Computer Architecture Comprehensive Exam

Exam Instructions

Answer each of the questions included in the exam. Write all of your answers directly on the examination paper, including any work that you wish to be considered for partial credit. The examination is open-book, and you may make use of the text, handouts, your own course notes, and a calculator.

On equations: Wherever possible, make sure to include the equation, the equation rewritten with the numerical values, and the final solution. Partial credit will be weighted appropriately for each component of the problem, and providing more information improves the likelihood that partial credit can be awarded.

On writing code: Unless otherwise stated, you are free to use any of the assembly instructions listed in the Appendix at the back of the book, including pseudoinstructions. You do not need to optimize your MIPS code unless specifically instructed to do so.

On time: You will have one hour to complete this exam. Budget your time and try to leave some time at the end to go over your work. The point weightings correspond roughly to the time each problem is expected to take.

THE STANFORD UNIVERSITY HONOR CODE

The Honor Code is an undertaking of the students, individually and collectively:

(1) that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;

(2) that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

I acknowledge and accept the Honor Code.

Magic Number _____

		Score	Grader
1. Short Answer	(15)	_____	_____
2. Average Memory Access Time	(15)	_____	_____
3. Bits and Bytes	(15)	_____	_____
4. MIPS	(15)	_____	_____
Total	(60)	_____	

Problem 1: Short Answer (15 points)
Please provide short, concise answers.

(a) [1 points] One benefit of RISC architectures is that they generally provide more general purpose registers. Why is this beneficial?

Reuse register values without spilling to memory

(b) [1 points] Ben Bitdiddle notices that many RISC architectures have 32 general purpose registers. Ben wonders if more registers are better, why not have even more? Explain to Ben the tradeoffs involved with increasing the number of registers to 64.

Smaller immediate values
OR
Slower register file access
OR
Opportunity cost for hardware for more registers than compiler can use
OR
More registers to context switch

(c) [1 points] One of the differences between RISC architectures and CISC architecture is supposed to be the reduced types of instructions available. Ben Bitdiddle thinks it would be a good idea to simplify the instruction set even more to remove the special case instructions that take immediate operands such as “li”, “addi”, etc. Explain to Ben why this might not be such a good idea.

Common operations will now require multiple instructions without any corresponding improvement in cycle time

(d) [2 points] Some procedure-linkage conventions require the caller (rather than the callee) pop procedure arguments from the stack on return. Why is it awkward for the callee to perform this step?

Callee does not always statically know the number of arguments passed, such as when there are variable number of arguments as in printf

(e) [2 points] A cache memory can speed up computation by eliminating some references to main memory. Does its use require clever programming techniques? Explain.

1 point – yes, optimal performance can require cache aware or cache oblivious code

1 point – no, program can benefit from cache without being modifed

2 points for yes and no explanation

(f) [2 points] Must a user program written for use with a non-paged memory system be modified in order to be used with a paged memory system? Explain.

1 point – yes, optimal performance can require page aware algorithm or working set management

1 point – no, program can use more than memory than physical memory with minimal impact if working set fits in physical memory

2 points for yes and no explanation (also points for page locality needed for high TLB hit rate)

(g) [2 points] Explain how a memory system that pages to secondary storage depends on locality of reference for efficient operation.

Without locality, the memory system would perform a disk speeds as every access could require an access to disk. A working set that fits within physical memory for efficient operations.

(h) [2 points] Program A consists of 1000 consecutive add instructions, while program B consists of a loop that executes a single add instruction 1000 times. You run both programs on a certain machine and find that program B consistently executes faster. Explain.

Program B fits easily in the instruction cache but program A takes more time to be fetched.

(i) [2 points] A simple strategy for aborting a CISC machine instruction when page faults occur would be simply to reset the program counter and stack pointer to the values they had when the current instruction began to execute. List a serious problem with this strategy.

Non-idempotent side effects need to be reversed.

Problem 2: Average Memory Access Time (15 points)

For this problem, assume that you have a processor with a cache connected to main memory via a bus. A cache access takes 1 cycle. A successful cache acces (a hit) finishes within that cycle. On an unsuccessful access (a miss) additional work must be performed to fetch a block from main memory over the bus. A bus transaction consists of one cycle to send the address to memory, four cycles of idle time for main-memory access, and then one cycle to transfer each word in the block to the cache. (Assume that the processor continues execution only after the last word of the block has arrived.)

Block size (B)	**Miss ratio (m), %**
1 word	3.40%
4 words	1.00%
16 words	0.40%
64 words	0.25%
256 words	0.19%

In other words, if the block size is B words (at 32 bits/word), a cache miss will cost 1 + 4 + B cycles. The adjacent table gives the average cache miss rates of a 1 megabyte cache for various block sizes.

(a) [3 points] Write an expression for the average memory access time for a 1-MByte cache and a B-word block size (in terms of m and B).

AMAT = 1 + m (1 + 4 + B)

(b) [4 points] What block size yields the best average memory access time?

For B == 1 1 + .0340(1+4+1) = 1.204
For B == 4 1 + .0100(1+4+4) = 1.09
For B == 16 1 + .0040(1+4+16) = 1.084 block size of 16 is best
For B == 64 1 + .0025(1+4+64) = 1.1725
For B == 256 1 + .0019(1+4+256) = 1.4959

(c) [4 points] If bus contention adds three cycles to the main-memory access time, which block size yields the best average memory access time.

