
(a) Give two different reasons why increasing the die (chip) size of a microprocessor increases its cost.

**Answer: [2 points]**
1. Larger die means fewer die per wafer (and much of the cost is per wafer).
2. Larger die have less chance of being defect-free (given random defects across the wafer).

(b) Different ISAs have different numbers of registers, and there is no consensus on what is the “right” number of registers. Give two reasons not to have too many registers (e.g., 1000 registers):

**Answer: [2 points]**
1. Requires more bits in the instructions (leading to larger instructions).
2. A larger register file will be slower to access.

Give one reason not to have too few registers (e.g., 4 registers):

**Answer: [1 point]**
Too few registers limits the compiler's ability to keep values in registers, increasing the number of loads and stores of temporary results, too few registers also makes it difficult to perform coding scheduling.

(c) The instructions in the ARM ISA were originally all 32-bit instructions. Later, ARM released the “Thumb” extension that mixes 16-bit instructions and 32-bit instructions. What issue does the Thumb ISA tackle?

**Answer: [2 points]**
A RISC ISA with all 32-bit instructions (such as ARM) is larger than a variable-length instruction set such as x86. By supporting smaller 16-bit instructions for common operations, the Thumb ISA improves code density (more instructions packed into fewer bytes).

(d) How can a compiler optimization increase performance yet hurt CPI?

**Answer: [2 points]**
It may have removed instruction (improving performance) but the instructions removed could have had lower than average CPI (thus increasing the average CPI of the remaining instructions).

(e) What is the main disadvantage of using the “millions of instructions per second” metric to compare the performance of two processors?

**Answer: [1 point]**
It can’t compare to processors with different ISAs.

(f) How do the SPEC CPU benchmarks avoid this problem?

**Answer: [1 point]**
The SPEC CPU benchmarks are source code programs that can be compiled for different ISAs, allowing the comparison of processors with different ISAs.
2. [ 10 Points ] Performance Calculations

(a) Assume a typical program has the following instruction type breakdown:
   - 35% loads
   - 10% stores
   - 50% adds
   - 3% multiplies
   - 2% divides

Assume the current-generation processor has the following instruction latencies:
   - loads: 4 cycles
   - stores: 4 cycles
   - adds: 2 cycles
   - multiplies: 16 cycles
   - divides: 50 cycles

If for the next-generation design you could pick one type of instruction to make twice as fast (half the latency), which instruction type would you pick? Why?

**Answer:** [2 points]
First, we calculate the CPI adder of each type of instruction:
loads: 1.4, stores: 0.4, adds: 1.0, multiplies: 0.48, divides: 1.0
As loads contribute the most to the CPI, we should half the latency of load operations.

---

**Cache Performance.** As a chip designer, you have two choices: (1) a direct-mapped cache with a two-cycle hit latency or (2) a set-associative cache with a three-cycle hit latency. In either case, the miss latency is 10 cycles.

(a) What is the average memory latency of the direct-mapped cache with a 15% miss rate?

**Answer:** [2 points]
2 + (15% * 10) = 2 + 1.5 = 3.5 cycles

(b) What miss rate must the set-associative cache obtain to have a lower average latency than the direct-mapped cache with a 15% miss rate?

**Answer:** [2 points]
3 + (x * 10) = 3.5
(x * 10) = 0.5
x = 0.05 = 5%

---

**Pipelining and Clock Frequency.** Consider a workload with 30% branches and a 66.66% branch prediction accuracy (33.33% misprediction rate). You may ignore memory operations.

(a) For a simple five-stage pipeline, what is the CPI of this workload assuming a three-cycle mis-prediction penalty and a single-cycle latency for all other instructions?

**Answer:** [2 points]
1 + (30% * 33.33% * 3) = 1.3 CPI

(b) Calculate under what condition a much deeper pipeline with double the core clock frequency will outperform the shallower pipeline?

**Answer:** [2 points]
At double the clock frequency, it could double the CPI (2.6 CPI) and still perform the same:
1 + (30% * 33.33% * N) = 2.6 CPI
Solving for N, give N = 16 cycles. Thus, only when the mis-prediction penalty is 16 cycles or less is the processor with the faster clock is faster.
3. **[9 Points] Carry-Select Adders.** When analyzing carry-select adders in class and in the homework assignment, we assumed that (i) a multiplexer (mux) had a delay of two gate delays and (ii) an \( n \)-bit ripple-carry adder has a delay of \( 2n \) gate delays. Assume for this question that a mux has just a single gate delay. That is, a mux has a delay of just 1 (rather than 2) but a \( n \)-bit ripple-carry adder still has a delay of \( 2n \).

(a) Using the above assumptions, calculate the delay of a 12-bit carry-select adder with two six-bit segments?

