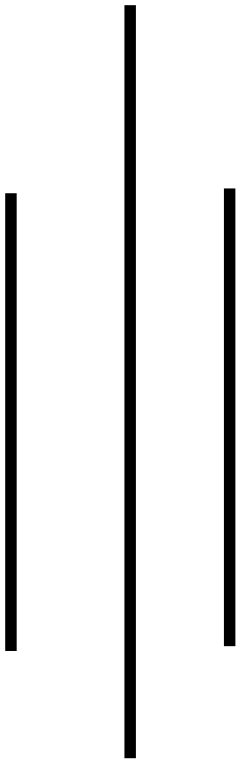
DEERWALK INSTITUTE OF TECHNOLOGY



# MULTIPROGRAMMING OPERATING SYSTEM (MOS)


**Submitted By:**

Sagar Giri

Anish Thakuri

Ujjwal Upreti

Krishna Chauhan

**Submitted To:**

Bikash Balami

(Lecturer at DWIT)

**Contents** **Page No.**

## Objective

The main objective of this project is to get acquainted with the fundamentals of the operating system and to some extent to have an idea of the computer system architecture. This project clearly falls into the category of simulator of an OS. Simulation based OS intentionally shields the bare essentials of any particular machine architecture and allows focusing on implementing operating system concepts like memory management, Input Output management and process management. The intension of this project is to focus on CPU hardware and its behavior (fetch–decode–execute) in simulation mode using C programming language.

## Introduction

- **Operating System**

  An operating system is an organized collection of system program that control the overall operation at a computer system and allocates the resources for other programs. The other programs are called applications or application programs. The application programs make use of the operating system by making requests for services through a defined application program interface (API). In addition, users can interact directly with the operating system through a user interface, such as a command language or a graphical user interface (GUI).

- **Processes**

  A process is an instance of a program in execution. Batch systems work in terms of "jobs". Many modern process concepts are still expressed in terms of jobs, (e.g. job scheduling). It usually includes (but is not restricted to):

  - Registers
  - Stack
  - Memory (global variables and dynamically allocated memory)
  - Open file tables
  - Signal management information

- **States**

  In multiprogramming OS, all jobs in main memory are in one of the following states:

  - New
    The process is in the state of being created.
  - Ready
    The process has all the resources available that it needs to run, but the CPU is not currently working on this process's instructions. i.e. the process is waiting for the CPU.
  - Running
    The CPU is working on this process's instructions. i.e. the process is in the state of using the CPU.
  - Waiting or in Blocked state
    The process cannot run at the moment, because it is waiting for some resource to become available or for some event to occur. For example the process may be waiting I/O operations.
  - Terminated
    The process has completed.

