K J's Educational Institute

K J College of Engineering & Management Research

(Accredited by NAAC)

Sr. No. 25 & 27, Kondhwa - Saswad Road, Near Bopdev Ghat, Pune 411048.

Department Of Computer Engineering

Lab Manual

Laboratory Practice(LP-1)

Subject Code: 310248

Class: - TE (2019 PAT) Branch: - Computer

Prepared by: - Prof. Satish L Yedage

Examinations: - PR [25M] and TW [25M]

Required H/W and S/W: -Linux/windows.

Programming Tool: Java programming tool

INDEX

Assignment No. 01

Problem Statement:

Design suitable data structures and implement pass1 and pass2 of a two pass assembler for pseudo-machine. Implementation should consist of a few instructions from each category and few assembler directives. The output of pass1 (intermediate code file and symbol table) should be input for pass2

Objectives:

- Understand the concept and working of assemblers.
- Design data structures for symbol table, opcode table, and literal table.
- Implement Pass 1 to generate intermediate code and symbol table.
- Implement Pass 2 to generate final machine code using the outputs of Pass 1.

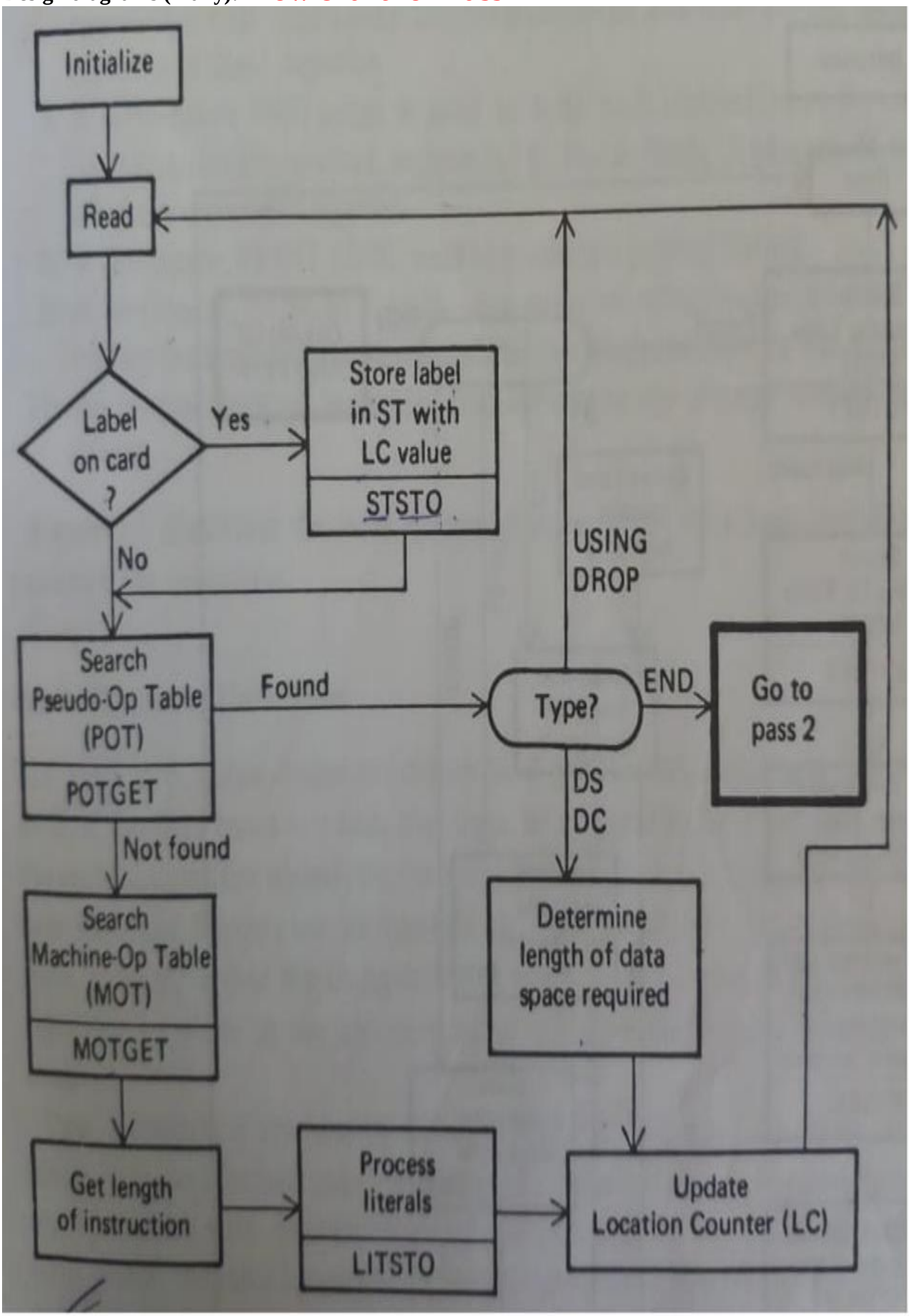
Theory:

1. Assembler is a low level translator which translates source code to machine code
2. It works in two phases : analysis phase and synthesis phase
3. In analysis phase, source assembly code is analyses to generate some intermediate data structures
4. In synthesis phase, machine code is generated
5. There are well defined system level standard algorithms to design the assembler as a two pass assembly, namely PassI and PassII algorithms of assembler
6. PassI takes the source code in assembly language as input and generates intermediate data structures.

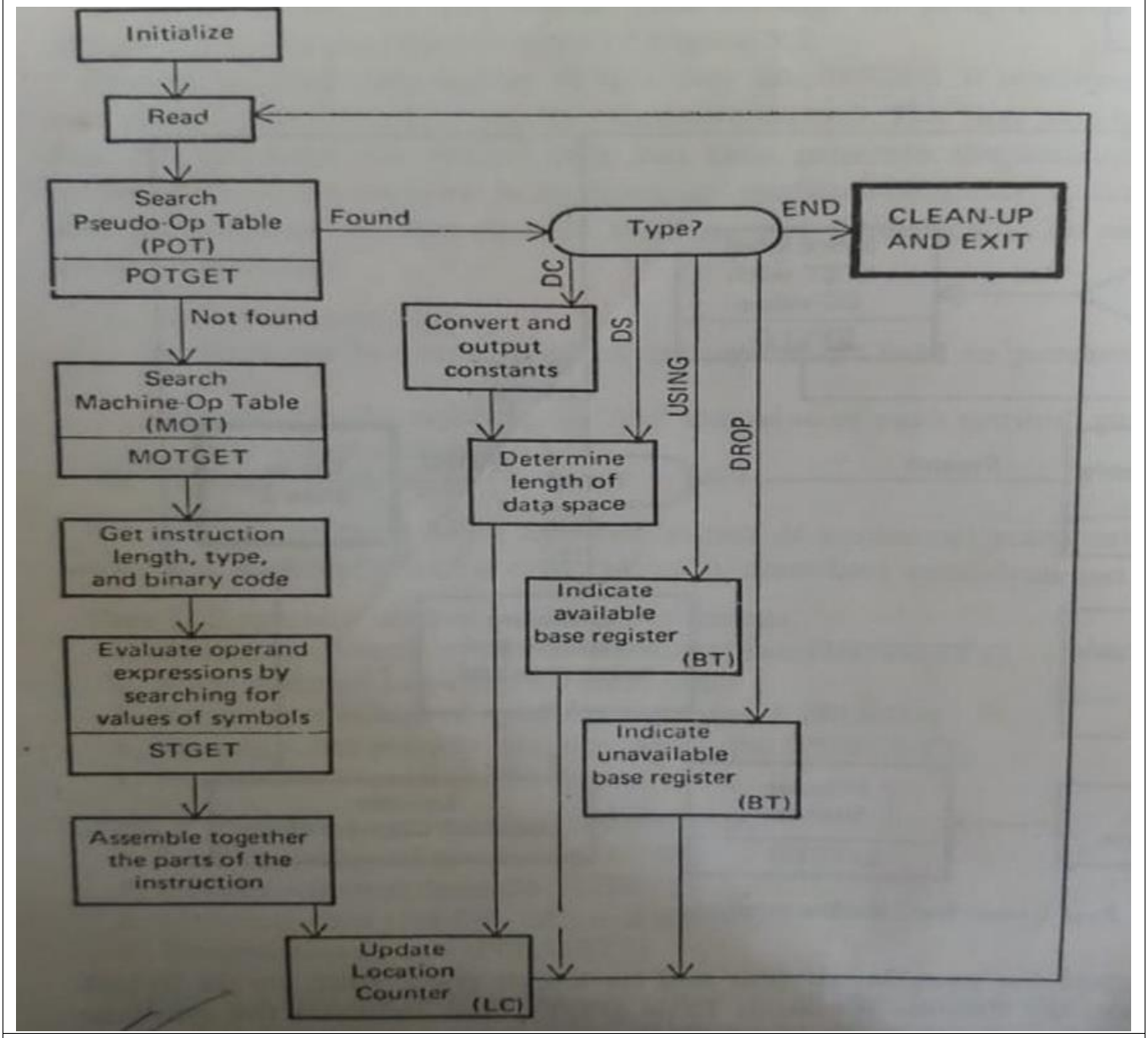
Data Structures Used:

- **Symbol Table**: Stores labels with their assigned addresses.
- **Opcode Table**: Stores instruction types and opcodes.
- **Intermediate Code**: Contains processed instructions in symbolic/mnemonic form.

Design diagrams (if any): **Flow chart for Pass-1**



Flow chart for Pass-2



Input for PassI of assembler:

1. Assembly language program in hypothetical language as per the Author Dhamdhere
2. OPTB
3. Condition code table
4. Register table

Input for PassII of assembler:

1. IC
2. SYMTAB
3. LITTAB
4. POOLTAB
5. OPTAB

Output of PassI of assembler:

1. IC
2. SYMTAB
3. LITTAB
4. POOLTAB

Output of PassII of assembler:
Machine code

Software Requirement:
Operating System recommended :- 64-bit Open source Linux or its derivative
Programming tools recommended: - Eclipse IDE /

Hardware Requirement:I3 and I5 machines

Frequently Asked Questions:

1. What is two pass assembler?
2. What is the significance of symbol table?
3. Explain the assembler directives EQU, ORIGIN.
4. Explain the assembler directives START, END, LTORG.
5. What is the use of POOLTAB and LITTAB?
6. How literals are handled in pass I?
7. What are the tasks done in Pass I?
8. How error handling is done in pass I?
9. Which variant is used in implementation? Why?
10. Which intermediate data structures are designed and implemented in PassI?
11. What is the format of a machine code generated in PassII?
12. What is forward reference? How it is resolved by assembler?
13. How error handling is done in pass II?
14. What is the difference between IS, DL and AD?
15. What are the tasks done in Pass II?

Conclusion:

1. Input assembly language program is processed by applying PassI algorithm of assembler and intermediate data structures, SYMTAB, LITTAB, POOLTAB, IC, are generated.
2. The intermediate data structures generated in PassI of assembler are given as input to the PassII of assembler, processed by applying PassII algorithm of assembler and machine code is generated

Instructions :

Handwritten write-up as follows :

- Name of Student:
- Subject:LP-1
- Assessment table
- Date of Performance:_____ Date of Completion_____
- Title
- Objectives
- Problem statement
- Software and hardware requirements
- Theory –What is Assembler?
- Types of Statement format. and Explain each Statement.
- Flowchart of Pass-1 and Pass-2
- Conclusion

