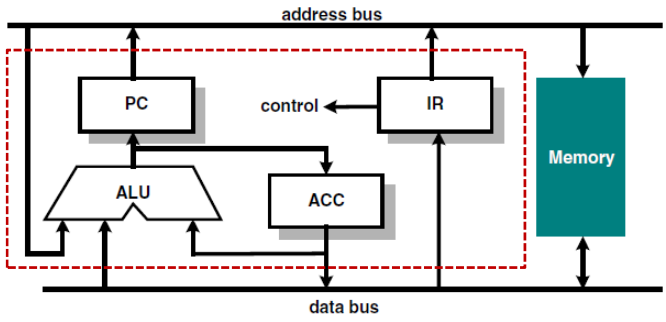
EEM332 Lecture1.

Design a simple CPU
Read/Write Steps
Machine cycle

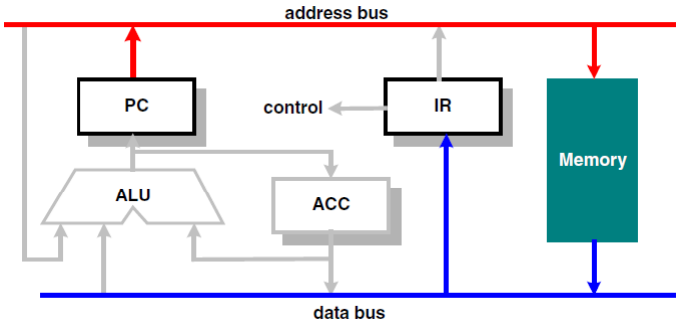
Design a simple CPU



Let us design a simple processor MU0 with 16-bit instruction and minimal hardware:-

- Program Counter (PC) - holds address of the next instruction to execute
- Accumulator (ACC) - holds data being processed
- Arithmetic Logic Unit (ALU) - performs operations on data
- Instruction Register (IR) - holds current instruction code being executed

Instruction execution step 1: Instruction Fetch

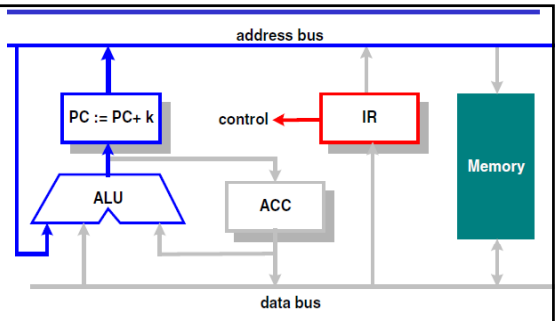


- MPU outputs value of program counter (PC) on address bus.
- Memory puts contents at the instruction address on the data bus.
- Instruction is stored in instruction registers (IR).

Step two: Instruction Decoder

The instruction word stored in IR is decoded by internal logic to provide control signals to ALU and other internal circuits inside MPU.

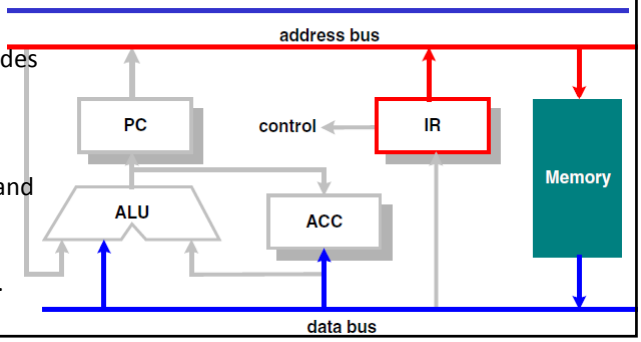
◆ Program counter value (PC) is pushed onto the address bus, the ALU increment this value by **k and put it back into the Program Counter.**



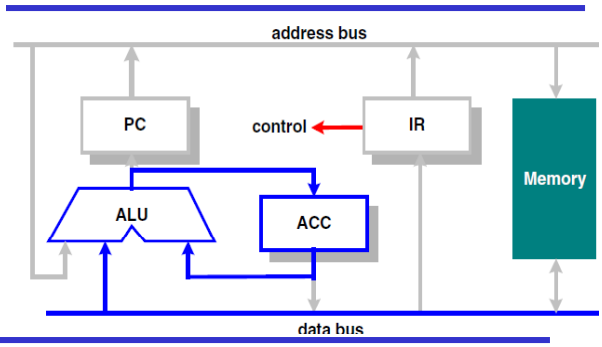
Step 3: Operand Fetch:

The instruction register provides the address of the data to be processed (i.e. **operand address).**

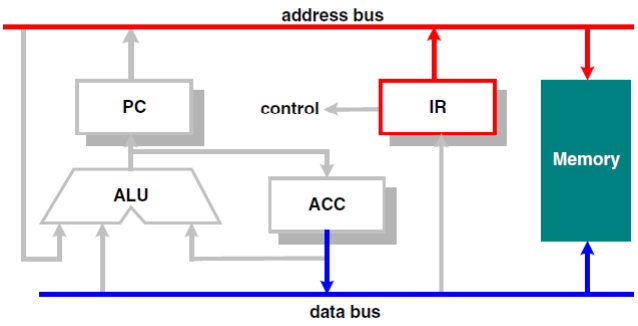
*.Memory supplies the operand data on the data bus to the MPU, ready for processing either by the ALU or the ACC.