For B == 1 1 + .0340(3+1+4+1) = 1.306
For B == 4 1 + .0100(3+1+4+4) = 1.12
For B == 16 1 + .0040(3+1+4+16) = 1.096 block size of 16 is best
For B == 64 1 + .0025(3+1+4+64) = 1.18
For B == 256 1 + .0019(3+1+4+256) = 1.5016

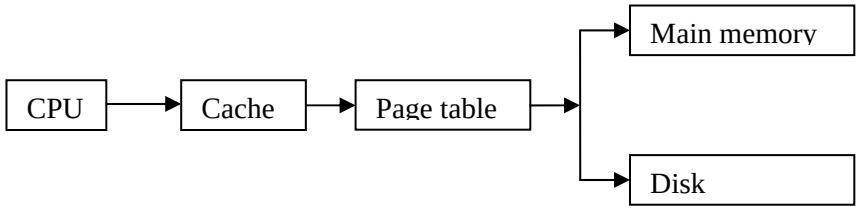
(d) [4 points] Ignoring bus contention, if the bus width is doubled to 64 bits, what is the optimal block size?

For B == 1 1 + .0340(1+4+1) = 1.204
For B == 4 1 + .0100(1+4+2) = 1.07
For B == 16 1 + .0040(1+4+8) = 1.052 block size of 16 is best
For B == 64 1 + .0025(1+4+32) = 1.0925
For B == 256 1 + .0019(1+4+128) = 1.2527

Problem 3: Bits and Bytes (15 points)

The figure below shows a CPU and its memory system. The computer features

- a single processor
- 32-bit virtual addresses
- a cache of 2^{10} sets that are four-way set-associative and have 8-byte blocks
- a main memory of 2^{26} bytes; and a page size of 2^{12} bytes.



(a) [1 points] Does this system cache virtual or physical addresses?

virtual addresses

(b) [3 points] How many bytes of data from memory can the cache hold? (Don't count tags)

2^{10} sets * 4 lines/set * 8 bytes/line = 2^{15} bytes = 32kb

(c) [4 points] In the cache, each block of data must have a tag associated with it. How many bits long are these tags?

32 bits in virtual address – 10 bits in index – 3 bits in offset = 19 bits for tag

(d) [4 points] How many comparators are needed to build this cache while allowing single cycle access?

1 per each set way = 4

(e) [3 points] At any one time, what is the greatest number of page-table entries that can have their valid bit set to 1?

$2^{32}/2^{12}$ bytes/page = 2^{20} pages.
All of them can valid because then can all alias the same physical page.

Problem 4: MIPS (15 points)

(a) [6 points] Ben Bitdiddle has been hired by Syl Valle to recover some code lost in a hard drive crash. He was able to recover some fragments of both the C code and its compiled MIPS assembly code from the drive. Help him out by filling in the parts he was unable to recover. Remember, in a MIPS function call, ra contains the return address, a0 contains the first argument, and upon return v0 contains the return value. The s# registers must be preserved across procedure calls. Also remember that standard MIPS has a branch delay slot.

```
int f(int n) {
    int f2;
    if (n<2)
        return n;
    f2 = f(n-2);
    return <A>;
}
```

```
00000000 <f>:
   0:   addiu   sp,sp,-40
   4:   sw      ra,36(sp)
   8:   sw      s8,32(sp)
   c:   move    s8,sp
  10:   sw      a0,40(s8)
  14:   lw      v0,40(s8)
  18:   nop
  1c:   slti    v0,v0,2
  20:   beqz    v0,3c <f+0x3c>
  24:   nop
  28:   lw      v0,40(s8)
  2c:   nop
  30:   sw      v0,<B>
  34:   j       84 <f+0x84>
  38:   nop
  3c:   lw      v0,40(s8)
  40:   nop
  44:   addiu   v0,v0,-2
  48:   move    a0,v0
  4c:   jal     0 <f>
  50:   nop
  54:   sw      v0,16(s8)
  58:   lw      v0,40(s8)
  5c:   nop
  60:   addiu   v0,v0,-1
  64:   move    a0,v0
  68:   jal     0 <f>
  6c:   nop
  70:   move    v1,v0
  74:   lw      <C>,16(s8)
  78:   nop
  7c:   addu    v1,v1,v0
  80:   sw      v1,24(s8)
  84:   lw      v0,24(s8)
  88:   move    sp,s8
  8c:   lw      ra,36(sp)
  90:   lw      s8,32(sp)
  94:   addiu   sp,sp,40
  98:   jr      ra
  9c:   nop
```

<A>: f(n-1) + f2

<B>: 24(s8)

<C>: v0

(b) [3 points] Ben is pleased with your work and asks you to help him optimize the assembly.
Optimize instructions 0x44 – 0x50.

44: jal 0 <f>
48: addui a0, v0, -2
52: nop
56: nop

(c) [6 points] Ben wants to take out some of the nops in the code and is interested in seeing how the resulting code is executed on a pipelined processor with a 5-stage pipeline. Branches are resolved in the decode stage and have a delay slot. All memory accesses take 1 clock cycle.

The table below shows the instruction stream starting at instruction 0x14 with some nops taken out (the branch is taken). Fill in the remainder of the pipeline diagram. Show all cases of data forwarding (or "bypassing") using arrows to connect the source and destination stages.

Instruction	Cycle															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lw v0,40(s8)	F	D	X	M↘	W											
slti v0,v0,2		F	D	-	X↘	M	W									
beqz v0,3c			F	-	-	D										
nop						F	D	X	M	W						
lw v0,40(s8)							F	D	X	M↘	W					
addiu v0,v0,-2								F	D	-	X↘	M	W			
move a0,v0									F	-	D	X	M	W		
jal 0											F	D				