OPERATING SYSTEMS

TAKE HOME ASSIGNMENT I

(Itish Agarwal
18CS30021)

Q 1. The instrudions that can only run in Kernel Mode are called Priviliged Instructions.

Whereas, the instructions that can run only in User Mode are called Non-Priviliged Instructions.

~~(a) Non Priviliged:~~

~~User Processes raise an Interrupt putting CPU to priviliged mode.~~

(a) Priviliged:

Otherwise a particular user process ~~to~~ may hog up the CPU resources for itself.

(b) Priviliged:

The process can change the values in the registers and create issues in memory allocation.

(c) Non- Priviliged:

The process requires to load ~~RTO~~ values in a CPU register to perform necessary computations.

(d) Priviliged:

Only OS should be allowed to do this otherwise a malicious user program may disrupt the whole system.

(e) Non-priviliged:

A user process cannot manipulate the I/O process by simply reading the status of an I/O device.

Q2.

→ In a scenario where many memory requests are not combined to a single one, & there will be context switch occuring at each syscall. This will lead to high context switching overhead (due to multiple context switches). This ~~leads~~ leads to delay execution.

→ If all the memory requests are completed in one go, then the process needs the CPU resources for computational tasks only. This speeds up the execution.

→ Further, memory accesses are quite slow (Von Neumann architecture). Thus if ~~the~~ we have less memory requests, ~~execute~~ execution ~~can~~ will be faster. Thus, it is beneficial to execute all requests in one go.

PTO

Q3. (a) CPU with double the speed is basically having the double clock speed. So, number of threads that can be allocated are doubled (Provided that memory doesn't cause constraint).

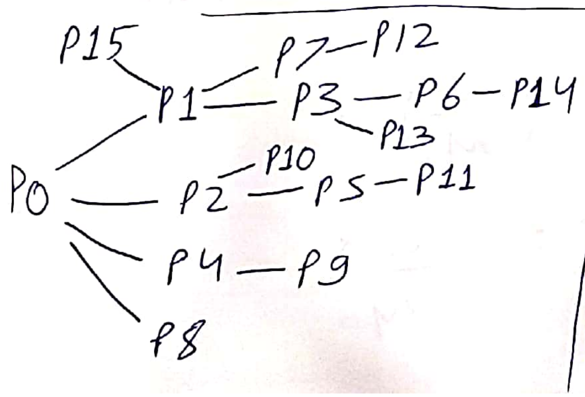
(b) This will not be beneficial as by expanding the main memory the speed will not improve but capacity to handle more threads will increase.

(c) It can be beneficial when I/O operations of large size are required by a process.

(d) It is the best option of the lot, because this means approx. double threads executed as well stored at the same time, hence approximately doubling the throughput.

Q4. At every fork call all the processes at a time will split into two (Child and the parent), hence doubling the number of processes. at each call.

∴ Total number of processes created $= 2^4 = \boxed{16}$.



Here,
Main Process ⇒ P0
Processes created by 1st fork: P1
" " by 2nd fork: P2, P3
" " by 3rd fork: P4, P5, P6, P7
Processes created by 4th fork: P8, P9, P10, P11, P12, P13, P14, P15

Q5. Difference between system call and exception:

System Call	Exception
→ Issued by user program	→ Issued by kernel (OS)
→ Syscall returns controll to user program	→ Exception does not return controll to user program.
→ Syscalls are intentional (ie, made according to will of user when user requires help of kernel).	→ These are unintentional (raised when something goes wrong).

Q6. (a) Let X be the optimal batch size.
Then,

$$\text{cost of serving time} = \frac{\left(\frac{S}{T}\right)}{X} = \frac{S}{TX}$$

Cost of waiting time per user

$$= \left(\frac{W}{M}\right) X$$

$$= \frac{WX}{M}$$

$\therefore$ Total cost $= \frac{S}{TX} + \frac{WX}{M} = f(X)$

To minimise $f(X)$, put $\frac{d f(x)}{dX} = 0$

$\Rightarrow \frac{-S}{Tx^2} + \frac{W}{M} = 0$

$\Rightarrow X = \sqrt{\frac{MS}{TW}}$

Also,

$\frac{d^2 f(x)}{dx^2} = \frac{+2S}{Tx^3} > 0 \left(at\ X = \sqrt{\frac{MS}{TW}} \right)$

$\therefore X = \sqrt{\frac{MS}{TW}}$ is a point of local minimum.

Hence proved

(b) We have, $M = 5$ mins
$T = 1$ min
$S = \$200/hr$
and $X = 50$.

