A1. Architecture (25 points)

Consider a classic five-stage integer pipeline extended to handle floating-point operations. The five stages are: Instruction Fetch (IF), Instruction Decode / Register Fetch (ID), Execution (EX), Memory (MEM), and Write Back (WB). Assume the following:

* All floating-point operations operate on double values and take 4 clock cycles in the execution stage. All other stages as well as integer operations, including memory accesses to load and store double values, take 1 clock cycle.
* The functional units are fully pipelined or replicated, so that an operation of any type (either integer or floating point) can be issued on every clock cycle and there are no structural hazards.
* Data forwarding.
* One-cycle delayed branches.

(a) (5 points) For each of the following pairs of instructions, calculate the latency in clock cycles between the instruction producing the result and the instruction using the result.

Instruction producing results	Instruction using results
Integer ALU op	Another Integer ALU op
Integer ALU op	Store Integer
Load Integer	Integer ALU op
Load Integer	Store Integer
Integer ALU op	Conditional branch
FP ALU op	Another FP ALU op
FP ALU op	Store double
Load double	FP ALU op
Load double	Store double

(b) Consider the following C language code and the straightforward MIPS/DLX assembly code generated by an unoptimized C compiler.

C code:

```
for (i=1000; i>0; i=i-1)
      x[i] = x[i] + d;
```

MIPS/DLX code:

```
Loop: L.D    F0, 0(R1)        ; Load the double value in x[i] to F0
      ADD.D  F4, F0, F2       ; Add two double values: F4=F0+F2
                              ; (F2 contains d)
      S.D    F4, 0(R1)        ; Store the double value F4 in x[i]
      DADDUI R1, R1, #-8      ; Integer subtraction: R1=R1-8
      BNE    R1, R2, Loop     ; Jump to Loop if R1 != R2
      NOP                     ; No operation
```

(i) (6 points) Show the pipeline timing diagram for this instruction sequence.

(ii) (7 points) Schedule the code to minimize pipeline stalls. Again show the pipeline timing diagram for the improved code sequence.

(iii) (7 points) Unroll the loop so that there are four copies of the loop body. Eliminate any obviously redundant computations and do not reuse any of the registers. The goal is to minimize the number of cycles needed to execute the unrolled loop. You don't need to draw the pipeline timing diagram here.

CORRECTION TO A1. Architecture Question

ASSUMPTION NO. 2 should state:

*The functional units are fully pipelined or replicated, so that an operation of any type (either integer or floating point) can be issued on every clock cycle and there are no structural hazards except that the memory system can allow at most one instruction fetch and one data access in a single cycle.

A2. Architecture (25 points)

(a) (13 points) Consider a directory-based cache-coherence protocol in which the directory entry that corresponds to each memory block may have one of three states: "uncached," "shared," and "exclusive." Draw the basic finite state machine for the state transitions for such a directory entry. You do not need to consider the necessarily complex implementation details such as deadlock issues and you may assume the transitions are atomic. For each transition, clearly describe the triggering condition and message(s) exchanged. For each message, clearly identify the source and destination which could be "home directory," "local cache," or "remote cache."

Briefly discuss the pros and cons of such a protocol compared to a snoopy cache-coherence protocol.

(b) (12 points) This problem is on hardware support for program synchronization. Briefly describe how Load Linked (LL) and Store Conditional (SC) instructions are used in newer architectures to support synchronization.

Write a simple sequence of code in a MIPS/DLX like assembly language to illustrate how Load Linked & Store Conditional can be used to implement atomic exchange. The code should keep on trying in a loop until the atomic exchange succeeds. Make sure you comment your assembly code. You may find sample MIPS/DLX assembly code in Problem A1.

What are the advantages of using LL and SC compared to older instructions, such as "atomic exchange."

A3. Architecture (25 points)

(a) (7 points) Explain what is meant by
i. structural hazards,
ii. data hazards, and
iii. control hazards
and the effect of such hazards upon pipeline speedup.

(b) (12 points) Briefly discuss how Tomasulo's Algorithm copes with non-memory related data hazards and comment on the effectiveness of the algorithm in dealing with each data hazard type. Organize your answer according to the following structure.

	Issue	Execute	Write Result
A. RAW			
B. WAR			
C. WAW			

(c) (6 points) Briefly explain what is "branch folding", and draw a pipeline timing diagram for a classic five-stage pipeline to show how "branch folding" improves performance.

A4. Architecture (25 points)

Consider a processor with in-order execution that runs at 1 GHz. The only instructions that read or write data from memory are loads (20% of all instructions) and stores (5% of all instructions). The memory system for this computer is composed of a split L1 cache, a unified L2 cache, and main memory.

The L1 cache imposes no penalty on hits. Both the I-cache and D-cache are direct mapped and hold 32 KB each. The I-cache has a 2% miss rate and 32-byte blocks, and the D-cache is write through with a 5% miss rate and 16-byte blocks. There is a write buffer on the D-cache that eliminates stalls for 95% of all writes.

The hit time for the unified L2 cache is 19 ns for the references made by the L1 D-cache and 23 ns for the references made by the L1 I-cache. Of all memory references sent to the L2 cache in this system, 80% are satisfied without going to main memory. The L2 cache is write-back and 50% of all blocks replaced are dirty. It takes 68 ns to load or replace a L2 cache block.

(a) (6 points) What is the overall miss rate for the split L1 cache? What are the local miss rate and global miss rate for the L2 cache?
(b) (7 points) What is the average memory access time for data reads?
(c) (7 points) What is the average memory access time for data writes?
(d) (5 points) Contrast the use of Miss Rate versus Average Memory Access Time (AMAT) as measures of cache performance. What are the strengths and weaknesses of each measure? Give two examples of how a change in cache design that results in improvement in the miss ratio might actually degrade the AMAT?