3.2 Von Neumann architecture

Institute for Advanced Studies (IAS) with Stored Program concept.

Input and output equipment operated by control unit

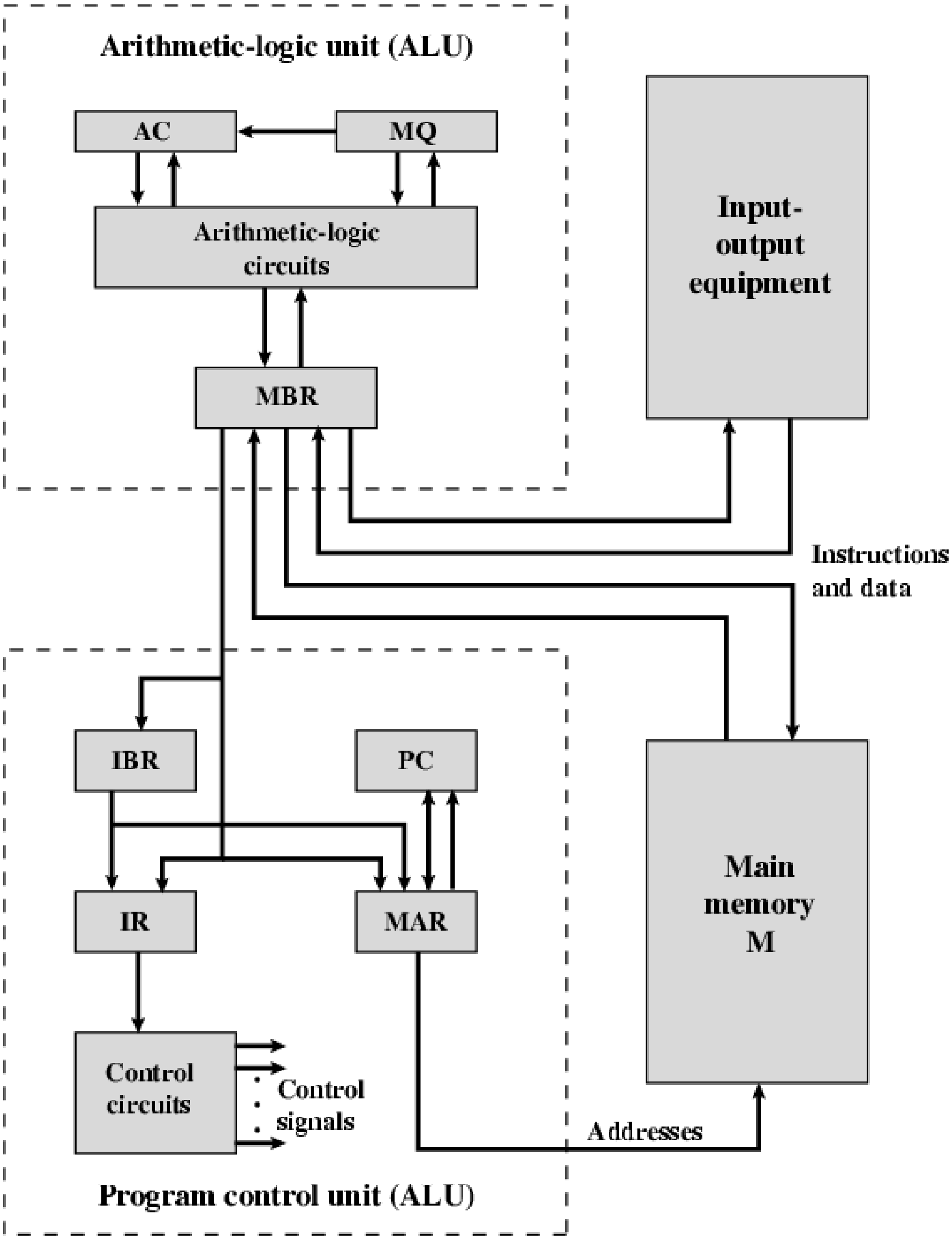
It has1000 x 40 bit words

2 x 20 bit instructions

Set of registers (storage in CPU)

- — Accumulator (**AC**)
- — Multiplier Quotient (**MQ**)
- — Memory Buffer Register (**MBR**)
- — Memory Address Register (**MAR**)
- — Instruction Buffer Register (**IBR**)
- — Instruction Register (**IR**)
- — Program Counter (**PC**)