Step 4: Execute instruction:
Processing is performed on the operand by the ALU according to the instruction.
◆ The result is put back into the Accumulator (ACC).



Step 5: Write-back
(may not exist):
The result from the Accumulator is written back into memory.



- A step-by-step analysis of CPU processes to add three numbers, with steps & code shown.
  - Assume a CPU has registers A, B, C, and D.
    - An 8-bit data bus and a 16-bit address bus.
  - The CPU can access memory addresses 0000 to FFFFH.
    - A total of 10000H locations.

Action	***Code***	***Data***
Move value 21H into register A	B0H	21H
Add value 42H to register A	04H	42H
Add value 12H to register A	04H	12H

If the program to perform the actions listed above is stored in memory locations starting at 1400H, the following would represent the contents for each memory address location…

Memory address	***Contents of memory address***
1400	(B0)code for moving a value to register A
1401	(21)value to be moved
1402	(04)code for adding a value to register A
1403	(42)value to be added
1404	(04)code for adding a value to register A
1405	(12)value to be added
1406	(F4)code for halt

The CPU puts the address 1400H on the address bus and sends it out.Memory finds the location while the CPU activates the READ signal, indicating it wants the byte at 1400H. The content (B0) is put on the data bus & brought to the CPU.

Memory address	*Contents of memory address*
1400	(B0) code for moving a value to register A
1401	(21) value to be moved
1402	(04) code for adding a value to register A
1403	(42) value to be added
1404	(04) code for adding a value to register A
1405	(12) value to be added
1406	(F4) code for halt

The CPU decodes the instruction B0 with the help of its instruction decoder dictionary. Bring the byte of the next memory location into CPU Register A.

Memory address	*Contents of memory address*
1400	(B0) code for moving a value to register A
1401	(21) value to be moved
1402	(04) code for adding a value to register A
1403	(42) value to be added
1404	(04) code for adding a value to register A
1405	(12) value to be added
1406	(F4) code for halt

From memory location 1401H, the CPU fetches code 21H directly to Register A.After completing the instruction, the program counter points to the address of the next instruction - 1402H.Address 1402H is sent out on the address bus, to fetch the next instruction

1400	(B0) code for moving a value to register A
1401	(21) value to be moved
1402	(04) code for adding a value to register A
1403	(42) value to be added
1404	(04) code for adding a value to register A
1405	(12) value to be added
1406	(F4) code for halt

From 1402H, the CPU fetches code 04H. After decoding, the CPU knows it must add the byte at the next address (1403) to the contents of register A. After it brings the value (42H) into the CPU, it provides the contents of Register A, along with this value to the ALU to perform the addition. Program counter becomes 1404, the next instruction address

1400	(B0) code for moving a value to register A
1401	(21) value to be moved
1402	(04) code for adding a value to register A
1403	(42) value to be added
1404	(04) code for adding a value to register A
1405	(12) value to be added
1406	(F4) code for halt

Address 1404H is put on the address bus and the code is fetched, decoded, and executed. Again adding a value to Register A. The program counter is updated to 1406H

Memory address	Contents of memory address
1400	(B0) code for moving a value to register A
1401	(21) value to be moved
1402	(04) code for adding a value to register A
1403	(42) value to be added
1404	(04) code for adding a value to register A
1405	(12) value to be added
1406	(F4) code for halt

The contents of address 1406 (HALT code) are fetched in and executed.The HALT instruction tells the CPU to stop incrementing the program counter and asking for the next instruction. Without HALT, the CPU would continue updating the program counter and fetching instructions.

Memory address	Contents of memory address
1400	(B0) code for moving a value to register A
1401	(21) value to be moved
1402	(04) code for adding a value to register A
1403	(42) value to be added
1404	(04) code for adding a value to register A
1405	(12) value to be added
1406	(F4) code for halt

- Microprocessors performs operations depending on instruction codes stored in memory.
- Instruction usually has two parts:
  - **Opcode - determines what is to be done**
  - **Operand - specifies where/what is the data**
- **Program Counter (PC) - address of current instruction code**
- **PC incremented automatically** each time it is used.
- The number of clock cycles taken by an instruction is the same as the number of memory access it makes.

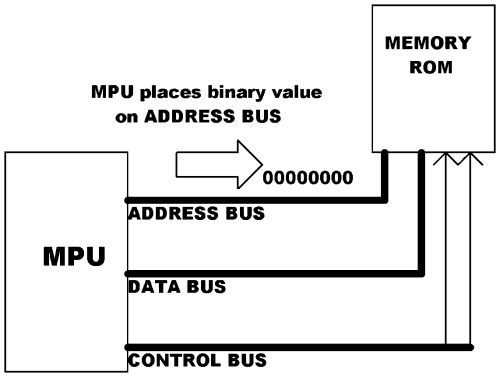
Memory contains both program and data. A peek into memory will tell you very little except a bunch of '1's and '0's.

