Computer Science Comprehensive Examination
Computer Architecture
[100 points]

This examination is open book. Please do all of your work on these sheets. Do not do your work in a blue book.

Number: ____

Problem	Max Score	Your Score
1	33	
2	33	
3	34	
TOTAL	100	

Problem 1: Memory Systems [33 Points 4 points each + 1 free]

A. Compared to a 16K direct-mapped cache, what type of misses will a 16K fully associative cache have fewer of? Circle all that apply.

(a) compulsory
(b) conflict
(c) capacity

(b) conflict misses

B. Suppose that cache A, a 16K-byte four-way set associative cache, cache B, a 16K-byte direct-mapped cache, and cache C, a 4K-byte direct-mapped cache are all referenced with an identical address sequence. All caches use a true least-recently used (LRU) replacement policy. Which of the following statements are true: (circle all that apply)

(a) A will contain a superset of the data in B
(b) A will contain a superset of the data in C
(c) B will contain a superset of the data in A
(d) B will contain a superset of the data in C

(b) A contains a superset of C. Cache C is equivalent to one "way" of A .
and (d) B contains a superset of C since both are direct mapped and B is larger.

C. In a 64K-byte four-way set-associative cache with 128-byte blocks, how large is the index field used to address the cache array? (write down the number of bits)

__*7*_________________

D. If the cache of question 1.C is physically tagged and physical addresses are 40-bits long, what is the minimum possible length the tag may be for correct operation? (write down the number of bits)

___*26 = 40-7 (index) – 7 (block)*________________

E. A processors needs to support multiple processes running simultaneously with complete isolation (each process cannot access the memory of any other process). What is the minimum set of features in the architecture required to implement this complete isolation? (Circle minimal subset)

(a) Atomic branch and entry to privileged mode (in which all of memory can be addressed)
(b) A single base and length (segment) register
(c) Multiple base and length (segment) registers, one per process
(d) A translation look-aside buffer with a trap to a privileged mode handler routine on miss.

(a) and (b) suffice, as does (d) by itself (either gets full credit)

F. Replacing a single memory bank with multiple interleaved memory banks has which of the following effects? (circle all that apply)

(a) Increases latency
(b) Increases bandwidth
(c) Increases reliability
(d) Decreases latency

(b)

G. A bus-based cache coherence protocol "snoops" the directory of each processor's level-2 cache on each bus access. For this scheme to work, what relationship between the level-1 cache and level-2 cache must be maintained? (one word)

inclusion

H. A cache coherent multiprocessor has four processors, each with a 16Mbyte direct-mapped level-2 cache that is organized into 128-byte lines. Each processor makes single-word (8 byte) references (read and write in equal numbers) to widely-spaced single word locations in memory in a repeating pattern of 128K references. The patterns of the processors are disjoint – there is no sharing – and small enough to entirely fit in the cache. Rank the following three cache coherence protocols in order the amount of memory or bus traffic generated after the first iteration through the patterns?

(a) No cache – all data fetched and written is from pages marked uncached.
(c) Write-through – all cache lines have two states I and V.
 Lines in state V are always in sync with memory – no dirty lines.
(d) Write-back – lines have three states, I, V, and D. The first write to a V line invalidates all other copies and make the line exclusive (and dirty), the D state.

The important thing to note here is that the working set is 128K 128Byte lines, so the working set is 16Mbytes which fits in the cache.

Thus (d) has the least traffic followed by (c) and then (a). If the working set were larger than the cache, (a) would have had the least traffic.

Problem 2: Instruction-Set Architecture [33 Points]

After graduate school, you work for JIPS Technologies, a company that designs embedded MIPS systems optimized for Java applications. The main product of JIPS is a MIPS processor and a just-in-time (JIT) compiler. The processor uses the classical 5-stage pipeline. The compiler runs along with any Java application and translates on-demand its Java code to MIPS instructions. The JIT compiler is kept very simple in order to minimize its overhead. It does implement the stack storage defined by the Java architecture, but tries to eliminate the number of push and pop operations to the stack by allocating as many temporary results as possible to the registers available in the MIPS processor.

After analyzing the typical workload (application & JIT), you come up with the following frequencies and CPIs for each type of instructions supported by the processor:

Type	Frequency	CPI
Load	30%	1.6
Store	20%	1.4
Arithmetic operations	35%	1.0
Branches & jumps	15%	1.5

Your analysis also indicates that 50% of the stores are used to push values into the stack. 33.33% of the loads are used to pop values from the stack. Assume that all pushes and pops operate on 32b integer numbers.

