IT5002 Computer Systems and Applications 2425 Sem 1 26 Nov 2024

Your Name: _______________ **Your ID:** _______________

# of Questions: 14

Date and Time of Exam Creation: Mon, Nov 25, 2024 @ 11:55:16

Notices:

1. On-campus Exam: Duration &gt; 1 HR

NATIONAL UNIVERSITY OF SINGAPORE

IT5002 - Computer Systems and Organizaiton

Semester 1 : AY 2024/25

Time Allowed: 120 MINUTES

INSTRUCTIONS TO STUDENTS

1. This assessment contains **FIVE** questions.

2. Please answer **ALL** questions.

3. This is a **SECURE** assessment.

4. You are allowed one A4 cheat sheet printed on both sides.

5. You are allowed to have a normal, scientific, business or graphing calculator, in addition to the laptop you are using to answer this assessment.

6. You are not permitted to leave the venue in the first hour of the assessment as well as in the last 15 minutes.

7. Please place your student card or identification document (NRIC, driving license, etc.) on the top right hand corner of your desk.

8. Please switch off your personal devices with communication features and leave it on the floor next to your desk at all times.

9. If you wish to communicate with an invigilator, go to the washroom, or leave before the end of the assessment, please raise your hand to inform the invigilator.

10. You will be liable for disciplinary action which may result in expulsion from the University if you are found to have contravened any of the clauses below,

•Violation of the NUS Code of Student Conduct (in particular the part on Academic, Professional and Personal Integrity), NUS IT Acceptable Use Policy or NUS Examination rules.

•Possession of unauthorized materials/electronic devices.

•Bringing your mobile phone or any storage/communication device with you to the washroom.

•Unauthorized communication e.g. with another student.

•Reproduction of any exam materials outside of the exam venue.

•Photography or videography within the exam venue.

•Plagiarism, giving or receiving unauthorised assistance in academic work, or other forms of academic dishonesty.

11. Once you have completed the assessment,

•Click on the "Exam Controls" button and choose "Submit Exam".

•Check off “I am ready to exit my exam” and click on "Exit" to upload your answers to the server.

•You will see a green confirmation window on your screen when the upload is successful. **Please keep this window on your screen.**

•If you do not see a green window, please disconnect and reconnect your WIFI and try again.

•Please be reminded that it is your responsibility to ensure that you have uploaded your answers to the Software.

•Please remain seated quietly in the meantime.

12. By clicking on the "Continue" button below, I acknowledge that I have read and fully understood this Exam Notice.

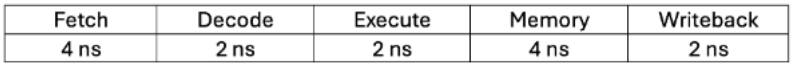
Thank You

I acknowledge that I have read this notice __

Total Exam Points: 70.00

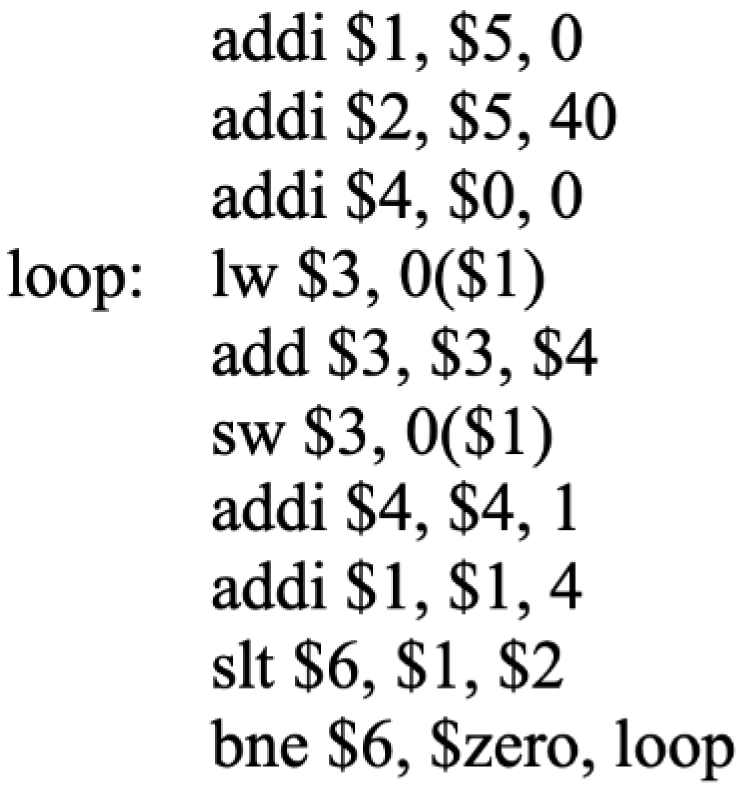
Question #: 1

In this question we assume the stages with the following timings in nanoseconds (1 ns =10^{-9} seconds)



