Department of Computer Science

Computer Architecture Qualifying Exam

Fall, 2000

This examination is open book and open notes. The questions are equally weighted. On all questions, the reasoning you use to obtain your answer is more important than the actual answer; if your answer to any question is simply a number without justification, it is wrong (even if the number happens to be correct).

1. Suppose a byte-addressed computer system has a 58 bit virtual address, a 16K byte page size, and a 40 bit physical address. In addition to the physical address, the page table entry must contain present, dirty, user-accessible, writeable, page cache disable, and page writethrough bits.

   A. What is a reasonable size for a page table entry?

   B. Give a reasonable division of a virtual address for this computer into a multiple level paging scheme.

   C. This machine uses a 64K byte L1 cache, and does cache lookups simultaneously with TLB lookups. What is the minimum set-associativity the cache can have?

2. A four-processor computer system using an invalidation-based cache coherence strategy, and with a four-byte cache block size, performs the following sequence of operations:

Operation Number	***Processor***	***Operation***	***Address***
1	P1	Read	`0x100`
2	P2	Read	`0x104`
3	P1	Write	`0x100`
4	P3	Read	`0x100`
5	P4	Write	`0x104`
6	P3	Invalidate	`0x100`

   A. Describe the bus transactions and memory state changes that occur as a result of each of these operations, assuming an MSI coherence strategy. Assume all of the processors‘ caches are initially empty. Remember it‘s a *sequence* of operations; each operation depends on the state of the machine left behind by the previous operation.

   B. If the cache block size were increased to eight, what would the new sequence of transactions and state changes be?

3. What are the comparitive advantages and disadvantages of bus, ring, crossbar, binary hypercube, and two-dimensional mesh interconnection topologies? How would wormhole routing affect this?

4. A computer system uses a standard five-stage (IF, ID/RR, EX, MEM, WB) pipeline. If a register is read and written on the same cycle, the read returns the new value. It uses forwarding to resolve any conflicts it can, and stalls when necessary. For this entire question, you will be considering the same code sequence; I've rewritten it in each subpart to save you time.

   All of the subparts of this question should be written on this page.

   A. Show all of the dependencies in this code; for instance, since the first instruction provides r1 as a result and the second instruction uses r1, I've drawn a line from the r1 in the first instruction to the r1 in the second instruction.

```
lw   r1, 100(r2)  ; load contents of memory location 100+r2 into r1
lw   r3, 200(r1)  ; load contents of memory location 200+r1 into r3
sw   r3, 100(r4)  ; store r3 into memory location 100+r4
lw   r4, 100(r5)  ; load contents of memory location 100+r5 into r4
add  r6, r3, r4   ; add r3 and r4, putting result into r6
sw   r6, 300(r5)  ; store r6 into memory location 300+r5
addi r2, r2, 4    ; add 4 to r2
add  r4, r2, r4   ; add r2 to r4
```

   B. Show all of the antidependencies in the code; for instance, since the first instruction uses r2 and the seventh instruction changes r2, I've drawn a line from the r2 in the first instruction to the r2 in the seventh instruction.

```
lw   r1, 100(r2)  ; load contents of memory location 100+r2 into r1
lw   r3, 200(r1)  ; load contents of memory location 200+r1 into r3
sw   r3, 100(r4)  ; store r3 into memory location 100+r4
lw   r4, 100(r5)  ; load contents of memory location 100+r5 into r4
add  r6, r3, r4   ; add r3 and r4, putting result into r6
sw   r6, 300(r5)  ; store r6 into memory location 300+r5
addi r2, r2, 4    ; add 4 to r2
add  r4, r2, r4   ; add r2 to r4
```

   C. Draw a timing chart for this code. Wherever forwarding takes place, draw an arrow to show the source and destination of the forwarded data. Reading and writing a register on the same cycle is not considered forwarding.

lw r1, 100(r2)																		
lw r3, 200(r1)																		
sw r3, 100(r4)																		
lw r4, 100(r5)																		
add r6, r3, r4																		
sw r6, 300(r5)																		
addi r2, r2, 4																		
add r4, r2, r4																		