اصحى صحصح فووق
جمع
ضرب
٢ ميموري (بفر و أدرس)
٢ إنستركشن(بفر و بس)
واحد بروجرام



3.2.1 Control unit

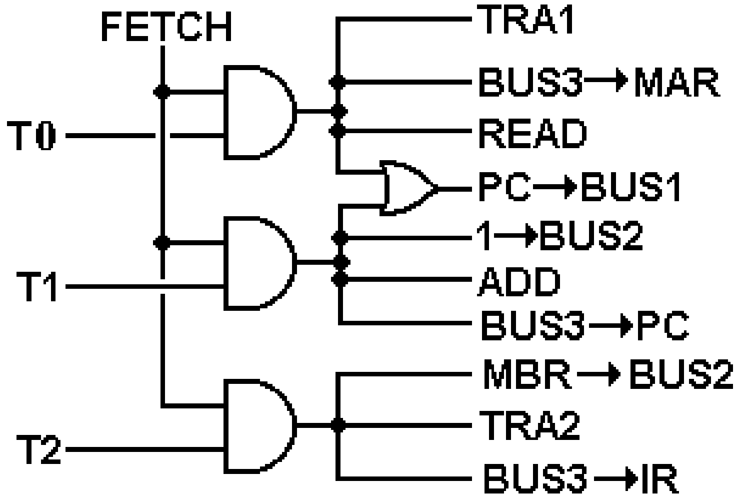
We now examine very briefly the two most common methods for building a control unit. Recall that the only function of the control unit is to emit control signals, so that the design of a control unit is just an investigation of how to generate control signals. There are two major classes of control units: **hardwired** and **microprogrammed** (or **microcoded**). In order to see the difference, let's write the above control signals for the common fetch sequence in a more compact notation.

T0: PC → Bus1, TRA1, Bus3 → MAR, READ.
T1: PC → Bus1, +1 → Bus2, ADD, Bus3 → PC.
T2: MBR → Bus2, TRA2, Bus3 → IR.

Here we have used ten control signals. Remember that the ALU has two inputs, one from Bus1, one from Bus2, and outputs its results on Bus3. The control signals used are:

PC → Bus1 Copy the contents of the PC (Program Counter) onto Bus1
+1 → Bus2 Copy the contents of the constant register +1 onto Bus2.
MBR → Bus2 Copy the contents of the MBR (Memory Buffer Register) onto Bus2
TRA1 Causes the ALU to copy the contents of Bus1 onto Bus3
TRA2 Causes the ALU to copy the contents of Bus2 onto Bus3
ADD Causes the ALU to add the contents of Bus1 and Bus2, placing the sum onto Bus3.
READ Causes the memory to be read and place the results in the MBR
Bus3 → MAR Copy the contents of Bus3 to the MAR (Memory Address Register)
Bus3 → PC Copy the contents of Bus3 to the PC (Program Counter)
Bus3 → IR Copy the contents of Bus3 to the IR (Instruction Register)

All control units have a number of important inputs, including the system clock, the IR, the PSR (program status register) and other status and control signals. A **hardwired control unit** uses combinational logic to produce the output. The following shows how the above signals would be generated by a hardwired control unit.



Here we assume that we have the discrete signal FETCH, which is asserted during the fetch phase of the instruction processing, and discrete time signals T0, T1, and T2, which would be generated by a counter within the control unit.

Note here that we already have a naming problem: there will be a distinct phase of the Fetch/Execute cycle called “FETCH”. During that cycle, the discrete signal FETCH will be active. This discrete signal is best viewed as a Boolean value, having only two values: Logic 1 (+5 volts) and Logic 0 (0 volts).
We next consider how a microprogrammed unit would generate the above signals. In this discussion, we shall present a simplified picture of such a control with a number of design options assumed; these will be explained later in the text.

The central part of a microprogrammed control unit is the **micro-memory**, which is used to store the **microprogram** (or **microcode**). The microprogram essentially interprets the machine language instructions in that it causes the correct control signals to be emitted in the correct sequence. The microprogram, written in microcode, is stored in a read-only memory (ROM, PROM, or EPROM), which is often called the **control store**.

