ID - 180041120

Name - Md Farhan Ishmam

CSE 4501

Date - 20-Jun-2021

Dept - CSE

Ans. to Q.no. 2(a)

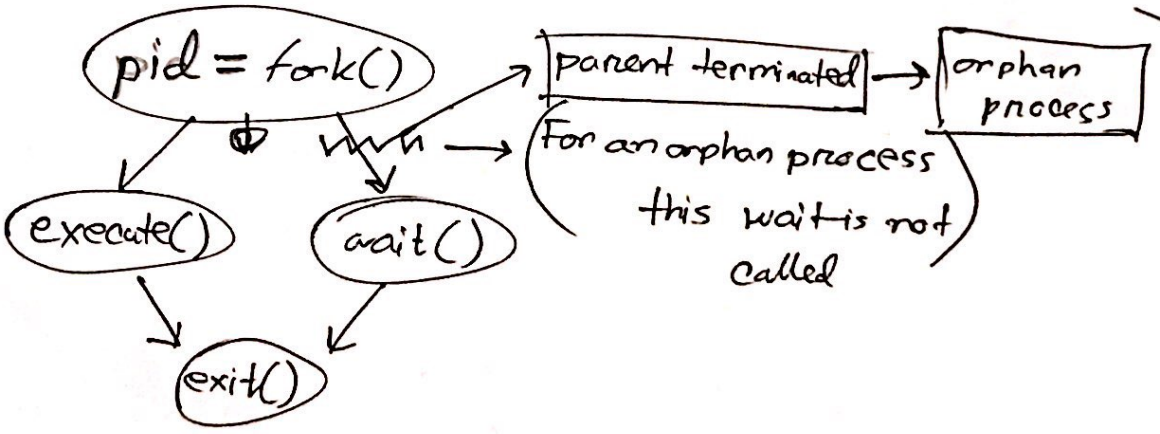
Zombie Process: A child process that has completed execution and waiting for the parent process to invoke wait() is called zombie process.

Orphan Process: A child process whose parent did not invoke wait() and got terminated due to some erroneous situation, then the child process is called an orphan process.

Every process transition to the state of orphan process after completing execution. After becoming a zombie process, the child process can exit ~~terminate~~

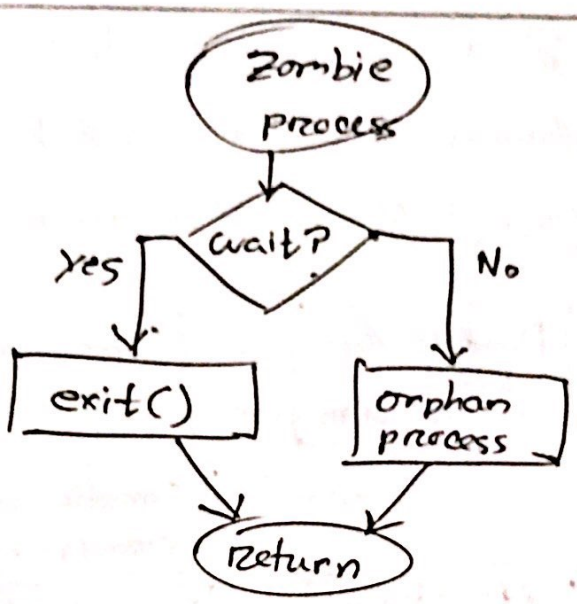
normally if the parent ~~zinko~~ invokes wait(). But for some reason if the parent gets terminated before doing it, only then, the ~~child~~ zombie process will become an orphan process. So, an orphan process is a special kind of zombie process. Thus at some point of the process cycle, an orphaned process remains a zombie process and we can say EVERY orphan process are zombie processes.

But it is possible for a zombie process to ~~terminate~~ exits normally when parent calls wait(). So, every zombie processes are NOT orphan processes.



