LPI - SPOS

Date:- / /
Page No.:-

Experiment No. 1

Title: Design suitable data structures & implement pass-I of a two pass assembler for pseudo-machine in Java using object oriented feature. Implementation should consists of a few instructions from each category and few assembler directives.

Objectives: To understand data structures to be used in pass-I of an assembler.
- To implement pass-I of an assembler.

Problem statement :- write a program to create pass-I assembler.

Outcomes: After completion of this assignment students will be able to.
- understand of the concept of pass I assembler.
- understand the progeamming language of Java.

Software requirements: linux OS, JDK 1.7.

Hardware requirements: 4 GB Ram, 128 GB SSD.

Theory Concepts: A language translator bridges an execution gap to machine language of computer system. An assembler is a language translator whose source language is assembly language.

- language processing activity consists of two phases, analysis phase and synthesis phase. Analysis of source program consists of three components, lexical rules Syntax rules and Semantic rules. lexical rules govern the function of valid statements in source language. Semantic rules associate the formation meaning with valid statements, which have the same meaning as source language statements. This consists of memory allocation and code generation.

Two pass translation scheme :-

- In a 2-pass assembler, the first pass constructs an intermediate representation of the source program of use by the second pass. This representation consists of two main components - data structures like symbol table, literal table & processed form of the source program called as intermediate code (IC). the intermediate code is represented by the syntax of variant - I.

- Analysis of source program statements may not be immediately followed by snythesis of equivalent target statements. this is due to forward references issue concerning memory requirements and organisation of language processor (LP).

language Processor Pass:

- It is the processing of every statement in a source program or its equivalent representation to perform language - processing function.

Assembly language statements:

- there are three types of statements imperative. declarative, assembly directives. An imperative statement indicates an action to be performed during the execution of assembled program. Each imperative statement usually translates into one machine instruction.
- Declare statement eg: DS reserves area of memory and associates names with them. DC constructs memory word containing constants. Assembler directives instruct the assembler to perform the certain actions during assembly of a program.

function of analysis and synthesis phase:

Analysis phase:

- Isolate the label: operation code and operand fields of a statements.
- Enter the symbol found in labled field (if any) and address of next available machine code into symbol table. validate the mnemonic operation code by looking it up in the table. determine the machine storage requirements of the statement by considering statements.

Date:- / /
Page No.:-

Synthesis phase:
- obtain the machine operation code corresponing to the mnemonic operation code by searching the mnemonic table.
- obtain the address of the operand from the symbol table synthesize the machine instruction or the machine form of the constant as the cas may be.

Datastructures of a two pass assembler:
Data structures of assembler:
a) Operation code table (OPTAB): This is used for storing mnemonic, operation code and class of instruction structure of OPTAB is as follows,
b) Data structure update during translation: Also called as translation time data structure. they are
1] Symbol table (SYMTAB): It contains enteies such as Symbol, its address and value.

2] Pool table (POOLTAB): It contains entries number of the starting literal of each literal pool.

3] literal Table (LITTAB): It contains enteies such as literal and its value.

4] location counter which contains address of next instruction by calculating length of each instruction.

Date:- / /
Page No.:-

Design of a Two Pass Assembler:

Tasks performed by the passes of two pass assember are as follows.

Pass - I:

Seperate the symbol, mnemonic opcode and operand fields.

Determine the storage required for every the assembly language statement and update the location counter.

Build the symbol table and the literal table.

Construct the intermediate code for every assembly language statement.

Pass - II:

Synthesis the target code by processing the intermediate code generated during pass I.

Intermediate code representation:

The intermediate code consists of a set of IC units each IC units consisting of the following three fields:

1. Address
2. representation of the mnemonic opcode.
3. Representation of operands.

Mnemonic field:

- The mnemonic field contains a parity of the form: (Statement class, code).

Date:- 1 1
Page No.:-

- for imperative statement, code is the instruction opcode in the machine language.
where Statement class can be one of IS, DL and AD standing for imperative statements, declaration statement and assembly directive respectively.