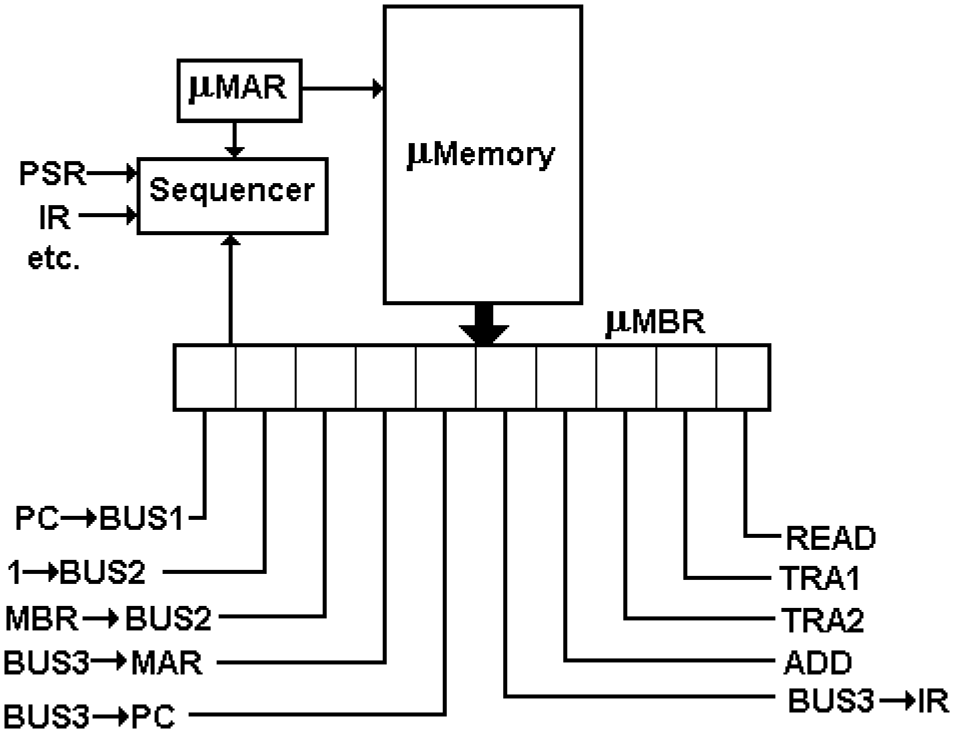
A microprogrammed control unit functions by reading a sequence of control words into a microinstruction buffer that is used to convert the binary bits in the microprogram into control signals for use by the CPU. To do this, there are several other components

the µMAR the micro-address of the next control word to read

the µMBR this holds the last control word read from the micro-memory

the sequencer this computes the next value of the address for the µMAR.

The figure below shows the structure of a sample microprogrammed control unit.

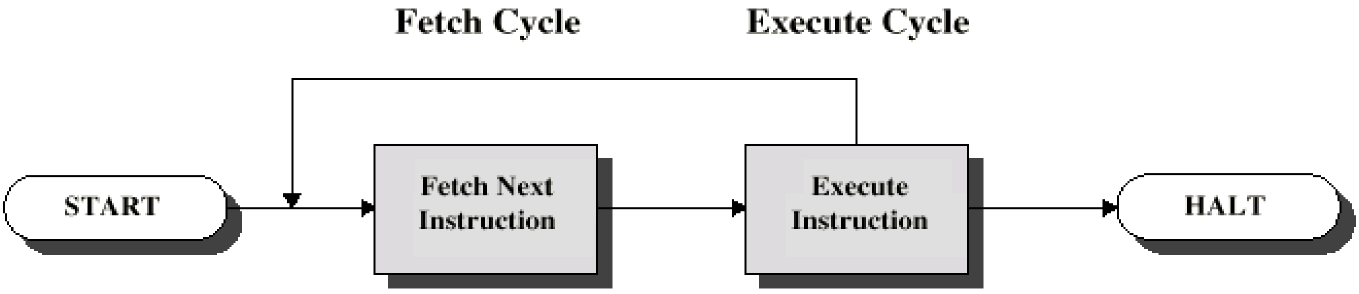


The microprogram for the three steps in fetch would be

```
10010 00011
11001 01000
00100 10100
```