- Program area and data area in memory are usually well separated.
- **ALU is responsible for arithmetic and logic functions.**
- There is always at least one register known as **accumulator where** the result from ALU is stored.
- There is usually one or more **general purpose register for storing** results or memory addresses.
- Fetching data from inside the CPU is much faster than from external memory.
- The processor must start from a **known state. Therefore, there is** always a reset signal to initialise the processor on power-up.

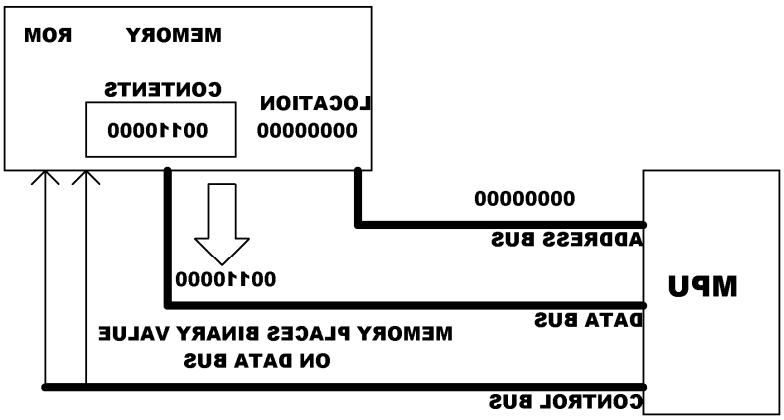
A Simplified view of MPU program execution
A MPU program is stored in memory as a sequence of binary values.
These binary values represent instructions. Each instruction has an unique binary value.
The MPU operation is controlled by reading and executing each instruction one at a time.

1. The MPU places an address on the ADDRESS BUS.

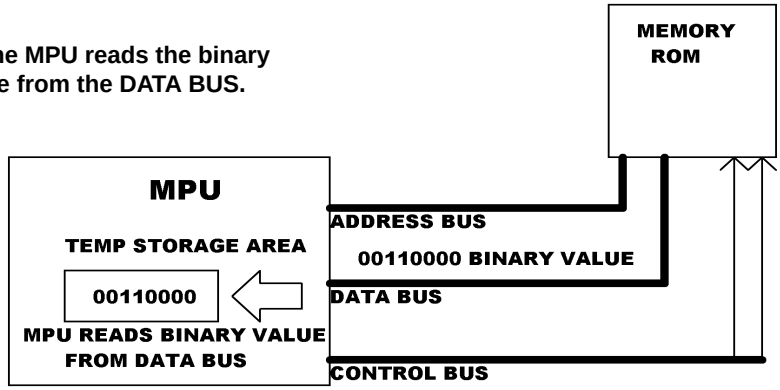
The address is a binary value which represents a unique memory location.



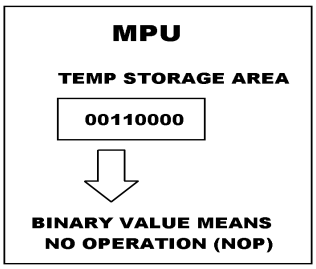
2. The Memory places the contents of the addressed location onto the DATA BUS. The contents is a binary value.



3. The MPU reads the binary value from the DATA BUS.



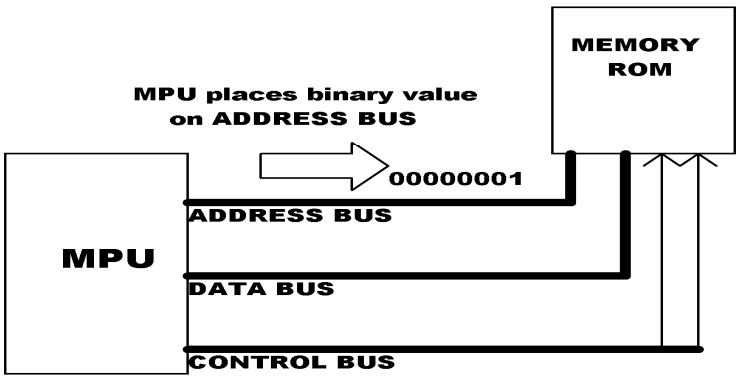
4. The MPU interprets what operation the BINARY Value represents and completes (executes) this operation