Declaration Statements		Assembler Directives	
DC	01	START	01
DC	02	END	02
		ORIGIN	03
		EQU	04
		LTORG	05

Algorithm (procedure):

Pass I :

- Initialize location counter, enteies of all tables as zero.
- read statements from input file one by one.
- while next statement is not END statements.
- If label is present insert label into symbol tabel.
- for declaractive statement update code, size & location counter.
- Generate intermediate code.
- pass this intermediate code to pass-2.

Conclusion :-

Thus, I have studied visual programming and implemented dynamic link library application for arithmatic operation.

Experiment NO-2

Date:- / /
Page No.:-

Title : Implement Pass-II of two pass assembler for pseudo-machine in Java using object oriented features. The output of assignment-I (intermediate file & Symbol table) should be input for the assignment.

Objectives : To understand data structures to be used in pass II of an assembler.
- To implement pass-I of an assembler.

Problem statement : write a program to create pass-2 assembler.

Outcomes : After completion of this assignment students will be able to:
- understand the concept of pass-II assembler.
- understand the programming language of Java.

Software requirements :- Linux OS, JDK 1.7.

Hardware requirements :- 4 GB Ram, 256 GB SSD.

Theory concepts :-
* Design of a two pass Assembler -
- TASKS performed by the passes of two-pass assembler are as follows.

Pass I : Seperate the symbol, mnemonic opcode and operand fields.
determine the storage - required for every assembly language.

statement and update the location counter. Build the symbol table and the literal table.
- Construct the intermediate code for every assembly language statements and update the location counter. Build the symbol table and the literal table.

Pass II :- Synthesis the target code by processing the intermediate code generated during pass I.

Data structures used by pass II :

1. OPTAB : A table of mnemonic opcodes and related.
2. SYMTAB : the symbol table.
3. POOL-TAB and LITTAB : A table of literals used in program.
4. Intermediate code generated by Pass I
5. Output file containing target code/ error listing.

Algorithm :-

1) Code_area_address = address of code area.
   Pooltab_ptr = 1;
   LOC_cntr = 0;

2) while next Statement is not an END statement.
   a) clear the machine_code_buffer.
   b) if an LTORG statement.

I) Process literals in LITAB [POOLTAB [pooltab-ptr]]..
LITTAB [POOLTab [pooltab-ptr+1]] - 1 similar to processing of constraints in a dc statement.

II) Size = Size of memory area required for literals.

III) pooltab-ptr = pooltab-ptr + 1

c) IF a START or ORIGIN statements then
I) loc-cnte = value specified in operand field.
II) Size = 0 ;

d) IF a declaration statement.
I) IF a DC statement then assembler the constant in machine-code-buffer.
II) Size = Size of memory area required by DC or DS.

e) if an imperative statement then
I) get operand address from SYMTAB or LITTAB.
II) Assembler instruction in machine code buffer.
III) Size = Size of instruction.

f) IF size ≠ 0 then..
I) move contents of machine-code-buffer to the address code-area-address + loc-cnte;
II) loc-cnte = loc-cnte + size;

3. Processing OF END Statements.

Conclusion :- thus, I have studied visual programming and implemented dynamic link library application for arithematic operation.

Practide NO 3

Date:- / /
Page No.:-

Title : Write a program for pass-II of a two-pass macro processor.

Objectives :
- To identify and create the data structures required in the design of macro processors.
- To learn parameter processing in macro.
- To implement pass II of macroprocessor.

Problem statement : Design suitable data structure & implement Pass-I and Pass-II of a two-pass macroprocessor. The output of pass-I (MNT, MDT and intermediate code file without any macro definitions).

Outcomes : After completion of this assignment students will be able to understand the concept of pass-II macro-processor.

Software requirements : Linux OS, JDK 1.7.

Hardware requirements : 4 GB RAM, 500 GB SSD/HDD.

theory concepts :
Pass II : replace every occurence of macro call with macro definition.
ere are four basic tasks that any macro instruction rocess must perform.

Date:- / /
Page No.:-

1) Recognize macro definition.
- A macro instruction processor and must recognize macro definition identified by the MACRO and MEND pseudo-ops. This tasks can be complicated when macro definition appears within macros. When MACRO's and MENDS are nested, as the macro processor must recognize the nesting and correctly match the last or outer MEND with first MACRO.