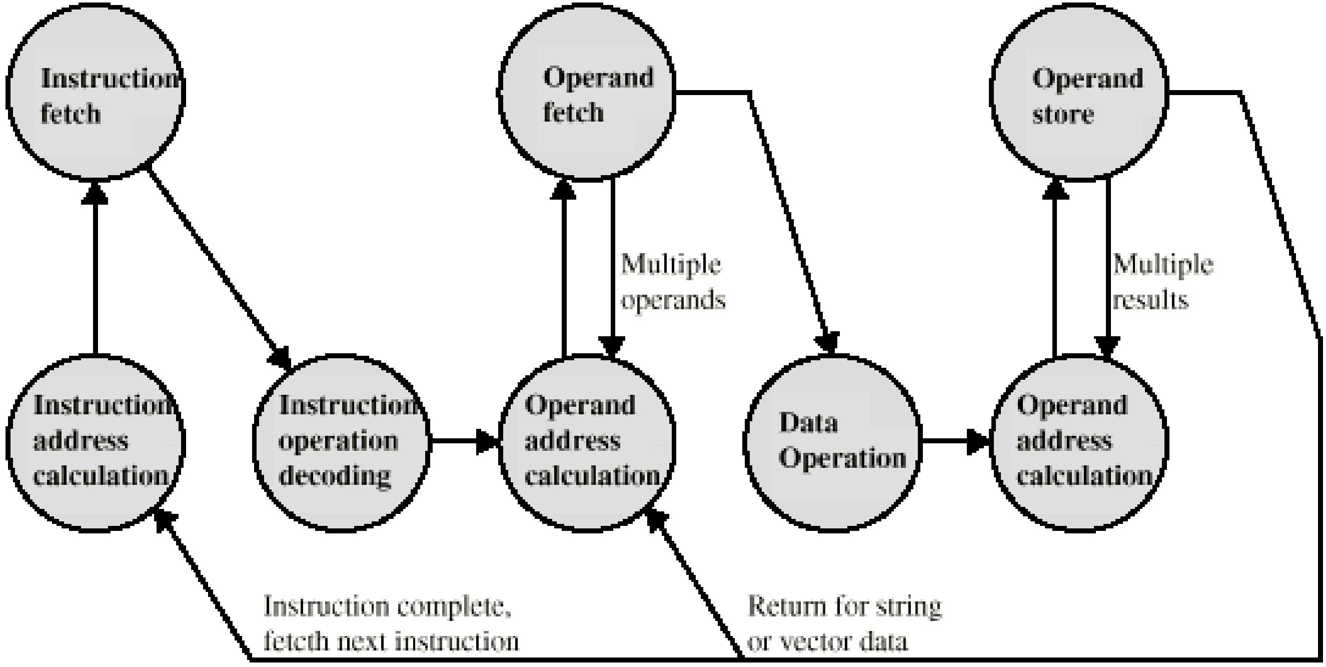
3.2.2 Instruction Cycle

Two steps: Fetch and Execute

