CS3103 2021B midterm

Operating Systems (City University of Hong Kong)

CITY UNIVERSITY OF HONG KONG

Course code & title	:	CS3103 Operating Systems
Session	:	Semester B 2020/21
Time allowed	:	80 minutes

---

1. This paper consists of 2 sections (a total of 80 marks)
2. Answer ALL questions in both sections
3. Section A (32 marks): multiple choice and fill-in-the-blank
4. Section B (48 marks): long questions
5. Specify the Section and Question number clearly for EACH answer in the answer script.
6. Submit ONE pdf file of the answer script to Canvas.
7. Use your Student ID to name the pdf file.

---

*This is an **open-book** quiz.*

Only the course teaching materials are allowed.

***No** access to the Internet and other materials.*

Candidates are allowed to use the following materials/aids:

Approved Calculator

Copy-and-paste the following honesty pledge in the beginning of the answer script and sign your student ID and name to reaffirm your academic honesty pledge.

"I pledge that the answers in this exam/quiz are my own and that I will not seek or obtain an unfair advantage in producing these answers. Specifically,

- ***I will not plagiarize (copy without citation) from any source;***
- ***I will not communicate or attempt to communicate with any other person during the exam/quiz; neither will I give or attempt to give assistance to another student taking the exam/quiz; and***
- ***I will use only approved devices (e.g., calculators) and/or approved device models.***

I understand that any act of academic dishonesty can lead to disciplinary action."

Student ID: _____

Student Name: _____

Section A (32 marks)

Attempt ALL questions from this Section. In case of the fill-in-the-blank questions, abbreviation or short form will NOT be accepted.

1. [2 marks]
At the beginning of each instruction cycle, the processor fetches the instruction whose address is stored in (i)_____ and the fetched instruction is loaded into (ii)_____.
a) accumulator
b) arithmetic and logic unit
c) instruction register
d) program counter

2. [1 mark]
In _____, a process has to busy wait for the I/O operation to be completed before proceeding.
a) Programmed I/O
b) Direct memory access (DMA)
c) Interrupt-driven I/O
d) None of the above

3. [3 marks]
Complete the following table by inserting "high", "middle" or "low".

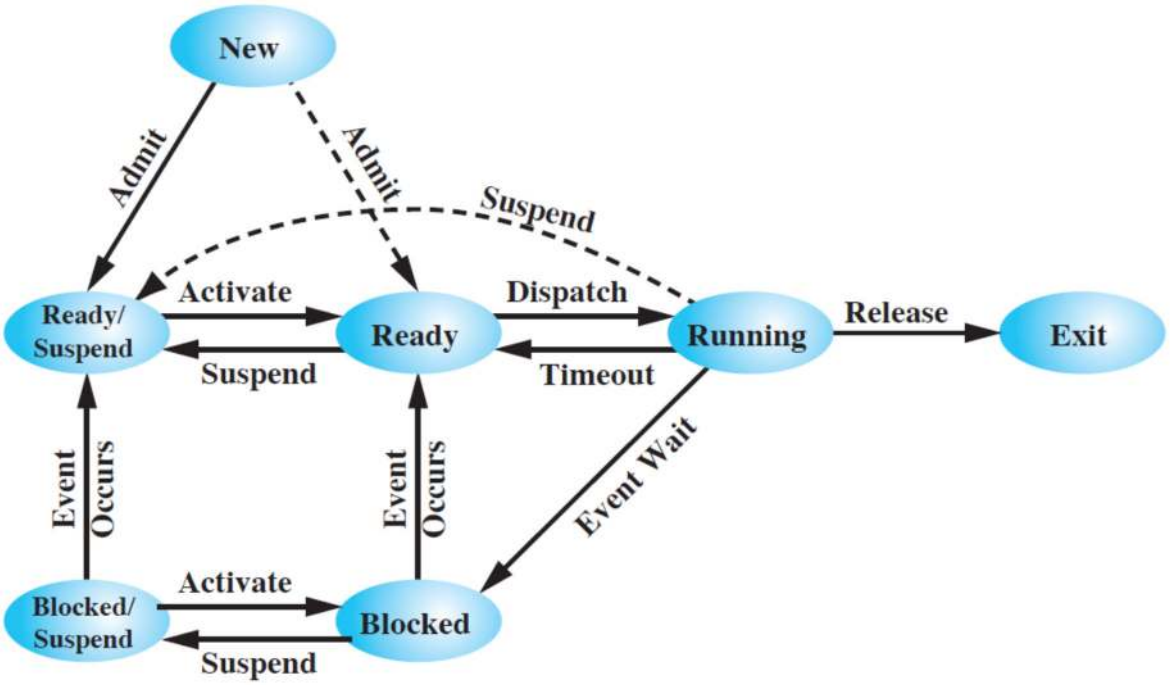
Type of memory	Capacity	Access frequency by processor
Disk	(i)	(iv)
Main memory	(ii)	(v)
Registers	(iii)	(vi)

4. [2 marks]
After the I/O device issues an interrupt signal to the processor, the following events will happen. Write down the correct sequence of those events.