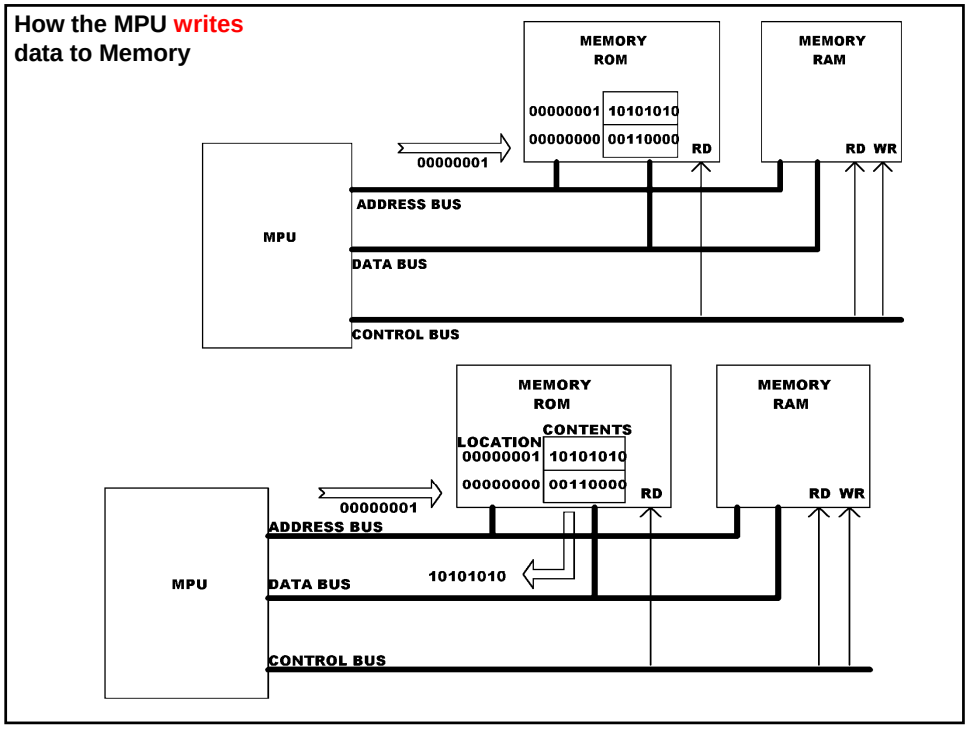


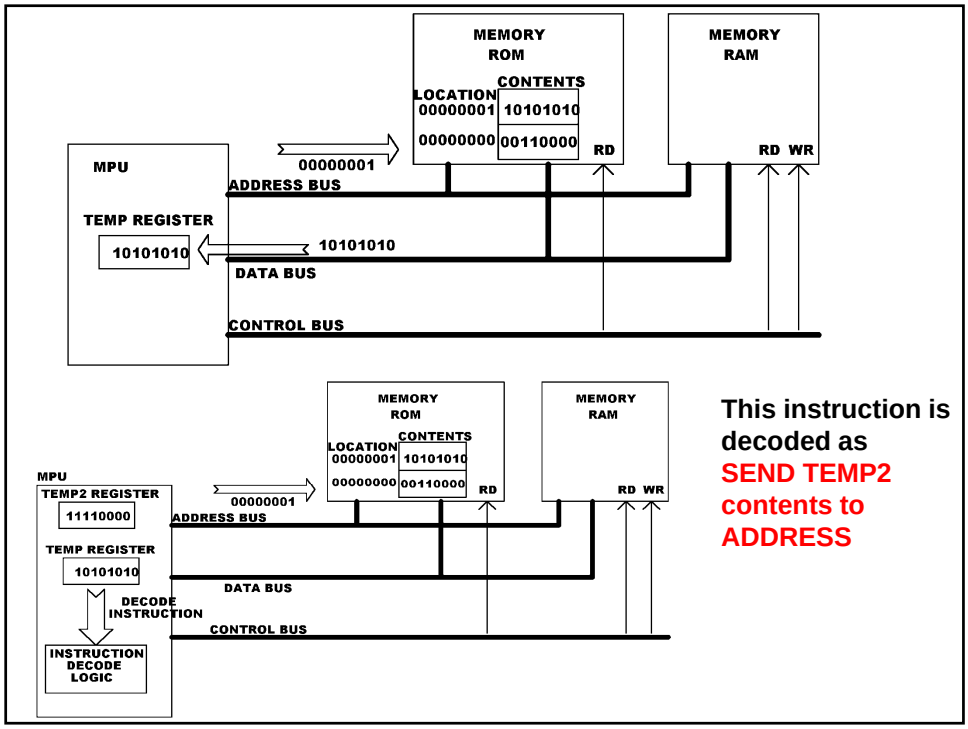
5. The MPU has completed its first instruction of the program stored as binary values in the memory.

The MPU needs to be told what to do next. The next instruction is in the next memory location. Stages 1. through 4. are repeated,

Note. At Stage 1. The MPU places a new binary value onto the ADDRESS BUS. This is the memory location of the next instruction. The binary value on the ADDRESS BUS has only changed by 1.