Using $X = \sqrt{\frac{MS}{TW}}$, we have,

$W = \frac{MS}{TX^2}$

$$\therefore \quad W = \frac{MS}{TX^2}$$

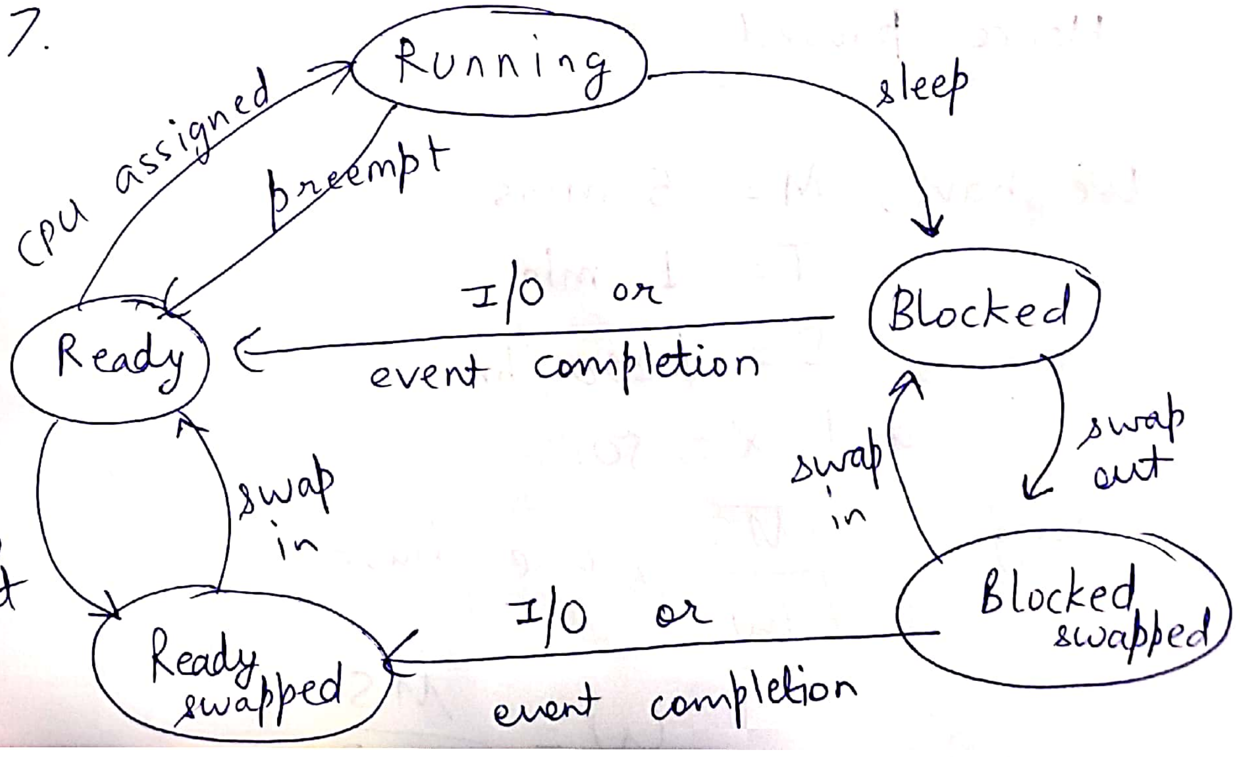
$$= \frac{(\cancel{5}\ \cancel{min})\ (\$\ 2\cancel{0}\cancel{0})}{(60\ min)\ (1\ \cancel{min})\ (\underset{5\cancel{0}\cancel{0}}{\cancel{2500}})}$$

$$= \frac{2}{5 \times 60}\ \$/min$$

$$= 0.0067\ \$/min$$

$$= \underline{\underline{0.4\ \$/hr}}$$

Q7.



NOW,

Ready to Ready Swapped:

There are large number of processes in their ready state, and this number is usually more than what the READY queue can hold. So some jobs are swapped out to READY-SWAPPED state processes.

Blocked to Blocked-swapped:

A large number of processes are waiting for an I/O or event completion ; thus some of them are swapped out to BLOCKED SWAPPED state.

Blocked to Ready:

When a process waiting for I/O or event completion gets the event completed, it is moved to READY state.

Ready-swapped to Ready:

When a process in READY SWAPPED state can be moved to READY queue, this is movement from READY SWAPPED to READY state

Blocked swapped to Ready swapped:

When a process waiting for I/O in Blocked swapped state, and when that I/O or event completes, process is moved to READY SWAPPED state.

Q8. PTO

Q8. Each process is represented in the operating system by a process control block (PCB). It contains many pieces of information associated with a specific process:

~~Process state: The counter indicates~~

→ Process state: The state may be new, ready, waiting, halted, etc.

→ Program counter: The counter indicates the address of next instruction to be executed for this process.

