Prof. Anand Gharu.
Assistant Professor
Computer Dept.
PVGCOE, Nasik.
8087777708.

QUESTION-ANSWER FOR ORAL

Subject ÷ System Programming & Operating System (SPOS)
Subject Code ÷ 310257 (SPOS LAB)
Total Practical ÷ 14.

TW - 25 Mark
PR - 50 Mark

UNIT-I

1) System Programming ÷
- it is the activity of writing and maintaing system s/w.

2) System s/w ÷
- it is computer s/w designed to operate and control the comp h/w. & provide platform for running appli^n s/w.
e.g. OS, utility s/w, device driver, compiler.

3) Application s/w ÷
- it is any program or group of program that is designed for end user.
e.g. d/b program, word processor, Web browser.

4) Software Development tools ÷
- 1) Editor 3) prog^n environment
2) Debug monitor 4) User Interface.

5) Types of text editor ÷
- 1) Line editor 3) word processor
2) Screen editor 4) Structure editors.

6) Basic function of loader ÷
- 1) Allocation 3) Relocation
2) Linking 4) loading

7) Assembler ÷
- it converts assembly code into m/c code.
Syntax ÷ Label, Mnemonics/opcode, operand1, operand2
e.g. tasm, masm, Nasm etc

8) Macro-processor ÷
- it allows sequence of src lang. code to be defined once & then refered to by its name.

Syntax ÷ macro name
macro body
⋮
End of macro def^n

ABC
ADD Areg, A
ADD Breg, B
Mend

9) Compiler ÷
- it convert high level lang into low level lang. prog.
e.g. C, C++, GCC, java compiler

10) Types of Compiler ÷
- 1) Cross compiler 3) Bootstrap compiler
2) Incremental 4) Native compiler.

11) Language Processor ÷
- it is s/w which bridges a specification ore exe. gap.

12) Types of Lang. processor ÷
- 1) Lang. translator 3) preprocessor
2) Detranslator 4) Lang. migrators

13) Interpreter ÷
- It is a program which scan prog. line by line & generate ICG code.
e.g. VB interpreter, java interpreter.

14) Compare Compiler vs Interpreter.

1) compiler scan whole code	1) scan line by line
2) code is optimised	2) No optimization
3) e.g. C, C++	3) VB, java

15) Von neuman Architecture
- 1) I/P unit 3) ALU 5) memory unit
2) O/P unit 4) Control unit 6) AC
7) DR 8) PC 9) IR 10) MAR.

16) Assembly lang. statement ÷
- 1) imperative statement (IS) - MOVER Areg, X
2) Declarative statement (DS) - X DS 1
X DS 1
3) Assembler Directive (AD) - START, END

Prepared By ÷ Prof. Anand Gharu

17) Assembler Directive ÷
- it gives direction to assemble which task is to be performed
e.g. START & END.

18) Literal and Constant ÷
- literal is an immediate operand seems in statement.
- e.g. x = 4+5 ; - 5 is known as literal
- Constant ÷ Z = 5 ; 5 is constant.

19) Literal vs Constant

1) literal can't be changed	1) Constant can be changed
2) e.g. x = 4+5	2) Z = 5
3) it is safe	3) it is not safe.
4) part of instruⁿ	5) not part of instruⁿ

20) Types of Assembler ÷
- 1) Load & go assembler.
2) pass-1 assembler 2) pass-2 assembler.

21) Forward References ÷
- values of variable is stored in forward instruⁿ
e.g. START 100
MOVER Areg, X — Here value of X is reference to variable X.
≡
X DC 1 ←
Note ÷ Backward reference is vice versa.

22) Advanced Assembler Directive ÷
- 1) ORIGIN 2) EQU 3) LTORG - literal origin.

23) Backpatching in Pass-1 assembler.
- Problem of forward reference is known as Backpatching. it is solved by Pass-2.

24) One pass Assembler ÷
- It scan src prog & generates
1) Symbol table. 2) Pool table
3) Literal table 4) ICG Table.
(problem of Backpatching & forward references)

25] Two-pass assembler
- accept I/P from pass-1, resolve problem backpatching & forward references & finally generates m/c code.