For zombie process, the PID is still ~~in~~ before wait() as it is waiting for parent to invoke this. Then there are two options → i) exitting normally
ii) being an orphan process

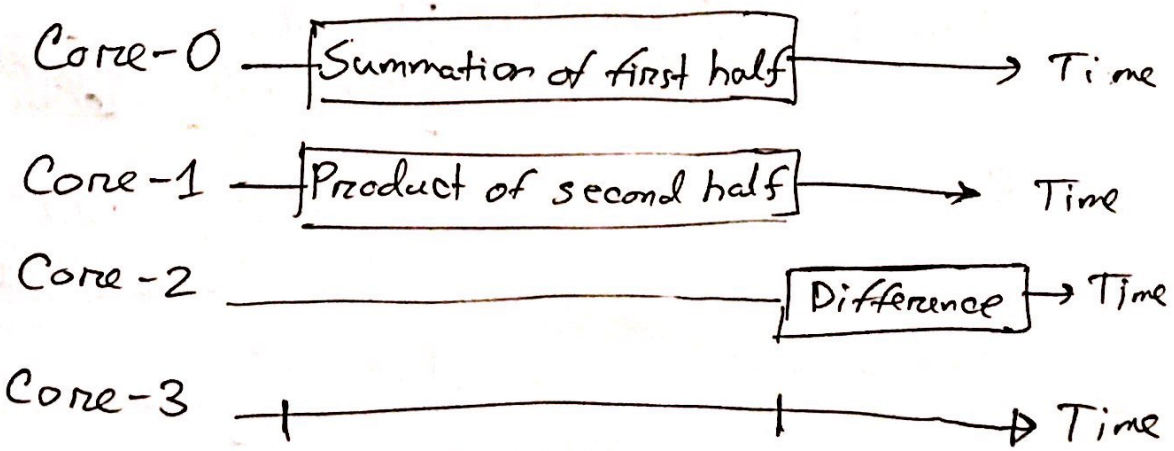
When (ii) is taken, the process becomes orphan process.



Ans. to Q.no. 2(b)

(i)

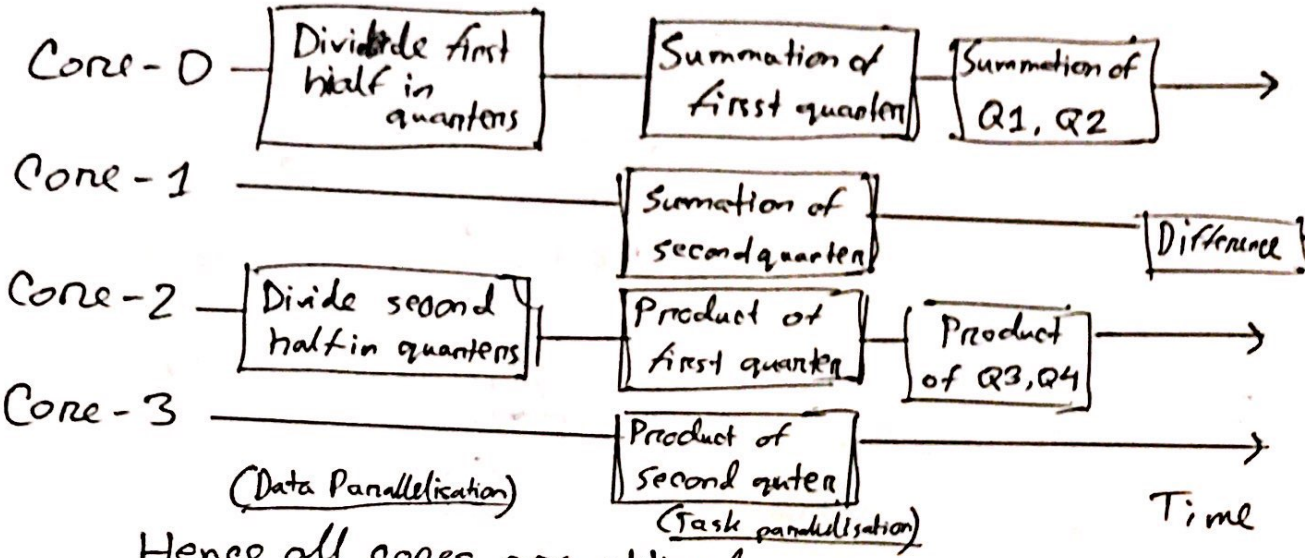
For scenario (i), we have 2 equal halves.
The first half will do summation and 2nd half do product. Then difference is found.
There are 4 cores.