**Answer: [1 point]** The delay of each six-bit adder (12 units) plus a multiplexor (1 unit) is a total of 13 units of delay.

(b) Using the above assumptions, calculate the delay of a 12-bit carry-select adder with three four-bit segments?

**Answer: [1 point]** The delay of each four-bit adder (8 units) plus two multiplexors (2 units) is a total of 10 units of delay.

(c) Using the above assumptions, calculate the minimal size (in terms of number of bits) in which the carry-select adder is faster than a ripple-carry adder? Give the configuration and total delay of this adder.

**Answer: [2 points]** A 2-bit carry-select adder (with two 1-bit segments) has a gate delay of 3 (whereas the ripple-carry would be 4 gate delays).

(d) Using the above assumptions, calculate the largest carry-select adder that can be built with 5 gate delay worst-case delay? How long is each of the segments? (non-uniform segment sizes are allowed.)

**Answer: [3 points]** Working from the 2-bit (1, 1) three-delay carry-select adder above, a four-delay adder could consist of only another 1-bit adder (1, 1, 1) for a total delay of four. For a five-delay adder, we can add a 2-bit segment (1, 1, 1, 2) for a total of 5 bits in a five-delay adder.

(e) Using the above assumptions, calculate the largest carry-select adder that can be built with 7 gate delay worst-case delay? How long is each of the segments? (non-uniform segment sizes are allowed.)

**Answer: [2 points]** Working from the 5-bit (1, 1, 1, 2) five-delay carry-select adder above, we add another segment of size 2 for a six-delay 7-bit (1, 1, 1, 2, 2) adder. For a 7-delay adder, we can add a three-bit segment, for a total 10-bit adder (1, 1, 1, 2, 2, 3) with 7 units of delay.
4. [ 14 Points ] Performance & ISAs.

Consider the implications of adding a three-input addition, \( \text{ADD3} \), to your favorite ISA (say, LC4 or MIPS). This new instruction operates on registers only and performs the computation \( A = B + C + D \).

(a) What are the two most significant changes to add such an instruction to a single-cycle LC4 datapath?

Answer: [4 Points] A second adder (or three-input adder) in the ALU plus an additional register read port. We also accepted “change the ISA instruction encoding to support four register specifiers”. (As an aside, LC4 has just enough space in its 16-bit instruction encoding to support a four-bit opcode plus four three-bit registers, so it would actually be feasible to add this instruction to LC4.)

(b) Beyond the two changes changes you listed for the single-cycle design, what are the two most significant additional changes to add such an instruction to our standard five-stage pipelined datapath (the number of stages is unchanged and \( \text{ADD3} \) executes entirely in the “X” stage)?

Answer: [4 Points] The stall logic needs to be extended to check for dependencies on a third input register. Bypassing datapath and control must be added for a third input register.

(c) An alternative implementation would be to split the execution of \( \text{ADD3} \) over two cycles (“X” and “M”, completing at the end of the “M” stage). Explain the potential performance advantage and disadvantage of such an implementation versus the implementation of \( \text{ADD3} \) executing entirely in the “X” stage.

Answer: [4 points] The main advantage is that it would likely have a faster clock, whereas performing back-to-back adds in a single cycle would likely hurt the clock. The main disadvantage is that it would hurt the CPI, because a \( \text{ADD3} \) would now have a two-cycle latency and introduce a stall cycle if the immediately following instruction is dependent (many student answers implied that \( \text{ADD3} \) would only impact load stalls, which is not quite right).

(d) Yet another way of implementing \( \text{ADD3} \) would be by using micro-ops. What would be the key advantage and disadvantage of implementing \( \text{ADD3} \) in this way?

Answer:[2 Points] Advantage: avoids all the modifications throughout the pipeline with a simpler change to the decode stage. Disadvantage: using the instruction wouldn’t be any faster than just doing two normal two-input \( \text{ADD} \) instructions back-to-back.

The standard pipeline discussed in class has five stages: fetch (F), decode (D), execute (X), memory (M), and writeback (W). Consider the following four-stage pipelines, formed by combining two stages of the classic five-stage pipeline.

(a) What specific positive impact would combining the “D” & “X” stages have on CPI?

**Answer:** [2 points] Reduce the branch misprediction penalty from two to one.

(b) What specific positive impact would combining the “X” & “M” stages have on CPI?

**Answer:** [2 points] Eliminate the load-to-use stall cycle.

(c) What specific positive impact might combining the “M” and “W” stages have on the clock frequency of the pipeline?

**Answer:** [2 points] By combining these stages, it removes a bypassing path, reducing the size of the bypass muxes in the “X” stage. If the “X” stage was the slowest stage, this could improve the clock frequency of the pipeline.

