University of California, Berkeley
College of Engineering
Computer Science Division — EECS

Fall 1997 D.A. Patterson

Midterm II
October 19, 1997
CS152 Computer Architecture and Engineering

You are allowed to use a calculator and one 8.5" x 1" double-sided page of notes. You have 3 hours. Good luck!

Your Name:	
SID Number:	
Discussion Section:	

1	/20
2	/20
3	/20
4	/20
Total	**/80**

Question 1

In addition to higher hardware costs, a larger cache can also have a longer access time. Beyond a certain size, a L1 cache will reduce the performance of the CPU. It is possible, however, to reduce the miss penalty without affecting cycle time. To do this, many modern computers use a second level cache. The L2 cache is not inside the pipeline and is accessed only when there is a miss in L1. Main memory access is required only when there is a miss in L2.

For this question, we will not distinguish reads from writes. Both the L1 and L2 caches are synchronized to the CPU clock. For both L1 and L2, a miss is handled with an access to the next lower level in the memory hierarchy, and after the miss the request is handled exactly like a hit. For example, in a read, L1 will first update its contents with data from L2, and then pass it on to the CPU.

Definitions:

- MR - Miss rate. Fraction of accesses that result in a miss.
- HT - Hit time. Access time during a cache hit.
- MP - Miss penalty. Additional access time incurred by a cache miss.

a) For a system with two levels of caching, L1 and L2, give the average access time of L1 in terms of $MR_{L1}, HT_{L1}, MR_{L2}, HT_{L2}, and MP_{L2}$, for the case where HT and MP are integral multiples of CPU cycle time.

Average access time$_{L1} = HT_{L1} + MR_{L1} \times (HT_{L2} + MR_{L2} \times MP_{L2})$

You are evaluating some proposals for improving performance of a system still in the design stage. For the applications that you have in mind, system performance is directly proportional to memory performance. Cost is of no object to you.

The question left out a statement indicating that L1 is the unified instruction and data cache. Fortunately, that's what everyone assumed.

The L1 and L2 caches available have the following characteristics:

- $HT_{L1} = 2$ns
- $MR_{L1} = 5\%$
- $HT_{L2} = 20$ns
- $MR_{L2} = 25\%$ (remember, L2 is accessed only by L1)
- $MP_{L2} =$ main memory access time $= 100$ns

The clock cycle time for the CPU is currently 2ns.

b) Consider a design that only uses a L1 cache. It has been proposed that doubling the size of the cache will decrease the miss rate to 4%, while increasing hit time to 2.4ns. Is this a good idea? What will be the percentage change in performance?

- original performance:

  Cycle time = 2 ns
  Hit time = 1 clock
  Miss rate = 0.05
  Miss penalty = 50 clocks

  Average access time$_{L1} = 1 + 0.05 \times 50 = 3.5$ clocks $= 7$ns

- performance after the change:

  Cycle time = 2.4ns
  Hit time = 1 clock
  Miss rate = 0.04
  Miss penalty = ceil$(\frac{100\ ns}{2.4\ ns}) = 42$ clocks

  Average access time$_{L1} = 1 + 0.04 \times 42 = 2.68$ clocks $= 2.68 \times 2.4 = 6.432$ ns

Change in performance $= \frac{\frac{1}{6.432} - \frac{1}{7}}{\frac{1}{7}} = +8.8\%$

For this performance comparison to make sense, we have to assume that MR_{L1} is actually the miss rate per instruction. Again, this is what most people assumed. No one should have lost points for making other reasonable assumptions.

c) With a larger transistor budget, we were able to add a L2 cache to the system. Is it a good idea to double the L1 cache now? What will be the change in performance?