(Not acceptable, 2 cores run in parallel)

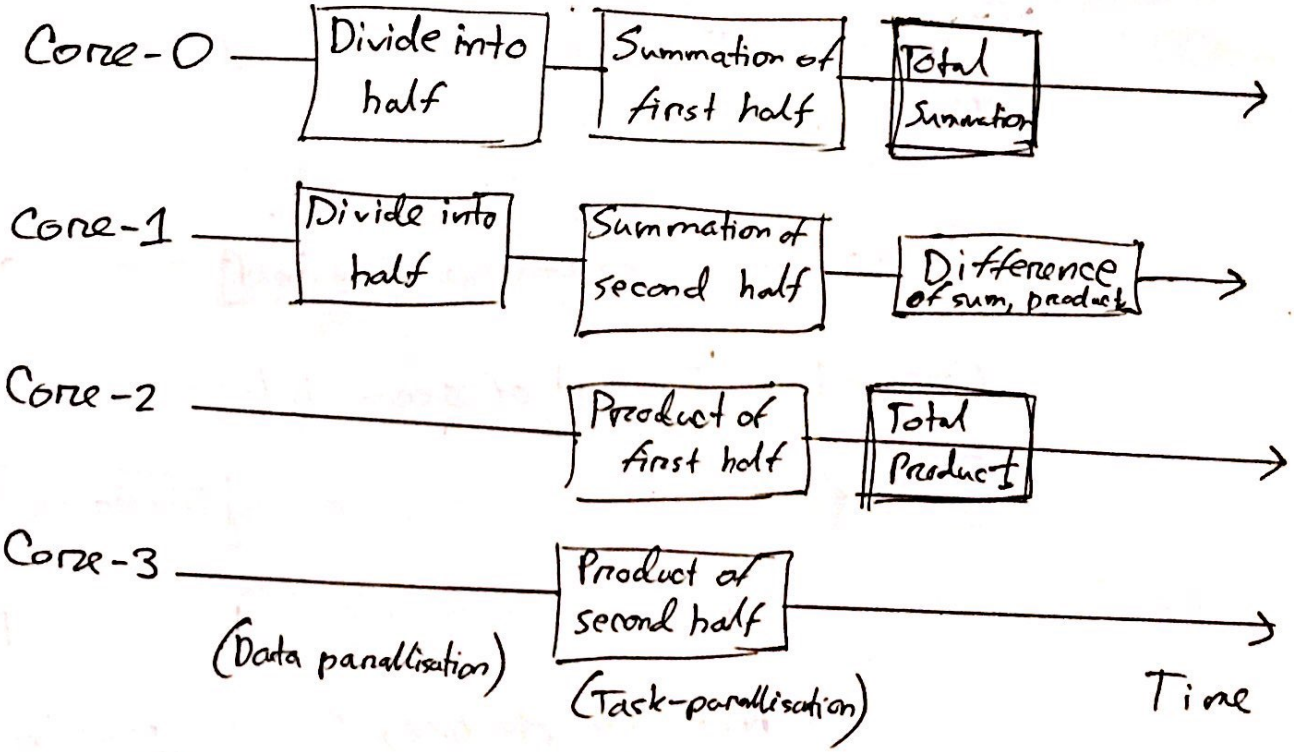
But here 2 cores aren't utilised.

Better option is 2 divide first half in two quarters and second half in 2 quarters.



Hence all cores are utilized.

For scenario (i),



All cores are utilised.

18004120

Ans to Qno 2(b)

(ii)

For scenario (i), the data parallisation is performed in the two equal halves, as we divide them in quarters. Then for summation and product, task parallisation is performed.

For scenario (ii), again data parallisation is performed as numbers are divided to halves and then we do task parallelisation in finding products and sums.

(iii)

In given scenario (i), we have 2 halves, which performs summation and products. So, for only 2 halves can be used by two cores, unless further data parallisation is performed. Then difference of summation and product is done. If S time is taken for serial work, then,

$$\text{Speedup} \leq \frac{1}{S + \frac{(1-S)}{N}} \quad \text{[Amdahl's law]}$$

Since, 2 cores are used, $N = 2$.

So, speedup will be

$$\text{speedup} \leq \frac{1}{S + \frac{(1-S)}{2}} = \frac{2}{S+1}$$

For scenario-(ii), again, the data is used for summation and product. Without dividing the data, core-0 can do summation and core-1 can do product.

