Test only • No solutions available

Computer Science Department
Stanford University
Comprehensive Examination in Computer Architecture
Autumn 1996

October 16, 1996

READ THIS FIRST!

1. You should write your answers for this part of the Comprehensive Examination in a BLUE BOOK. Be sure to write your MAGIC NUMBER on the cover of every blue book that you use.

2. This exam is CLOSED BOOK. You may not use notes, articles, or books.

3. Show your work, since PARTIAL CREDIT will be given for incomplete answers. For example, you can get credit for making a reasonable start on a problem even if the idea doesn't work out; you can also get credit for realizing that certain approaches are incorrect. On a true/false question, you might get partial credit for explaining why you think something is true when it is actually false. But no partial credit can be given if you write nothing.

TEST IS 7 pages. 1 HOUR long.

Computer Science Comprehensive Examination
Computer Architecture
(60 points)

Question 1 (15 points)

Early next year, Intel will add an extension to the x86 ISA aimed at improving performance of multimedia oriented programs on its Pentium™ processor chips. These instructions, collectively called the MMX extensions, can optimize execution of ALU instructions that operate on 8 / 16 / 32 bit signed or unsigned integers by "packing" several of these smaller integers into special 64-bit registers and then performing the same ALU operations on these integers in parallel with a single instruction.

We are interested in determining the effectiveness of this extension. A suite of benchmark programs has been selected with the following instruction counts and CPIs on a non-pipelined machine:

	# of instructions	CPI
Integer ALU		
(8-bit)	10,000	4
(16-bit)	20,000	4
(32-bit)	40,000	4
(64-bit)	10,000	4
Load	15,000	5
Store	15,000	4
Branches	5,000	3

After examination of the 8-bit and 16-bit integer ALU operations, we discover the following:

	8-bit ALU	16-bit ALU	32-bit ALU
% can be optimized	80%	70%	20%
average number of useful packed integers / MMX instruction	5	3	2

The CPI's of the MMX instructions are the same as the original ALU instructions. There is also an overhead of loading the integers into the special CPU registers. For this benchmark, an extra 5000 move instructions (CPI = 3) are required to perform the necessary register-register moves.

a. (5) What is the performance improvement on the benchmark with the ISA extension if the clock cycle does not change on the machine with extensions?

(24)

b. (5) If the benchmark is currently running on a 50MHz machine with MMX extensions, what would the new clock frequency on the original machine have to be in order to have the same performance as the machine with the ISA extension?

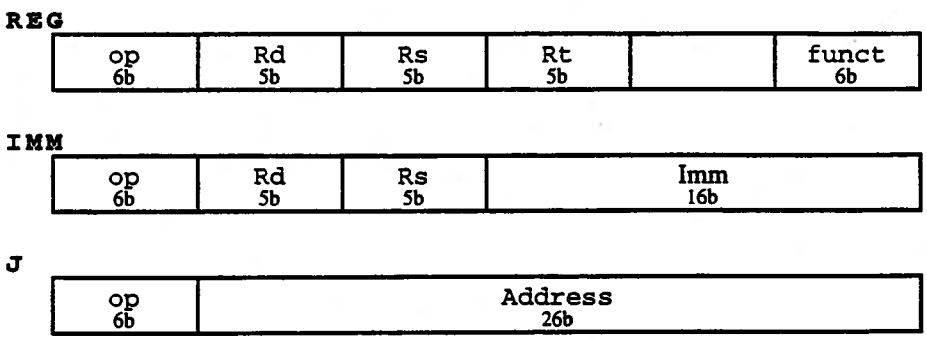
c. (5) What changes to the machine and issues must be considered to accommodate these new instructions? Based on these issues and changes associated with adding these instructions, do you think this is a good idea? Why?

(25)

Question 2 (25 points)

On the next page is a basic single-cycle MIPS processor datapath with some missing information:

- 32-bit byte-aligned PC
- 3 instruction formats:

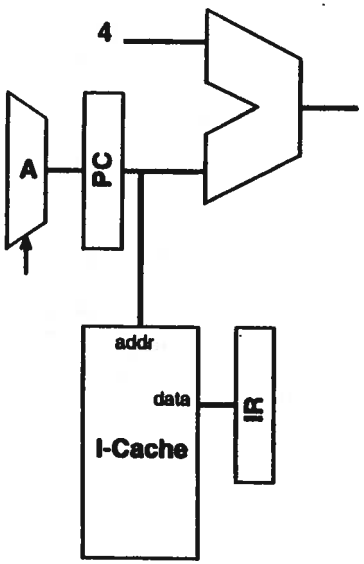


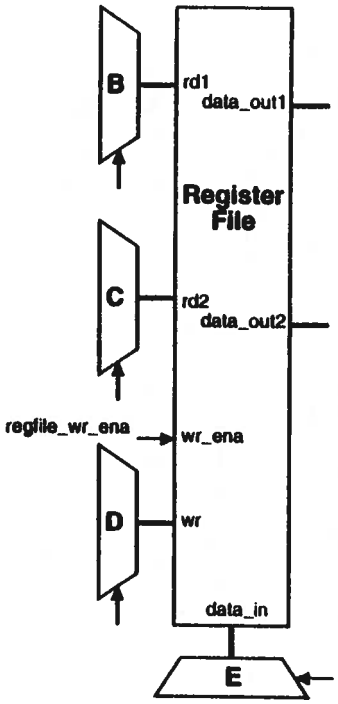
- The ALU can perform unsigned or signed addition or subtraction operations