Fetch	Decode	Execute	Memory	Writeback
4 ns	2 ns	2 ns	4 ns	2 ns

Consider the following program. Register $5 contains the base address of a 10-element array A:



```
         addi $1, $5, 0
         addi $2, $5, 40
         addi $4, $0, 0
loop:    lw $3, 0($1)
         add $3, $3, $4
         sw $3, 0($1)
         addi $4, $4, 1
         addi $1, $1, 4
         slt $6, $1, $2
         bne $6, $zero, loop
```

Q1a. (2 marks out of 14) How many instructions are executed by this program?

Answer: ___1___ instructions

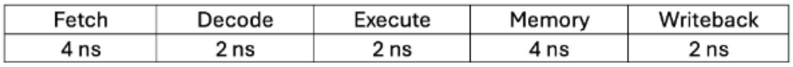
Q1b. (2 marks out of 14) In a single-cycle implementation, how long does it take to execute this programme in ns?

Answer: ___2___ ns

Item Weight: 4.0

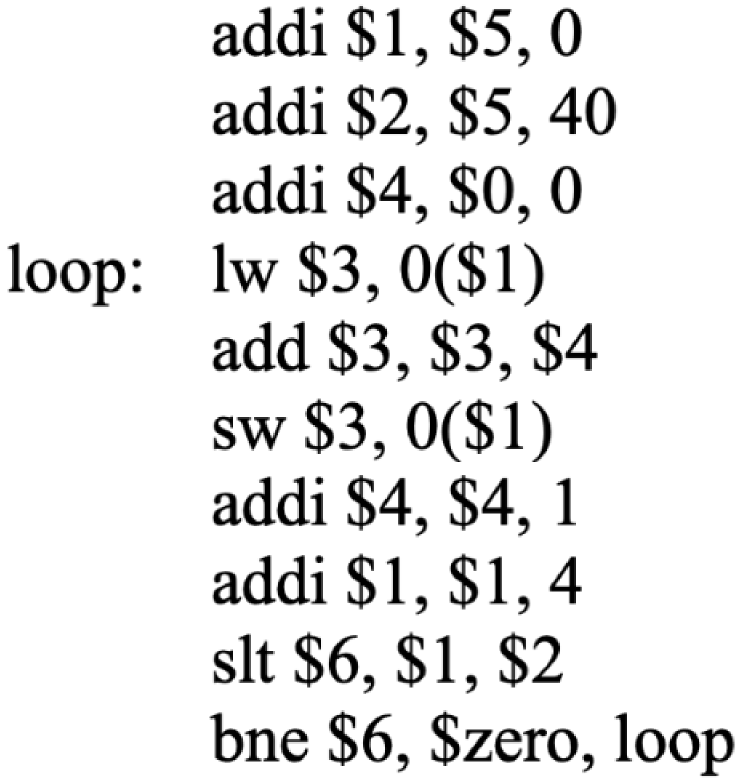
Question #: 2

In this question we assume the stages with the following timings in nanoseconds (1 ns =10^{-9} seconds)



Fetch	Decode	Execute	Memory	Writeback
4 ns	2 ns	2 ns	4 ns	2 ns

Consider the following program. Register $5 contains the base address of a ten-element array A.



```
       addi $1, $5, 0
       addi $2, $5, 40
       addi $4, $0, 0
loop:  lw $3, 0($1)
       add $3, $3, $4
       sw $3, 0($1)
       addi $4, $4, 1
       addi $1, $1, 4
       slt $6, $1, $2
       bne $6, $zero, loop
```

Q1c (1 mark of 14): What is the clock frequency of our single-cycle implementation? State your answer in MHz (1 MHz =10^6 Hz)

Answer: 1 MHz

Consider now a multi-cycle implementation, where instructions can skip stages it does not need.

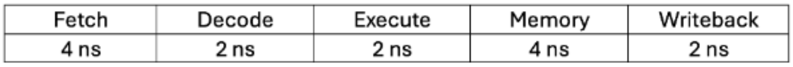
Q1d (1 mark of 14): What is the clock frequency of our multi-cycle implementation?