Then speedup is ~~same~~ similiar. But here, the work done in parallel is twice than that of (i). Because for (i) it was $n/2$ numbers and for (ii) it is n numbers. So, speedup is

~~speedup $\leq \frac{1}{\frac{S}{2} + \frac{1-S}{N}}$ [N=2 as 2 cores can work pallelly]~~

~~speedup $\leq \frac{1}{\frac{2S}{2} + \frac{1-2S}{2}} = \frac{2}{2S+1}$~~ $2 \cdot 2S = 1$

So, parallel work is $(1-S) \times 2$

serial work is $S/2$

So, speedup is $\leq \frac{1}{S/2 + \frac{(1-S)}{2\times 2}}$ [N=2]

$\Rightarrow \frac{4}{2S+1-S}$

$= \frac{4}{S+1}$

So, scenario (ii) has 2 times of scenario (i)'s speedup.

Ans. to Q.no. 1(a)

When an interrupt occurs, the current state of the processes being executed are saved, and the CPU loads another process by loading its state, and this is called context switching. It's a CPU overhead.

A PCB or process control ~~board~~ block is used to ~~repre~~ represent the process by OS.

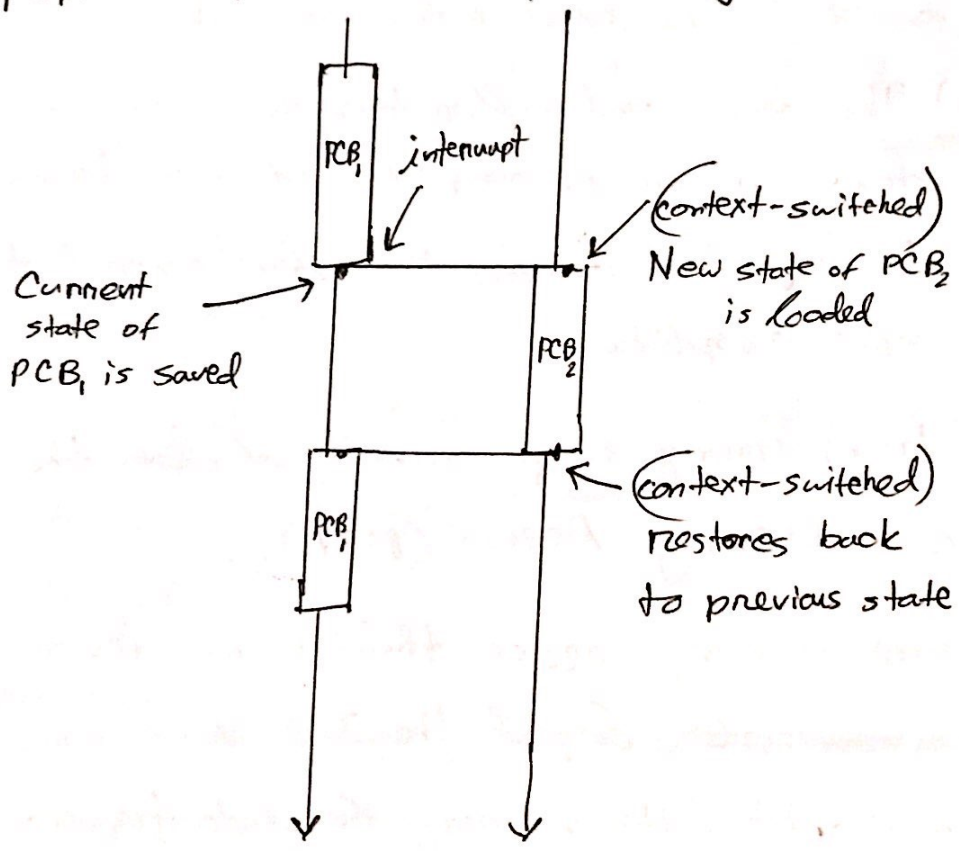


Fig:- Context Switching

180041120

Ans. to Q no. 1(b)

The steps of interrupt driven I/O is:

(i) The device driver loads the registers in the device controllers.

(ii) The device controller then determines the necessary steps to be taken based on the loaded register values.

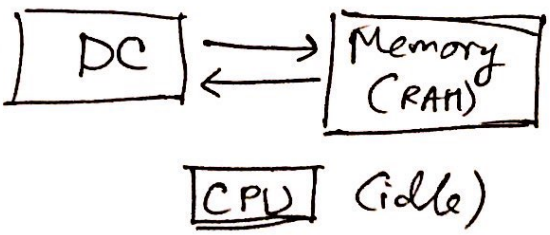
(iii) The device controller transfer data from the device to the local buffer of host.

(iv) The device controller then sends an interrupt to device driver for completion of data transfer.

(v) Finally, the driver transfers control to OS upon completion.

