Computer Science 152 Fall 1996 Midterm 2

Professor D.E. Culler

Problem 1 [25 points] : Basic concepts

(a) State the five major components of a computer. "The Big Picture"

(b) State the six primary steps in the instruction processing cycle.

(c) Specify the formats for and the algebraic value of 1) IEEE normalized and 2) IEEE denormalized numbers. Explain why denormalized are valuable.

(d) Give a formula for the miss rate at which the average memory access time is twice the hit time.

(e) Explain the three basic types of cache misses.

(f) If a 200 MHz processor executes one instructions per cycle, except on cache misses (for which it stalls), the instruction miss rate is 1%, the data miss rate is 5%, 30% of the instructions are loads or stores, 10% are jumps or branches, and the miss penalty is 10 cycles. What fraction of time is spent stalled for memory? What is the CPI?

Problem 2 [30 points]

Your company is having trouble with a new 32-bit computer system and it is believed to have been isolated to a two-instruction sequence on a scaled down version of the machine. In order determine the problem you must first generate a detailed account of what should happen to serve as a test vector.

Your machine has four 32-bit registers, with R0 always zero. It has sixteen 32-bit words of memory. It is byte addressed, big endian. The machine has a page size of sixteen bytes. It has a unified cache containing four 2-word blocks. It is direct mapped, write-back, no-alocate. It has a fully-associative four entry TLB using round-robin replacement. A special NEXT register points to the entry that will be filled on the next TLB miss and is incremented (mod 4) after each fill. The ormat of these organizational components and the initial state is described in the figure below.

The contents of memory are displayed in symbolic form, with all integers in decimal. The page table occupies one page and is resident at physical location zero. It contains four valid page table entries - the contents of the fileds are as shown. Addresses are shown in binary.

The meaning of the exception cause is as follows:

Value	Meaning
0	overflow
1	page fault
2	unaligned access
3	external interrupt

Part a: [2 points] For the machine in the following figure, specifiy the number of bits in each of the cache fields.

Part b: [28 points] On the page following the figure, write down in sequence every reference and update made to any of the components, i.e., the TLB, the cache, physical memory, and the registers. Also, alter the figure to show the effects of each update.

cause	
EPC	
PC	000000
R0	0
R1	010000
R2	
R3	

TLB

	VPN	PFN	R	W	X	V	Next
0							0
1	2	2	1	0	0	1	
2							
3							

Physical Memory

000000	valid 1, X
000100	valid 3, RW
001000	valid 2, R
001100	valid 1, R
010000	LW R2 <= R1, 4
	SW R1, 24 <= R2
100000	152
	16
	437
	"ABCD"
110000	15
	17
	40
	150

Cache

	v	d	tag	word 0	word 1
0					
1					
2					
3					

Page Table Entry
[Valid, PPN, R, W, X]

Page Table Entry Legend
Valid = Valid
PPN = Physical Page Number
R = Readable
W = Writable
X = Executable

References and updates to ALL components (not just memory):

Problem 3 [20 points]:

You are working on the follow-on to a system designed around a 32-bit bus operating at 50 MHz. It is a simple bus with no overlap among the arbitration, address, and data phases. You have been able to deduce by instrumenting the bus that 50% of the transfers are single word transfers and the rest are four-word transfers.

The maximum delivered bandwidth of the bus is 91 MB/s.

What is the utiliation of the bus?

If the arbitration phase is a single cycle, how many cycles are spent in the address phase of each transfer (including possible waits), with the above workload?

What would be the bus speedup if you could shave one cycle off the address phase?