Attach Program code and Its Output

Assignment No. 02

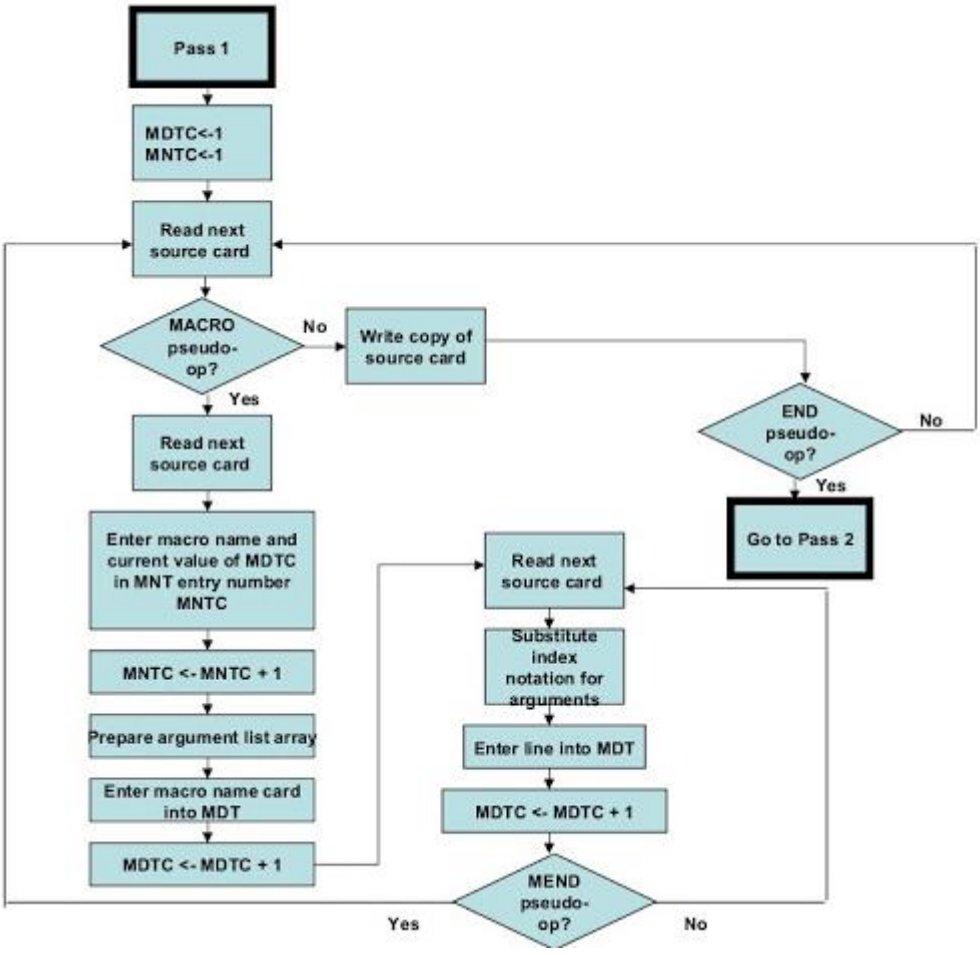
Problem Statement:

Design suitable data structures and implement Pass-I of a two pass macro processor using OOP features in Java/C++. The output of Pass-I (MNT, MDT, ALA & Intermediate code file without any macro definitions) should be input for Pass-II.

Objectives:

1. To identify and design different data structure used in macro-processor implementation
2. To apply knowledge in implementation of two pass microprocessor.

Algorithm/Flowchart:



Design diagrams (if any):

1. Flow Chart for Pass1 and Pass2
2.

Input:

Small assembly language program with macros written in file input.asm.

```
        MACRO
&lab    ADDS &arg1,&arg2
&lab    L 1, &arg1
        A 1, &arg2
        MEND
PROG START 0
        BALR 15,0
        USING *,15
LAB ADDS DATA1, DATA2
        ST 4,1
DATA1 DC F'3'
DATA2 DC F'4'
        END
```

Output:

Assembly language program without macro definition but with macro call.