26) Data Structure for Pass-1
- 1) Machine Opcode Table (MOT
2) Symbol Table. (ST)
3) Literal Table (LT)
4) Pool Table (PT)
5) ICG Table (Intermediate Code)

27] Error reporting in assembler
- 1) Syntax error like missing comma
2) Invalid opcode
3) Undefined symbol
4) missing START or END.
5) Symbol defined But not used.

27] Is Literal processed in Pass-2 ?
- NO.

29] Undefined symbol are detected in Pass-1
- Yes.

30] Format of Intermediate Code ÷
- Each mnemonic field is represented by (statement class, machine code)

NOTE ÷ UNIT-1 most Q&A mentioned.
Even you should prepare all question of unit
- PASS-I Algorithm & Flowchart
- PASS-II Algorithm & Flowchart
- Block diag. for PASS-1 & PASS-2.
1) Practical 1 & 2 based on 1st unit
Prepare ÷ How to run Assemble code.
Javac - Java Compiler.
- Prepare pass-1 & pass-2 examples & Programs of file

Prepared By - Prof. Anand Gharu

UNIT-II

1) MACRO VS SUBROUTINES.

-1) macro is expanded 1) it cannot be expanded
2) exe. speed is more. 2) less
3) cannot handle label 3) handle label.
e.g. e.g.

2) Defining macro, Calling, Expansion :-

- Calling - name of macro, argument
  e.g. INCR X.
- expansion - MOVEM Areg, X
  ADD Breg, X
  mend.

3] Types of parameter in MACRO :

-1) keyword parameter - INCR var=A, Incr=B
2) positional 3) mixed parameter.
e.g. =INCR &var, &Ic,b) — both.

4] Nested Macro Call :-

- it is macro call within macro.

e.g. MACRO
ABC &Arg1
PQR &Arg2
MEND.

5] Advanced MACRO facility

-1) AIF - Advance If
2) AGO - Advance Go.

6] Issues related to Macro preprocessor -

-1) AIF 3) expansion time variable
2) AGO 4) sequencing symbol.
- Recognize MACRO defⁿ
- Save macro defⁿ
- Recognise MACRO call
- expand MACRO call.

7] MACRO-PREPROCESSOR :-

- it take src prg containing macro defⁿ & call & translate into assembly lang prog. w/o any macro defⁿ.

8] Database / Datastructure of PASS-1 & 2 macro-processor

- Macro Defⁿ Table (MDT).
- Macro Name Table (MNT)
- Argument list Array (ALA)
- MNT pointer (MNTP)
- MDT pointer (MDTP)

9] Methods of handling Nested macro call :-

- Several level of expansion
- Recursion expansion
- Use of stack during expansion.

NOTE :-
Practical 3 & 4 depend on UNIT-2.
So you should prepare all question of Unit 2
Some other questions :-
* Block diagram Pass 1 & 2 macro processor
* Algo & Flowchart of Pass-1 & 2 Macro.
* Examples of Macroprocessor Pass 1 & 2

LOADER.

1] Loading schemes or Types of loader.

-1) Compile & Go 4) Subroutine Linkage.
2) General loader 5) Relocating loader.
3) Absolute 6) Direct linking loader.

2] Overlay structure.

- it is part of prog. which have same load origin as some other part of prog.

LINKER.

1] Linker :-
it is prog. which links multiple object module together for exe. of prog.

2] Object Module :-
- it contains all infoⁿ necessary to relocate & link different modules.

By - Prof Anand Gharu

PROF. ANAND GHARU

3] Static & Dynamic Link libraries

- static linker takes object file produced by compiler. exe. file contain copy of every subroutine ...
- static linker is fixed, can't be changed runtime.
- Dynamic linking ÷
- it reference to an external module during runtime.
- perform reloc^n during runtime
- changes can be possible in dynamic

4] Dynamic Link Libraries (DLL) ÷

- DLL is microsoft imple^n of shared library in window. file format for DLL is same as window EXE.

A DLL can contain 1) code, data, Resource. Shared code is placed into a single, seperate file, The prog that call file are connected to it at runtime, with OS performing linking.