- original performance:

  Cycle time = 2ns
  $HT_{L1} = 1$ clock
  $HT_{L2} = 10$ clocks
  $MR_{L1} = 5\%$
  $MR_{L2} = 25\%$
  $MP_{L2} = 50$ clocks
  $MP_{L1} = 10 + 0.25 \times 50 = 22.5$

  Average access time$_{L1} = 1 + 0.05 \times 22.5 = 2.125$ clocks $= 2.125 \times 2 = 4.25$ ns

- performance after changes:

  Cycle time = 2.4ns
  $HT_{L1} = 1$ clock
  $HT_{L2} = \text{ceil}(\frac{20}{2.4}) = 9$ clocks
  $MR_{L1} = 4\%$
  $MR_{L2} = 25\%$
  $MP_{L2} = \text{ceil}(\frac{100}{2.4}) = 42$ clocks
  $MP_{L1} = 9 + 0.25 * 42 = 19.5$

  Average access time$_{L1} = 1 + 0.04 \times 19.5 = 1.78$ clocks $= 1.78 \times 2.4 = 4.27$ ns

Change in performance $= \frac{\frac{1}{4.27} - \frac{1}{4.25}}{\frac{1}{4.25}} = -0.5\%$

d) In general, you tend to care more about miss ratio for L1 caches and hit times for L2 caches. True or False?

False. Miss penalty for L1 is significantly reduced by the presence of a L2 cache. L1 hit time is more important because it directly affects cycle time. Hit time is less important for L2 because it does not affect cycle time.

e) Compare your answer to part b) with part c). Does the presence or absence of a L2 cache influence design decisions for L1? Explain why.

Designing the L1 cache involves a trade-off between miss rate and cycle time. The best design balances Cycles Per Instruction (increases with miss rate) with Seconds Per Clock (increases with cycle time) to obtain the best Instructions/Second. The addition of a L2 cache makes a L1 miss less expensive. Intuitively, this shifts the optimal balance toward a higher L1 miss rate and shorter cycle time.

Question 2

You have recently been recruited by InDec Corp. to work on a second version of their "Alphium" processor. Alphium is a simple ISC processor with 5 pipeline stages, just like the one presented in class. Its clock frequency is 200MHz and the chip power supply (Vdd) is 3.3V. For the highly appreciated MisSpec benchmark, Alphium achieves a IPC of 0.8 instructions per cycle ($IPC = \frac{1}{CPI}$).

Your assignment is to evaluate the improvement in execution time, power and energy consumption for four proposed new versions of the Alphium processor. Here is a description of the alternatives:

- **Low Power Alphium (LPA)**: This would be the same design as the original Alphium but it would be clocked at 133MHz to save power. The power supply (Vdd) would remain 3.3V and the achieved IPC for MisSpec would be 0.8 again.
- **Superscalar Alphium (SSA)**: This would be a 4-way superscalar version of the Alphium. It would have the ability to issue up to 4 instruction per cycle. The power supply (Vdd) would remain 3.3V but the clock frequency would be reduced to 166MHz, due to the complexity of the issuing logic. The achieved IPC for MisSpec would be 2. Assume that the effective capacitance switched in the superscalar design would be 4 times that of the original.
- **Low Voltage Alphium (LVA)**: This would be the same design with the original Alphium again but both the power supply and the clock frequency would be reduced. Power supply (Vdd) would be 2V and clock frequency would be 133MHz. The IPC for MisSpec would be 0.8 once again.
- **Low Voltage Superscalar Alphium (LVSSA)**: This would be a 4-way superscalar version of with reduced power supply. The clock frequency would be 100MHz, the power supply (Vdd) would be 2V and an IPC of 2 would be achieved for MisSpec. Assume that the effective capacitance switched in the superscalar design would be 4 times that of the original.

Here is some information that you may find useful:

Power: When we measure power for a system, we care about the maximum instantaneous power the system can consume. This is important as it determines the maximum current that the power supply must be able to supply to the system and the amount of heat that has to be removed from the system.

Energy: $E=C\times Vdd^2$ is just the energy per transaction. This is not interesting. We care about the energy consumed from the power supply to execute a task (or perform some computation). Once the task is executed, the processor can be turned off and no further energy is needed. The energy per task determines how many tasks you can execute before the battery runs out.

Question 2 (cont)

a) Complete the basic formulas that will allow you to compare the 5 alternatives to the original Alphium. You will need the formulas for execution time, power consumption, energy consumption for the MisSpec benchmark, performance per power ratio for the MisSpec benchmark and performance per energy ratio for the MisSpec benchmark.