2) Save the definition.
- The processor must store the macro instruction definition, which it will need for expanding macro calls.

3) Recognize calls.
- the processor must recognize the macro calls that appear as operation mnemonics. This suggests that macronames be handled as a type of op-code.

4) Expand calls and suitable arguments.
- the processor must substitute for dummy or macro definition arguments the corresponding arguments form a macro calls, the resulting symbolic text is then substitute for macro calls. this text may contain additional macro definition or call.

Implementation of a 2 pass algorithm.
1). we assume that our macroprocessor is functionally independent of the assembler, and that the output text from the macro..

Date:- 1.1
Page No.:-

processor will be fed into the assembler.
2. The macro processor will make two independant scans or passes over the input text searching first for macro definitions & then for macro calls.
3. the macro processor cannot expand a macro call before having found & saved the corresponding macro.
4. Thus we need two passes over the input text, one to handle macro definitions & other to handle macro calls.
5. the firstpass examines every operation code, will save all macro definitions in a macro definition table & save a copy of the input text, minus macro definitions on the secondary storage.
6. The first pass also prepares a macroName Table along with macroDefinition table as seen in the previous assignment that successfully implemented pass-I of macro-pre-processor.

Specification of DataBase.

Pass-2 Database.

1. The Copy of the input source deck obtained from Pass-I.
2. The output expanded source deck to be used as input to the assembler.
3. The macro definition Table (mDT), created by PassI.
4. The macro Name Table (mNT), created by Pass I.
5. The macro definition Table (mDT) counter, used to indicate the next line of text to be used during Macro expansion.

6. the Argument list Array (ALA), used to substitute macro call arguments for the formal markers in stored macro definition.

Conclusion : thus we have successfully implemented pass - II of a two-pass assembler.

Practicle NO 4

Date:- / /
Page No.:-

Title: write a program to create dynamic link library for any mathematical operation & write an application program to test it.

*Objectives:
- to understand dynamic link libraries concepts.
- to implement dynamic link below concepts.
- to study about visual basic.

* Problem statement:-
- write a program to create dynamic link library to arithematic operation in VB net.

* outcomes:-
- After completion at assignment student well be able to:
- understand programming languages at visual basics.

* Software requirements:-
Visual Studio 2010.

* Hardware requirements:-
M/C lenovo think center M700, Ci3 6100, 6th gen H81, 4 GB RAM, 500 GB HDD.

Theory:

* Dynamic Link Libraries :-
  - A DLL is a collection of small programs that can be loaded when needed by larger program, and used at the same time DLL files that supports specific, device the operations are known as device drivers.

A DLL file is often given a 'dll' file name sufix DLL files are dynamically linked with the program that was them during compiled into the main program.

* feature of DLL :-
- DLL are essentially the same as EXE the choice of which to produce as parts of linking process is for clarity since it is possible to export function and data from either.

- DLL's execute in memory space at calling process and with the same access permision which means there is little overhead in their use but also that there is no potential for calling exe of the DLL a has any sort of bug.

Date:- / /
Page No.:-

* Difference betn Application vs DLL ?

- An application can have multiple instances at itself running in system simultaneously, whereas a DLL can be only one instance.

- An application can own thing, such as stack global memory, file handlers and a message queue but a DLL a cannot.

* Executable file think to DLL :-
- An executable file links to a DLL in one at two ways.

i) Implicit linking.
ii) Explicit linking.

- calling DLL function from which visual basic application.

In the DEF file.
EXPORTS
MYFUNC = myfunc @12
INITCode = Initcode @ 0.

* Advantages
1) code reusability.
2) efficient updates.

Date:- / /
Page No.:-

3) memory efficiency.
4) versioning.
5) seperating of concerns
6) encapsulation
7) multithreading.

* Disadvantage of DLL
1) dependancy hell.
2) Security risk.
3) lack of isolation.
4) performance overhead.
5) portability Issues.
6) debugging challenges.