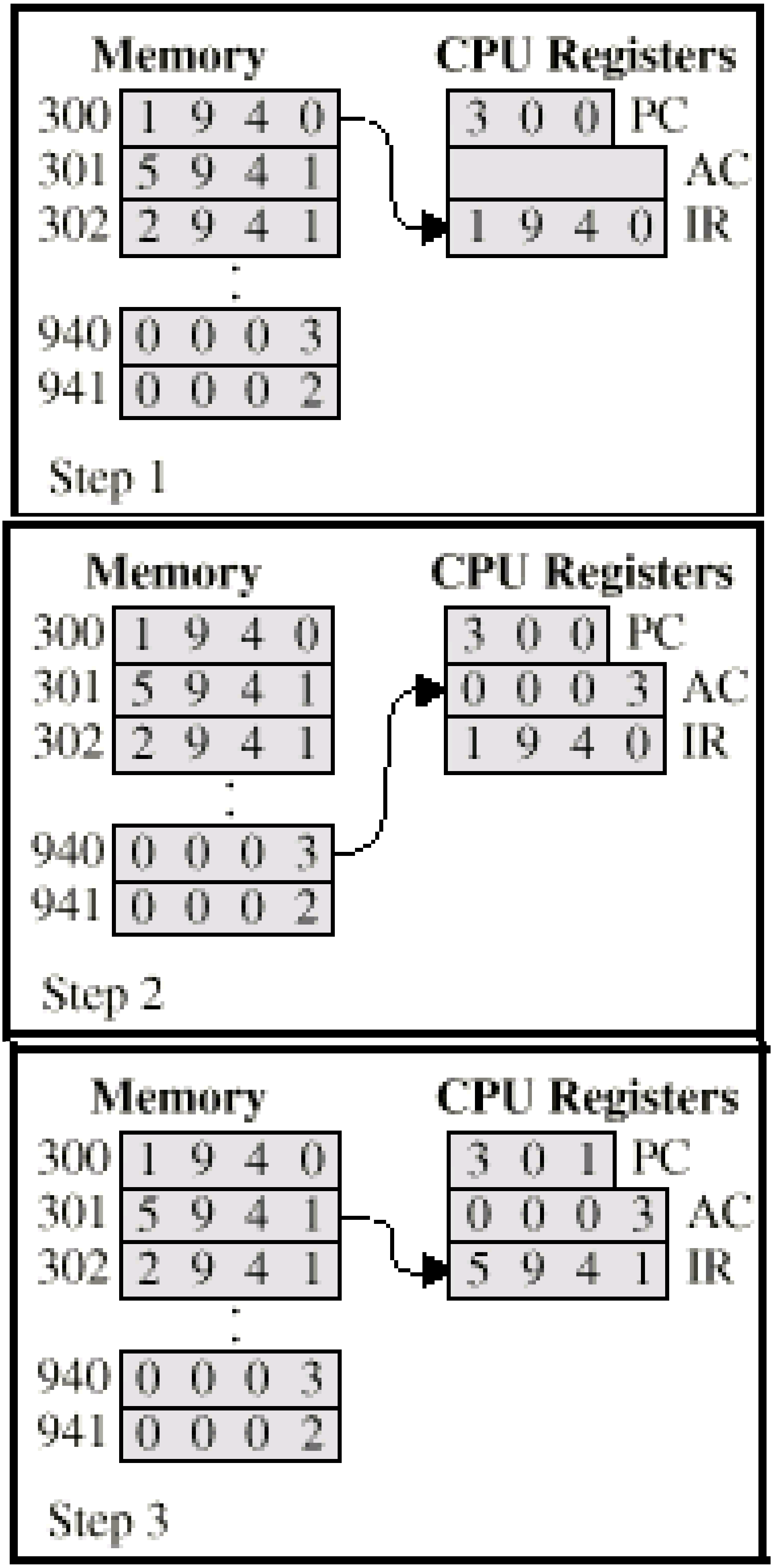


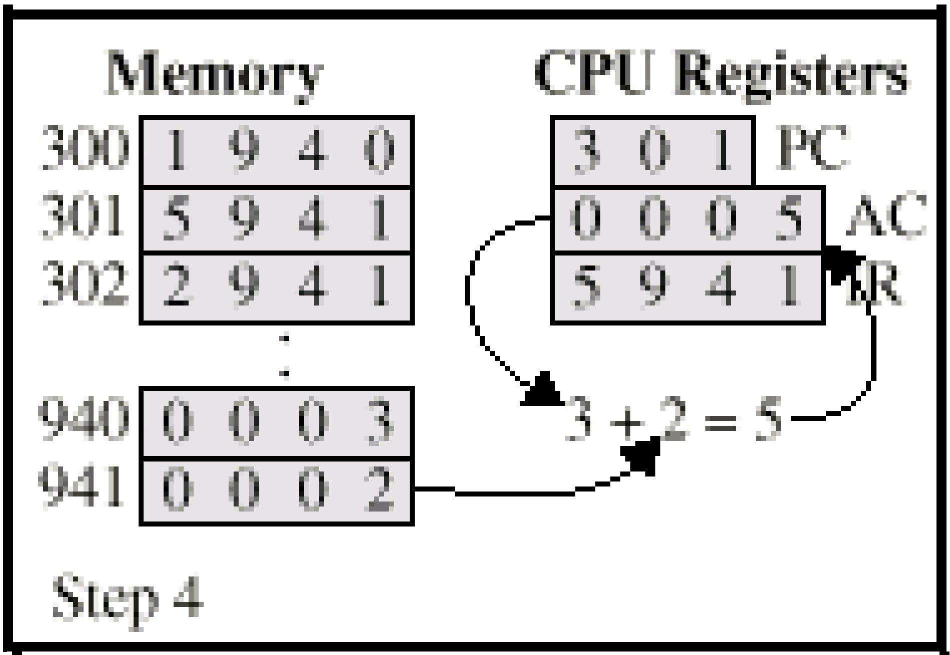
Or more precisely: Fetch, decode, and execution;