(2 points for formula and calculations in b))
Execution Time = $\frac{InstructionsCount \cdot CPI}{Freq} = \frac{InstructionsCount}{IPC \cdot Freq}$

(2 points for formula and calculations in b))
Power = $C \cdot VDD^2 \cdot Freq$

(5 points for formula and calculations in b))
Energy = $Power \cdot ExecutionTime$

(2 points for formula and calculations in b))
Performance per Power = $\frac{Performance}{Power} = \frac{1}{ExtecutionTime \cdot Power} = \frac{1}{Energy}$

(2 points for formula and calculations in b))
Performance per Energy = $\frac{Performance}{Energy} = \frac{1}{ExtecutionTime \cdot Energy}$

Question 2 (cont)

b) Fill in the two following tables. In each box write how the proposed new version compares to the original Alphium for that feature (e.g. $\frac{ExecTime_{new}}{ExecTime_{original}}$, $\frac{Power_{new}}{Power_{original}}$ etc). Two fractional digits per entry are enough. Use the following (blank) page as scratch paper.

	IPC	Freq	Vdd	Relative ExecTime	Relative Power	Relative Energy
LPA	0.8	133MHz	3.3V	1.50	0.66	1.0
SSA	2.0	166MHz	3.3V	0.48	3.32	1.59
LVA	0.8	133MHz	2.0V	1.50	0.24	0.36
LVSSA	2.0	100MHz	2.0V	0.80	0.73	0.58

	IPC	Freq	Vdd	Relative Performance/Power	Relative Performance/Energy
LPA	0.8	133MHz	3.3V	1.00	0.66
SSA	2.0	166MHz	3.3V	0.62	1.30
LVA	0.8	133MHz	2.0V	2.77	1.85
LVSSA	2.0	100MHz	2.0V	2.08	2.60

Question 2 (cont)

c) Circle the right answer (**true** or **false**):

1. (1 point) Without any other changes, lowering the clock frequency of a processor leads to energy savings **False**

2. (1 point) Increasing performance always leads to energy wasting **False**

3. (1 point) Lowering supply voltage can be combined with increasing performance **True**

4. (1 point) Comparing processors using performance per Power is the same as using performance per Energy **False**

d) (1 point) For a server, which of the above metrics (execution time, power, energy, performance/power, performance/energy) would you use to pick a processor? Why?

For a server all you care about is performance. So the execution time is the best metric.

(1 point) For a **laptop computer**, which of the above metrics (execution time, power, energy, performance/power, performance/energy) would you use to pick a processor? Why?

For a laptop we care about acceptable performance at low energy or acceptable energy at high performance. So, performance per energy is the best metric.

(1 point) For **Personal Digital Assistant (PDA)**, which of the above metrics (execution time, power, energy, performance/power, performance/energy) would you use to pick a processor? Why?

Current PDAs have minimal computing requirements so all you care about is energy (battery life). So energy is the best metric.

Question 3

The DiSPlacement is a hypothetical DSP variation of the MIPS architecture. Here are the 3 changes from MIPS:

1. Load and store instructions are changed to have **ONLY** the following two addressing modes:
   (a) Register indirect: the address is the contents of the register. For example:
   `lwi r5, r1` # r5 ← Mem[r1]
   (b) Register autoincrement (ai): the address is the contents of the register; as part of this instruction, increment this register by the size of the data in bytes. Note that the memory address is the ordinal value of the register before incrementing. For example:
   `lwai r5, r1` # r5 ← Mem[r1]; r1 ← r1 + 4
2. There is a new 64-bit register called `Acc`, standing for accumulator.
3. There is a multiply accumulate instruction (`MAC`), which both adds the contents of the `Hi:Lo` to `Acc` and multiplies two 32-bit registers and puts the 64-bit product into the existing registers `Hi:Lo`. For example:
   `mac r3, r4` # Acc ← Acc + Hi:Lo; Hi:Lo ← r3×r4