(d) What specific impact would combining the “F” and “D” stages have on the implementation of branch prediction?

**Answer:** [2 points] As decode is part of the fetch cycle, a branch target buffer (BTB) wouldn’t be needed to determine which instructions are branches or what their target would be.

(e) If a single-cycle datapath with delay of \( n \) nanoseconds is converted into a pipelined datapath with \( p \) stages, the clock period will be longer (slower) than \( n/p \) nanoseconds. Give two reasons for this:

**Answer:** [2 points] (1) The overhead of the pipeline latches. (2) The logic is likely not evenly divided among all the pipeline stages (that is, not all pipeline stages will have the same delay).

(f) What is a structural hazard? Describe two approaches a hardware designer might use to tackle the problem of structural hazards.

**Answer:** [3 points] A structural hazard occurs in a pipeline when two instructions wish to use the same resource (or “structure”) in the same cycle. There are three general solutions to work around structural hazards:

1. Detect & stall. Allow one instruction to proceed and stall the others. For example, if a load instruction conflicts with an instruction fetch, the processor can stall the fetch and allow the load to proceed.
2. Add more resources. Replicate the resource so that both instructions can proceed in parallel. This could be by having separate instruction and data memories, as discussed in class, or by having separate ports to a single memory.
3. Pipeline organization. Schedule access to the resource so that conflicts don’t arise. The 5-stage pipeline discussed in class does this for the register file write port by delaying all register writes until the W stage (ALU instructions could write the register file in M, but this could cause a structural hazard with a preceding load instruction trying to write the register file in W).
6. [ 9 Points ] Caches

Consider a processor whose ISA uses 16-bit addresses and byte-granularity addressing (each address specifies a byte in memory). The processor has a direct-mapped 16KB cache with 16-byte cache blocks. The cache is initially empty (all blocks invalid). As a reminder, for such a cache the 16-bit address is divided into a 2-bit tag, a 10-bit index, and a 4-bit offset.

(a) Consider a short (two-instruction) program that loads from the following two addresses. The cache starts empty, so the first load will always miss. However, will the second load hit or miss in the cache? (circle the correct answer)

Load from address: 0010 1010 0111 1000 (miss)
Load from address: 0010 1010 0101 1010 hit? --or-- miss?

(b) Now assume the processor’s cache has 256-byte blocks (but the size of the cache is unchanged). Will the second load hit or miss in the cache? (circle the correct answer)

Load from address: 0010 1010 0111 1000 (miss)
Load from address: 0010 1010 0101 1010 hit? --or-- miss?

(c) Now assume the processor’s cache has 4096-byte blocks (but the size of the cache is unchanged). Will the second load hit or miss in the cache? (circle the correct answer)

Load from address: 0010 1010 0111 1000 (miss)
Load from address: 0010 1010 0101 1010 hit? --or-- miss?

Answer: [2 points] (a) miss, (b) hit, (c) hit. Grading: 2 points for all three correct, 1 point for two correct, zero points otherwise.

(d) What program property is responsible for the decrease in miss rate?

Answer: [1 point] Spatial locality

(e) Consider a different program that loads from the following three addresses. As before, the cache starts empty, so the first load will always miss. However, will the second and third loads hit or miss in the cache with 16-byte blocks? (circle the correct answers)

Load from address: 0010 1010 0111 1000 (miss)
Load from address: 1010 1100 0101 1110 hit? --or-- miss?
Load from address: 0010 1010 0101 1010 hit? --or-- miss?

(f) Now assume the processor’s cache has 256-byte blocks, but the size of the cache is unchanged. (circle the correct answers)

Load from address: 0010 1010 0111 1000 (miss)
Load from address: 1010 1100 0101 1110 hit? --or-- miss?
Load from address: 0010 1010 0101 1010 hit? --or-- miss?

(g) Now assume the processor’s cache has 4096-byte blocks, but the size of the cache is unchanged. (circle the correct answers)

Load from address: 0010 1010 0111 1000 (miss)
Load from address: 1010 1100 0101 1110 hit? --or-- miss?
Load from address: 0010 1010 0101 1010 hit? --or-- miss?

Answer: [3 points] (a) miss, miss, (b) miss, hit, (c) miss, miss. Grading: each part is one point, and both answers must be correct to get the point.
(h) What hardware phenomenon is responsible for the increase in miss rate?

**Answer:** [1 point] Cache conflict

(i) Other than adjusting the block size, what is the most reasonable way to change the cache hardware to remedy this degradation in miss rate?

**Answer:** [2 points] Implement a set-associative cache

### Distribution

Mean: 46.4 points (70%) — Median: 47 points (71%) — High: 64 (97%)