A1 –Architecture: Memory Performance (25 points)

The average memory access time formula (given below) provides us with a framework to present cache optimizations and techniques for improving cache performance.

Average memory access time = hit time + miss rate * miss penalty

where:

Hit time is the time to hit in the cache
Miss rate is the fraction of access that is not in the cache
Miss penalty is the additional time to serve the miss

IMPORTANT NOTES: Write clearly, be concise, and report only the information asked. Additional information (not required) will be considered as the indication that you have not correctly understood the question and therefore will penalize your final score.

a. (6 points) Briefly explain why a smaller and simpler cache can reduce the hit time.

Make sure that in your explanation you address both the impact of a smaller cache and the impact of a simpler cache.

b. (6 points) Briefly explain why larger block sizes and higher associativity can decrease miss rate.

Make sure that in your explanation you address both the impact of larger block sizes cache and the impact of higher associativity.

c. (6 points) Briefly explain why a multi-level cache can reduce miss penalty.

d. (7 points) Explain why a fully associative schema for block placement increases the hit time and therefore the vast majority of processor caches today are direct mapped, two-way set associative, or four-way set associative.

A2 Architecture: Dynamic Programming with Scoreboard (25 points)

A major limitation of the simple five-stage pipeline is that it uses in order instruction issue, execution, and completion. In scoreboard, instructions execute out of order when there are sufficient resources and no data dependencies. Each instruction undergoes four steps, which replace the ID, EX, and WB steps in the standard five-stage pipeline.

IMPORTANT NOTES: Write clearly, be concise, and report only the information asked. Additional information (not required) will be considered as the indication that you have not correctly understood the question and therefore will penalize your final score.

a. (8 points) List the four steps and briefly describe them. Clearly identify what hazards are resolved in each step (if any).

b. (8 points) A scoreboard uses the available ILP to minimize the number of stalls arising from the program's true dependencies. In eliminating stalls, a scoreboard is limited by four factors. List the four factors and briefly describe them.

c. (9 points) Assume a basic scoreboard architecture with scoreboard controlling instruction execution. All data flows between the register file and the functional units over the busses. There are two FP multipliers, an FP divider, two FP adders, and an integer unit for memory references, branches, and integer operations. Also assume the following execution cycles latencies for the FP functional units:

- Adder: 1 clock cycles
- Multiplier: 3 clock cycles
- Divider: 10 clock cycles

Consider the code below:

```
MUL.D     F0, F6, F4
SUB.D     F8, F0, F2
ADD.D     F2, F10, F2
```

Show the state of the instruction status table and register result status table at the end of cycle 6 (right before cycle 7). Briefly describe the final status of the three instructions.

IMPORTANT NOTES:

- In the instruction status table, identify any stage that is stalled (use "stall" when needed) and use numbers to represent the cycles when the instruction was able to complete the stage.
- Assume that an instruction waiting for a result from another functional unit CANNOT pass through read operands at the same time the result is being written (in other words, no forwarding is available).

A3 Architecture - Dynamic Programming with Tomasulo's Approach (25 points)

A major limitation of the simple five-stage pipeline is that it uses in order instruction issue, execution, and completion. The Tomasulo's schema combines key elements of the scoreboard schema with the introduction of register renaming.

IMPORTANT NOTES: Write clearly, be concise, and report only the information asked. Additional information (not required) will be considered as the indication that you have not correctly understood the question and therefore will penalize your final score.

a. (8 points) List the three steps in the Tomasulo's approach and briefly describe them. Clearly identify what hazards are resolved in each step (if any).

b. (8 points) Describe how the Tomasulo's schema eliminates WAW and WAR.

c. (9 points) Tomasulo's algorithm has a major disadvantage versus scoreboard: only one result can complete per clock, due to the CDB. Assume the hardware configuration in Figure 1 and Figure 2 for the scoreboard and the Tomasulo's schema respectively. Assume the latencies from Table 1.

Give a code fragment that uses no more than 10 instructions and shows the problem, i.e., the scoreboard does not stall and the Tomasulo's algorithm stalls. Show the Tomasulo's instruction status table with the status of the instructions immediately after the problem occurs. Explain with your words how the execution will be handled and the conflict will be solved.

Table 1. Pipeline latencies where latency is the number of cycles between producing and consuming instruction.

Instruction producing result	Instruction using result	Latency in clock cycles
FP multiply	FP ALU op	6
FP add	FP ALU op	4
FP multiply	FP store	5
FP add	FP store	3
Integer operation (including load)	Any	0

HELP: In such a code, two instructions attempt to write their results in the same cycle, and do not use the same group of functional units in the scoreboard system.