Answer: 2 MHz

Item Weight: 2.0

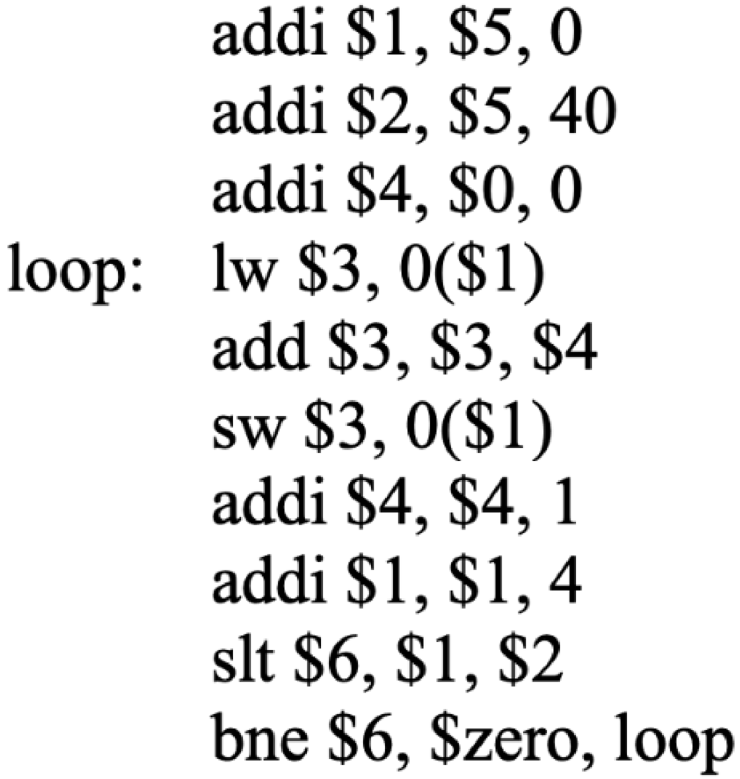
Question #: 3

In this question we assume the stages with the following timings in nanoseconds (1 ns =10^{-9} seconds)



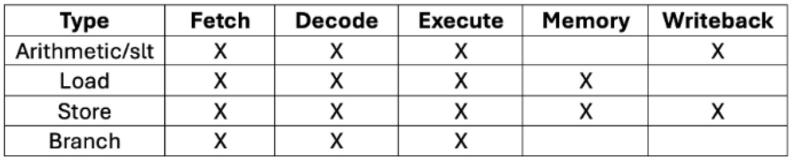
Fetch	Decode	Execute	Memory	Writeback
4 ns	2 ns	2 ns	4 ns	2 ns

Consider the following program. Register $5 contains the base address of a ten-element array A.



```
       addi $1, $5, 0
       addi $2, $5, 40
       addi $4, $0, 0
loop:  lw $3, 0($1)
       add $3, $3, $4
       sw $3, 0($1)
       addi $4, $4, 1
       addi $1, $1, 4
       slt $6, $1, $2
       bne $6, $zero, loop
```

Q1e (4 marks of 14): How long does it take to execute the programme above? To help you, the table below shows the stages used by various types of instructions:



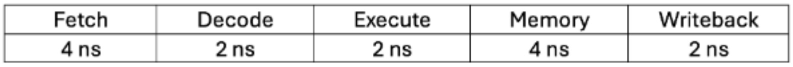
Type	Fetch	Decode	Execute	Memory	Writeback
Arithmetic/slt	X	X	X		X
Load	X	X	X	X	
Store	X	X	X	X	X
Branch	X	X	X		