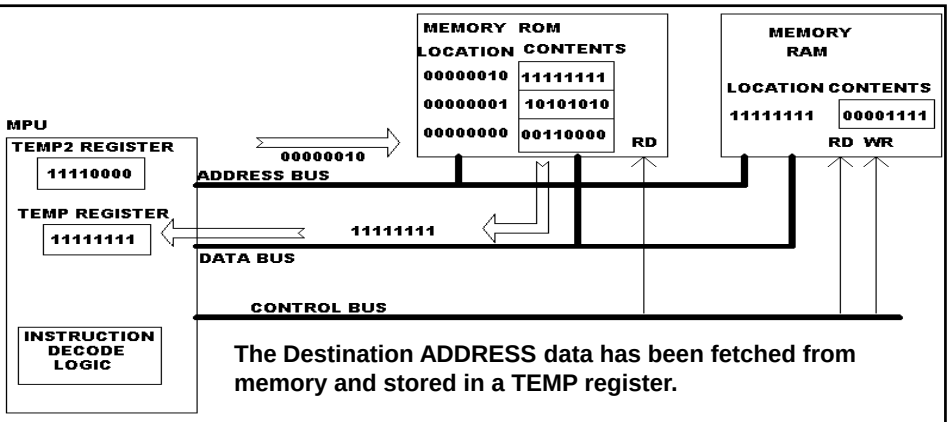

How the MPU writes
data to Memory
MEMORY
ROM
00000001 10101010
00000000 00110000
RD
MEMORY
RAM
RD WR
00000001
MPU
ADDRESS BUS
DATA BUS
CONTROL BUS
MEMORY
ROM
CONTENTS
LOCATION
00000001 10101010
00000000 00110000
RD
MEMORY
RAM
RD WR
00000001
ADDRESS BUS
MPU
DATA BUS
10101010
CONTROL BUS


MEMORY
ROM
CONTENTS
LOCATION
00000001 10101010
00000000 00110000
RD
MEMORY
RAM
RD WR
MPU
00000001
ADDRESS BUS
TEMP REGISTER
10101010
10101010
DATA BUS
CONTROL BUS
MEMORY
ROM
CONTENTS
LOCATION
00000001 10101010
00000000 00110000
RD
MEMORY
RAM
RD WR
MPU
TEMP2 REGISTER
11110000
00000001
ADDRESS BUS
TEMP REGISTER
10101010
DATA BUS
DECODE
INSTRUCTION
CONTROL BUS
INSTRUCTION
DECODE
LOGIC
This instruction is
decoded as
SEND TEMP2
contents to
ADDRESS

The MPU instruction decode logic now need to execute this instruction
SEND TEMP2 to an ADDRESS

TO WHAT ADDRESS …

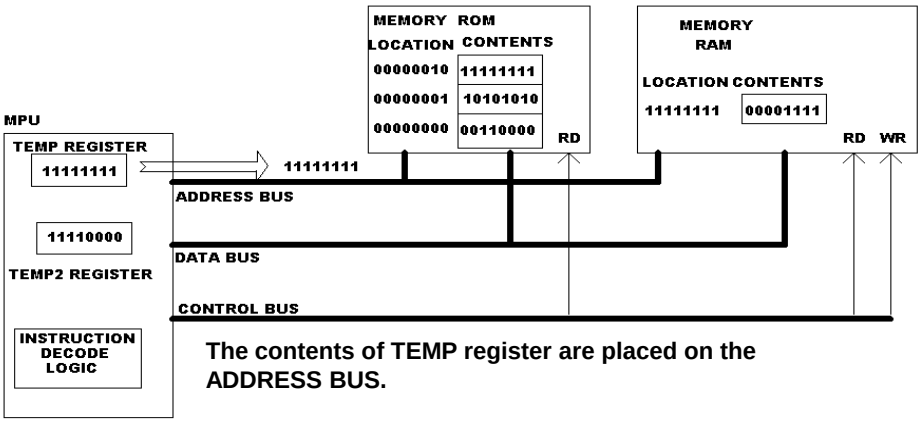
This particular MPU instruction decodes as saying that the DESTINATION ADDRESS will be in the next memory location. The MPU then fetches the binary value from the next location, as this will be the ADDRESS where TEMP2 contents will be SENT.



The TEMP register holds the DESTINATION ADDRESS where we wish the SEND our DATA (Binary Value) which is stored in TEMP2 register.

We do not try to decode the contents of TEMP because it is not an instruction, it is data associated with a specific operation, it is known as an OPERAND

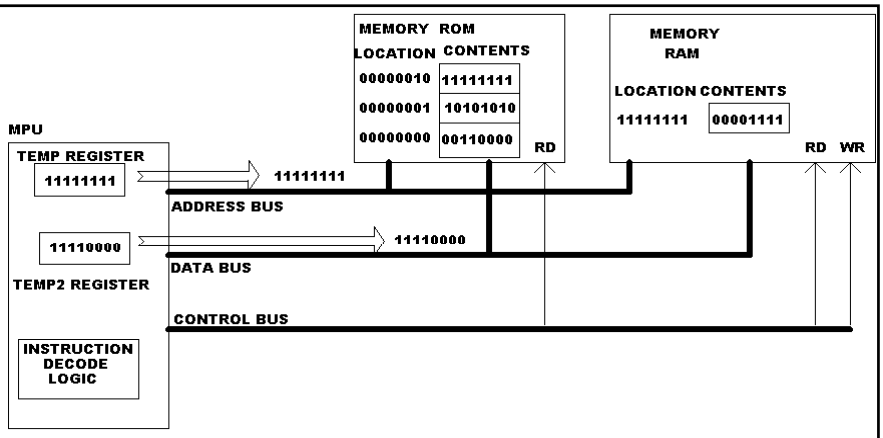
WHAT HAPPENS NEXT .. Remember we want TEMP2 contents to be sent to the MEMORY ADDRESS which is currently held in TEMP.



