Question 1

a) [1 mark]

Timesharing; **also accept** pre-emption

b) [4 marks; one for each correct point]

1. Thread block/sleep
2. Thread yield
3. Thread exit
4. preemption

c) [2 marks; one for each point]

1. semaphores have a counter instead of a boolean indicating they are held
2. semaphores have no ownership

d) [2 marks; one for each point --- if combined but correct, also accept]

thread_fork is passed a function pointer
the new thread executes that function

e) [3 marks; one for each point]

1. The process has no living parent
2. When the parent process has called waitpid on the process, which has terminated
3. When the parent dies, zombie children can be fully deleted

ALSO ACCEPT for full marks

1. Any situation where a parent process is unable to call waitpid on the child

f) [2 marks; accept any reasonable answer]

an error code or 0

g) [2 marks; one for each point]

1. Security
2. Abstract design

h) [2 marks; one for each point]

1. Advantage: efficient OR easy to implement
2. Disadvantage: external fragmentation

Question 2

a) **[1 mark]**

1 + 10 + 10 = 21

b) **[6 marks; one for each correct answer]**

1. NO
2. YES
3. NO
4. NO
5. NO
6. NO

c) **[1 mark]**

45

d) **[1 mark]**

No

Question 3
[12 marks total; deduct a mark for each mistake]

User	**Kernel**
Application frames (or user code, or …)	trapframe
open()	mips_trap
	syscall
	sys_open
	trapframe
	mips_trap
	mainbus_interrupt [OPTIONAL, deduct no points if missing]
	timer_exception_handler
	thread_yeild
	thread_switch
	switchframe

Question 4

a) **[1 mark]**

2^32/2^12 = 2^20

b) **[1 mark]**

2^48/2^12 = 2^36

c) **[1 mark]**

12

d) **[1 mark]**

20

e) **[1 mark]**

36

f) **[4 marks; one for each correct answer; accept hex answers]**

1.	0x0000 0000	0
2.	0x0000 0ACE	0
3.	0x0110 EA5E	0x110E
4.	0x0000 00C5	0

g) **[4 marks; one for each correct answer]**

2^13/2^12 = 2 pages of memory (0 and 1)

1. VALID
2. VALID
3. INVALID
4. VALID

Question 5
[8 total marks; one for each point]

1. Acquire procTableLock
2. if ProcTable contains a process with name procName
    a. Get pointer to process
    b. Release lock
    c. Return process PID
3. Otherwise
    a. Release lock
    b. Return ENOPROC

Question 6

a) [7 marks; one for each point]

```
bool try_acquire( lock *lk )
    1. acquire lk->spinlock
    2. If lock available
        a. Take lock
        b. Release lk->spinlock
        c. Return true
    3. Release lk->spinlock
    4. Return false
```

b) [4 marks]

Yes (there is deadlock) **[1 mark]**

Corrections, identified with [], [1 mark each].

```
int total = 0;
   int account = 0;
   lock mutex = lock_create( "mutex" );

  void FuncA(int acc)
   {
     lock_acquire( mutex );
        for ( i = 0 to N )
        {
             total ++;
       }
  ~~   lock_acquire( mutex );~~
     account = acc;
     [lock_release(mutex)]
   }

  void FuncB(int acc, int val)
   {
     for ( i = 0 to N )
      {
          FuncA( acc );
          [lock_acquire(mutex)]
          total = total - val;
          [lock_release(mutex)]
      }
}
```

Question 7

a) **[2 marks; one for offset and one for segment number]**

Segment Number = 2 bits
Segment Offset = 30 bits

b) **[2 marks; one for each point]**

1. Add a register for each segment to indicate if the segment is read-only or not
2. If writing to a read-only segment; MMU must raise exception

c) **[2 marks]**

0

d) **[2 marks; one for each point]**

1. Clear relocation and limit registers
2. Load new process relocation and limit register values

e) **[BONUS 2 marks; one for each point]**

1. Clear STACK relocation and limit register
2. Load new THREAD STACK relocation and limit register values