Answer: 1 ns

Item Weight: 4.0

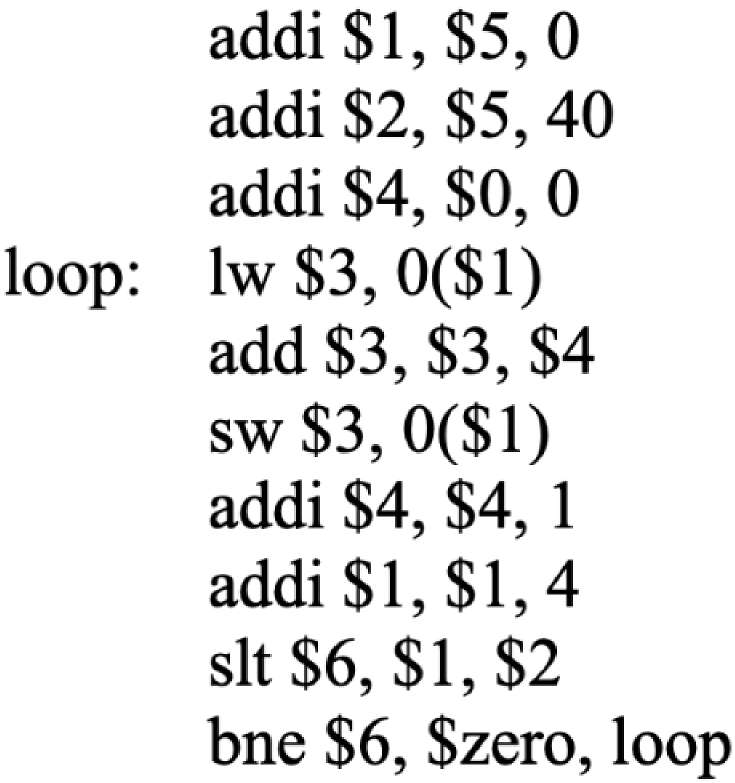
Question #: 4

In this question we assume the stages with the following timings in nanoseconds (1 ns =10^{-9} seconds)



Fetch	Decode	Execute	Memory	Writeback
4 ns	2 ns	2 ns	4 ns	2 ns

Consider the following program. Register $5 contains the base address of a ten-element array A.



```
       addi $1, $5, 0
       addi $2, $5, 40
       addi $4, $0, 0
loop:  lw $3, 0($1)
       add $3, $3, $4
       sw $3, 0($1)
       addi $4, $4, 1
       addi $1, $1, 4
       slt $6, $1, $2
       bne $6, $zero, loop
```

Now consider an ideal pipelined implementation, with no data dependencies, branch hazards or pipeline register latencies.

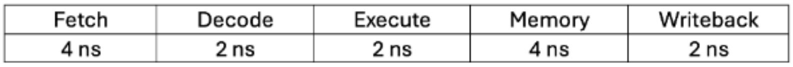
Q1f (1 mark of 14): What is the frequency of our pipelined CPU? As before state your answer in MHz (1 MHz =10^6 Hz)

Answer: 1 MHz

Item Weight: 1.0

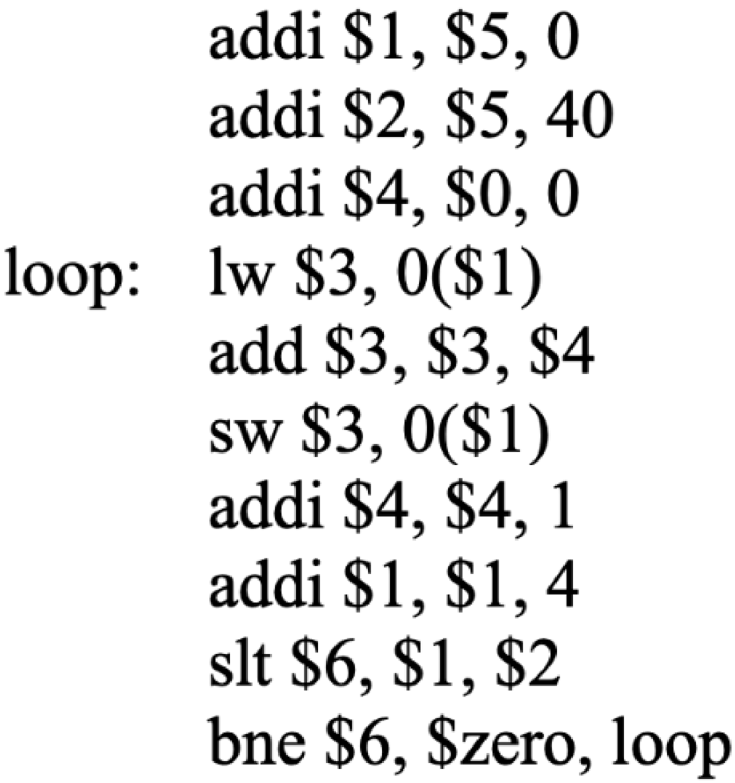
Question #: 5

In this question we assume the stages with the following timings in nanoseconds (1 ns =10^{-9} seconds)



Fetch	Decode	Execute	Memory	Writeback
4 ns	2 ns	2 ns	4 ns	2 ns

Consider the following program. Register $5 contains the base address of a ten-element array A.



```
        addi $1, $5, 0
        addi $2, $5, 40
        addi $4, $0, 0
loop:   lw $3, 0($1)
        add $3, $3, $4
        sw $3, 0($1)
        addi $4, $4, 1
        addi $1, $1, 4
        slt $6, $1, $2
        bne $6, $zero, loop
```

Now consider an ideal pipelined implementation, with no data dependencies, branch hazards or pipeline register latencies.