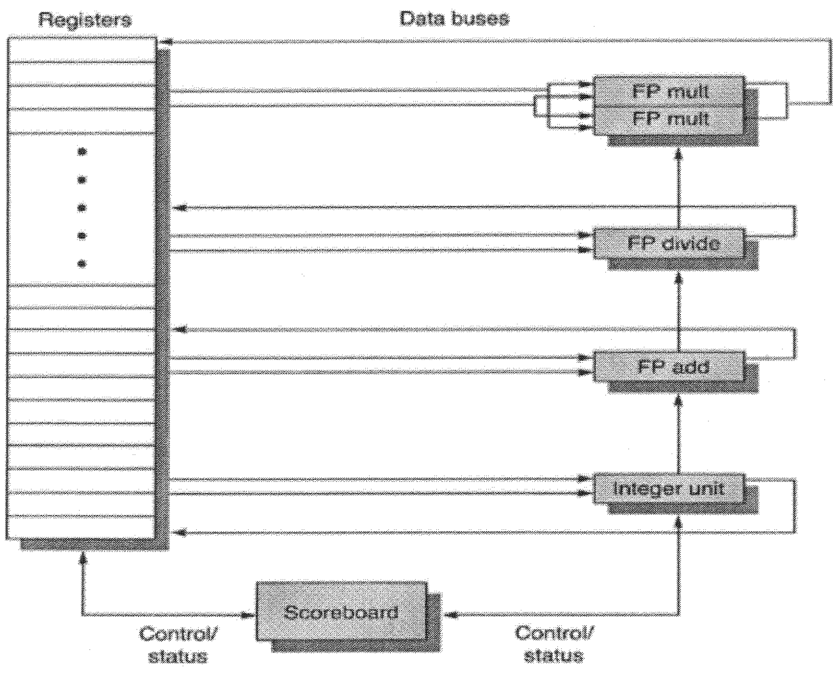

Figure 1. Scoreboard schema

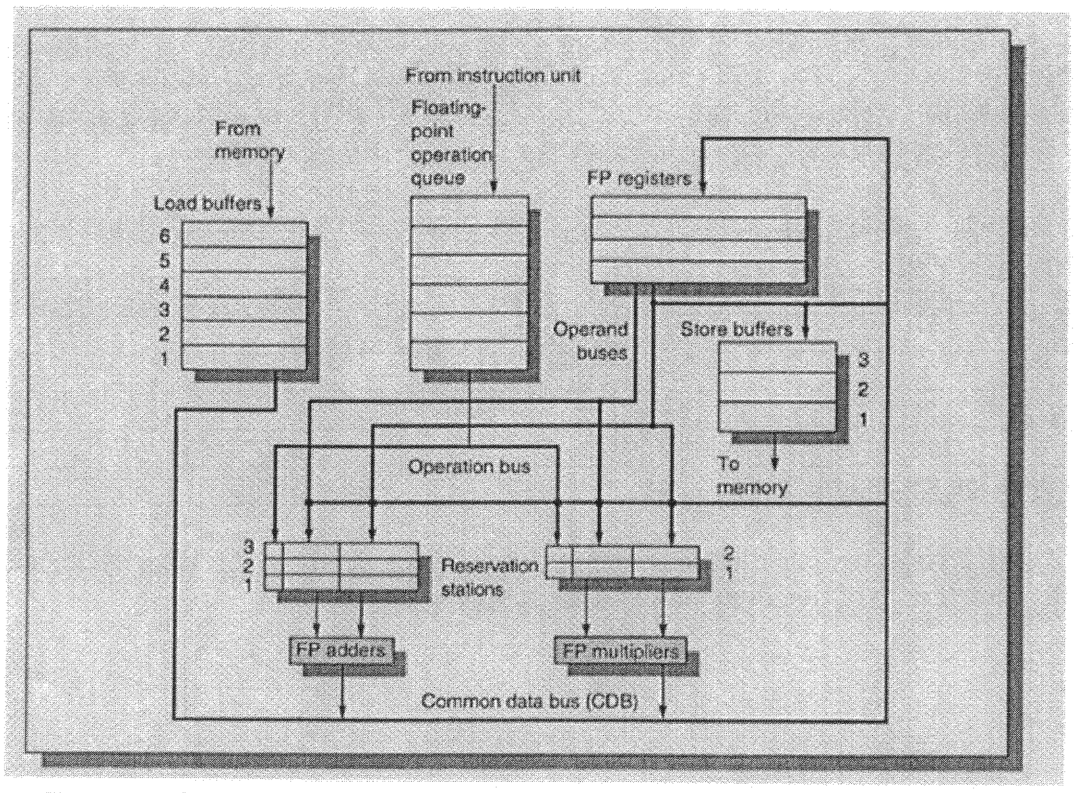

Figure 2. Tomasulo's schema

A4 Architecture – Instruction Set Architectures (25 points)

a. (9 points) For each of the three classes of instruction set architectures listed below, describe its characteristics in terms of operand locations and operating style. Illustrate each architecture using a simple diagram.

- Accumulator
- Stack
- Load-store (or register-register)

b. (10 points) Assume that instructions are an integral number of bytes in length. The opcode is 1 byte. Memory access uses direct addressing. All variables are in memory initially.

For each instruction set architecture, write the best assembly language code for the following high-level-language code sequence:

A = A+B;
C = D-A;

IMPORTANT NOTES:

- Write clearly and comment each line of your code fragments.
- Use A, B, C, and D to represent the memory addresses of the variables.
- For the accumulator code, assume that these instructions are available:
  - loada X: load variable X from memory in accumulator
  - adda X: add variable X in memory to accumulator
  - storea X: store accumulator to variable X in memory
  - negatea: negate value in accumulator
- For the stack code, assume these instructions are available:
  - push X: push X from memory onto stack
  - pop X: pop X from stack in memory
  - add: top stack = X + Y
  - sub: top stack = X-Y

  where X represents a variable stored in memory.
- For the load-store code, assume these instructions are available:
  - load r1, X
  - add r1, r2, r3: r1=r2+r3
  - store X, r1:
  - sub r1, r2, r3: r1=r2-r3

  where X represents a variable stored in memory
- All results must be stored in memory at the end of the code fragment execution.

c. (6 points) Assume again that instructions are an integral number of bytes in length. The opcode is 1 byte. Memory access uses direct addressing. All variables are in memory initially.

Assume the memory address and the data operands are both 16 bits in length. And, there are 16 registers for architecture(s) with many general-purpose registers.

For each architecture, answer the following questions:

- How many instruction bytes are fetched?
- How many data bytes are transferred between memory and CPU?

Show and explain your work.