For transferring large amount of data, the alternate is Direct Memory Access (DMA).

In direct memory access, there is no device driver, ~~device controller~~. Instead the I/O device controller directly communicates with memory for data transfer. The CPU stays idle. When data transfer is complete, the control is transferred to OS.



The benefits are –

(i) Capable of transferring large amount of data in lower time i.e. more efficient.

(ii) Doesn't depend on device driver or CPU cycles.

(iii) Bypassing local buffer and driver saves further time.

Ans to Q.no. 1(c)

Graceful degradation: Over time, the performance of a machine decreases due to degradation of hardware components. It is called graceful degradation.

Ex- 2-year old CPU will perform worse than when it was new.

Thread pool: A collection of threads created at startup, which are activated upon request is called thread pool.

Thread cancellation: The procedure of terminating a thread before it is completed is called thread cancellation. Can be done using another thread or the same thread repeatedly checking.

Ex – when a website is stopped from loading, some threads are cancelled.

Ans. to Q.no 3(a)

~~created p-id is the~~

Line - A 0. -1 -4 -9 -16

Line - B 0 -1 -4 -9 -16

Line A will be executed in pid of child.
Because for child pid == 0.

Line B will be executed in parent space as pid > 0.

180091120

Ans. to Q no 3(b)

The differences between RPC and RMI is given below:

RPC	RMI
1) Stands for Remote Procedure Call.	1) Stands for Remote Method Invocation.
2) It is a standard OS feature.	2) It is a Java feature.
3) Allows only function calls on remote machine.	3) Allows invoking methods on remote machine.
4) Ordinary parameters	4) Objects can be passed as parameters.

RPC has 3 issues that must be ensured.

i) Difference in data representation
Server/client
→ ~~Sender/receiver~~, can use any of big or little endian for data representation. If the representation is not same, integrity will be lost. So, data is converted to XDR (external data representation).

ii) Semantics of the call

iii) Communication between server and client.

180041120

Ans. to Q.no.3(c)

I/O bound process: A process that spends more time doing I/O operation.

CPU bound process: A process that spends more time doing computational work.

If most processes are I/O bound, the ready queue will be usually empty. Thus short term scheduler will stay idle. Again, for more CPU bound process, the ready queue will be filled but I/O queue will usually be empty. So, a combination of both of desirable.

The middle term scheduler ensures a mix of both kinds of processes. It uses swapping method, to take out processes from memory and introduce other processes.

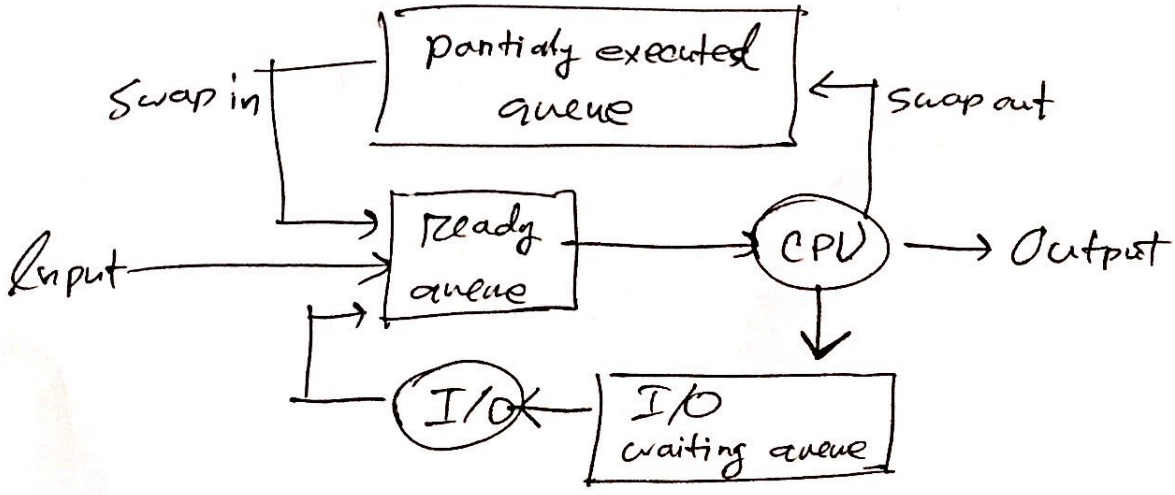


Fig: Mid term scheduler