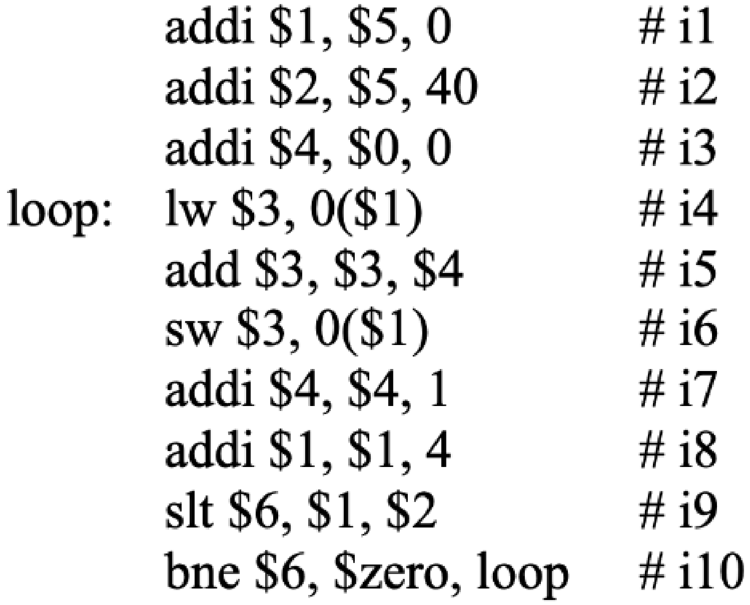
Q1g (3 marks of 14): How long does it take to execute the program above on an IDEAL pipelined CPU? State your answer in ns. (1 ns =10^{-9} s)

Answer: ___1___ ns

Item Weight: 3.0

Question #: 6

Consider again the same programme as in the previous question, this time with the instructions labeled from i1 to i10. You can use these labels to help you work out the answers.



```
       addi $1, $5, 0           # i1
       addi $2, $5, 40          # i2
       addi $4, $0, 0           # i3
loop:  lw $3, 0($1)             # i4
       add $3, $3, $4           # i5
       sw $3, 0($1)             # i6
       addi $4, $4, 1           # i7
       addi $1, $1, 4           # i8
       slt $6, $1, $2           # i9
       bne $6, $zero, loop      # i10
```

Our system has 24-bit addresses and 32-bit words. It takes 50ns to read/write the main memory, and 2 ns to read/write the cache.

Assuming that the address of the first instruction is at **0x2C1D04**, and that $5 contains the value **0x10FC48**, answer the following questions:

Q2a (4 marks): Our instruction cache is a direct-mapped cache with 4 words per block. The total cache size is 64 bytes, excluding tag bits, etc. What is the hit-rate of our instruction cache for our programme above? Express your answer in percent, to two decimal places (e.g. 26.4%)

Answer: 1 %

Q2b (4 marks): Our data cache has 64 bytes in total as well, excluding tag bits etc. This is a 2-way set-associative cache with least-recently-used block replacement policy, and each block is 2 words long. What is the hit-rate of our data cache for our programme above, for both reading and writing? Assume a write-back cache with write-allocate miss policy. Express your answer in percent, to two decimal places (e.g. 26.4%)

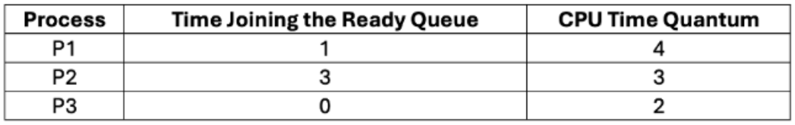
Answer: 2 %

Item Weight: 8.0

Question #: 7

A time-slicing system is one where processes are given a fixed time quantum to run, after which it is preempted and another process is picked to run for its time quantum.

We have three processes P1, P2 and P3 in a time-slicing system with the following characteristics:



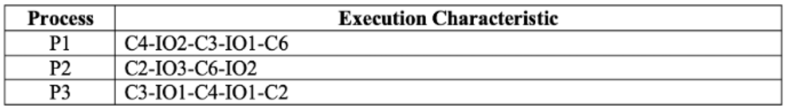
Process	Time Joining the Ready Queue	CPU Time Quantum
P1	1	4
P2	3	3
P3	0	2

"Time Joining the Ready Queue" shows the time in time units that the process first becomes ready to run. The CPU Time Quantum shows the amount of time units a process gets to run continuously on the CPU before it is pre-empted.

There is one I/O device on our system, and at most one process can use it at any point in time. Any other process that requires use of the I/O device must wait until the other process finishes using the I/O.