5) Loading phases using java ÷

1) Loading
2) Linking
3) Initializing

$$\begin{tikzcd} & \text{Bytecode verification} \\ \text{Linking} \arrow[ru] \arrow[r] \arrow[rd] & \text{class preparation} \\ & \text{Resolving} \end{tikzcd}$$

NOTE ÷ UNIT-2 notes are completed. you should read complete UNIT-2. with examples, Algorithm & Flowchart.

Prepared By ÷ Prof. Anand Gharu

UNIT-III

LANGUAGE TRANSLATOR

1) Token, pattern, lexemes & Error ÷

- Token - String of character in prog. e.g. identifier, keyword etc
- lexemes ÷ is seq. of char in src prog. that is ~~pattern~~ matched by pattern for token. e.g. int xy = 5 so xy is lexemes for token.
- pattern ÷ set of rules to match token.
- Lexical error ÷

error occures when pattern not matched e.g. ; missing, rules not matched. etc.

2] General model of compiler ÷ (diagram)

Phases of compiler ÷

- 1) lexical analyzer 4) ICG
- 2) syntax analyzer 5) Code optimization
- 3) Semantic analysis 6) Code Generation.

3] Representation of ICG ÷

- 1) Three Address Code 4) postfix notation
- 2) Quadruple 5) Syntax tree
- 3) Triple. 6) DAG Representation

4] Code optimization techniques ÷

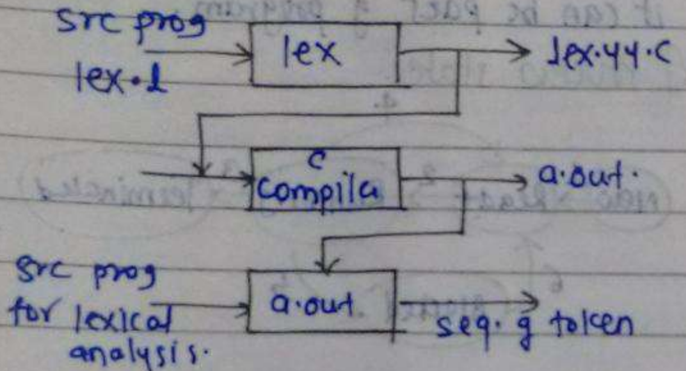
- 1) Compile time evaluation
- 2) Elimination of common sub expre^n
- 3) Dead Code Elimination.
- 4) Freq^n reduction 5) Strength Reduction.

5] Design issues of Code Generator ÷

- 1) I/P to code generator
- 2) Target prog.
- 3) Memory management
- 4) Instru^n selection
- 5) Register Allocation
- 6) Choice of evaluation order.
- 7) Approaches to code generation.

6] Software tools for compiler construction ÷
-1) Lex - analyzer generator
2) Yacc - parser generator

7] LEX ÷
- it is also tool, which scan src prog & generates token as keyword, identifier etc.



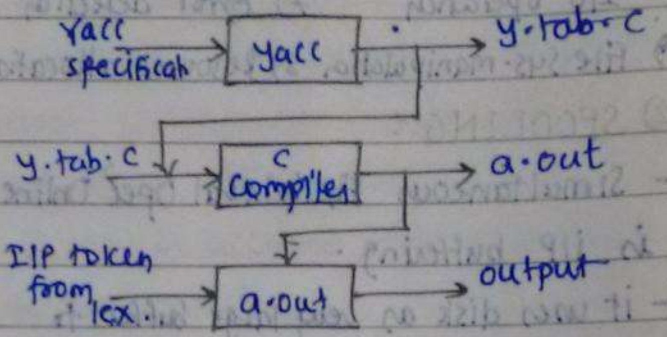
* Lex specification ÷
declaration
%.%.
translation rules
%.%.
user function.

* Function of LEX ÷
- 1) yylex() 4) yylval()
2) yytext() 5) yywrap()
3) yyleng() 6) YYin() 7) YYout()

* How to run Lex program
- Lex program.l
cc lex.yy.c
./a.out ↵

NOTE ÷ UNIT-3 rules are completed you should prepare all question of UNIT-3. you must prepare LEX & YACC program which we have studied.