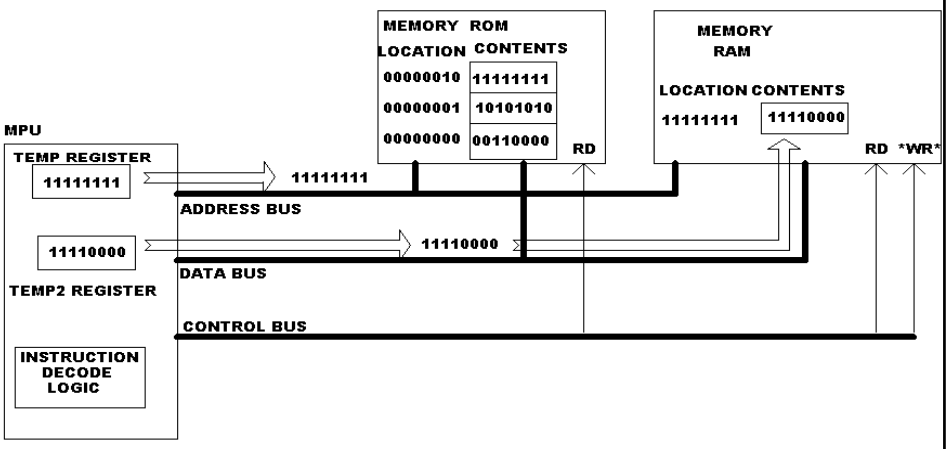
The contents of TEMP register are placed on the ADDRESS BUS.

The MEMORY WRITE part of our instruction has started. In this phase, since we are sending DATA to MEMORY or WRITING TO MEMORY we must be accessing a MEMORY store which allows us to do so.

ROM is read only memory, where our program is stored, we cannot write to it. The RAM random access memory ,(read/write memory) is where our DATA will be written. Notice the control signal lines on each of the memory types.



The ADDRESS is a memory location within the RAM, the TEMP2 register places its contents onto the DATA BUS.



The RAM reads the DATA BUS value and stores it at the location which the ADDRESS value is accessing. NOTICE THE WR signal is ACTIVE

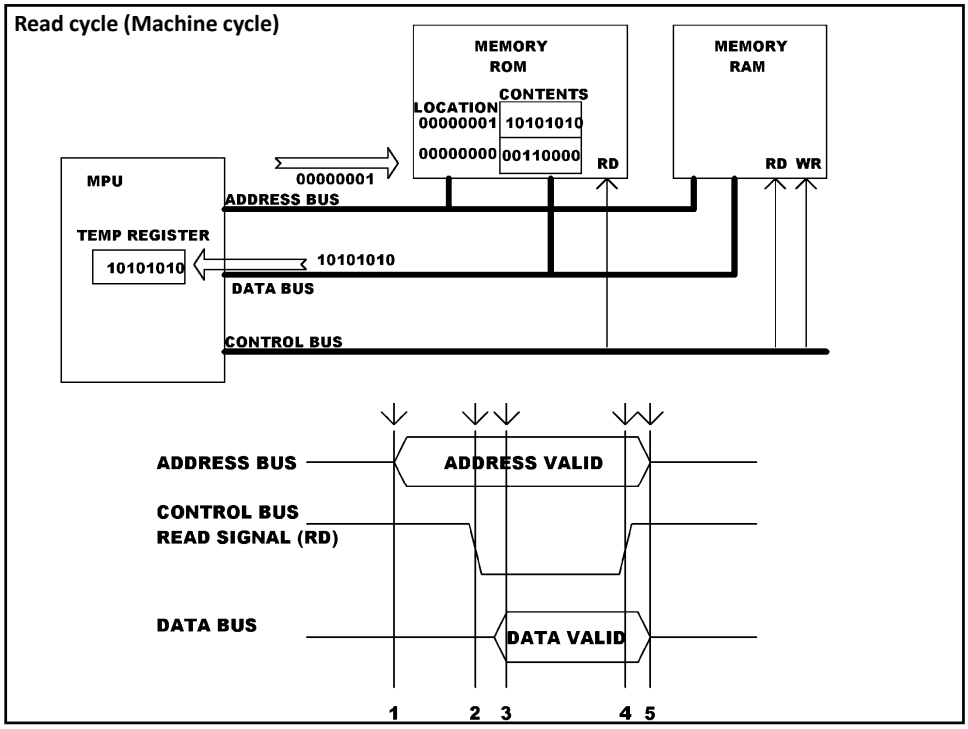
The instruction has now completed executing
The result of this operation is that the contents of TEMP2 register have been WRITTEN to a RAM Memory location 11111111.

Prior to the DATA WRITE the contents of RAM location was 00001111. After the MEMORY WRITE RAM Memory location now contains 11110000 the previous DATA value has been overwritten.

A Quick Review :

The MPU fetches instructions one at a time from memory
The instruction is interpreted by MPU and then this operation is executed. This called the FETCH DECODE EXECUTE CYCLE. The program, sequence of instructions are held in consecutive memory locations.

