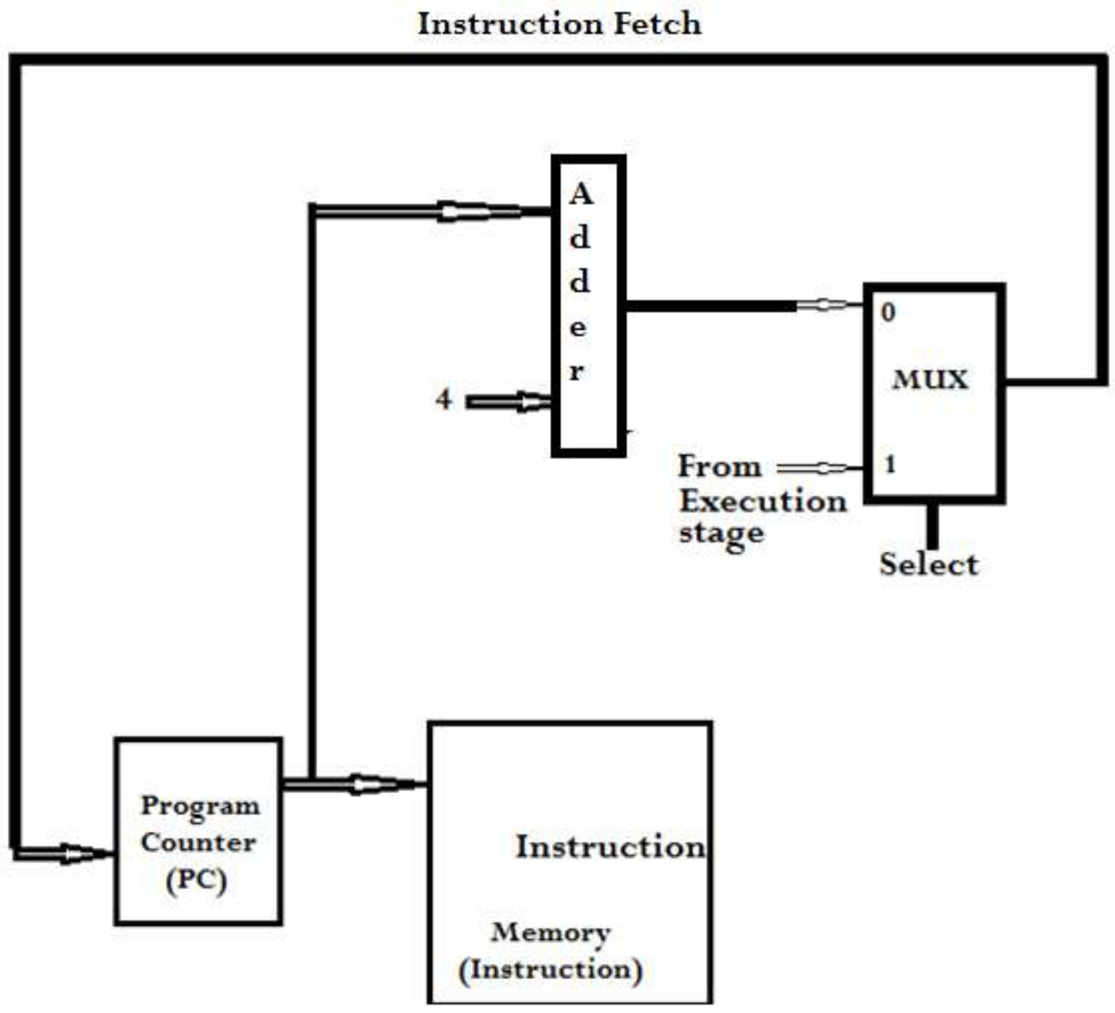
Instruction Fetch (each word is 4 bytes)

Instruction Fetch:

- An Instruction pointed by Program Counter is loaded from the main memory into the CPU. The PC is incremented 4 times, because the word size is 4 bytes.

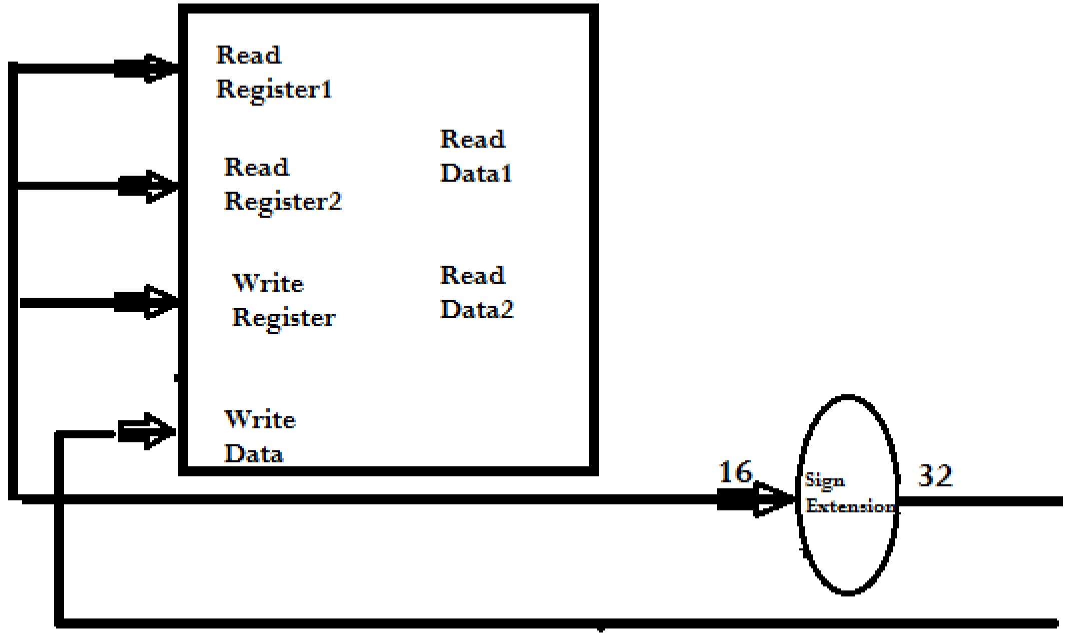


Using “Adder” instead of ALU to add 4 to the content of the program counter



Instruction Decode/ Register File Read

- When decoding the instruction, first we must find out what is the instruction, then we must access the register file to read the required registers. A more profound explanation of all the stages and overall function of a CPU will be discussed later.


Read Register1
Read Register2
Write Register
Write Data
Read Data1
Read Data2
16
Sign Extension
32