8] YACC ÷
- it stands for Yet Another Compiler Compiler it creates C-prog for parser.
- Yacc accepts token generated by lex and match grammer of regular expres° in YACC prog with it's specified rules.



* YACC specification ÷
declaration
%.%.
translation rules
%.%.
C-function.

- The declaration section consist of token declaration & C-code by -
%{
%}.

- The context free grammer is placed in the rule section
- User function are added in last section
- Yacc generates file Y.tab.c.
- Yacc generates an header file y.tab.h for lex. this file contain integer values & it's each type of token.
- The y.tab.h file should be added in lex file

function of YACC ÷
1) yyparse() 2) yywrap() 3) yyerror()
4) yylex() 5) yylval() 6) yytext()

PROF. ANAND GHARU

Prepared By - Prof-Anand Gharu

UNIT-IV

OPERATING SYSTEM

1) Operating System ÷

- it provides interface betⁿ user & h/w.

e.g. window 7, ubuntu, Apple OS etc.

2] Function of Operating System

-1). prog. Development. 5) Communication
2) Prog. execution 6) Resource sharing
3) I/O operation. 7) Error detection
4) File sys. manipulation 8) Resource allocation.

2) SPOOLING ÷

- Simultaneous Pheripheral Opeⁿ Online is I/P buffering.
- it uses disk as very large buffer to store & read O/P file.

- spooling allows CPU to overlap I/P of one job with O/P of other job.

3) Types of Operating system ÷.

-1) Batch OS. 4) N/w OS
2) Multiprogramming 5) Distributed OS
3) Real time OS 6) time sharing OS.

4) Operating System Component ÷

-1) processes 4) signal
2) Files 5) Cmd Interpreter.
3) system call

4] System call ÷

it provides interface to user program with operating sys.

e.g. open(), close, fork, exit() etc.

5]. Command Interpreter (shell) ÷

- It provides cmd-line interface.

It allow user to enter cmd on cmd-line.

it interprete the cmd entered by user & executes it.

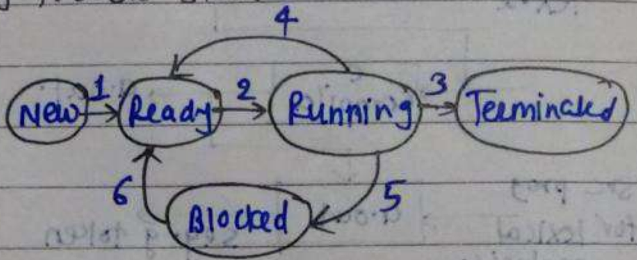
6] Types of Operating System Structure ÷

-1) Monolithic System
2) layered System
3) Virtual machine
4) Client Server model.

7] process ÷

- it is program while it is being executed it can be part of program.

8] Process state ÷



9] Process Control Block (PCB) ÷

- PCB keeps track of all information concerning a process.

i.e. process no, process state, Prog. Counter, Registers, etc.

10] Thread ÷

- Thread can be part of process.
- OS support multiple thread execution.
- thread use memory of process.

so it is lightweight process/thread.

Two types - 1) Single threaded process
2) multithreaded processes

11] Process schedulling ÷

It is set of policies & mechanism supported by OS to control the order in which work to be done is completed.

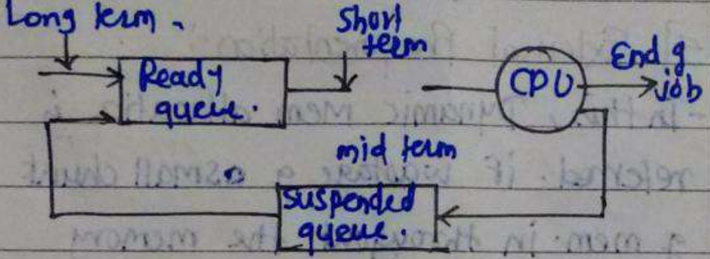
12] Schedular ÷

- it is OS program that select next job to be admitted for exe.