- **Project**
  - **SPECIFICAIONS OF MACHINE SEEN BY USERS**

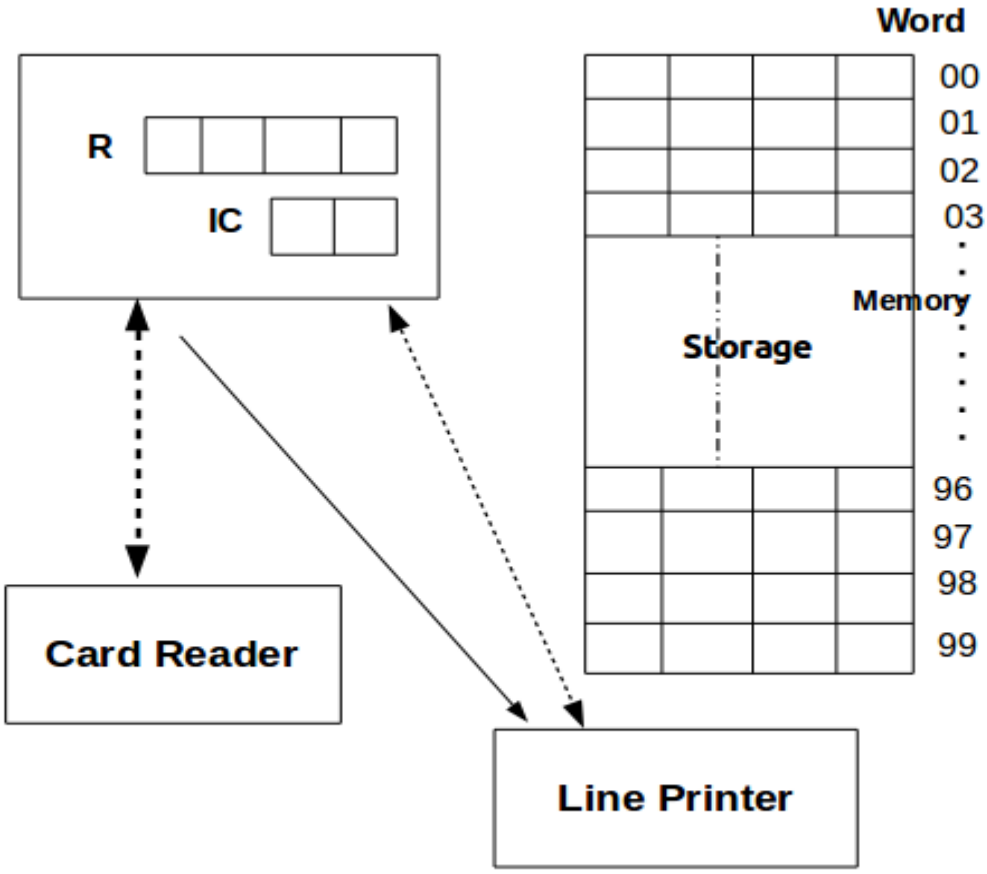



Figure :- User Virtual machine

- **Algorithm**

**BEGIN**

INITIALIZATION

SI = 3

MOS (Master Mode)

Case SI of

1. READ
2. WRITE
3. TERMINATE

End Case

READ

IR [4] ← 0

Read next (data) card from input file in memory locations IR [3, 4] through IR [3, 4] + 9

EXECUTEUSERPROGRAM

WRITE

IR [4] ← 0

Write one block (10 words of memory) of data from memory locations IR[3, 4] through IR [3, 4] + 9 to the output file

EXECUTEUSERPROGRAM

TERMINATE

Write 2 blank lines in output file

LOAD

LOAD

M ← 0

While not eof

Read next (Program, Control) card from the input file in a buffer

Control card : $AMJ, End While

$DTA STARTEXECUTION

$END, End While

Program card : if M = 100, abort (memory exceeded)

Store buffer in memory location M through M + 9

M ← M + 9

End While

STOP

STARTEXECUTION

IC ← 00

EXECUTEUSERPROGRAM

**END** (MOS)

EXECUTEUSERPROGRAM (SLAVE MODE)

Loop

IR ← M [IC]

IC ← IC + 1

Examine IR [1, 2]

LR : R ← M [IR [3, 4]]

SR : R → M [IR [3, 4]]

CR : Compare R and M [IR [3, 4]]

If equal C ← T else C ← F

BT : If C = T then IC ← IR [3, 4]

GD : SI = 1 (Input Request)

PD : SI = 2 (Output Request)

H : SI = 3 (Halt)