(i) Processor loads new PC value based on interrupt
(ii) Processor restores PSW and PC from control stack
(iii) Processor pushes PSW and PC onto the control stack
(iv) Process the interrupt
(v) Processor finishes execution of current instruction

5. [1 mark]
Comparing with uniprogramming, multiprogramming generally achieves _____ mean response time and _____ throughput.
a) higher, higher
b) higher, lower
c) lower, higher
d) lower, lower

6. [3 marks]
Consider the following 7-state process model. Complete the table by filling the state transition (may or may not be shown on the diagram). The first one is an example.



Event	Transition
The dispatcher chooses a process to run.	Ready → Running
A process has reached its time slice.	(i)
OS brings in a high-priority ready process from disk into memory.	(ii)
A parent process terminates its child process waiting for an event to occur in memory.	(iii)

7. [1 mark]
The collection of process identification, processor state information and process control information is referred to as the _____ .

8. [1 mark]
When a child process is spawned, the `fork` system call returns _____ to the child process and _____ to the parent process.
a) a value of zero, the PID of the child process
b) a value of one, the PID of the parent process
c) the PID of the child process, a value of zero
d) the PID of the child process, a value of one

9. [1 mark]
Which of the following events would **not** lead to a ***trap*** being generated?
a) divided by zero
b) completion of a disk I/O
c) illegal file access attempt
d) reference a protected memory location

10. [1 mark]
Multithreading is important because it can _____.
a) facilitate software design and promote good programming practices
b) improve performance and scalability
~~c)~~ facilitate cooperation/synchronization of activities
d) all of the above

11. [1 mark]
Which of the following statements are correct description of kernel-level threads (KLTs)?

(i) The kernel is aware of the existence of KLTs
~~(ii)~~ KLTs are created by invoking an application-level function.
~~(iii)~~ The transfer of control from one thread to another within the same process requires a mode switch to the kernel is one of the advantages of the KLT approach.
~~(iv)~~ Using KLTs cannot provide a better performance than a single-threaded solution on a single-processor system.

a) (i), (iii) and (iv)
b) (i) only
c) (ii) only
d) (i) and (iv)

12. [2 marks]
Complete the following table by indicating whether the state is for process only, thread only, or both.

State	process/thread/both
Ready	(i)
Suspend	(ii)

13. [1 mark]
_____ restricts access to a shared variable to only one thread at any given time.

14. [1 mark]
A semaphore whose definition includes the policy that the process that has been blocked the longest is released from the queue first is called a _____ semaphore.
a) strong
b) binary
c) counting
d) weak

15. [1 mark]
When a process is in the critical section, it must be running.
a) True
b) False

16. [1 mark]
Which of the following statements about critical sections is false?
a) Only one thread at a time can execute the instructions in its critical section for a particular resource.
b) If one thread is already in its critical section, another thread must wait for the executing thread to exit its critical section before continuing.
c) Once a thread has exited its critical section, a waiting thread may enter its critical section.
d) All threads must wait whenever any critical section is occupied.

17. [1 mark]
A binary semaphore can always be replaced by a mutex.
a) True
b) False

18. [2 marks]
What happens when a thread calls `semWait(S)` before entering its critical section, where `S` is a binary semaphore with current value of 0?
a) The thread is allowed to enter its critical section and `S` is set to 1.
b) The thread is blocked and `S` remains no change.
c) The thread is blocked and `S` is set to 1.
d) The thread is allowed to enter its critical section and `S` remains no change.

19. [1 mark]
If a semaphore was initialized to be 2, its current value is -3. The number of processes blocked on the semaphore is _____

20. [1 mark]
A _____ send is an example of _____ communication that requires the sender to wait for receipt notification before continuing program execution.
a) blocking, synchronous
~~b)~~ blocking, asynchronous
c) nonblocking, synchronous
~~d)~~ nonblocking, asynchronous

21. [1 mark]
In a resource allocation graph, an arrow points from a _____ to a _____ to indicate that the system has allocated a specific instance of resource to the process.
a) square, circle
b) circle, dot in a square
c) circle, square
d) dot in a square, circle

22. [1 mark]
If a resource allocation graph can be reduced by all its processes, then _____.
a) there is no deadlock
b) starvation occurs
c) the system is in an unsafe state
d) none of the above

23. [1 mark]
If we have a cycle in a resource allocation graph, then _____.
a) there is a deadlock
b) there is no deadlock
c) a deadlock might exist
d) none of the above

24. [1 mark]
Which of the following condition cannot be disallowed for non-sharable resources in order to prevent deadlocks?
a) circular wait
b) no preemption
c) hold and wait
d) mutual exclusion