1) [7 points] You decide to improve performance by adding push and pop instructions to the MIPS ISA:

```
push rs   # MEM[r30+4]←rs; r30←r30+4
pop  rs   # rs←MEM[r30];   r30←r30-4
```

Register r30 stores a pointer to the top of the stack by default. What additional resources will you need in the 5-stage pipeline to support the execution the two new instructions? Explain.

Pop writes both rs and r30. Hence, we need a 2nd write port to the register file.

No adder is needed for (r30+4) or (r30-4). Push and pop can perform this operation on the integer ALU in the EX stage of the pipeline.

2) [6 points] A colleague notices that **push** and **pop** are similar to loads and stores with autoincrement and autodecrement addressing respectively. Hence, she suggests introducing autoincrement/autodecrement addressing (in which the register used to provide the address is incremented/decremented as a side effect of the load or store instruction) to the MIPS ISA instead of **push** and **pop**. Explain to her why **push** and **pop** are cheaper to implement.

General autoincrement/autodecrement loads/stores have two disadvantages

- They require additional bits in the opcode to encode the increment/decrement value (register specifier or immediate)
- They require 2 integer ALUs: one for the effective address calculation (register + offset) and one for the autoincrement/autodecrement.

3) [6 points] The JIPS design group comes to the conclusion that the introduction **push** and **pop** will increase the clock cycle time of the processor by 5% due to the additional resourced needed to implement them. Calculate the overall performance improvement if the new instructions are added to the design.

$$Speedup = \frac{ExecutionTimeOld}{ExecutionTimeNew} = \frac{ICold * CPIold * CCTold}{ICnew * CPInew * CCTnew}$$

CPIold = Sum(individual CPIs) = 0.3*1.6+0.2*1.4+0.35*1+0.15*1.5 = 1.335
CCTnew = 1.05*CCTold

Each push/pop instruction will replace one store/load instruction in the original program and the arithmetic operation used to increment/decrement the stack pointer. Hence, the introduction of push/pop eliminates a number of arithmetic operations from the original program. Therefore:
ICnew = (1-#push-#pop)*ICold = (1-0.5*0.2*0.33*.3)*ICold = 0.8*ICold

The CPI of a push/pop instruction is equal to that of a store/load instruction in the original program. However, our CPInew calculation must take into account that we eliminated some of the arithmetic instructions in the original program.
CPInew = (0.3*1.6+0.2*1.4+0.15*1+0.15*1.5)/0.8 = 1.41875

Hence, the overall speedup is 1.335/(0.8*1.41875*1.05) = 1.12 or 12%

4) [7 points] A third colleague notices that for the same clock cycle time increase (5%) you can double the associativity of the data cache used with the processor. The improved cache hit rate will reduce CPI for all loads and stores in the workload. JIPS Technologies has enough engineers to either implement the new instructions or improve the cache, but cannot do both. Calculate the minimum percentage improvement of the CPI for the loads and stores that is necessary in order to decide to go with the improved cache. Assume that the percentage of CPI improvement for loads and stores is exactly the same.

The improved cache must lead to overall performance improvement of at least 12% or a speedup of 1.12. We can use the speedup equation of part 3). In this case:
ICnew = ICold
CCTnew = 1.05*ICold
CPIold = 1.335

Speedup > 1.12⇔ 1.335 > 1.12 *CPInew*1.05 ⇔ CPInew < 1.135

Assume that the new cache design changes the CPI of loads and stores by a factor of x. I.e. the CPI for loads is 1.6*x and the CPI for stores is 1.4*x
CPInew < 1.135 ⇔ 0.48*x + 0.28*x + 0.35 + 0.225 < 1.135 ⇔ 0.76*x< 0.56 ⇔ x < 0.73.

Hence, the new cache decrease the CPI of load and stores by 0.73 or 27%. The new CPI for loads must be 1.17 and the new CPI for stores must be 1.02.

[7 points] JIPS Technologies decides to implement the push and pop after all. You propose to use 16-bit encoding for the two instructions. This will lead to better code density which is important for embedded applications. All other instruction retain their 32-bit encoding. A senior colleague quickly points out that mixing 32-bit and 16-bit instructions requires an instruction set change that may limit the code density benefits from your proposal. What is the change and how does it hurt code density?

With 32-bit fixed-length instructions, all instructions addresses are aligned to 4-byte boundaries. In other words, the 2 least significant bits of the PC are always 0. Hence, the displacement of branch and jump instructions does not have to specify these last 2 bits.