Processes are picked to run in the order that they join/re-join the ready queue.

The processes have the following execution characteristics:



Process	Execution Characteristic
P1	C4-IO2-C3-IO1-C6
P2	C2-IO3-C6-IO2
P3	C3-IO1-C4-IO1-C2

"Cn" means that the process uses the CPU for n time units and IOm means that the CPU uses the I/O for m time units. Hence C4-IO3-C2 means that the process runs on the CPU for 4 time units, does IO3 for 3 time units, then runs on the CPU for 2 time units.

A process that uses I/O before it uses its entire time quantum will rejoin the ready queue once it finishes I/O, and will be picked to continue using its remaining quantum when its turn comes.

Q3a (6 marks of 21): What is the first time step that each process is picked to run?

P1: 1

P2: 2

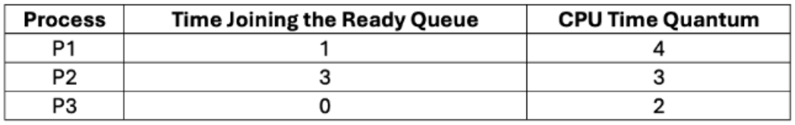
P3: 3

Item Weight: 6.0

Question #: 8

A time-slicing system is one where processes are given a fixed time quantum to run, after which it is preempted and another process is picked to run for its time quantum.

We have three processes P1, P2 and P3 in a time-slicing system with the following characteristics:



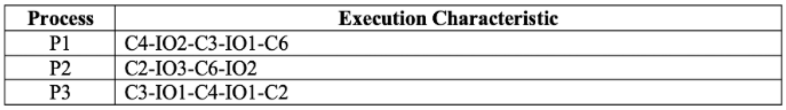
Process	Time Joining the Ready Queue	CPU Time Quantum
P1	1	4
P2	3	3
P3	0	2

"Time Joining the Ready Queue" shows the time in time units that the process first becomes ready to run. The CPU Time Quantum shows the amount of time units a process gets to run continuously on the CPU before it is pre-empted.

There is one I/O device on our system, and at most one process can use it at any point in time. Any other process that requires use of the I/O device must wait until the other process finishes using the I/O.

Processes are picked to run in the order that they join/re-join the ready queue.

The processes have the following execution characteristics:



Process	Execution Characteristic
P1	C4-IO2-C3-IO1-C6
P2	C2-IO3-C6-IO2
P3	C3-IO1-C4-IO1-C2

"Cn" means that the process uses the CPU for n time units and IOm means that the CPU uses the I/O for m time units. Hence C4-IO3-C2 means that the process runs on the CPU for 4 time units, does IO3 for 3 time units, then runs on the CPU for 2 time units.

A process that uses I/O before it uses its entire time quantum will rejoin the ready queue once it finishes I/O, and will be picked to continue using its remaining quantum when its turn comes.

Q3b (9 marks of 21): At what time step does each process finish?

P1: 1

P2: 2

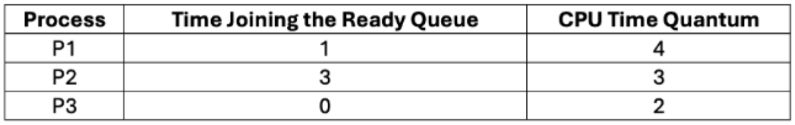
P3: 3

Item Weight: 9.0

Question #: 9

A time-slicing system is one where processes are given a fixed time quantum to run, after which it is preempted and another process is picked to run for its time quantum.

We have three processes P1, P2 and P3 in a time-slicing system with the following characteristics:



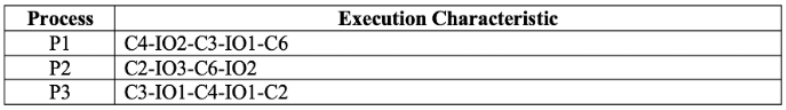
Process	Time Joining the Ready Queue	CPU Time Quantum
P1	1	4
P2	3	3
P3	0	2

“Time Joining the Ready Queue” shows the time in time units that the process first becomes ready to run. The CPU Time Quantum shows the amount of time units a process gets to run continuously on the CPU before it is pre-empted.

There is one I/O device on our system, and at most one process can use it at any point in time. Any other process that requires use of the I/O device must wait until the other process finishes using the I/O.

Processes are picked to run in the order that they join/re-join the ready queue.