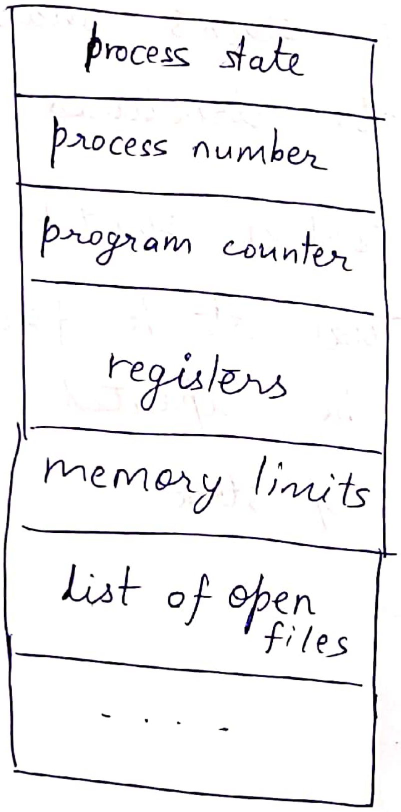
→ CPU registers: The registers vary in number and type, depending on computer architecture.

→ CPU-scheduling information: This information includes a process priority, pointers to scheduling queus, etc.

→ Memory-management information

→ Accounting information: This information includes the amount of CPU and real time used, time limits, etc.

→ <u>I/O status information</u> : Includes list of I/O devices allocated to the process, etc.



Process Control Block (PCB)

During a context switch, stack pointer must be updated, process state has to be changed, program counter has to be updated, and register values need to be received.

We have the following 2 design modifications:

I) Since some processes may not use registers, we can have a check bit that can signify whether we have to receive them or not.

II) Virtual stacks can be implemented such that all stacks share common memory using data structures like linked list, during context switch only top pointer of virtual stack for that new process has to be updated instead of individual elements of stack.

Q9.

CPU-bound processes:

→ search algorithms
→ Video streaming
→ mathematical calculations

I/O Bound Processes:

→ Copying/ Moving/ Transferring/Downloading files

Q10

SHORT TERM	MEDIUM TERM	LONG TERM
Fast speed compared to others	Medium speed compared to others	Lesser speed
~~Long term is~~ Short term is also known as CPU scheduler	Medium ~~Long~~ term is also Called swapping scheduler	Long term is known as job scheduler.
It is insignificant in the time-sharing order.	This scheduler is an element of time-sharing systems.	Either absent or minimal in a time-sharing system.
Offers less control	Reduce the level of multiprogramming.	Offers full control
It only selects processes that are in a ready state of execution.	It helps you to send processes back to memory.	Allows you to select processes from the loads and pool back into memory.

Q11. PTO

Q 11. <u>Code segment</u>:

```
int main ()  {

    int pip[2];   // used to store 2 ends of
                                         pipe
    char str[] = "football";
    if (pipe(pip) == -1) {
        exit (1);
        // no free descriptors or pip
        // arry is not valid.
    }
    pid pid1 = fork(); // creates a new child
                                       process
    if (pid1 < 0 )  {
        exit ( 1);
        // forking was unsuccessful

    }
    if (pid1 > 0 ) {
        // pid 1 is process id of child
        char answer [100];
        write ( pip[1], str, str(len) + 1);
        // write message to pipe
```

```
      close (pip[1]);
      // close end of pipe
       wait (NULL);
       // wait for child to execute
       read (pip[0], answer, 100);
        // read reply from child in 'answer'.
       printf ("Answer is : %s \n," answer);
        close (pip[0]); //close reading end
        close (pip[1]); //close writing end
   }
   else  {
       char answer[100] = "Done";
        char reply [100];
         read ( pip [0], reply, 100);
        //read message from parent 'reply'.
         printf ("Message is : %s \n", reply);
          ~~wa~~
          ~~wait~~
          write (pip [1], reply, 100);
           exit (0);
    }
}
```

Q12. These instructions are associated with the operating system. ~~the~~ Hence, they are typically at a specific location so that they can be loaded during boot.

Q13. Data structures suitable for linking of PCB's:

ⓐ Doubly linked list:

Advantages: → Easier implementation
→ Constant time insertion & deletion
→ Memory efficient

Disadvantages: → Search is slow, overhead might increase if next scheduled process's PCB is far away.

ⓑ Priority queue: (Heaps):

Advantages: → Suited for pre-emptive scheduling where priority matters.

Disadvantages: → Search takes logarithmic time. Overhead might increase if traversal leads to a child node.

(c) Dynamic Arrays:

Advantages: → Random access available

→ Constant time insertion & deletion

Disadvantages: → Many a times, large amount of memory may be allocated, leading to wastage or shortage.