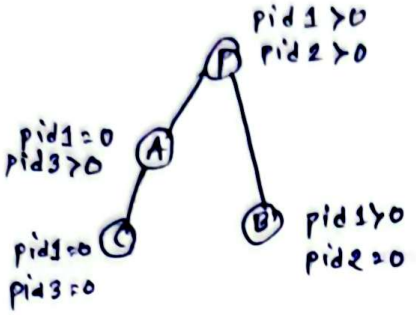
OS 251 Mid Solution

1(a) [2 marks] The operating system scheduler schedules a process and let it run for a time-slice. After that it deschedules a process and schedules another process to run. Thus there is a continuous loop b/w Ready and Running State.

1(b) [6 marks]



[2 marks]

this order is fixed

Output

- Horcrux destroyed: Slytherin's Ring (B)
- At least one Horcrux gone! (P)

this order is fixed

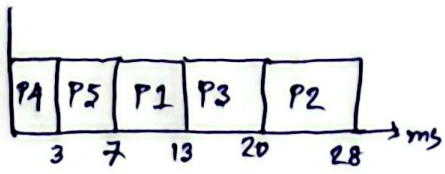
- Horcrux destroyed: The Riddle's Diary! (A)
- Horcrux destroyed: Hufflepuff's cup (C)

1(c) Data segment → Global and static variables that have been initialized.
Stack segment → Storing local variables, function arguments, return addresses etc.

(2) **2(a)** Caching is used to store frequently used data. This component is faster than RAM but slower than register and can hold more data than register.

(2) **2(b)** Interrupt driven I/O is generally better, especially when dealing with frequent or high-speed events. Polling involves the CPU constantly checking for events, consuming CPU cycles even when nothing is happening.

(3) **3(a)**

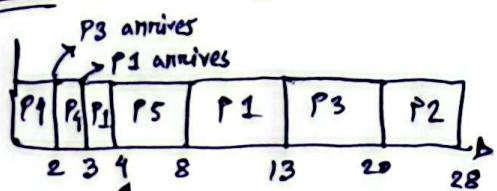


Turnaround time for each process

$$= \frac{3+7+13+20+28}{5} = 14.2 \text{ ms}$$

If individual turn around time is calculated, that is also accepted

3(b) [3 marks]



Avg. ~~turnaround~~ response time = $\frac{(3-3)+(20-0)+(13-2)+0+0}{5}$

$= 6.2$ ms

4(a) [3 marks]

Output: thread 0
thread 2
thread 1

4(b) [3 marks]

~~problems~~ same conditional variable used in both producers & consumers. Ambiguity in providing signal

if condition is used before waiting. Need to use while loop

4(c)

```
void *guest(void *args){
    lock(m);
    guest_count++;
    if(guest_count == N)
    { signal(cv_host);}
    wait(cv_guest, m);
    enterHouse();
    signal(cv_guest);
    unlock(m);
}
```