Putting these extensions together, the unrolled loop of the FIR filter looks like this (assume that `Acc` and `Hi:Lo` are initialized to 0):

```
lwai r5, r1          # r5 ← Mem[r1]; r1 ← r1 + 4
mac r2, r5           # Acc ← Acc + Hi:Lo; Hi:Lo ← r2×r5
lwai r5, r1          # r5 ← Mem[r1]; r1 ← r1 + 4
mac r2, r5           # Acc ← Acc + Hi:Lo; Hi:Lo ← r2×r5
...
```

Since the memory accesses are based on the contents of registers only, the designers of DiSPlacement decided to change the 5-stage pipeline by swapping the EX and MEM stages:

1. Instruction Fetch
2. Instruction Decode/Register Fetch
3. Memory Access
4. Execute
5. Write Back

Assume that the execute stage has a 1 clock cycle multiplier and that the ALU can perform 64-bit additions. The figure on the next page shows the modified pipeline datapath.

Replace this page with displacement pipeline

Question 3 (cont)

As an internationally famous computer designer (this is in 2 years), you are brought in to comment on DiSPlacement.

Here are the questions the management has about DiSPlacement. Please answer clearly and show how you got your answers.

a) What are the pipeline hazards in this modified pipeline?

1. (1 point) Any structural hazards?

   Yes: there is only 1 register write port, but 2 are needed for `lwai`.

2. (1 point) Any control hazards?

   Yes: branches lead to a control hazard, just as they do in the regular MIPS pipeline.

3. For data hazards, look at the following interactions. Which are hazards? When?

   (a) (1 point) Load then Load (same address register)
   There are two: when an autoincremented address is used to address memory in the next instruction, and when the data value loaded from memory is used to address memory in the next instruction.

   (b) (1 point) Load then Branch
   Since there is no forwarding path that will resolve this dependency, there is a hazard.

   (c) (1 point) Load then Arithmetic-logical
   Dependencies can be resolved through forwarding, unless the arithmetic-logical instructions uses the autoincremented address from the load instruction.

   (d) (1 point) Load then Store
   Since there is no forwarding path that will resolve this dependency, there is a hazard.

   (e) (1 point) Arithmetic-logical then Arithmetic-logical
   Dependencies can be resolved through forwarding.

   (f) (1 point) Arithmetic-logical then Store
   There is a hazard, since a later instruction's mem stage can occur before an earlier instruction's execute stage has completed.

   (g) (1 point) Arithmetic-logical then Branch
   There is a hazard, since a later instruction's decode stage can occur before an earlier instruction's execute stage.

Question 3 (cont)

b) (8 points) Remove as many of these hazards as you can, but you are limited to changes in the datapath from the following list:

1. Change the number of read or write ports on the register file;
2. Add one more adder (of whatever width you need);
3. Add multiplexors to the inputs of memory, multipliers, ALUs, or adders.

Do not worry about the control of any changes. In the table below, list the original hazard, hardware changes, and why the change resolves the hazard.

Hazard	Hardware Changes	Why It Resolves
LW → LW (data reg)	Mux and path from Mem/Ex	Allows forwarding
LW → LW (autoincremented reg)	Add adder in Mem stage	Allows incremented value to be available in time for second LW
LW → BR	Add mux to B output of reg file	Allows forwarding
LW → SW	Add mux in Mem stage	Allows forwarding
Structural	Add 2nd regfile write port	Allows two regfile writes
(incorrect) ALU → BR		Can't add a forwarding path to fix this; must stall
(incorrect) ALU → ST		Can't add a forwarding path to fix this; must stall

Question 3 (cont)

c) What is the impact of these changes for non-DSP applications? Specifically, how would the changes affect the clock rate of DiSPlacement, instruction count of traditional programs running on DiSPlacement, or the CPI of the original MIPS instruction set. State assumptions, then estimate instruction count and CPI impact quantitatively.