Problem 4: Instruction Sets and Processor Design [50 Points]:

In CS152 you became very familiar with the MIPS R3000 load-store architechture and how to design a fast pipeline processor. You've just joined a startup called Less-is-Better that plans to sell an ultra-reduced instruction set computer. They have discovered that if you just add an addressing mode bit or two for each operand, they can eliminate many of the instructions in the instruction set. No more nasty loads and stores. No more special immediate instructions. Their instruction set has a single, fixed 32-bit format for all instructions, shown below:

4	1	4	1	4	2	16
opcode		Rd		Rs		Rt
	Md		Ms		Mt	

The mode bits have the following meaning:

Value	**Meaning**
0	register direct (low 4 bits for Rt)
1	register indirect
2	immediate (Rt only) - sign ext. for arith, zero ext for logical
3	upper immediate (Rt only) - immediate in upper 16 bits like mips LUI

opcode	**instruction**	**meaning**
0	add Rd Rs Rt	$Mode_d$(Rd) <= $Mode_s$(Rs) + $Mode_t$(Rt) ; pc <= pc+4
1	sub Rd Rs Rt	$Mode_d$(Rd) <= $Mode_s$(Rs) - $Mode_t$(Rt) ; pc <= pc+4
2	and Rd Rs Rt	$Mode_d$(Rd) <= $Mode_s$(Rs) AND $Mode_t$(Rt) ; pc <= pc+4
3	or Rd Rs Rt	$Mode_d$(Rd) <= $Mode_s$(Rs) OR $Mode_t$(Rt) ; pc <= pc+4
4	neg Rd Rs Rt	$Mode_d$(Rd) <= NOT $Mode_t$(Rs) ; pc <= pc+4
5	JMP	pc <= $Mode_t$(Rt); Rd <= pc +4

6	BLT	if $Mode_s(Rs) < 0$ then pc <= $Mode_t(Rt)$ else pc <= pc+4
7	BZ	if $Mode_s(Rs) == 0$ then pc <= $Mode_t(Rt)$ else pc <= pc+4

(a) Generate a generic code sequence to perform a subroutine call (to lable F) with arguments in registers R1 through R5, the return address in R15 and the result used in R1. Explain the additional assumptions that will be needed in order to compile working code for programs and libraries. Generate the code to return from F. Use the back of the previous sheet.

(b) Now its time to desgin a processor for this instruction set.

Your design goal is to produce a machine that will execute sequences of register-to-register arithmetic and logic instructions at an average CPI of 1 with clock rate of 100 MHz, i.e. you will need to pipeline your datapath. Instructions that use the memory addressing modes will take a larger number of cycles. Take your time and think through the design. It is not just a rehash of your MIPS design, you need to apply the design concepts that you have learned to a *new* problem.

WORK OUT YOUR DESIGN ON SCRATCH PAPER AND COPY IT NEATLY TO YOUR EXAM. IT MAY TAKE A COUPLE OF TRIES TO GET A GOOD DESIGN. WE CAN ONLY GIVE IT A GOOD GRADE IF WE CAN UNDERSTAND IT.

TAKE IT STEP BY STEP AND DOCUMENT EVERY STEP. BE CAREFUL OF THE ADDRESSING MODES AND THE MEMORY.

You may assume as building blocks with the given worst-case propagation delays: 32-bit adders (4ns), 32-bit ALU (5 ns, op: add, sub and, or), latches (2 ns), registers (1 ns setup, 1 ns prop), register file (2 ns setup, 2 ns prop), muxes (1 ns), boolean logic gates (0.2 ns), and wires. In addition, you have one ideal single-ported data memory (6 ns), which can either be read or written in a cycle, but not both, and one instruction memory (6 ns).

- Step 1: analyze the instruction set and determine what storage components and combinational components you will need in your datapath. Include your final parts list. (Hint: be a little careful, since some component may be used multiple times per instruction.)
- Step 2: construct a datapath capable of supporting the data transfers required by each of the instructions.
- Step 3: determine the control settings for the register-transfers involved in each of the instructions on the datapath of step 2.
- Step 4: detrmine the controller specification, design the controller structure, and generate the detailed control logic.

FINAL List of Components.

FINAL Datapath Design (Be especially neat here)

FINAL Definition of control point and signals (What each control point does and what each signal means.)

FINAL Controller design (Block diagram of control structure, Detailed control description for each instruction, Logic equation or truth table for each control point.)