※注意：請統一於答案卷之「非選擇題作答區」內依序作答，並應註明作答之大題及其題號。

1. Suppose we have a 32 bit MIPS-like RISC processor with the following arithmetic and logical instructions (along with their descriptions):

Addition	
`add rd, rs, rt`	→ Put the sum of registers `rs` and `rt` into register `rd`.
Addition immediate	
`addi rt, rs, imm`	→ Put the sum of register `rs` and the sign-extended immediate into register `rt`.
Subtract	
`Sub rd, rs, rt`	→ Register `rt` is subtracted from register `rs` and the result is put in register `rd`.
AND	
`and rd, rs, rt`	→ Put the logical AND of register `rs` and `rt` into register `rd`.
AND immediate	
`and rt, rs, imm`	→ Put the logical AND of register `rs` and the zero-extended immediate into register `rt`.
Shift left logical	
`sll rd, rt, imm`	→ Shift the value in register `rt` left by the distance (i.e. the number of bits) indicated by the immediate (`imm`) and put the result in register `rd`. The vacated bits are filled with zeros.
Shift right logical	
`srl rd, rt, imm`	→ Shift the value in register `rt` right by the distance (i.e. the number of bits) indicated by the immediate (`imm`) and put the result in register `rd`. The vacated bits are filled with zeros.

Please use at most one instruction to generate assembly code for each of the following C statements (assuming variable a and b are unsigned integers). You can use the variable names as the register names in your assembly code.

(a) (5 pts)
```
b = a / 8;  /* division operation */
```

(b) (5 pts)
```
b = a % 16;  /*  modulus operation */
```

接背面

2. Assume a RISC processor has a five-stage pipeline (as shown below) with each stage taking one clock cycle to finish. The pipeline will stall when encountering data hazards.

IF	ID	EXE	MEM	WB

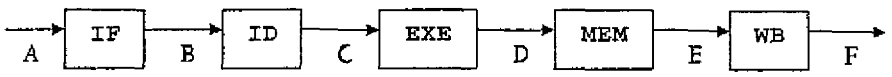
`IF`: Instruction fetch
`ID`: Instruction decode and register file read
`EXE`: Execution or address calculation
`MEM`: Data memory access
`WB`: Write back to register file

(a) (5 pts) Suppose we have an add instruction followed immediately by a subtract instruction that uses the add instruction's result:

```
add  r1 ← r2, r3
sub  r5 ← r1, r4
```

If there is no forwarding in the pipeline, how many cycle(s) will the pipeline stall for?

(b) (5 pts) If we want to use forwarding (or bypassing) to avoid the pipeline stall caused by the code sequence above, choosing from the denoted 6 points (A to F) in the following simplified data path of the pipeline, where (from which point to which point) should the forwarding path be connected?



(c) (5 pts) Suppose the first instruction of the above code sequence is a load of `r1` instead of an add (as shown below).

```
load r1 ← [r2]
sub  r5 ← r1, r4
```

Assuming we have a forwarding path from point E to point C in the pipeline data path, will there be any pipeline stall for this code sequence? If so, how many cycle(s)? (If your first answer is yes, you have to answer the second question correctly to get the 5 pts credit.)

3. (5 pts) Cache misses are classified into three categories- compulsory, capacity, and conflict. What types of misses could be reduced if the cache block size is increased?

接次頁

4. (5 pts) Consider three types of methods for transferring data between an I/O device and memory: polling, interrupt driven and DMA. Rank the three techniques in terms of lowest impact on processor utilization.
5. (15 pts) Assume an instruction set that contains 5 types of instructions: load, store, R-format, branch and jump. Execution of these instructions can be broken into 5 steps: instruction fetch, register read, ALU operations, data access, and register write. Table 1 lists the latency of each step assuming perfect caches.

Instruction class	Instruction fetch	Register read	ALU operation	Data access	Register write
Load	2ns	1ns	1ns	2ns	1ns
Store	2ns	1ns	1ns	2ns	
R-format	2ns	1ns	1ns		1ns
Branch	2ns	1ns	1ns		
Jump	2ns				

Table 1

(a) (5pts) What is the CPU cycle time assuming a multicycle CPU implementation (i.e., each step in Table 1 takes one cycle)?

