Read all of the following information before starting the exam:

- Please keep your written answers brief; be clear and to the point. You do not need to fill the whole space provided for answers.
- This test has 6 problems and 9 pages (including the cover and a blank page). Make sure that you have all of the pages!
- This is a **closed book** exam. No additional material is allowed.
- Good luck!

Question	**Marks**	**Score**
1	12	
2	12	
3	12	
4	12	
5	14	
6	18	
Total	**80**	

Problem 1. (*12 points*)

a. (*4 pts*) What resources does a thread share with other threads of the same process?

b. (*4 pts*) What resources are associated with each thread that it does not share with other threads (of the same process)?

c. (*4 pts*) Give **two** examples of how a thread can go directly from *running* state to *ready* state.

Problem 2. (*12 points*)

a. (*3 pts*) What are the requirements that must be satisfied for shared access to a *critical section*?

b. (*4 pts*) Consider the following solution to the critical section problem involving only two threads. Would it work? Explain your answer.

```
/* shared variable */
boolean flag[2]              /* Shared. Initially false */

...

flag[i] = true;              /* Declare intent to access */
while (flag[1-i]) {}         /* Busy wait */

  < Critical Section >

flag[i] = false;
```

c. (*5 pts*) Suppose we have the following two techniques to implement mutual exclusion:

i Disabling Interrupts

ii Special hardware instruction (e.g. test-and-set)

State **<u>one</u>** advantage of each over the other. Which one is used by OS/161?

Problem 3. (*12 points*)

a. (*4 pts*) There are only two processes in the system P_A and P_B containing one thread each. Assume that P_A is running and P_B is on the ready to run queue. Describe the steps required to save and restore or manipulate the processes context (including MMU) if P_A performs a non-blocking system call and it has not used up all of its quantum. Be sure to explain where the context is saved to and restored from. Provide a general description, you dont need to give specific register names.

b. (*8 pts*) There are only two processes in the system P_C and P_D containing one thread each. Assume that P_C is running and P_D is on the ready to run queue. Describe the steps required to save and restore or manipulate the processes context (including MMU) if P_C performs a blocking system call and it has not used up all of its quantum. Be sure to explain where the context is saved to and restored from. Provide a general description, you dont need to give specific register names.

Problem 4. (*12 points*)

a. (*2 pts*) Why is it important to have a separate kernel stack for each thread?

b. (*3 pts*) List three ways in which execution can switch from user space to kernel space.

c. (*3 pts*) Consider the following factors:

- Size of the page table
- Internal fragmentation
- I/O overhead

Which of these factors may argue for a large page size and which could be used to argue for a smaller page size? Explain (**one sentence each**).

d. (*4 pts*) Explain in brief how OS uses the *timer interrupt* for scheduling.

Problem 5. (*14 points*) Consider a virtual memory system that uses paging. Virtual and physical addresses are both 32 bits long, and the page size is 4KB = 2^{12} bytes. We have a TLB that can hold **eight** entries and during execution of a process P_1, it has the following entries.

Virtual Page#	Physical Frame#	Valid	Dirty
AC	2CD	1	1
DA	5DF	1	1
0	1F	1	0
29	6C1	0	1
21	32	0	0

Note: All numbers in this question are in **hexadecimal**. Also consider the *Dirty* bit has the same meaning as in an OS/161 TLB entry.

a. (*8 pts*) If possible, explain how the MMU will translate the following virtual addresses into physical address. If it is not possible, explain what will happen and why. **Show and explain how you derived your answer.**

Load from virtual address = 0x000A CF91.

Store at virtual address = 0x0000 03F8.

b. (*6 pts*) Now consider we have the same entries in the TLB and the following is part of the page table for process P_1.

	Physical Frame#	Read/Write
...	...	...
1	3D2	1
...	...	...
13	1F6	1
...	...	...
132	259	1
...	...	...
1328	981	1
...	...	...
13289	FFC	1
...	...	...

Note: Consider the *Read/Write* flag in page table entry has the same meaning as the *Dirty* flag in the TLB entry.

If possible, explain how the MMU will translate the following virtual addresses into physical address. If it is not possible, explain what will happen and why. **Show and explain how you derived your answer.**

Load from virtual address = 0x0001 3289.

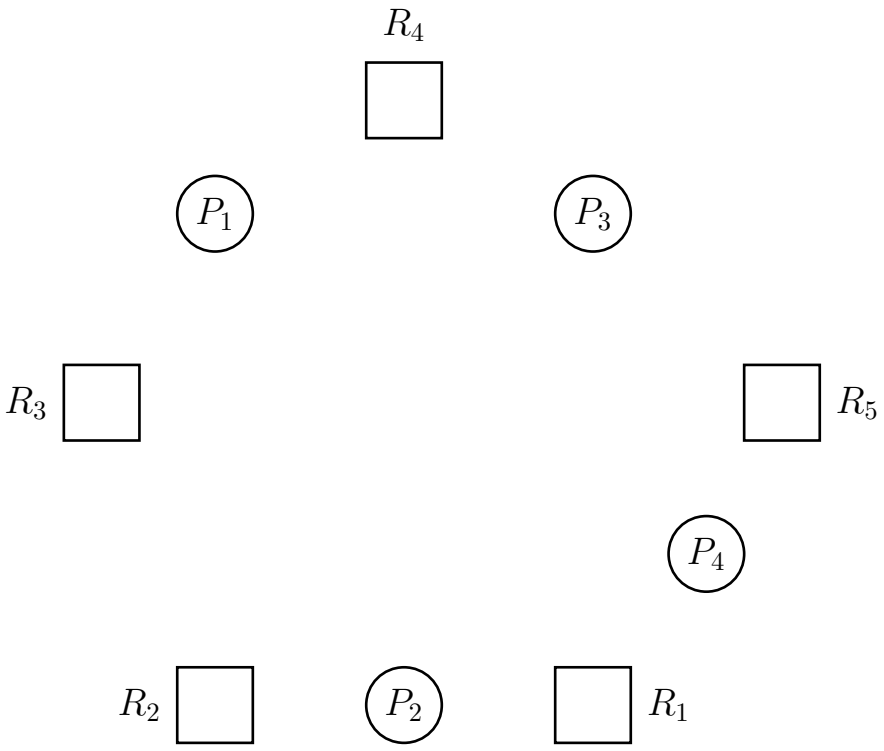
Problem 6. (*18 points*) This question uses the following notation (as used in the course notes) to describe resource allocation in a computer system:

- D_i : demand vector for process P_i
- A_i : current allocation vector for process P_i
- U : unallocated (available) resource vector

Given the scenario below, fill in the details for the resource allocation graph, **indicate if the system is deadlocked, and justify your answer**.

PLEASE USE SOLID LINES WITH ARROWS FOR ALLOCATION EDGES AND DASHED OR DOTTED LINES WITH ARROWS FOR REQUEST EDGES.

$U = (0, 0, 0, 0, 1)$
$D_1 = (0, 1, 0, 0, 1), D_2 = (0, 0, 1, 0, 1), D_3 = (0, 0, 0, 0, 1), D_4 = (0, 0, 0, 0, 1)$
$A_1 = (1, 0, 1, 1, 0), A_2 = (1, 1, 0, 0, 0), A_3 = (0, 0, 0, 1, 0), A_4 = (1, 0, 0, 0, 0).$



Space for **Problem 6**. Please mention if you answer other questions here.