Section B (48 marks)

Attempt ALL questions from this Section

Q1. [10 marks]

Consider the following program fragment.

```
long prog_global = 3;

void *myThreadFun(void *in)
{
        long fun_local = 2;
        ++fun_local;
        ++prog_global;
        cout << "Thread: " << fun_local << " " << prog_global << endl;
        pthread_exit(NULL);
}

int main()
{       // irrelevant variable declarations are removed here
        long main_local = 1;
        pid = fork();
        if (pid != 0) {
                wait(NULL);
                pthread_create(&tid, NULL, myThreadFun, NULL);
                pthread_join(tid, NULL);
                ++main_local;
                ++prog_global;
                cout << "Process1: " << main_local << " " << prog_global << endl;
        }
        else {
                ++main_local;
                ++prog_global;
                cout << "Process2: " << main_local << " " << prog_global << endl;
        }
        pthread_exit(NULL);
}
```

(i) [6 marks]
What would be the output of the program? Show the order and values clearly.

(ii) [4 marks]
Explain the value(s) of the global variable, prog_global, in the above output.

Q2. [7 marks]

Consider the following program running in a uniprocessor system and the two processes, `P1` and `P2`, are running concurrently.

Line no.	
	`int x;`
	`void P(int i)`
	`{`
1	`x = 10;`
2	`x = x + 1;`
3	`x = x - 1;`
4	`if (x != 10)`
5	`printf ("x is %d", x);`
	`}`
	`void main()`
	`{`
	`parbegin (P(1), P(2));`
	`}`

Use the above program to illustrate the quote: "In a race condition, the loser wins." by showing a sequence of interleaved statements such that "`x is 10`" is printed. For each statement, specify the process (`P1` or `P2`) that executes the statement, statement line number and value of `x`. The first one is given below.

Process	Line number	x
P1	1	10

Q3. [11 marks]

Refer to the following solution to the bounded-buffer producer/consumer problem using semaphore. Assume that buffer size is **10**.

```
/* program boundedbuffer */
const int sizeofbuffer = /* buffer size */;
semaphore s = 1, n= 0, e= sizeofbuffer;
void producer()
{
     while (true) {
          produce();
          semWait(e);
          semWait(s);
          append();
          semSignal(s);
          semSignal(n);
     }
}
void consumer()
{
     while (true) {
          semWait(n);
          semWait(s);
          take();
          semSignal(s);
          semSignal(e);
          consume();
     }
}
void main()
{
     parbegin (producer, consumer);
}
```

Consider the following two cases, assuming no process is waiting for the time being. Explain how **synchronization** can be achieved by showing all changes of the values of the three semaphores until both processes are done with a data item.

(i) A consumer wants to take a data item from an **empty** buffer.

(ii) Following (i), a producer wants to append a data item to the buffer.

Q4. [10 marks]

Consider the following solution using ***semaphores*** to the *one-writer many-readers* problem.

```
int         readcount;
Semaphore   sem_x, sem_y;
```

Writer

```
semWait(sem_x);
/* writing performed */
semSignal(sem_x);
```

Readers

```
semWait(sem_y);
readcount++;
if readcount==1 then semWait(sem_x);
semSignal(sem_y);
/* reading performed */
semWait(sem_y);
readcount--;
if readcount==0 then semSignal(sem_x);
semSignal(sem_y);
```

Suppose a *writer* is now writing in the system. No other readers are waiting for the time being.

(i) [2 marks]
What are the current values of the two semaphores?

(ii) [2 marks]
What will happen when a *reader* wants to read while the *writer* is still writing?

(iii) [2 marks]
Following the *first reader*, what will happen when a *second reader* wants to read while the *writer* is still writing?

(iv) [4 marks]
What will happen when the *writer* finishes writing?

Q5. [10 marks]

In the code below, three processes, `P0`, `P1`, and `P2` are competing for six resources `A` to `F`.

```
void P0()
{
  while (true) {
    get(A);
    get(B);
    get(C);
    // critical section:
    // use A, B, C
    release(A);
    release(B);
    release(C);
  }
}
```

```
void P1()
{
  while (true) {
    get(D);
    get(E);
    get(B);
    // critical section:
    // use D, E, B
    release(D);
    release(E);
    release(B);
  }
}
```

```
void P2()
{
  while (true) {
    get(C);
    get(F);
    get(D);
    // critical section:
    // use C, F, D
    release(C);
    release(F);
    release(D);
  }
}
```

(i) [6 marks]
Draw a ***resource allocation graph*** to show the possibility of a deadlock in this implementation. Show the cycle, if any, in the resource allocation graph.

(ii) [4 marks]
Which of the condition for deadlock can be prevented by modifying the order of some of the `get` requests? Show **one** possible modification.

- END -