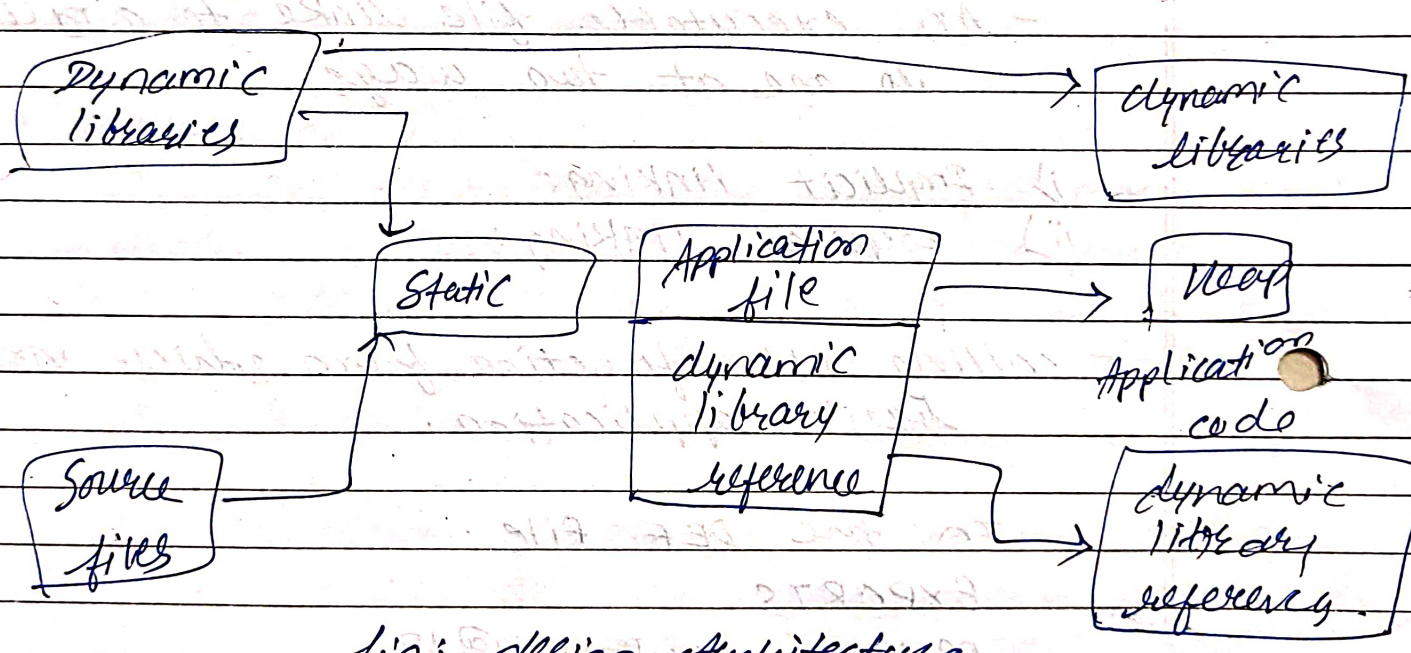


fig: design architecture

Conclusion :- Thus, I have studied visual programming & implemented all, application for arithmetic operation.

LP-I-
SPOS

Experiment No. 5

Date:- / /
Page No.:-

Title: Write a Java program (using oop features) to implement following scheduling algorithms: FCFS, SJF (preemptive), Priority (non-preemptive) and Round Robin (Preemptive).

Objectives: To understand OS & SCHEDULING concepts
- To implement scheduling FCFS, SJF, RR & priority algorithms.
- To Study about Scheduling & Shedular.

Problem Statement: write a Java program (using oop features) to implement following scheduling algorithms FCFS, SJF, Priority and round robin.

outcomes: After Completion of this assignment Students will be able to.
- Knowledge scheduling policies.
- Compare diffrent scheduling algorithms.

Software requirements: JDK / eclipse.

Hardware requirements: m/c lenovo think center, m700 Ci3, 6100, 6th Gen, H81, 4GB RAM, 500 GB HDD.