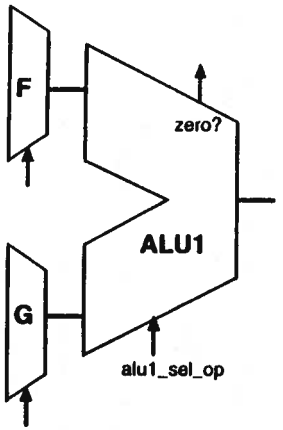
Show the necessary inputs into the multiplexers and the bit widths of the missing wires necessary to support the following list of instructions in the diagram. Also show the correct setting of the multiplexers and control wires for each instruction in the table given on the following page:

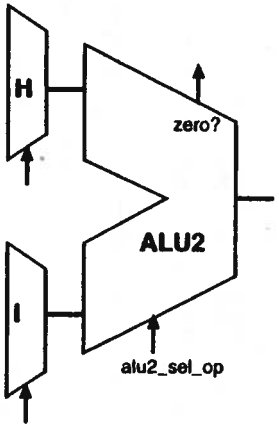
ADDI - Add signed Rs and Imm and place result in Rd
SUBU - Subtract unsigned Rt from Rs and place result in Rd
BEQAL - Branch to PC + Imm (Imm is a signed offset in number of instructions) and place the address of the next instruction after the current PC in R31 if Rs and Rt are equal, otherwise go to the next sequential PC.
J - Jump to the word address in the instruction concatenated with the upper bits of the next instruction after the current PC.
LW - Load the contents of the address at Rs + Imm (Imm is a signed offset in bytes) into Rd
SW - Store Rd to the address at Rs + Imm (Imm is a signed offset in bytes)

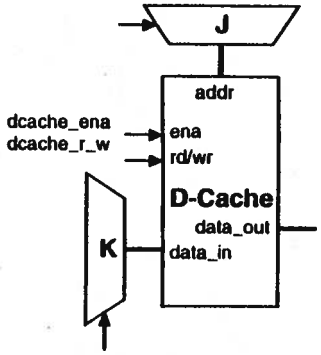
(26)


4
A
PC
addr
data
I-Cache
IR


B
rd1
data_out1
Register File
C
rd2
data_out2
regfile_wr_ena
wr_ena
D
wr
data_in
E


F
zero?
ALU1
G
alu1_sel_op


H
zero?
ALU2
I
alu2_sel_op


J
addr
dcache_ena
ena
dcache_r_w
rd/wr
D-Cache
data_out
K
data_in

	ADDI	SUBU	BEQAL	J	LW	SW
A						
B						
C						
D						
E						
F						
G						
H						
I						
J						
K						
alu1_sel_op						
alu2_sel_op						
regfile_wr_ena						
dcache_ena						
dcache_r_w						

(25)

Question 3 (20 points)

Shown below are the contents the TLB, page table and cache of the MicroMachines MiniCPU™, a miniaturized version of the modern CPU. The MiniCPU™ has the following characteristics:

- 16-bit virtual address, 15-bit physical address, both byte aligned
- TLB
  - 4 entries, direct mapped
  - 5% miss rate
  - No translation penalty for a hit
  - Each miss requires an average of 30~ (cycles) to process
- Page table
  - 16 entries total to map all possible pages in a process' virtual memory space
  - Page table accesses are accounted for in the TLB miss time
- Cache
  - Write back, write allocate
  - There is a infinitely large write buffer to main memory which can hide all write latencies to main memory
  - 50% of the lines are dirty
  - 16 entry, 2-way set associative
  - Each cache line is 32 bytes long
  - 10% miss rate
  - No penalty on a hit
  - A lookup in the cache returns a word (4 bytes)
- Main memory
  - It first takes 80ns to lookup a random address in main memory. Once this location is found, the bus can transfer data at 50Mbytes/s.
  - Sequential memory access latencies are hidden by the bus transfer time.
  - An entire cache line must be transferred before a miss in the cache can be satisfied.

a. (5) Show the tag, index and offset bits necessary to access the TLB with a virtual address and to access a word in the cache with a physical address:

VIRTUAL

PHYSICAL

b. (5) Suppose we wanted to do the cache and TLB lookup in parallel. What restrictions does this place on the TLB/cache sizes, if any?

c. (5) What is the AMAT (average memory access time) of this memory system if the CPU is running at 25MHz? How much would the AMAT improve if the cache lookup is done in parallel with the TLB lookup (assuming it is possible with the current configuration)?

d. (5) Discuss the tradeoffs between a direct mapped and a highly associative cache (e.g. 4-way set associative with LRU replacement). What sort of reference streams would benefit from an associative cache? If you only had a direct mapped cache available, what could you do to your code and data structures at compile time to improve performance on these reference streams?

(30)