End Examine

End Loop

- Flowchart

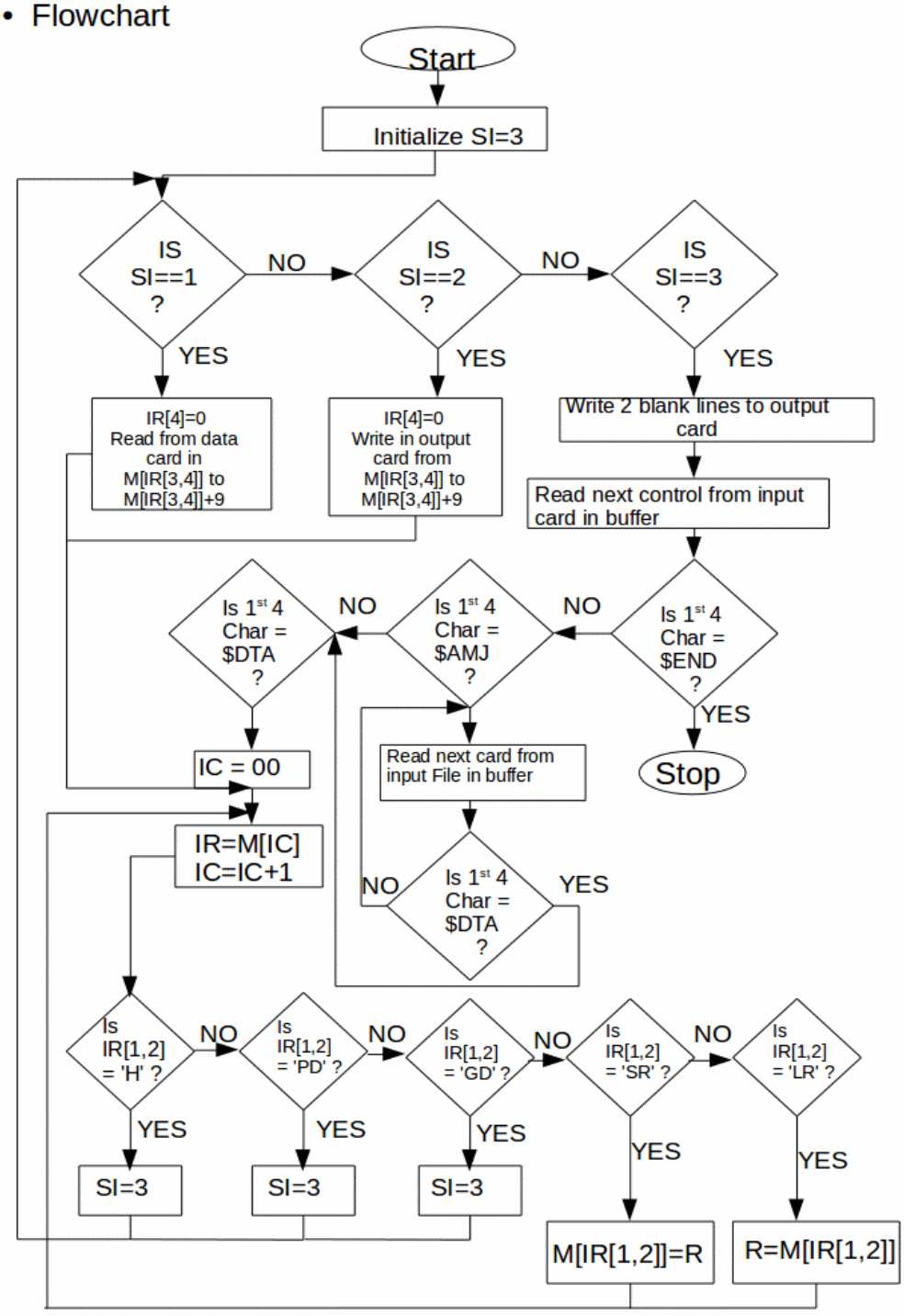



Fig. Flowchart for MOS

**Implementation of Job**

- **MOS CPU Instruction Set**

  LR = Load a virtual memory location's contents into R
  SR = Store contents of R into virtual memory location
  GD = Get data
  PD = Put data
  H = Halt user program

- **Data**

  Simulating Queue
  ENQ DEQ 1  2  3
  CALL ME: First In First Out

- **Input Program**

  $AMJ060204000009
  GD30PD30GD40LR40SR50LR42SR51PD50LR43SR51
  PD50LR44SR51PD50LR41SR60LR42SR61PD60LR43
  SR61PD60LR44SR61PD60GD60PD60H
  $DTA
  Simulating Queue
  ENQ DEQ 1  2  3
  CALL ME : First In First Out
  $END0602

- **Result in Output Card**

  Simulating Queue

  ENQ 1
  ENQ 2
  ENQ 3
  DEQ 1
  DEQ 2
  DEQ 3
  CALL ME: First In First Out

- **Conclusion**

  After the 2 months of work, the project is finally completed. Our all team member were assigned a task of the project. We made our new job and implemented in the simulated OS to obtain the desired result. Hence, Implementation of simulation of MOS (Multiprogramming operating system) was implemented using C programming language.