Theory Concepts:

CPU Scheduling: CPU scheduling refers to a set of policies and mechanisms built into the OS that govern the order in which the work to be done by a computer system is completed.

- Scheduler is an OS module that selects the next job to be admitted into the system & next process to run.
- The primary objectives of Scheduling is to optimize system performance in accordance with the criteria deemed most important by the system designers.

What is scheduling ?
- Scheduling is defined as the process that governs the order in which the work is to be done.
- Scheduling is done in the area as where name of jobs or works are to be performed. then it requires some plan.

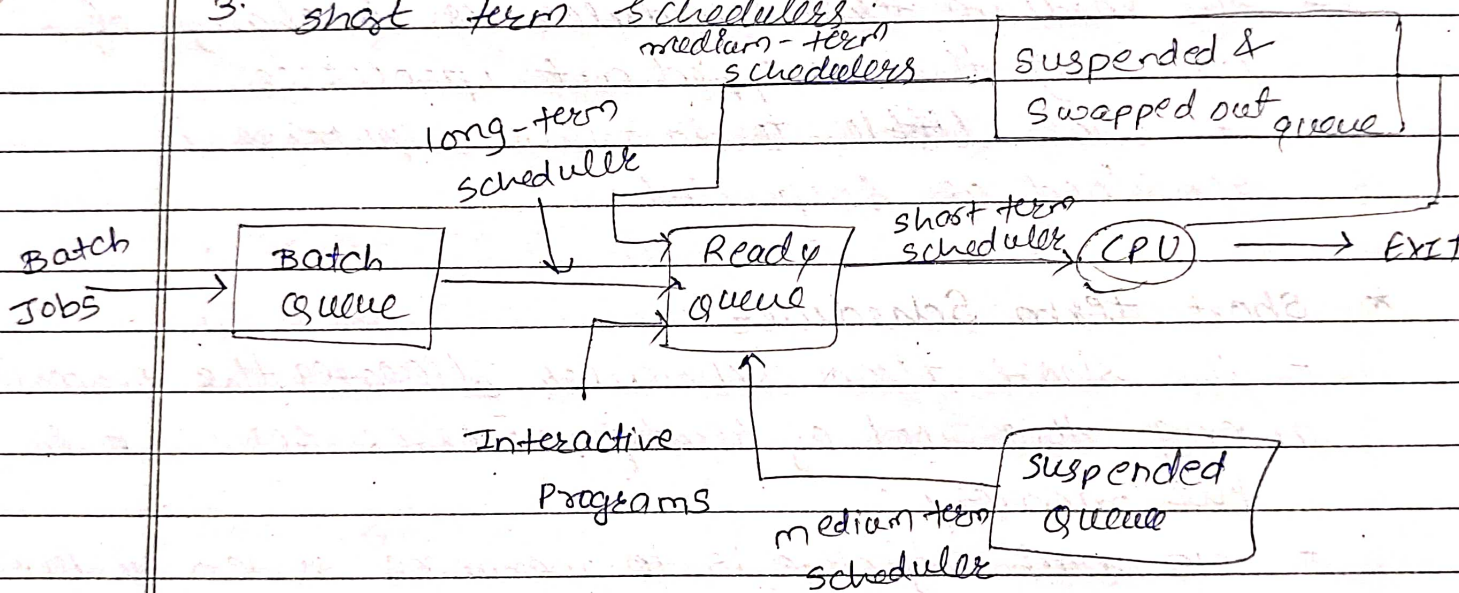
what is Scheduler ?
1] Scheduler in an OS module that selects the next job to be admitted into the system & the next process to run.
2] primary objective of the scheduler is the optimize system performed in accordnce with the criteria decmed by the system designers.

Necessity of Scheduling :
- Scheduling is required when no. of jobs are to be performed by CPU.
- scheduling provides mechanisms to give order to each work to be done.
- Primary objective of scheduling is to optimize system performance.

Date:- / /
Page No.:-

* Scheduling provides the case to CPU to execute the processes in efficient manner.

Types of Schedulers:
- In general, there are three different types of schedulers which may co-exists in a complex operating system.
1. long term schedulers.
2. medium term schedulers.
3. short term schedulers.