Note: Follow the following templates during implementation

Macro Name Table (MNT) :

Index	Macro Name	MDT Index
1	ADDS	15

Macro Definition Table (MDT) :

Index	Macro Definition Card entry
15	&lab ADDS &arg1, &arg2
16	#0 L 1, #1
17	A 1, #2
18	MEND

Argument List Array (ALA) :

Index	Arguments
0	&lab
1	&agr1
2	&arg2

Test Cases:

1. Check macro end not found.
2. Duplicate macro name found.
3. Check program output by changing macro name and parameter list.
4. Handle label in macro definition.
5. Handle multiple macro definitions and calls

Software Requirement:

1. Fedora
2. Eclipse
3. JDK

Hardware Requirement: N/A

Frequently Asked Questions:

1. Define macro?
2. Define purpose of pass-1 of two pass macro processor
3. List out types of macro arguments
4. What is the use of MDT-index field in MNT?
5. What we store in ALA?

Conclusion: We have successfully completed implementation of Pass-I of macro processor.

Problem Statement: Design suitable data structures and implement Pass-II of a two pass macro processor using OOP features in Java/C++. The output of Pass-I (MNT, MDT, ALA & Intermediate code file without any macro definitions) should be input for Pass-II.

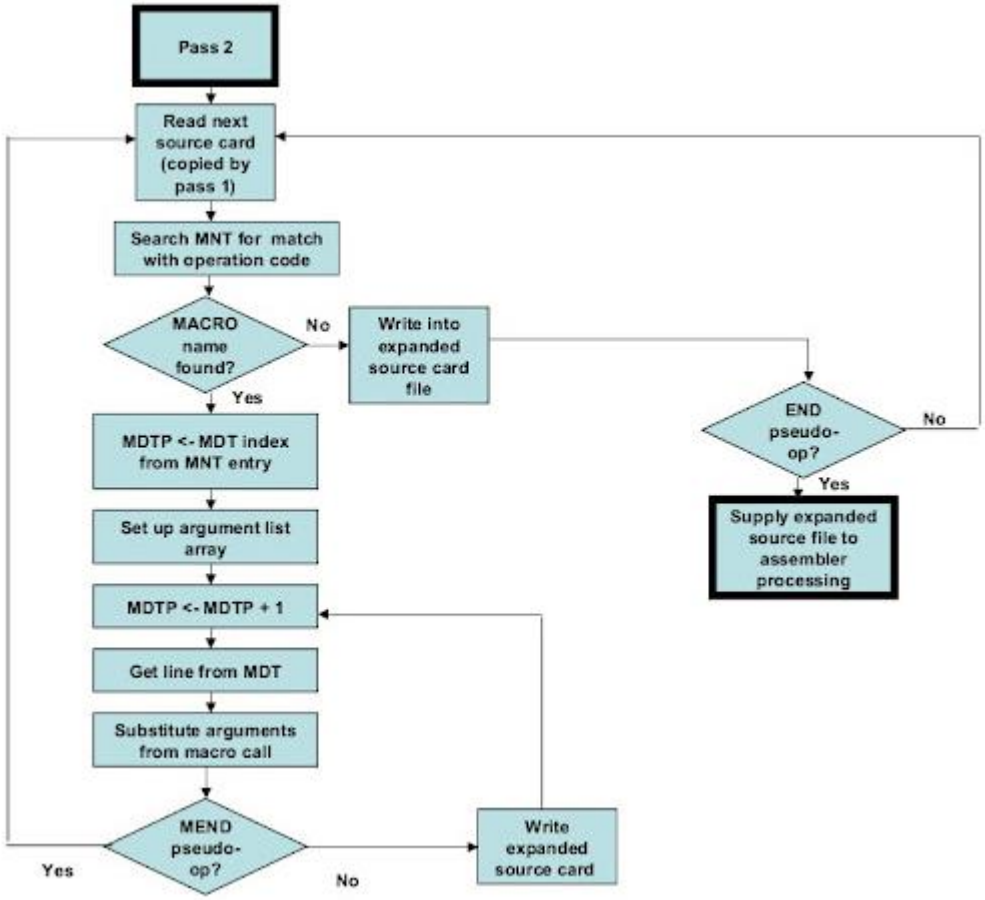
Objectives:

1. To identify and design different data structure used in macro-processor implementation
2. To apply knowledge in implementation of pass-2 of two pass microprocessor.

