98 學年度第 2 學期博士班資格考試　計算機結構

1. (18%) Consider the execution of the following loop, which increments each element of an integer array, on a two-issue processor, once without speculation and once with speculation:

```
Loop:   LD       R2,0(R1)      ;R2=array element
        DADDIU   R2,R2,#1      ;increment R2
        SD       R2,0(R1)      ;store result
        DADDIU   R1,R1,#8      ;increment pointer
        BNE      R2,R3,LOOP    ;branch if not last element
```

Assume that there are separate integer functional units for effective address calculation, for ALU operations, and for branch condition evaluation. Create a table for the first three iterations of this loop for both processors. Assume that up to two instructions of any type can commit per clock. Table 1 shows the time of issue, execution, and writing result for a two-issue processor without speculation. Please fill in Table 2 with the time of execution and writing result for a two-issue processor with speculation.

Table 1

Iteration number	Instructions		Issues at clock cycle number	Executes at clock cycle number	Memory access at clock cycle number	Write CDB at clock cycle number	Comment
1	LD	R2,0(R1)	1	2	3	4	First issue
1	DADDIU	R2,R2,#1	1	5		6	Wait for LW
1	SD	R2,0(R1)	2	3	7		Wait for DADDIU
1	DADDIU	R1,R1,#8	2	3		4	Execute directly
1	BNE	R2,R3,LOOP	3	7			Wait for DADDIU
2	LD	R2,0(R1)	4	8	9	10	Wait for BNE
2	DADDIU	R2,R2,#1	4	11		12	Wait for LW
2	SD	R2,0(R1)	5	9	13		Wait for DADDIU
2	DADDIU	R1,R1,#8	5	8		9	Wait for BNE
2	BNE	R2,R3,LOOP	6	13			Wait for DADDIU
3	LD	R2,0(R1)	7	14	15	16	Wait for BNE
3	DADDIU	R2,R2,#1	7	17		18	Wait for LW
3	SD	R2,0(R1)	8	15	19		Wait for DADDIU
3	DADDIU	R1,R1,#8	8	14		15	Wait for BNE
3	BNE	R2,R3,LOOP	9	19			Wait for DADDIU

Table 2

Iteration number	Instructions		Issues at clock cycle number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number
1	LD	R2,0(R1)	1				
1	DADDIU	R2,R2,#1	1		╳		
1	SD	R2,0(R1)	2		╳	╳	
1	DADDIU	R1,R1,#8	2		╳		
1	BNE	R2,R3,LOOP	3		╳	╳	
2	LD	R2,0(R1)	4				
2	DADDIU	R2,R2,#1	4		╳		
2	SD	R2,0(R1)	5		╳	╳	
2	DADDIU	R1,R1,#8	5		╳		
2	BNE	R2,R3,LOOP	6		╳	╳	
3	LD	R2,0(R1)	7				
3	DADDIU	R2,R2,#1	7		╳		
3	SD	R2,0(R1)	8		╳	╳	
3	DADDIU	R1,R1,#8	8		╳		
3	BNE	R2,R3,LOOP	9		╳	╳	

a 100 ns time to handle reference to a remote memory. For this application, assume that all the references except those involving communication hit in the local memory hierarchy, which is slightly optimistic. Processors are stalled on a remote request, and the processor clock rate is 2 GHz. If the base CPI (assuming that all references hit in the cache) is 0.5, how much faster is the multiprocessor if there is no communication versus if 0.3% of the instructions involve a remote communication reference?

3. (18%) The simple, bus-based multiprocessor illustrated in Fig. 1 represents a commonly implemented symmetric shared-memory architecture. Each processor has a single, private cache with coherence maintained using the snooping coherence protocol of Fig. 2. Each cache is directed-mapped, with four blocks each holding two words. The cache-address tag contains the full address and each word shows only two hex characters, with the least significant word on the right. The coherence states are denoted M, S, and I for Modified, Shared, and Invalid.

The performance of a snooping cache-coherent multiprocessor depends on many detailed implementation issues that determine how quickly a cache responds with data in an exclusive or M state block. For the multiprocessor illustrated in Fig. 1, consider the execution of a sequence of operations on a single CPU where

- CPU read and write hits generate no stall cycles.
- CPU read and write misses generate N_{memory} and N_{cache} stall cycles if satisfied by memory and cache, respectively.
- CPU write hits that generate an invalidate incur $N_{invalidate}$ stall cycles.
- a writeback of a block, either due to a conflict or another processor's request to an exclusive block, incurs an additional $N_{writeback}$ stall cycles.

Consider two implementations with different performance characteristics summarized in Table 3. Moreover, a sequence of one or more CPU operations is specified in the following form:

P#: <op> <address> [← <value>]

where P# designates the CPU (e.g., P0), <op> is the CPU operation (e.g., read or write), <address> denotes the memory address, and <value> indicates the new word to be assigned on a write operation.