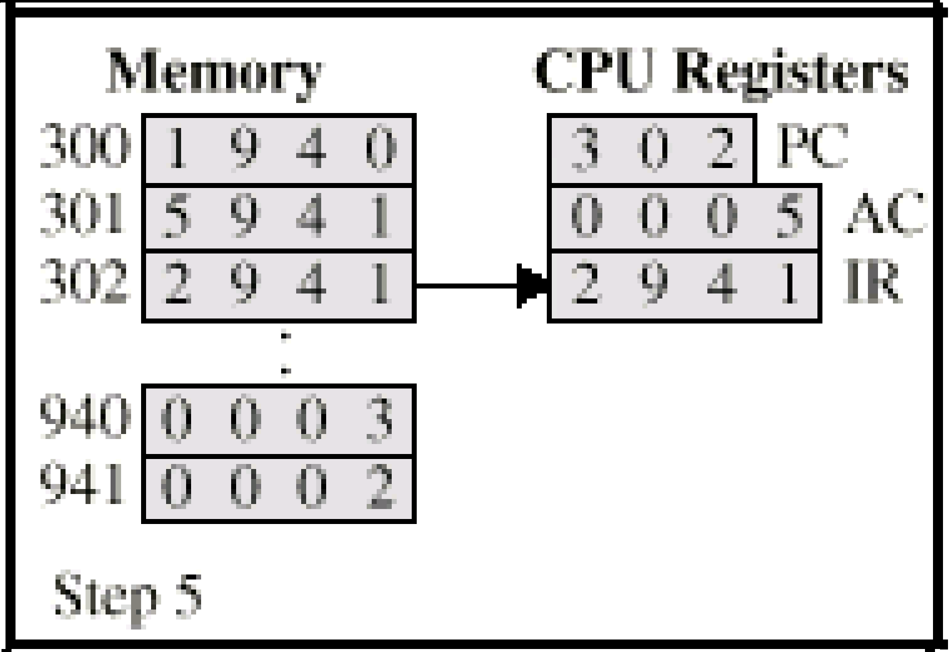
Instruction Cycle - State Diagram

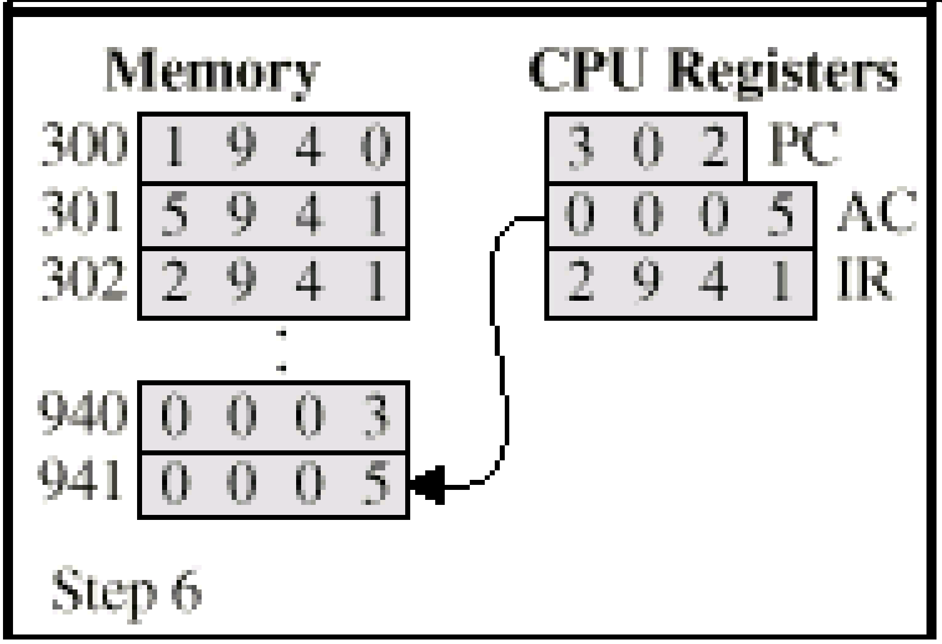

Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Instruction complete, fetcth next instruction
Return for string or vector data

Example of Program Execution (**Op-Codes**: 1=load, 2=store, 5=add, …)









References

William Stallings, Computer Organization and Architecture (9th Ed), 2012, Pearson Publisher