We have seen that not every binary value read from memory is an instruction. Some of the values can be DATA associated with an instruction.



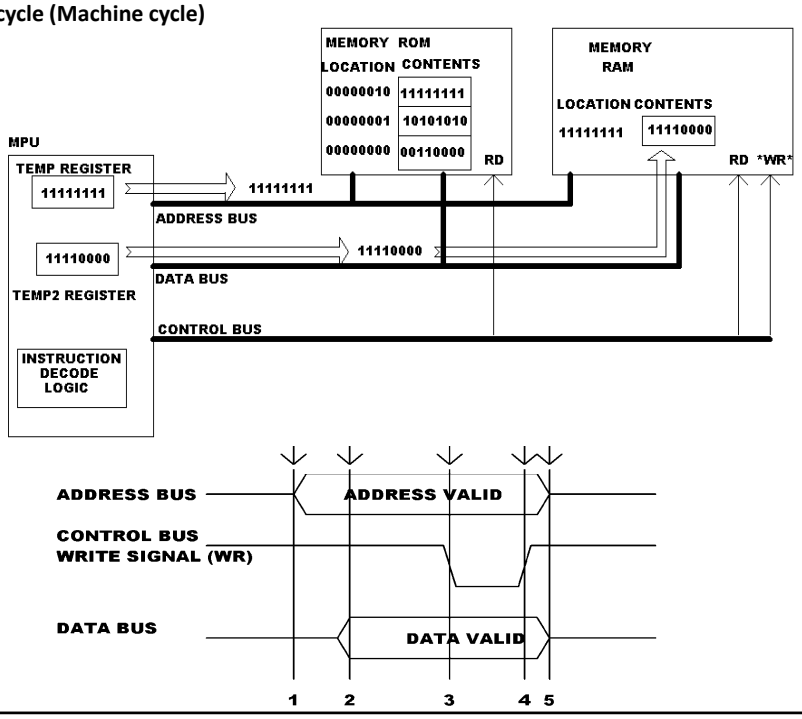
Simple Read Cycle

1. Address onto Address Bus
2. READ (RD) Control Signal Asserted by MPU
3. Data Placed from Memory onto the Data Bus
4. RD signal De-Asserted, Data into MPU temp register.
5. Read Cycle Ends

Write Cycle

1. Address placed on ADDRESS BUS
2. Data placed onto DATA BUS
3. Write (WR) Control Signal Asserted by MPU
4. Write (WR) Control Signal De-Asserted and Data Written into Memory
5. Write Cycles completes.

Write cycle (Machine cycle)



Pipeline Timing Diagram

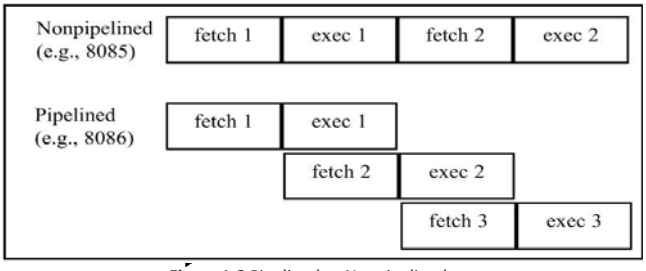
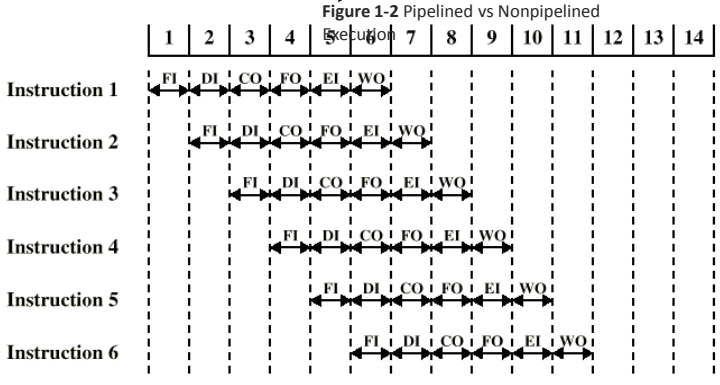


Figure 1-2 Pipelined vs Nonpipelined Execution



Abstract
•CPU clock frequency determines the speed of the microcomputer.
•The number of data and address pins on the microprocessor chip make up the microcomputer's word size and maximum memory size. The microcomputer's I/O and interfacing capabilities are determined by the control pins on the microprocessor chip.
•The microprocessor is a single chip which is capable of processing data and controlling all of the components which make up the microcomputer system. It contains all of the circuits needed to create the 'brain of the microcomputer'.
•1. Temporary storage locations in the form of a number
•of registers which can hold binary values (information),
•representing program instruction or data.
•2. The Arithmetic Logic unit (ALU). This part of the MPU
•performs both arithmetic and logical operations
•3. Timing and Control Circuits : that keep all of the other
•parts of system (memory & I/O) working together in the right time sequence.
•A Bus is a common communications pathway used to carry information between the various elements of a computer system The term BUS refers
•to a group of wires or conduction tracks on a printed circuit board (PCB)
•though which binary information is transferred from one part of the microcomputer to another

Data Bus:
The Data Bus carries the data which is transferred throughout the system. (bi-directional)
Address Bus:
An address is a binary number that identifies a specific memory storage location or I/O port involved in a data transfer

The Address Bus is used to transmit the address of the location to the memory or the I/O port.
The Address Bus is unidirectional (one way): addresses are always issued by the MPU.