Theory:

1. Explain design steps of two pass microprocessor, types of statements, data structures required and flowcharts.

Algorithm/Flowchart:



Design diagrams (if any):

1. Class diagram
2. Sequence diagram
3.

Input: Output of pass-1 (Intermediate File) given as a input to pass-2.

```
PROG START 0
BALR 15,0
USING *,15
LAB ADDS DATA1, DATA2
ST 4,1
DATA1 DC F'3'
DATA2 DC F'4'
END
```

Output:

Assembly language program without macro definition and macro call.

```
PROG START 0
BALR 15,0
USING *,15
LAB L 1, DATA1
A 1, DATA2
ST 4,1
DATA1 DC F'3'
DATA2 DC F'4'
END
```

Macro Name Table (MNT):

Index	Macro Definition Card entry		
15	&lab	ADDS	&arg1, &arg2
16	#0	L	1, #1
17		A	1, #2
18		MEND	

Macro Definition Table (MDT):

Index	Macro Name	MDT Index
1	ADDS	15

Argument List Array (ALA) :

Index	Arguments
0	LAB
1	DATA2
2	DATA3

Test Cases:

1. Check macro definition not found.
2. Check program output by changing parameter list in macro call.

Software Requirement:

1. Fedora
2. Eclipse
3. JDK

Hardware Requirement: N/A

Frequently Asked Questions:

1. What is macro expansion?
2. Define purpose of pass-2 of two pass macro processor
3. What is positional arguments?
4. What is the use of MDT-index field in MNT?
5. What is the use of MNT table while processing macro call?

Conclusion: We have successfully completed implementation of Pass-II of macro processor.

Instructions :

Handwritten write-up as follows :

- Name of Student:
- Subject:LP-1
- Assessment table
- Date of Performance:_____ Date of Completion_____
- Title
- Objectives
- Problem statement
- Software and hardware requirements

Theory:

1. What is a macro processor?
2. Differentiate Macro and Function?
3. Explain the design of Pass- I of macro-processor with the help of flowchart?
4. Explain the design of Data structure used in Pass-I?
5. Explain the design of Pass- II of the macro-processor with the help of flowchart?
6. Conclusion

Attach Program code and Its Output

Assignment No. 03

Problem Statement:

Write a program to simulate CPU Scheduling Algorithms: FCFS, SJF (Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive).

Objectives:

1. To study the process management and various scheduling policies viz.Preemptive and Non preemptive.
2. To study and analyze different scheduling algorithms.

Input:

1. Enter the number of processes:
2. Enter burst time and arrival time of each process

Theory:

FCFS (First Come First Serve)

- Non-preemptive scheduling algorithm.
- Processes are executed in the order of their arrival.
- Simple and fair but may lead to **convoy effect**.

➤ SJF (Preemptive)

- Also known as **Shortest Remaining Time First (SRTF)**.
- The process with the shortest remaining burst time is executed next.
- Preemptive: ongoing process can be interrupted if a shorter process arrives.

➤ Priority Scheduling (Non-Preemptive)

- Each process has a priority, and the CPU is assigned to the process with the highest priority (lowest number).
- Non-preemptive: once a process starts, it finishes before another takes over.

➤ Round Robin (Preemptive)

- Time-sharing scheduling.
- Each process is given a fixed time quantum.
- If a process doesn't finish in its time quantum, it's sent to the end of the queue.

Test Cases:

1. Check arrival time of all process should not be same.

Software Requirement:

1. Fedora
2. Eclipse
3. JDK

Hardware Requirement: for simulation no dependency

Frequently Asked Questions:

i) What are the types of CPU scheduler?

ii) What is the difference between long and short term scheduling?

iii) Logic of program?

iv) What is preemptive and non-preemptive scheduling?

v) What are types of scheduling algorithms?

vi) Why Priority scheduling may cause low-priority processes to starve?

vii) What are the goals of scheduling?

viii) Define the difference between preemptive and nonpreemptive scheduling.

ix) Which scheduling algorithm is best? Why?

