1 Potpourri

1.1 Processor Design [20 points]

Circle the lines including terms that are compatible with each other and it makes sense for a processor design to include both.

- [✓ superscalar execution — in-order execution] 2 points
- [✓ superscalar execution — out-of-order execution] 2 points
- single-cycle machine — branch prediction 2 points
- [✓ reservation station — microprogramming] 2 points
- [✓ fine-grained multithreading — single-core processor] 2 points
- Tomasulo's algorithm — in-order execution 2 points
- precise exceptions — out-of-order instruction retirement 2 points
- branch prediction — fine-grained multithreading 2 points
- direct-mapped cache — LRU replacement policy 2 points
- [✓ fine-grained multithreading — pipelining] 2 points

1.2 Pipelining [6 points]

What are the three major causes of pipeline stalls?

Data/Control Flow dependences (other possible answer: Data flow dependence) 2 points

Multi-cycle operations (other possible answer: Control flow dependence) 2 points

Resource contention 2 points

1.3 Caches I [5 points]

Please reason about the following statements about a possible processor cache one can design.

Can a cache be 5-way set associative?

[✓ YES] NO

Explain your reasoning. Be concise. Show your work.

Answer: we just need 5 tag comparators.
Explanation: Nothing wrong with a non-power-of-two associativity.

1.4 Caches II [10 points]

Assume a processor where instructions operate on 8-byte operands. An instruction is also encoded using 8 bytes. Assume that the designed processor implements a 16 kilo-byte, 4-way set associative cache that contains 1024 sets.

How effective is this cache? Explain your reasoning. Be concise. Show your work.

Answer:
1) The cache requires two accesses to be effective. (5 points)
2) The cache cannot exploit spatial locality. (5 points)
Explanation: The cache has $4 * 1024 = 4096$ cache lines in total. That means, each cache line is $16KB/4096 = 4$ bytes. With 4-byte cache lines, each operand and each instruction needs to be stored in two cache lines, which will require 2 accesses to the cache for each load/store operation and instruction fetches. The cache cannot exploit spatial locality, but only can provide benefit by exploiting temporal locality (albeit requiring two accesses).

1.5 Performance Analysis [15 points]

A multi-cycle processor executes *arithmetic instructions* in **5 cycles**, *branch instructions* in **4 cycles** and *memory instructions* in **10 cycles**. You have a program where 30% of all instructions are arithmetic instructions, 35% of *all instructions* are memory instructions, and the rest are branch instructions. You figured out that the processor cannot execute the program fast enough to meet your performance goals. Your goal is to reduce the execution time of this program by at least 10%. Hence, you decide to change the processor design to improve the performance of **arithmetic instructions**.

In the new processor design, **at most** how many cycles should the execution of **a single arithmetic instruction** take to reduce the execution time of the *entire program* by **at least** 10%? Show your work.

Answer: 2 cycles. (10 points)
Explanation: Let the total number of instructions be X.
The processor will execute the program in:
$5 * 0.3 * X + 4 * 0.35 * X + 10 * 0.35 * X = 6.4 * X$ cycles.
To improve the execution time by 10%, the program should complete in:
$6.4 * X * 0.1 = 0.64 * X$ less cycles.
The cost of executing the arithmetic instructions was $5 * 0.3 * X = 1.5 * X$ cycles. To improve the program's performance by 10%, the arithmetic instructions should complete execution at least in $1.5 * X - 0.64 * X = 0.86 * X$ cycles. Hence, $A * 0.3 * X <= 0.86 * X, A <= 2.87$, where A is the new number of cycles that the processor should execute an arithmetic instruction in. So, to improve the overall performance by 10%, an arithmetic instruction needs to execute 3 cycles faster. Hence, it should take at most 5 - 3 = 2 cycles. (a correct explanation that proves the student's understanding may receive 13/14 points.)

1.6 Microprogrammed Design [4 points]

In lecture, we discussed a design principle for microprogrammed processors. We said that it is a good design principle to generate the control signals for cycle $N + 1$ in cycle N.

Why is this a good design principle? Be concise in your answer.

Answer: Likely keeps the critical path short (it follows the critical path design principle).
Explanation: By generating the control signals in advance, we can make the critical path of the circuit likely shorter. Shorter critical path can increase the frequency of the processor.

1.7 Processor Performance [10 points]

Assume that we test the performance of two processors, A and B, on a benchmark program. We find the following about each:

- Processor A has a CPI of 2 and executes 4 Billion Instructions per Second.
- Processor B has a CPI of 1 and executes 8 Billion Instructions per Second.

Which processor has higher performance on this program? Circle one.
Recall that CPI stands for Cycles Per Instruction.

```
A. Processor A
B. Processor B
C. They have equal performance
```
[✓ D. Not enough information to tell]

Explain concisely your answer in the box provided below. Show your work.

Answer: Neither of these metrics nor their combination provide execution time.
Explanation: Although information about the CPI and the instructions/second is provided, it is not enough to reason about the processors' performance. The processors may support different Instruction Set Architectures, in which case the benchmark program will be compiled into a different assembly code. The fact that one of the processors execute more instructions per second does not necessarily mean that the processor makes more progress on the benchmark.