The processes have the following execution characteristics:



Process	Execution Characteristic
P1	C4-IO2-C3-IO1-C6
P2	C2-IO3-C6-IO2
P3	C3-IO1-C4-IO1-C2

“Cn” means that the process uses the CPU for n time units and IOm means that the CPU uses the I/O for m time units. Hence C4-IO3-C2 means that the process runs on the CPU for 4 time units, does IO3 for 3 time units, then runs on the CPU for 2 time units.

A process that uses I/O before it uses its entire time quantum will rejoin the ready queue once it finishes I/O, and will be picked to continue using its remaining quantum when its turn comes.

Q3c (6 marks of 21): What is the turn-around-time (TAT) for each process? The TAT is defined to be the amount of time taken for a process to run to completion, from the time it is first ready to run.

P1: __1__

P2: __2__

P3: __3__

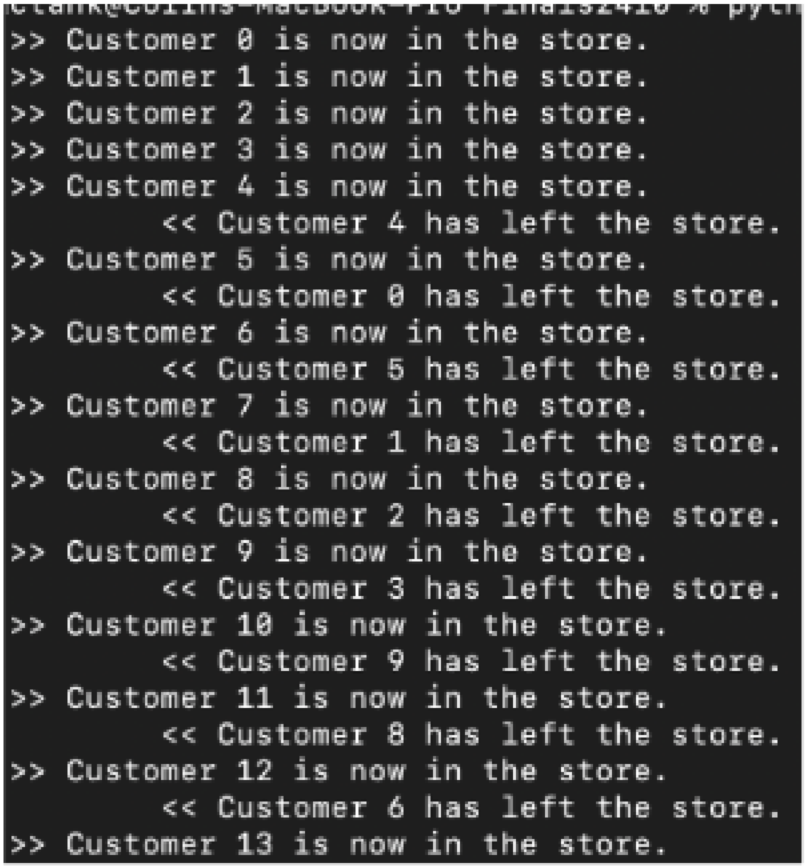
Item Weight: 6.0

Question #: 10

Back in 2020 to 2022 the world was undergoing one of the worst pandemics in recent memory, and venues were not allowed to hold more than a certain number of people.

In this question we are looking simulating a shop with a maximum capacity of 5 customers, on a day with 50 customers visiting. Customers spend between 2 and 7 time units in the store before leaving.

When running the output should look similar to this:



```
ctank@Collins-MacBook-Pro Final32410 % pyth
>> Customer 0 is now in the store.
>> Customer 1 is now in the store.
>> Customer 2 is now in the store.
>> Customer 3 is now in the store.
>> Customer 4 is now in the store.
        << Customer 4 has left the store.
>> Customer 5 is now in the store.
        << Customer 0 has left the store.
>> Customer 6 is now in the store.
        << Customer 5 has left the store.
>> Customer 7 is now in the store.
        << Customer 1 has left the store.
>> Customer 8 is now in the store.
        << Customer 2 has left the store.
>> Customer 9 is now in the store.
        << Customer 3 has left the store.
>> Customer 10 is now in the store.
        << Customer 9 has left the store.
>> Customer 11 is now in the store.
        << Customer 8 has left the store.
>> Customer 12 is now in the store.
        << Customer 6 has left the store.
>> Customer 13 is now in the store.
```

In python, the statement:

s =threading.Semaphore(n)

Creates a semaphore with an initial value of n. If a process executes s.acquire(), it will decrement s if s is larger than 0. Otherwise the process blocks.

If a process executes s.release(), if there are processes blocking on s, one will be unblocked. Otherwise s is incremented by 1.