The Control Bus: is another group of signals whose functions are to provide synchronization (timing control) between the MPU and the other system components. Control signals are unidirectional, and are mainly outputs from the MPU. Example Control signals

- ***Registers*** - to store information temporarily. 8, 16, 32, bit, depending on CPU.
  - ***Program counter*** - to point to the address of the next instruction to be executed.
    - In the IBM PC, a register called IP or *instruction pointer.*
  - ***Instruction decoder*** - to interpret the instruction fetched into the CPU.

Bus Sizes: Buses are specified in width, number of bits.
The width of the external Data bus generally gives a very good indication of the internal MPU architecture, particularly the registers size and the ALU.
The Data Bus width limits the bandwidth of the data that can be transferred per bus operation.
Address Bus :
The MPU Address bus size specifies how much memory and I/O locations may accessed.
Number of locations = 2^N where N is number of Address lines The Address Bus may be specified using the following 16 Bits Wide, (A15 to A0), 16 Address Lines : **In this instance $2^N = 2^{16} = 65536$ locations**

Microprocessors are classified by

• The semiconductor technology of their design (TTL, transistor-transistor logic; CMOS, complementary-metal-oxide semiconductor; or ECL, emitter-coupled logic),

•The width of the data format (4-bit, 8-bit, 16-bit, 32-bit, or 64-bit) they process; and by their

•instruction set (CISC, complex-instruction-set computer, or RISC, reduced-instruction-set computer; see RISC processor). Von Neumann or Harvard (Memory usage)

What is Microprocessor ?

It is a program controlled semiconductor device (IC}, which fetches, decode and executes instructions.

What is the function of microprocessor in a system?

The microprocessor is the master in the system, which controls all the activity of the system. It issues address and control signals and fetches the instruction and data from memory. Then it executes the instruction to take appropriate action.

What are the basic units of a microprocessor ?

The basic units or blocks of a microprocessor are ALU, an array of registers and control unit.

what is Software and Hardware?

The *Software* is a set of instructions or commands needed for performing a specific task by a programmable device or a computing machine.

Without software the Hardware is an idle machine.

What is assembly language?

The language in which the mnemonics (short -hand form of instructions) are used to write a program is called assembly language. The manufacturers of microprocessor give the mnemonics.

What are machine language and assembly language programs?

The software developed using 1's and *0's* are called machine language, programs. The software developed using mnemonics are called assembly language programs.

What is the drawback in machine language and assembly language, programs?

The machine language and assembly language programs are machine dependent. The programs developed using these languages for a particular machine cannot be directly run on another machine .

Assembly language is referred to as a *low-level* language because it deals directly with the internal structure of the CPU.

Define opcode and operand.

Opcode (Operation code) is the part of an instruction / directive that identifies a specific operation. Operand is a part of an instruction / directive that represents a value on which the instruction acts.

What is pipelined architecture?

In pipelined architecture the processor will have number of functional units and the execution time of functional units are overlapped. Each functional unit works independently most of the time.

What is fetch and execute cycle?

In general, the instruction cycle of an instruction can be divided into fetch and execute cycles. The fetch cycle is executed to fetch the opcode from memory. The execute cycle is executed to decode the instruction and to perform the work instructed by the instruction.

Assembly language is referred to as a *low-level* language because it deals directly with the internal structure of the CPU.

Define machine cycle.

Machine cycle is defined as the time required to complete one operation of accessing memory, I/O, or acknowledging an external request. This cycle may consist of three to six T-states.

What is processor cycle (Machine cycle)?

The processor cycle or machine cycle is the basic operation performed by the processor. To execute an instruction, the processor will run one or more machine cycles in a particular order.

What is Instruction cycle?

The sequence of operations that a processor has to carry out while executing the instruction is called Instruction cycle. Each instructioncycle of a processor indium consists of a number of machine cycles.

What is opcode fetch cycle?

The opcode fetch cycle is a machine cycle executed to fetch the opcode of an instruction stored in memory. Every instruction starts with opcode fetch machine cycle.

Why interfacing is needed for 1/0 devices?

Generally I/O devices are slow devices. Therefore the speed of I/O devices does not match with the speed of microprocessor. And so an interface is provided between system bus and I/O devices.

What is the need for Port?

The I/O devices are generally slow devices and their timing characteristics do not match with processor timings. Hence the I/O devices are connected to system bus through the ports.

.What is a port? **The port is a buffered I/O, which is used to hold the data transmitted from the microprocessor to I/O device or vice-versa.**

Flip Flop: A flip-flop stores one bit of information.

Register: When a set of *n flip-flops is used to store n bits of* information, such as an *n-bit number, we refer to these flip-flops as a register. A common* clock is used for each flip-flop in a register,

shift register.:A register that provides the ability to shift its contents is called a shift register.