Computer Science Comprehensive Examination Solution
Computer Architecture

Autumn 2001

(Total time = 60 minutes, Total Points = 60)

Magic Number:__________Solution Set____________________

In recognition of and in the spirit of the Stanford University Honor Code, I certify that I will neither give nor receive unpermitted aid on this exam.

Magic Number Signature:______________________________________

This examination is open book. Please do all your work on these sheets. Do not do your work in a blue book. You have one hour to complete the exam. Before starting, please check to make sure that you have all 10 pages.

1	10	
2	8	
3	6	
4	8	
5	10	
6	6	
7	12	
Total	60	

1. Short Answer Question (10 points)

Assume that you have written the following function:

```
double AddCond(double *a, double *b, double startVal)
{
   if (a != NULL) {
        startVal = startVal + (*a);
        if (b != NULL) {
             startVal = startVal + (*b);
        }
   }
   return startVal;
}
```

After you turn in the function to your professor you re-read the instructions and discover that the function could not include any branch instructions! You decide your best hope is to submit an addendum to your assignment that describes a compiler and architectural features that allow this function to be generated with no branch instruction.

Describe here the architectural features you are assuming the compiler has to work with. Show the branch-less assembly language of this function in a pseudo assembly language of your choosing.

Let's assume we have some kind of predicated or conditional instructions.
Assume that a is passed in Ra, b is passed in Rb, and startVal is passed in Fv.

```
SetNe R1, Ra,0          // Set R1 with (a != NULL)
SetNe  R2,Rb,0          // Set R1 with (b != NULL)
SetAnd  R3,R1,R2         // Set R3 with (a != NULL) && (b != NULL)
CondLoad  F1,(Ra),R1   // if (R1) then load F1 with *a
CondLoad  F2, (Rb),R3  // if (R3) then load F2 with *b
CondAdd  Fv,F1,R1       // if (R1) then add F1 to Fv
CondAdd  Fv,F2,R3       // if (R3) then add F2 to Fv
Ret Fv                  // return startVal;
```

18

More space for question 1 (if needed)

2. Short Answer Question (8 points total)

Assume you are given two caches: A and B. Cache A is an 64 byte direct-mapped cache. Cache B is an 64 byte 4-way set associative cache with LRU replacement. Both caches use a 16 byte line size. Assume that you have a memory reference address stream that is listed in column one of the table below. Fill in the table using the following notion:

Confl - The cache will take a conflict miss.
Capac - The cache will take a capacity miss.
Compul - The cache will take a compulsory miss.
Hit - The cache will hit.

Reference Address	Cache A	Cache B
0x0000	***Compul***	***Compul***
0x0148	***Compul***	***Compul***
0x028c	***Compul***	***Compul***
0x03c0	***Compul***	***Compul***
0x0004	***Conflic***	***Hit***
0x0408	***Compul***	***Compul***
0x014c	***Capac***	***Capac***
0x000c	***Confl***	***Hit***

3. Short Answer Question (6 points)

Although numerous studies of cache architectures have shown limited miss-rate benefits and high implementation cost of having a very high degree of associativity in primary data caches, successful machines have been constructed with 32way and higher associativity in their data cache. Explain why an implementation might use a high degree of associativity that isn't justified by cache miss rate reduction.

You do this when you want to increase the cache capacity without increasing the number of bits used as an index. This is most frequently done so the cache can be index using the virtual memory page offset bits allowing the TLB lookup to be done in parallel with cache access.

4. Short Answer Question (8 points)

Traditionally the TLB of a CPU has not been as transparent to the OS as other caches such as the instruction and data caches. This lack of transparency means that implementations that add a TLB frequently extend the architecture to make the TLB visible to the software and require the operating system be aware of this cache. Describe how an implementation could transparently add an effective TLB to existing paged architecture without requiring OS changes.

The TLB need to maintain consistency with the in-memory page tables of the virtual memory system. Having the TLB snoop on changes made to memory and detect if the cached PTE is being modified. This is similar to an invalidation-based multiprocessor cache coherency protocol.

2-2

5. Short Answer Question (10 points)

The design of the MIPS and SPARC RISC architectures were done when a simple 5-stage pipeline was a popular implementation technique for their first implementation. Although both designs support register+offset addressing, the MIPS choose not to implement register+register addressing except for loads and stores accessing the registers of the floating-point unit.

(a) Describe how the initial pipeline design would have influenced the MIPS architects to make this decision on addressing modes.

(b) In comparison between the MIPS and SPARC architecture, the SPARC architects claim that better CPI seen for the MIPS architecture was caused in part by the existence of register+register addressing in SPARC and not in MIPS. Explain this.

(A) ***Having register+register on a store instruction with a 5-stage pipeline can require 3 registers (value being stored and two address register) fetch at once. No other instruction needs 3 read ports into the register file to fetch its operands in a single cycle. Note that the supporting register+register to the FP register still only requires two ports in the integer register file.***

(B) ***Without register+register the MIPS architecture requires separate add instructions to do the add while the SPARC can do the add with the store instruction that stalls for one additional cycle. Since MIPS runs more instructions and those instructions take only a single cycle it results in a lower CPI that SPARC can executes fewer but instructions with more stalls.***

6. Short Answer Question (6 points)

Explain how scatter/gather I/O interfaces to high bandwidth I/O devices such as disks is important to support large bulk data transfers in modern computer systems.

Modern computer systems use virtual memory so that large data blocks are likely to be discontinuous in physical memory. Scatter/gather DMA is used to put together these pages for transfer to and from the I/O device.

7. True and False Questions (3 points each, 12 points total)

Answer the following questions as either True or False and provide a brief justification of your answer.

(a) An architecture with condition codes (e.g. a flags register) in which every instruction sets every condition code will show no instruction level parallelism on a dynamic scheduled super-scalar architecture because of the dependencies on the condition code register. (True or False, justify)

False, a dynamic scheduled architecture can rename the flags register like any other register allowing ILP.

(b) A VLIW architecture will have better instruction density than a standard 32-bit RISC instruction set. (True or False, justify)
False, VLIW architectures frequently have worse density because of the need to pad with NOPs when ILP is lacking.

(c) It is always possible to build a program that will defeat any branch predictor so that the branch prediction rate will be less than 10%. (True or False, justify)
False, a random branch predictor could not be defeated in this way.

(d) WAW hazards are only possible if instructions take a variable number of cycles to execute. (True or False, justify)
True, fixed number of cycles mean the instructions finish in order so WAW is not possible.