* long Term Scheduler:
- The long term schedulers, when present works with the batch queue and selects the next batch-job to be executed.
- Batch is usually reserved for resource intensive low priority programs that may be used fillers of low activity of interative jobs.

* medium term scheduler:
- After executing for a while, a running process may become suspended by making an I/O request or by issuing a system call.
- when a number of processes becomes suspended, the remaining supply of ready processes in systems where all suspended processes remain resident in memory may become reduced to a level that imparts functioning of schedulers.
- the medium term scheduler is in charge of handling the swapped out processes.
- it has little to do while a process is remained as suspended.

* Short term Scheduler:
- the short term scheduler allocates the processor among the pool of ready processes resident in the memory.
- Its main objective is to maximize system performance in accordance with the set of criteria.
- Some of the events introduced this for that cause rescheduling by virtue of their ability to change the global system states are:
  1. clock ticks.
  2. Interrupt and I/O completions.
  3. Sending & receiving of signals.
  4. Activation of interactive programs.
- whenever one of these events occurs, the OS involves the short term scheduler.

Scheduling Criteria :-

* CPU utilization : Keep the CPU as key busy as possible. It range from 0 to 100%. In practice, it range from 40 to 90%.

* Through put : throughput is the rate at which processes are completed per unit of time.

* Turnaround time : this is the how long a process takes to execute a process. It is process and its completion.

* waiting time : waiting time is the sum of the time periods spent in waiting in the ready queue.

* Response time : Response time is the time it takes to start responding from submission time. It is calculated as the amount of time it takes from when first response is produced.

* Non-premitive scheduling : In non-premtive mode, once if a process enters into running state, it continues to execute until its running or blocks itself to wait for input / output or by requesting some Operating system service.

* Preemptive scheduling : In premptive mode, currently running process may be interrupted and moved to the ready state by the operating system.
  - when a new process arrives or when an interrupt occurs.

- It is desirable to maximize CPU utilization and throughput and to minimize turnaround time, waiting time and response time.

* Types of Scheduling algorithms:
- In general, Scheduling desciplines may be preemptive or non-preemptive.
- In batch, non preemptive implies that once Scheduler a selected job turns to completion.
- There are different types of Scheduling algorithms such as:
  1. FCFS (first come first serve).
  2. SJF (Short JOB first).
  3. priority Scheduling.
  4. Round robin Scheduling algorithm.

* first come first serve algorithm.
- FCFS is working on the simplest Scheduling desciplive.
- the workload is simply processed in an order of their arrival, with no pre-emption.
- FCFS Scheduling may result into poor performance
- since there is no discrimination on the basis of required services, short jobs may considerable in turn around delay and waiting time.

* Advantages:

- Better for long processes.

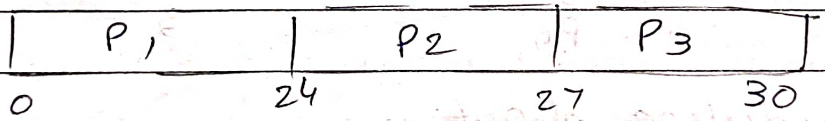
- Simple method (ie: minimum overhead on processor)
- No starvation.

* Disadvantages:
- convoy effect occurs. Even very small process should wait for its turn to come to utilize the CPU.
- Short process behind long process results in lower CPU utilization.
- Throughput is not emphasized.

* First come first served:

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1, P_2, P_3.
- The Gantt chart for the schedule is:



- waiting time for $P_1 = 0$, $P_2 = 24$, $P_3 = 27$.
- average waiting time: $(0 + 24 + 27)/3 = 17$.