	Reason For Change	Estimated Impact (Quantitative)
Clock rate (1 point)	More forwarding muxes: slower; 1 cycle multiply: slower; 64 bit adder: slower	
Instruction count (1 point)	No offset on LW/SW: higher; Autoincrement addressing mode: lower	
CPI (1 point)	Stall on ALU→BR and LD→BR: higher	

Question 4

The I/O bus and memory system of a computer are capable of sustaining 1000 MB/s without interfering with the performance of an 700-MIPS CPU (costing $50,000). This system will be used as a transaction processing (TP) system. TP involves many relatively small changes (transactions) to a large body of shared information (the database account file). For example, airline reservation systems as well as banks are traditional customers for TP. Here are the assumptions about the software on the system that will execute a TP benchmark:

- Each transaction requires 2 disk reads plus 2 disk writes.
- The operating system uses 50,000 instructions for each disk read or write.
- The database software executes 500,000 instructions to process a transaction.
- The amount of data transferred per transaction is 2048 bytes.

You have a choice of two different types of disks:

- A small disk (2.5") that stores 1000 MB and costs $60.
- A big disk (3.5") that stores 2500 MB and costs $150.

Either disk in the system can support on average 100 disk reads or writes per second.

You wish to evaluate different system configurations based on a transaction processing benchmark that uses a 20 GB database account file. Answer parts (a)–(e) based on this benchmark. Assume that the requests are spread evenly to all the disks, and that there is no waiting time due to busy disks. Show all work for all parts.

a) (3 points) Complete the table below. "Number of Units" refers to the minimum number of that item required for each organization; "Demand per Transaction" refers to the demand (in MIPS, bytes, or I/Os) that each transaction places on that component; and "TP/s Limit" refers to the maximum number of transactions per second that each subsystem (processor, bus, or disks) could support.

Units	Performance	Number of Units	Demand per Transaction	TP/s Limit
CPU	700 MIPS	1	0.7 MIPS	1,000
Bus	1000 MB/s	1	2048 bytes	512,000
2.5" disks	100 IOs/s	20	4 I/Os	$\frac{20\times100}{4} = 500$
3.5" disks	100 IOs/s	8	4 I/Os	$\frac{8\times100}{4} = 200$

Question 4 (cont)

b) (2 points) How many transactions per second are possible with each disk organization, assuming that each uses the minimum number of disks to hold the account file?

For a 2.5" disk configuration, the maximum TP/s is limited by the disk and is 500 TP/s. So, for a 3.5" disk configuration, the maximum TP/s is also limited by the disk and is 200 TP/s.

c) (5 points) What is the system cost per transaction per second of each alternative for the benchmark?

Since we have already computed the TP/s for each configuration, all we need to do is compute the cost and divide by the number of TP/s.

Organization	Cost	TP/s	Cost per TP/s
CPU + 2.5"	$51,200	500	$102
CPU + 3.5"	$51,200	200	$256

Question 4 (cont)

d) (5 points) How fast must a CPU be in order to make the 1000 MB/sec I/O bus a bottleneck for the benchmark? (Assume that you can continue to add disks.)

Since the current bus can handle 512,000 TP/s and the current CPU can handle 1,000 TP/s, the CPU would have to be: $\frac{512{,}000}{1{,}000} = 512$ times faster than the current CPU. Since the current CPU is 700 MIPS, the new CPU would have to be 358,400 MIPS.

e) (5 points) As manager of MTP (Mega TP), you are deciding whether to spend your development money building a faster CPU or improving the performance of the software. The database group says that they can reduce a transaction to 1 disk read and 1 disk write and cut the database instructions per transaction to 400,000. The hardware group can build a faster CPU that sells for the same amount as the slower CPU with the same development budget. (Assume you can add as many disks as needed to get higher performance.) How much faster does the CPU have to be to match the performance gain of the software improvement?

The software approach would reduce the number of instructions per transaction and the number of disk reads and writes required so that the new TP/s would be:

50,000 $\times$ (1 read + 1 write) + 400,000 = 500,000 inst/transaction

This changes the load on the CPU to 0.5 MIPS per transaction and the overall TP/s would be 1,400 for the CPU. On the other hand, the old approach needed 700,000 instructions per transaction and so the new CPU would have to be $\frac{700{,}000}{500{,}000}$ or 1.4 times faster than the original CPU.