By. Prof. A.N. Gharu

13] Performance criteria for Process schedulling :-
- 1) CPU Utillzation 4) waiting time
2) Throughput 5) Response time.
3) Turnaround time

14] Types of Schedular :-
- 1) Long-term
2) medium term
3) short term.



15] schedulling methodes :-
- 1) preemptive schedulling
2) Non-preemptive schedulling

16] schedulling Algorithms (Types) :-
- 1) FCFS schedulling 3) RR schedullig
2) SJF schedulling 4) Priority schedulling

17] Interprocess Communication :-
- it allows communicating processes to exchange data and infor^n.
There are 2 method g IPC
1) Through shared memory
2) Through Message Passing.

18] Race Condition :-
- Race cond^n is situation, where two or more processes are reading / writing some shared data and final result depend on who run precisely & when.

19] Critical section :-
In this, only single process can be run at a time in critical section to avoid deadlock.

20] Mutual Exclusion :-
In this, if one user is using shared variable or file then other process will be excluded from doing same thing.

21] Semaphore :-
- it is synchronization tool was developed by Dijshtra.
- semaphore is a variable which accepts non-negative integer value and except initialization.
- It may be accessed & manipulated by two operation.
1) Wait 2) Signal.

22] IPC problem (classical sync^n problem)
- 1) producer-consumer problem
2) Reader-writer problem
3) Dining philosopher problem.

23] Monitors :-
- monitor is mechanism that support the safe & effective sharing g resources among process in addition to concurrency & synchronization.
- it allow control access to shared variable. It is form g data abstraction.

24] Deadlock :-
- Deadlock is a situation, where nos g processes try access same variable at a time.

25] Condition for Deadlocks :-
1) mutual Exclusion
2) Hold & wait
3) No preemption
4) circular wait.

Prepared By - Prof. Gharu Anand

26] Deadlock Avoidance ÷

There are two approaches ÷

1) Do not start process if it's demand might lead to deadstock.

2) Do not grant incremental resources request to process if this allocation might lead to deadstock

e.g. Bankers algorithms.

27] Banker Algorithms ÷

already mentioned in your file.

NOTE ÷ UNIT-4 notes are completed.
you should prepare all question of unit-4.
you should prepare all example of scheduling & Bankers algorithm.

UNIT-V

MEMORY MANAGEMENT

1] Categories of 80386DX register ÷

- 1) General Purpose reg
- 2) Segment reg.
- 3) index, pointer & base reg.
- 4) Flag reg.
- 5) System Address reg.
- 6) Control register
- 7) Debug registers.

2] Memory Management Techniques ÷

- multiprog with fixed partition
- multiprog with Dynamic partition

3] Placement Algorithm ÷

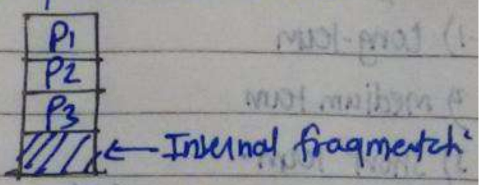
strategies to allocate free partition to program ÷

- 1) First fit
- 2) Best fit
- 3) worst fit.

4] Internal fragmentation ÷

- In this, it occurs fixed mem allo tech. if wastage mem. at the end of prog/Mem. then internal fragmentation occurs.

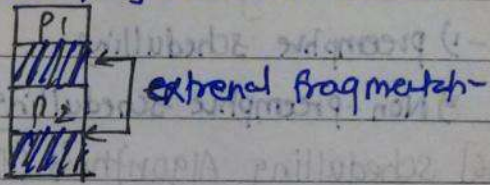
e.g.



5] External fragmentation ÷

- In this, Dynamic mem allocation is referred. if wastage of small chunk of mem. in throughout the memory then external fragmentation occurs

e.g.



6] Swapping ÷

- moving processes from main mem to disk and back is called as swapping.

7] Virtual Memory ÷

- it is mem tech. in which if physical memory is not sufficient to execute prog the virtual mem. swapped the pages to execute the program.