Complete the code below by selecting the correct statements

```
import threading
import time
import random
___1___

def customerThread(custNum):

  ___2___
   print(">> Customer %d is now in the store." % custNum)
   time.sleep(random.randint(2, 7))
   ___3___

   print("\t<< Customer %d has left the store." % custNum)
def main():
  threadList =[]
  for i in range(0, numCustomers):
    ___4___
    ___5___
if __name__ =='__main__':
   main()
```

Item Weight: 10.0

Question #: 11

Q6a (6 marks of 17)

Consider a FAT32 partition that uses 28-bit indexes. Special index values are set aside to indicate FREE blocks, BAD blocks, and to indicate End-Of-File (EOF). Each 28-bit index is represented by a 32-bit value as shown below:

0000	28-bit index

i) How much memory will this FAT consume when loaded from disk? State your answers in GiB (1 GiB = 2^{30} bytes) rounded to two decimal places.

Answer: 1 GiB

ii) Assuming 16 KiB (1 KiB = 2^{10} bytes) blocks, what is the maximum size of this partition? State your answer in TiB (1 TiB = 2^{40} bytes) to 2 decimal places.

Answer: 2 TiB

Item Weight: 6.0

Question #: 12

Q6a. (1 mark) Consider a FAT32 partition that uses 28-bit indexes. Special index values are set aside to indicate FREE blocks, BAD blocks, and to indicate End-Of-File (EOF). Each 28-bit index is represented by a 32-bit value as shown below:

0000	28-bit index

iii) What is the maximum size of a file on this partition? State your answer in TiB (1 TiB = 2^{40} bytes) to 2 decimal places.

Answer: 1 TiB

Item Weight: 1.0

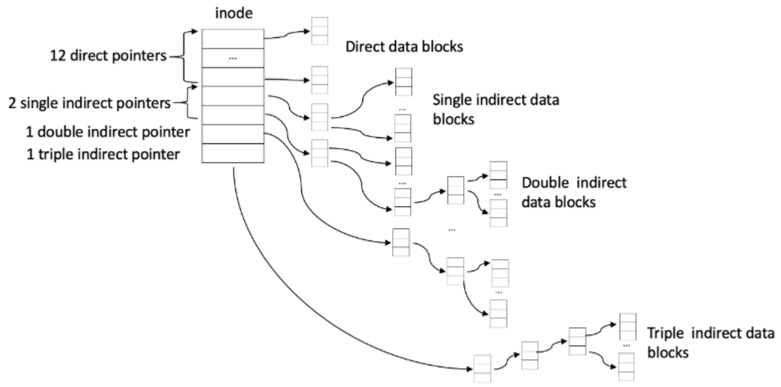
Question #: 13

Q5b (6 marks of 17)

b. We now consider a file system using inodes with the following table of contents (TOC) structure:

Number of direct pointers: 12
Number of single-level indirect pointers: 2
Number of double-level indirect pointers: 1
Number of triple-level indirect pointers: 1

This is shown in the following diagram:


inode
12 direct pointers
2 single indirect pointers
1 double indirect pointer
1 triple indirect pointer
...
Direct data blocks
Single indirect data blocks
Double indirect data blocks
Triple indirect data blocks

Further Information:

Block size: 4KiB (1 KiB = 2^{10} bytes)

Size of block index: 32 bits

The inode is the size of one block, though it does not fully use the block.

i) What is the maximum size of this partition? State your answer in TiB (1 TiB = 2^{40} bytes) to 2 decimal places.

Answer: 1 TiB

ii) What is the maximum size of a file on this partition? State your answer in TiB (1 TiB = 2^{40} bytes)

Answer: 2 TiB

Item Weight: 6.0

Question #: 14

b. We now consider a file system using inodes with the following table of contents (TOC) structure:

Number of direct pointers: 12
Number of single-level indirect pointers: 2
Number of double-level indirect pointers: 1
Number of triple-level indirect pointers: 1

This is shown in the following diagram:

1.
   Further Information:

Block size: 4KiB (1 KiB =2^{10} bytes)
Size of block index: 32 bits
The inode is the size of one block, though it does not fully use the block.

In the next two sub-questions we consider the case of loading a single integer from the largest file possible:

iii) In the best case, how much memory is consumed by the inodex/index blocks (do not count the data block), assuming that, if we need to load multiple index blocks, they are held in memory at the same time?
Answer: __1__ KiB
iv) In the worst case, how much memory is consumed by the inode/index blocks (do not count the data block), assuming that, if we need to load multiple index blocks, they are held in memory at the same time?

Answer: __2__ KiB

Item Weight: 4.0