OSSP Odd Sem 2023-24

T1 Solution

Q1.

a) [3 marks Total]

Consider the following CPU instructions, which can be found in modern CPU architectures such as x86. For each instruction, indicate whether you believe it is privileged or unprivileged, and defend your answer. If your response is "privileged," for example, provide a one-sentence example of what might go wrong if this command were executed in an unprivileged environment. If you answered "unprivileged," explain in one sentence why it is safe/necessary to execute the instruction in unprivileged mode.

(i) Write instruction to the interrupt descriptor table register.

(ii) A write instruction to a general-purpose CPU register.

Ans a): Privileged for both the instructions, because a user process may misuse this ability to redirect interrupts of other processes. **[1 marks for exact answer for each instructions and 0.5 marks for explanation for each instructions]**

Q1. b) two direct children of P are created and one other descendant of P are created
[1.5 for direct children, 1 mark for descendent, 0.5 mark for explanation or graph]

Q2. [Total Marks=3; 2 marks for count value and 1 mark for explanation]

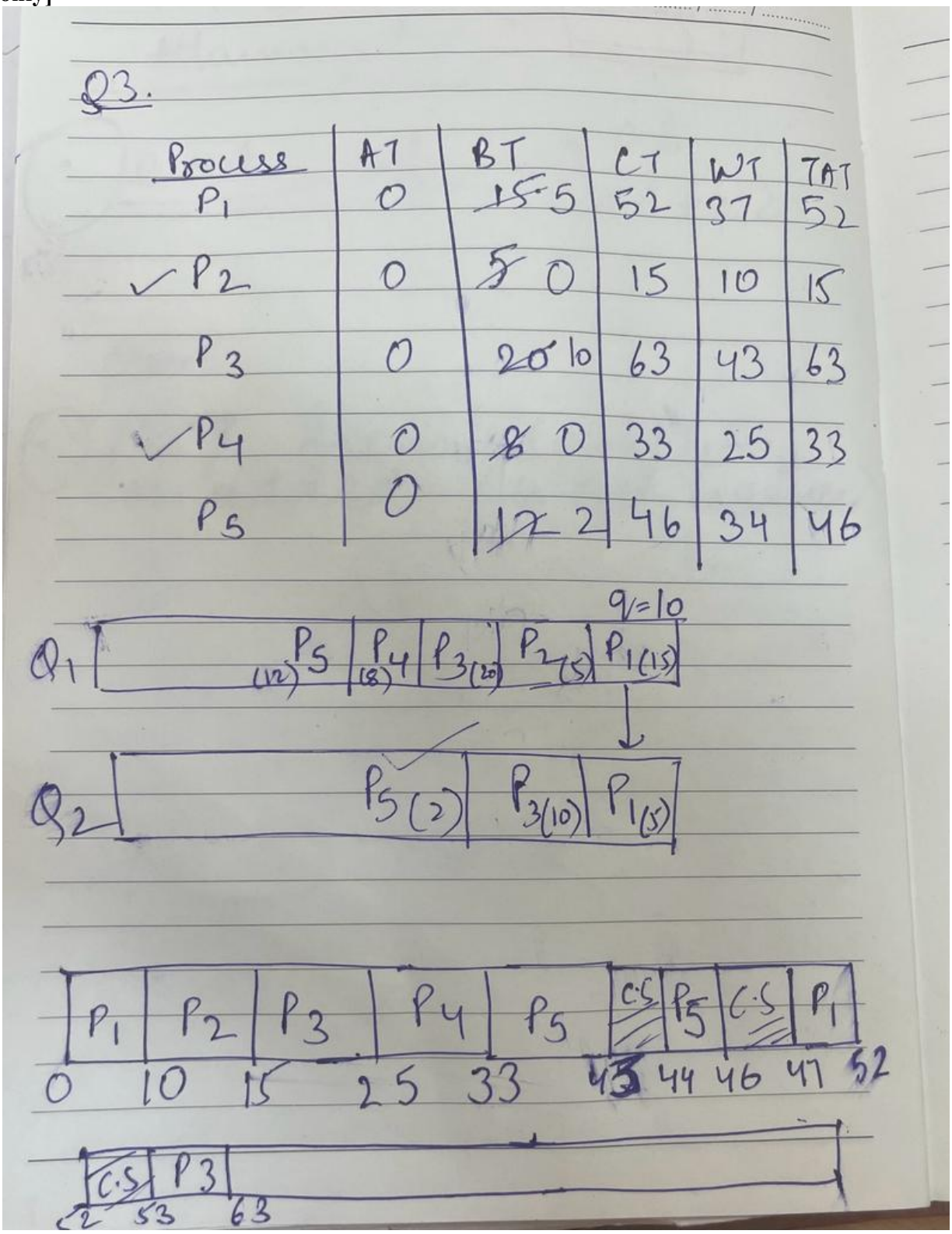
Consider a parent process that has forked a child in the code snippet below.

```
int count = 0;
ret = fork();
if(ret == 0)
{ printf("count in child=%d\n", count);
}
Else
{ count = 1;
}
```

The parent executes the statement ”count = 1” before the child executes for the first time. Now, what is the value of count printed by the code above?

Ans: 0 (the child has its own copy of the variable)

Q.3 [5 marks for fully correct, for steps marking maximum you can give 1-2 marks only]



Q4. a) What output do the following program produce and why? **[3 marks total]**

```
#include<stdio.h>
int counter;
static void * thread_func(void * _tn)
 {
int i;
 for (i = 0; i < 100000; i++)
counter++;
return NULL;
 }
 int main()
 {
int i, N = 5;
pthread_t t[N];
for (i = 0; i < N; i++)
pthread_create(&t[i], NULL, thread_func, NULL);
 for (i = 0; i < N; i++)
 pthread_join(t[i], NULL);
printf("%d\n", counter);
return 0;
}
```

Ans:

The output is not deterministic – 5 threads are accessing a shared but unprotected variable concurrently.

Q4. b) [3 marks total]

Suppose there are two threads executing simultaneously. One is printing numbers from 1 to 1000000 and another is printing strings with numbers as characters from 1 to 1000000. Which thread is going to complete its execution first.

Ans: 2nd thread completes its execution first **(Strict marking either 3 or 0)**