Conclusion:

Instructions :

Handwritten write-up as follows :

- Name of Student:
- Subject:LP-1
- Assessment table
- Title
- Objectives
- Problem statement
- Software and hardware requirements
- **Theory :**
- Define process.
- Explain need of process Scheduling
- Explain different scheduling criteria and policies for scheduling processes
- Explain possible process states
- Explain FCFS,SJF(Preemptive,priority(Non-preemptive) and Round Robin(Preemptive) and determine waiting time,turnaround time,average turn around time for each algorithm through example
- Conclusion

Attach Program code and Its Output

Assignment No. 04

Problem Statement:

Write a Java Program (using OOP features) to implement paging simulation using 1. FIFO 2. Least Recently Used (LRU) 3. Optimal algorithm

Objectives:

1. To study page replacement policies to understand memory management.
2. To understand efficient frame management using replacement policies.

Theory: CONCEPT OF PAGE REPLACEMENT:

1. Page Fault: Absence of page when referenced in main memory during paging leads to a page fault.

2. Page Replacement: Replacement of already existing page from main memory by the required new page is called as page replacement. And the techniques used for it are called as page replacement algorithms.

NEED OF PAGE REPLACEMENT:

Page replacement is used primarily for the virtual memory management because in virtual memory paging system principal issue is replacement i.e. which page is to be removed so as to bring in the new page, thus the use of the page replacement algorithms. Demand paging is the technique used to increase system throughput. To implement demand paging page replacement is primary requirement. If a system has better page replacement technique it improves demand paging which in turn drastically yields system performance gains.

PAGE REPLACEMENT POLICIES:

1. Determine which page to be removed from main memory.
2. Find a free frame.
   1) If a frame is found use it
   2) if no free frame found, use page replacement algorithm to select a victim frame.
   3) Write the victim page to the disk.
3. Read the desired page into the new free frame, change the page and frame tables.
4. Restart the user process.

PAGE REPLACEMENT ALGORITHMS:

1. FIFO

This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

2. OPTIMAL PAGE REPLACEMENT ALGORITHM: Replace the page that will not be used for longest period of time as compared to the other pages in main memory. An optimal page replacement algorithm has lowest page fault rate of all algorithm. It is called as OPT or MIN.

ADVANTAGE:

1) This algorithm guarantees the lowest possible page-fault rate for a fixed no. of frames.

DISADVANTAGE:

1) The optimal page replacement algorithm is very difficult to implement, as it requires the knowledge of reference strings i.e. strings of memory references.

3. LEAST RECENTLY USED (LRU): LRU algorithm uses the time of the page's last usage. It uses the recent past as an approximation of the near future, then we can replace the page that has not been used for the longest period of the time i.e. the page having larger idle time is replaced. ADVANTAGE: 1) The LRU policy is often used for page replacement and is considered to be good. DISADVANTAGES: 1) It is very difficult to implement. 2) Requires substantial hardware assistance. 3) The problematic determination of the order for the frames defined by the time of last usage

Algorithm:

1. FIFO :

1. Start the process
2. Read number of pages n
3. Read number of pages no
4. Read page numbers into an array a[i]
5. Initialize avail[i]=0 .to check page hit
6. Replace the page with circular queue, while re-placing check page availability in the frame Place avail[i]=1 if page is placed in the frame Count page faults
7. Print the results.

8. Stop the process.

2. LEAST RECENTLY USED

1.Start the process

2. Declare the size

3. Get the number of pages to be inserted

4. Get the value

5. Declare counter and stack

6. Select the least recently used page by counter value

7. Stack them according the selection.

8. Display the values

9. Stop the process

3. OPTIMAL ALGORTHIM:

1. Start Program

2. Read Number Of Pages And Frames

3.Read Each Page Value

4. Search For Page In The Frames

5. If Not Available Allocate Free Frame

6. If No Frames Is Free Replace The Page With The Page That Is Least Used

7.Print Page Number Of Page Faults

8.Stop process.