(b) (5pts) Assuming the instruction mix shown below, what is the average CPI of the multicycle processor without pipelining? Assume that the I-cache and D-cache miss rates are 3% and 10%, and the cache miss penalty is 12 CPU cycles.

Instruction Type	Frequency
Load	40%
Store	30%
R-format	15%
Branch	10%
Jump	5%

(c) (5pts) To reduce the cache miss rate, the architecture team is considering increasing the data cache size. They find that by doubling the data cache size, they can eliminate half of data cache misses. However, the data access stage now takes 4 ns. Do you suggest them to double the data cache size? Explain your answer.

接背面

6. [10pts] The following five questions are true-or-false questions. Please circle your answer. Each correct answer will receive two points.

No.	Question	T	F
(1.)	The purpose of timing sharing is to increase the utilization of system resources.	T	F
(2.)	Suppose that the performance metrics for scheduling algorithms is "average waiting time". When round-robin scheduling is used, the better system performance, the smaller time quantum.	T	F
(3.)	Memory pages that are shared between two or more processes can never be swapped out to the disk.	T	F
(4.)	Paging avoids the problem of external fragmentation of memory in a multi-programming environment.	T	F
(5.)	If a thread generates an exception while inside a critical section protected by a lock, the OS can just kill the thread and release the lock; other threads should be able to continue accessing that critical section without problem. (Assume that locks are implemented by the kernel so that the OS knows all the locks that any particular thread is holding at any point in time.)	T	F

7. [10pts] While writing a program for his system programming class, Alice writes two functions foo() and bar() each of which needs to acquire two locks LockA and LockB. Function Acquire() and Release() are used to acquire and release the locks. Functions foo() and bar() are shown in the following. What is wrong with the following code (5pts) and how might you fix it (5pts) ?

```
LockT LockA, LockB;

foo() {
    …// Other codes are here.
    Acquire(LockA);
    …// Other codes are here.
    Acquire(LockB);
    …// Other codes are here.
    Release(LockB);
    …// Other codes are here.
    Release(LockA);
    …// Other codes are here.
}
```

接次頁

```
bar() {
    ...
    Acquire(LockB);
    …// Other codes are here.
    Acquire(LockA);
    …// Other codes are here.
    Release(LockA);
    …// Other codes are here.
    Release(LockB);
    …// Other codes are here.
}
```

8. [5pts] Alice proposed a new scheduling algorithm and claims that the algorithm favors *new* processes and is fair to *old* processes. In Alice's scheduling algorithm, there are two queues: *new process queue* and *old process queue*. When a new process arrives, the process is inserted into the new process queue. When a new process has been in the new process queue for 5 milliseconds, the process is removed from the new process queue and inserted into the old process queue. When it is time to schedule, the scheduler selects one process from the head of the queues alternatively. Each process will be given a 5-milliseconds time slot. Is this algorithm starvation free? If yes, please explain your answer. If not, please give an example showing that a process may starve.

9. [10 pts] Disks: consider a disk with one surface and 80 tracks recorded evenly spaced between 0.5" to 3.5" from the center of the platter.

(a.) (5 pts) Assume that you have a disk that can store 20 sectors per 2π inches of magnetic media that passes under the disk head, that the disk drive can have a different number of sectors per track for each track, and fractional sectors per track. What is the maximum capacity of this disk in sectors?

(b.) (5 pts) Assume the disk can only record at a single density (i.e., the sectors per track will be the same for each track), and the maximum density is still 20 sectors per 2π inches, what is the capacity of the disk in sectors?

10. [15 pts] Security: suppose you are using RSA encryption with p= 3, q = 7, and e = 5. That is, the public key is a pair (e, n); the private key is a pair (d, n), where e, d, and n are positive integers.

(a) (5 pts) Find the decryption exponent d.

(b) (5 pts) Encrypt the message m = 4.

(c) (5 pts) Decrypt the cypher c = 2.