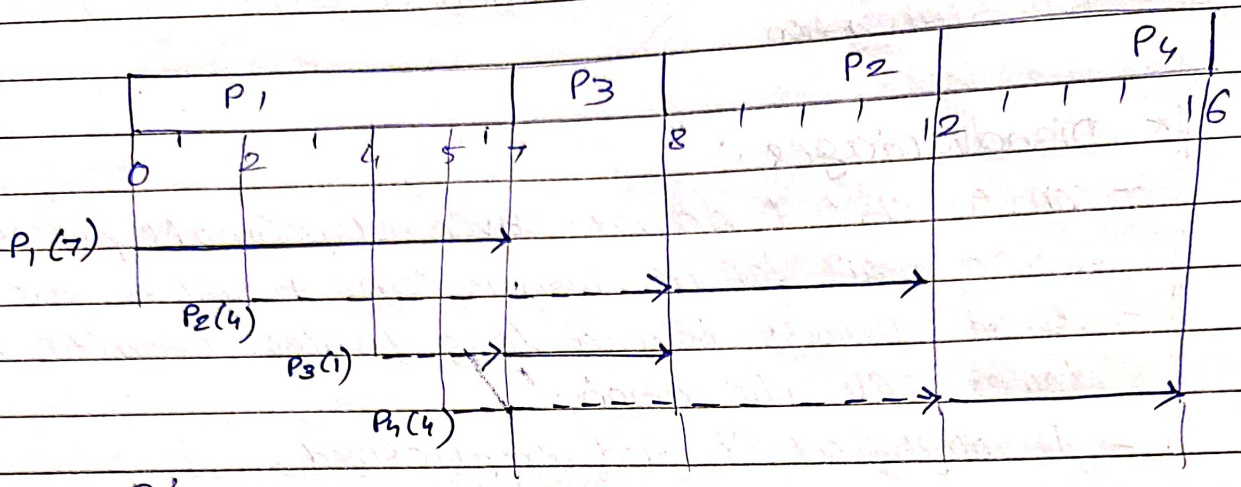
* Short Job first scheduling.

Example:

Process	Arrival time	Burst time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

Page No.:-

Non preemptive SJF Average waiting time = (0+6+3+7)/4 = 4

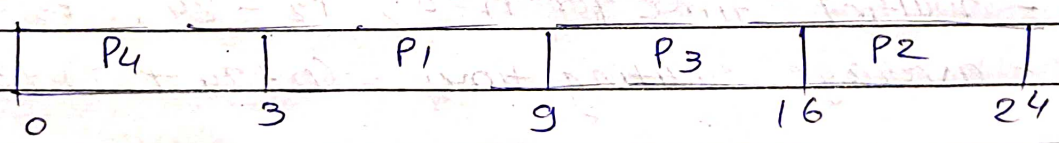


P1'S waiting time = 0
P2'S waiting time = 6
P3'S waiting time = 3
P4'S waiting time = 7

Example of SJF:

Process	Burst Time
P1	6
P2	8
P3	7
P4	3

SJF Scheduling chart



Average waiting time = (3+16+9+0)/4 = 7

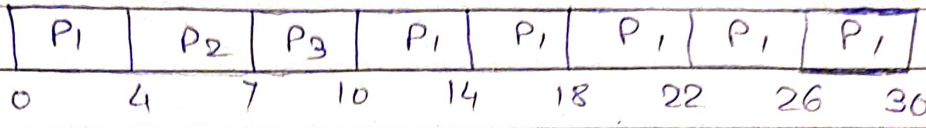
Round Robin

Process	Burst Time
P1	24
P2	3
P3	3

Date: 11
Page No.:

Quantum time = 4 miliseconds

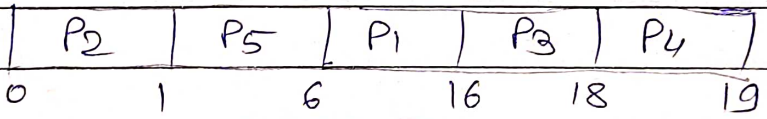
The Gantt chart is:



Priority.

Process	Burst time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

Gantt chart is:



Average waiting time = (6+0+16+18+1)/5 = 8.2

Conclusion :- Hence we have studied that -

- CPU scheduling concepts like context switching, types of scheduling, diffrent timing parameters, like waiting time turnaround time, burst time, etc.
- diffrent CPU scheduling algorithms like FIFO, SJF, etc.
- FIFO is the simplest for implementation but parameter large waiting times & reduces system performance.