Consider the following sequence of operations assuming the initial cache state in Fig. 1. For simplicity, assume that the second operation begins after the first completes (even though they are on different processors):

P1: read 110
P15: read 110

For Implementation 1, the first read generates 80 stall cycles because the read is satisfied by P0's cache. P1 stalls for 70 cycles while it waits for the block, and P0 stalls for 10 cycles while it writes the block back to memory in response to P1's request. Thus the second read by P15 generates 100 stall cycles because its miss is satisfied by memory. Thus this sequence generates a total of 180 stall cycles.

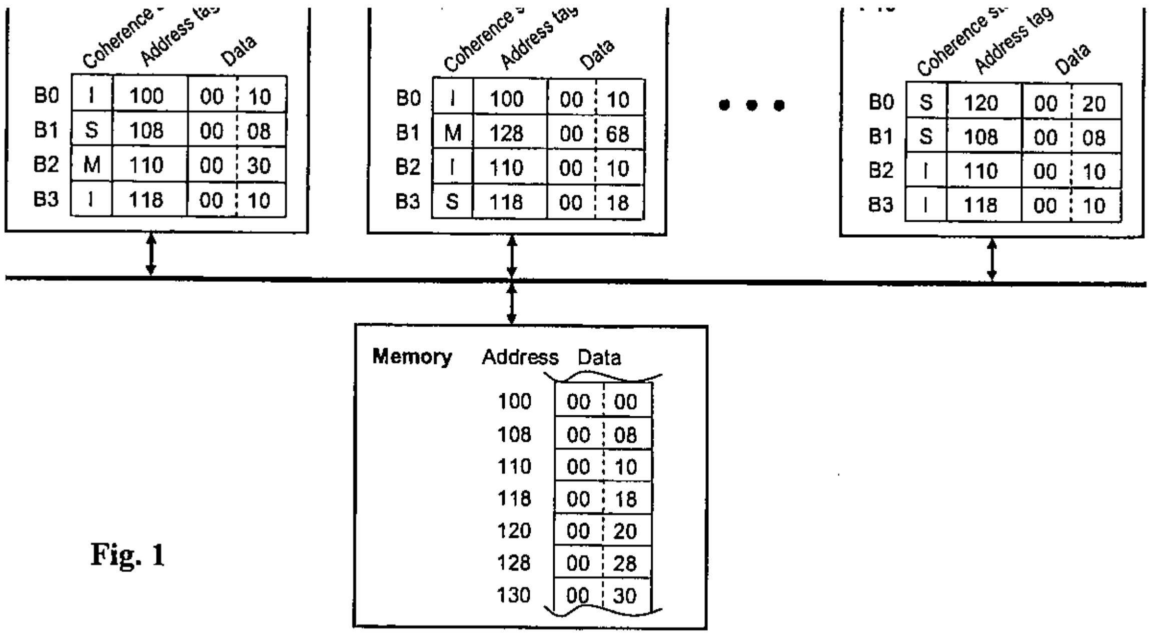
For the following sequences of operations, how many stall cycles are generated by each implementation?

(1) P0: read 120
P0: read 128
P0: read 130

(2) P0: read 100
P0: write 108 ← 48
P0: write 130 ← 78

(3) P1: read 120
P1: read 128
P1: read 130

(4) P1: read 100
P1: write 108 ← 48
P1: write 130 ← 78



	Coherence state	Address tag	Data
B0	I	100	00 10
B1	S	108	00 08
B2	M	110	00 30
B3	I	118	00 10

	Coherence state	Address tag	Data
B0	I	100	00 10
B1	M	128	00 68
B2	I	110	00 10
B3	S	118	00 18

	Coherence state	Address tag	Data
B0	S	120	00 20
B1	S	108	00 08
B2	I	110	00 10
B3	I	118	00 10

Memory Address	Data
100	00 00
108	00 08
110	00 10
118	00 18
120	00 20
128	00 28
130	00 30