8] Paging ÷

- paging is mem mngment tech that permits a prog. mem to be non-contiguous into physical memory. This allows prog to be allocated physical memory wherever it is possible.

Prepared By - Prof. Anand Gharu.

9] Demand Paging ÷
- it means that each page g process is brought in only when it is needed.
- when process is started, if there are page fault / not sufficient memory then pages are demand to O.S for exe.

10] Page Replacement Policies ÷
- First In First Out (FIFO)
- Least Recently used (LRU)
- Optimal (OPT)
- Not Recently Used (NRU)

11] Design issue for paging
- 1) The working set
2) Local vs Global allocation
3) Page Size.

12] Segmentation ÷
- segmentation means dividation / partitiong available memory into different partition.

13] Thrashing ÷
- This situation may arise in demand paging when there are too many active processes in the memory and a very few piece g any process is in memory.
- when OS bring in pagein a memory it swap out another page. if OS throw out a page just before it is about to be used. Too much g this lead to condition known as thrashing.

OR

when OS demand or swapped pages for exe. & if pages are not available at that memory then we can say thrashing.

UNIT-V

INPUT AND OUTPUT, FILE SYSTEM

1] Types g I/O devices.
- 1) Block devices - DISK, HDD
2) character devices - KBD, printer, terminal

2] Techniques g DMA (Data transfer) mode
- 1) programmed I/P / O/P.
2) Interrupt driven I/P O/P
3) Direct Memory Access.

3] Types g Interrupt ÷
- 1) program interrupt (S/W interrupt)
2) Timer Interrupt
3) I/O Interrupt
4) Hardware failure.

3] DMA (Direct Memory Access) ÷
- In DMA, there is less intervention g CPU. or no intervention g CPU if CD.Rom or other external devices tries to interact with system then DMA allow these devices to directly access memory w/o using CPU.

4] I/O software layers ÷
- 1) user processes.
2) Device Independant S/W
3) Device driver
4) Interrupt handler
5) Hardware.

5] Magnetic Disk ÷
- It is used to store data
platter, sector, track, latency etc

PROF. ANAND GHARU

Prepared By - Prof Anand Gharu.

6] RAID (Redundant Array of Inexpensive Disk) 7 level :-

- 1) Non-redundant
- 2) Mirrored
- 3) Redundancy thro: hamming Code.
- 4) Bit-Interleaved parity.
- 5) Block level parity.
- 6) Block level Distributed parity
- 7) Dual Redundancy.

7] Disk schedulling algorithms :-

- 1) First Come first served Schedulling (FCFS)
- 2) Shortest Seek time First (SSTF)
- 3) Scan Schedulling.
- 4) Circular Scan (C-SCAN)

8] File Operation :-

- 1) Creating
- 2) reading
- 3) writing
- 4) opening
- 5) closing
- 6) Renaming
- 7) Appending data to file
- 8) setting attribute.
- 9) getting attribute.

9] Types of file :-

- 1) Regular file
- 2) Directories
- 3) characters special files.
- 4) Block special files.

10] file Access Methods :-

- 1) The Pile
- 2) sequential file
- 3) Indexed file
- 4) Hashed (Direct) file.
- 5) Indexed sequential file.

11] Types of Directories :-

- 1) Flat directory
- 2) Hierarchical directory

12] Types of path :-

- 1) Absolute path
- 2) Relative path.

13] Tech of alloc of disk Blocks :-

- 1) Contiguous allocation.
- 2) Linked allocation.
- 3) Indexed allocation.

14] Method of Disk management

- 1) Linked list
- 2) Bit map.

NOTE :-

UNIT-6 notes are complete even you should prepare all questions of Unit-6.

* How to run lex & yacc program
* How to run java code & C-code.
* long form GCC compiler
* lex.yy.c - what is used of it.
* y.tab.h - what is use of it.
* long form atoi
* which s/w is used for java.
* How to install MS visual studio (VB)
* Lex & Yacc tool for window & Ubuntu.
* -d :- what is use of -d option.
* -ll - what is use of -ll parameter.
* stdio.h → Long form
*

*** BEST OF LUCK ***

Prepared By - Prof. Anand Gharu