Input:

1. Number of frames

2. Number of pages

3. Page sequence

Output:

1. Sequence of allocation of pages in frames (for each algorithm)

2. Cache hit and cache miss ratio.

Software Requirement:

1. Eclipse
2. JDK

Hardware Requirement: for simulation no dependency

Frequently Asked Questions:

i) 1. What is virtual memory?

ii) 2. Explain working of LRU page replacement algorithm

iii) 3. Explain working of OPTIMAL page replacement algorithm

iv) 4. Which Page replacement algorithm is best?

v) 5. Explain what is Belody's Anomaly?

vi) 6. Explain the scenario in which page replacement algorithm is used?

vii) 7. Explain what is page fault?

viii) 8. Explain what is paging scheme? 9. Explain what is counting based page replacement algorithms?

Conclusion: Successfully implemented all page replacement policies..

Instructions :

Handwritten write-up as follows :

- Name of Student:
- Subject:LP-1
- Assessment table
- Title
- Objectives
- Problem statement
- Software and hardware requirements
- **Theory :**
- 1. What is virtual memory?
- 2. Explain working of LRU page replacement algorithm
- 3. Explain working of OPTIMAL page replacement algorithm
- 4. Which Page replacement algorithm is best?
- 5. Explain the scenario in which page replacement algorithm is used?
- 6. Explain what is page fault?
- 7. Explain what is paging scheme? 9. Explain what is counting based page replacement algorithms?

Attach Program code and Its Output

Assignment No. 05

Problem Statement:

To implement the **Banker's Algorithm** for resource allocation and deadlock avoidance in an operating system and determine whether the system is in a safe state.

Objectives:

1 Understand the concept of **safe state** and **deadlock avoidance** in operating systems.

2 Implement the **Banker's Algorithm** for resource allocation to multiple processes and resource types.

3 Calculate the **Need** matrix and simulate resource allocation.

4 Determine whether the system remains in a safe state after resource requests.

Theory:

Banker's Algorithm is a deadlock avoidance algorithm used in operating systems. It helps in allocating resources to processes in such a way that the system always remains in a **safe state**.

- A system is in a **safe state** if there exists a **safe sequence** of all processes such that each process can finish with the currently available resources.
- The algorithm uses:
  - **Available**: Available instances of resources
  - **Max**: Maximum demand of each process
  - **Allocation**: Currently allocated resources
  - **Need**: Remaining resources needed = Max - Allocation

Algorithm Steps:

1. Input the number of processes and resource types.
2. Input the Allocation, Max matrices, and Available vector.
3. Calculate the Need matrix using:
   `Need[i][j] = Max[i][j] - Allocation[i][j]`
4. Find a process whose needs can be met with available resources.
5. If found, assume the process completes and release its resources back to the available pool.
6. Repeat steps 4–5 until all processes are finished or no further allocation is possible.
7. If all processes are completed, the system is in a **safe state** and the safe sequence is printed.

Software Requirement: • Java

Platform: Any IDE or text editor

Hardware: i3 or above processor, 2 GB RAM or more

Hardware Requirement: i3 or above processor, 2 GB RAM or more

Frequently Asked Questions:

1 What is the purpose of the Banker's Algorithm?

2 Define Safe State and Unsafe State.

3 What is the role of the Need matrix?

4 Can Banker's Algorithm work with dynamic resources?

5 What are the limitations of the Banker's Algorithm?

Conclusion: Successfully implemented Banker's Algorithm

Instructions :

Handwritten write-up as follows :

- Name of Student:
- Subject:LP-1
- Assessment table
- Title
- Objectives
- Problem statement
- Software and hardware requirements
- **Theory :**

Attach Program code and Its Output