Fig. 1

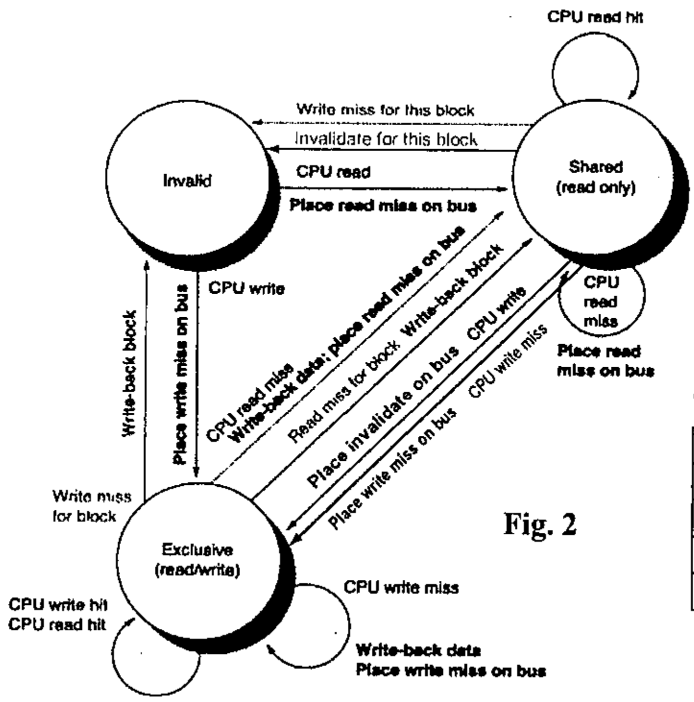


Fig. 2

Table 3

Parameter	Implementation 1	Implementation 2
N_{memory}	100	100
N_{cache}	70	130
$N_{invalidate}$	15	15
$N_{writeback}$	10	10

4. (14%) Which has the lower miss rate: a 32 KB instruction cache with a 32 KB data cache or a 64 KB unified cache? Use the miss rates in Table 4 to help calculate the correct answer, assuming 40% of the instructions are data transfer instructions. Assume a hit takes 1 clock cycle and the miss penalty is 100 clock cycles. A load or store hit takes 1 extra clock cycle on a unified cache if there is only one cache port to satisfy two simultaneous requests (a structural hazard). What is the average memory access time in each case? Assume write-through caches with a write buffer and ignore stalls due to the write buffer.

Table 4: Miss per 1000 instructions

Size	Instruction cache	Data cache	Unified cache
16 K	6	42	51
32 K	2	40	43
64 K	1	36	40

a write-back cache that does write allocate. The elements of a and b are 8 bytes long since they are double-precision floating-point arrays. There are 3 rows and 100 columns for a and 101 rows and 3 columns for b. Let's also assume they are not in the cache at the start of the program.

(1) For the code shown in Fig. 3, determine which accesses are likely to cause data cache misses and calculate the number of data cache misses. (8%)

(2) If prefetch instructions are inserted as shown in Fig. 4 to reduce misses. Calculate the number of prefetch instructions executed and the misses avoided by prefetching. (10%)

```
for ( i = 0;  i < 3;  i = i + 1)
    for ( j = 0;  j < 100;  j = j + 1)
        a[i][j] = b[j][0] * b[j+1][0];
```

Fig. 3

```
for ( j = 0;  j < 100;  j = j + 1) {
    prefetch(b[j+7][0];  /* b(j,0) for 7 iterations later */
    prefetch(a[0][j+7];  /* a(0,j) for 7 iterations later */
    a[0][j] = b[j][0] * b[j+1][0];};
for ( i = 0;  i < 3;  i = i + 1)
    for ( j = 0;  j < 100;  j = j + 1) {
        prefetch(a[i][j+7];  /* a(i,j) for +7 iterations */
        a[i][j] = b[j][0] * b[j+1][0];}
```

Fig. 4

6. (12%) Suppose a processor sends 40 disk I/Os per second, these requests are exponentially distributed, and the average service time of an older disk is 10 ms. Answer the following questions:

(1) On average, how utilized is the disk? (4%)

(2) What is the average time spent in the queue? (4%)

(3) What is the average response time for a disk request, including the queuing time and disk service time? (4%)

7. (10%) Please explain the differences in a processor's ability to exploit the resources of a superscalar for the following processor configurations (shown in Fig. 5): a superscalar with no multithreading support, a superscalar with coarse-grained multithreading, a superscalar with fine-grained multithreading, and a superscalar with simultaneous multithreading.

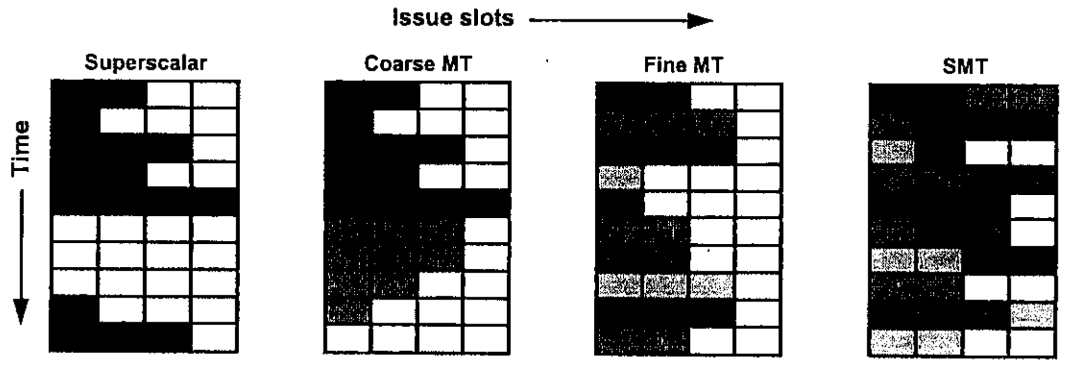


Fig. 5