If we interleave 16-bit and 32-bit instructions, an instruction address may be aligned to 4-byte or 2-byte boundaries. Only the very least bit of the PC is guaranteed to be 0. Compared to the original ISA, we need an extra bit of displacement with branches and jumps. However, we cannot go to 33-bit encoding for branches and jumps, hence with have to live with one less bit of displacement for these instructions.

If there is a branch in a program that uses all the bits of displacement with the original ISA (32-bit encoding only), this branch will require two instructions in the new ISA (16-bit & 32-bit instructions interleaved). If such branches are common, some of the code density benefits from encoding push and pop with 16-bits may be cancelled out.

Problem 3: Pipelining [34 Points]

A company has recently announced a processor running at 1.5GHz with the following pipeline:

IF1	First part of instruction fetch (TLB access)
IF2	Instruction fetch completes (instruction cache accessed)
RF	Instruction decoded and register file read
EX	Perform operation; compute memory address (base + displacement); compute branch target address; compute branch condition
MEM1	First part of memory access (TLB access)
MEM2	Memory access completes (data cache accessed)
WB	Write back results into register file

As in the simple MIPS 5-stage pipeline, the results are written back into the register file in the first half of the cycle, and registers are read in the second half of the cycle.

3a. (8 points)
(i) How many register read ports does this pipeline require to implement a MIPS-style instruction set? how many write ports?

```
2 read ports, 1 write port
```

(ii) How many adders does the machine require (to prevent structural hazards), and in which stages are they used?

```
2 adders; one in IF1 (for the PC), one in EX (for the ALU).
```

(iii) What is the branch delay? What is the load delay?

```
Branch delay is 3 cycles. Load delay is 2 cycles (MEM2->EX).
```

(iv) To implement complete forwarding, how many destination register numbers must the machine record? how many comparators does it require?

```
   3 destinations registers must be saved, and 6 comparators to compare
2 source register numbers with the 3 destinations
```

3b. (12 points)

Many processor designers favor a simple approach to branch prediction. One possibility considered for this pipeline was to use delayed branches. The compiler written for the processor is able to fill the branch delay slots with the following frequencies:

from before the branch, fill all slots:	for 10% of branches
from before the branch, fill two slots:	30%
from before the branch, fill one slot	10%
from the target path, fill one slot:	4%
from the sequential path, fill one slot:	3%

The compiler puts NOPs in unfilled delay slots. If the performance analysis group measures that 65% of branches in typical applications are taken, what is the average CPI of a branch?

```
Branches have three delay slots.
 # usefully filled = 10%*3 + 30%*2 + 10%*1 + 4%*(65% taken)*1
                + 4%*(35% not taken)*1 = 1.0365
 ==> Avg. # of NOP cycles = 3 - 1.0365 = 1.9635

 ==> Avg. CPI of branches = 1 + penalty cycles = 1 + 1.9635 = 2.96
```

3c. (14 points)

Comparing pipelines: some people advocate staying with the simpler 5-stage MIPS pipeline. They say they can get the clock rate as high as 1.2GHz and still have only one branch delay slot while eliminating half of the load-use stall cycles.Given the following statistics for the 7-stage processor, which pipeline will perform better (ignoring memory stalls)?

	% of total execution cycles
Branch Delay NOP cycles	5%
Load Delay stall cycles	4%

You may need to use your results and the compiler statistics from part (b)(i). If you were not able to solve part (b)(i), assume that on average 1.5 of a branch's delay slots are filled with NOPs.

```
  Since Exec.Time = IC * CPI / Clock Rate = # cycles / Clock rate

  For 100 cycles of execution on a 7-stage pipeline, there are 5 branch
delay NOP cycles, 4 load delay cycles, and hence 91 instruction
execution cycles.

     Exec. Time (7) = 100 cycles / 1.5 GHz = 66.7 ns

  The 7-stage pipeline requires 1.9635 NOP cycles per branch.

   The 5-stage pipeline needs only
     1 - (10% + 30% + 10% + 4%*65% + 3%*35%)*1 = 0.4635 NOP
cycles/branch (since if the compiler could fill 3 and 2 slots 10% and
30% of the time, respectively, it could certainly fill 1 slot for that
same fraction).
  ==> # branch delay stall cycles required is
      (0.4365/1.9635)*5 cycles = 1.11 cycles

   Also, eliminate half of load delay stall cycles, so only need 2:

   ==> Exec. Time (5 stage) = (91 + 2 + 1.11) / 1.2 GHz = 78.4 ns
  which is much worse (5-stage is 78.4/66.7=17.5% slower)
```