A1. Architecture (25 points)

We are considering pipelining the MIPS processor to enhance its performance. The planned pipeline will have six stages: IF, ID, ALU1, MEM, ALU2, WB. Effective address calculation and branch target calculation is performed in ALU1 stage. All arithmetical/logical operation and branch resolution (i.e. deciding if the branch is taken or not) are performed in ALU2 stage. The only instructions that can access the memory are LOAD and STORE.

a) (8 points) What is the maximum achievable speedup? List all possible types of hazards that can lower the speedup below its maximum value. Illustrate each of the hazards with a short code sequence. Assume that the first register specified in the instruction is the destination register.

b) (4 points) How large is the branch penalty in this processor? Explain what is Branch Target Buffer (BTB) and how it can be used to reduce the branch penalty.

How large is the branch penalty in this processor if we use BTB with branch folding?
How large is the penalty with BTB but without the branch folding?
Don't forget to consider both the correctly and incorrectly predicted branches.

c) (7 points) We are considering an enhancement so that the branch target address and the outcome are both calculated in the ID stage. This enhancement can only be applied to "zero-branches" - those branches that compare contents of a register with zero.
Assume that 40% of all instructions are branches. Effective CPI for the branches equals the branch penalty and the other instructions have the effective CPI o f 1.2.

Which percentage of branches must be zero-branches to achieve speedup of 2 in the enhanced processor when compared to the original processor (from part a)?

d) (6 points) Explain the difference between local and global (correlating) predictors. Give a sample code sequence where a local predictor is likely to generate more accurate prediction than a global predictor. Now give a sample code sequence where a global predictor is likely to generate more accurate prediction than a local predictor. You can use either C or MIPS instructions.

A2. Architecture (25 points)

You are given a system with 2 GHz processor clock rate and ideal CPI of 1.2. Load and store operate on 16-bit words only and instruction size is 16 bits. The system has 2-level cache organization. Both caches are unified and contain a mix of instructions and data. L1 cache is 512 KBytes, with block size 2 Bytes (16 bits) and miss rate 5%. L2 cache is 1 MByte with block size 8 Bytes (64 bits) and miss rate 30%. L1 is write-through cache and has a write buffer that eliminates 95% of all writes. L2 cache is write-back and on the average 50% of blocks are dirty.

Access time to L1 cache is 1 ns, and L1 is connected to CPU with a 800 MHz bus that can transfer 16 bits in one bus cycle.
Access time to L2 cache is 10 ns, and L2 is connected to L1 with a 400 MHz bus that can transfer 16 bits in one bus cycle.
Access time to memory is 100 ns, and memory is connected to L2 with a 100 MHz bus that can transfer 32 bits in one bus cycle.

a) (3 points) What is L1 hit time in case of a read?
b) (3 points) What is L2 hit time?
c) (3 points) What is L1 hit time in case of a write (this should include the overhead for write-through to L2 only)?
d) (3 points) What is L2 miss penalty?
e) (3 points) What is L1 miss penalty?
f) (3 points) On this processor what is the average (data or instruction) read time?
g) (3 points) On this processor what is the average data write time?
h) (4 points) Assume that in our common workload, we have 20% of loads and 10% of stores.
Also assume that every instruction must pay the ideal CPI. What is the average CPI?

A3. Architecture (25 points)

a) (10 points) Explain how Tomasulo's algorithm eliminates WAW and WAR hazards.

b) (10 points) Explain how Tomasulo's algorithm uses reorder buffer to support speculative execution.

c) (5 points) Which problems does speculative execution bring with regard to exception handling?

A4. Architecture (25 points)

Consider the following loop:

```
      ADD R2, R0, R0
      DADDUI R6, R0, #100
Loop: LD R3, 100(R1)
      MUL R5, R4, R3
      ADD R8, R5, R7
      SD R8, 100(R1)
      DADDUI R1, R1, #-8
      DADDUI R2, R2, #1
      BNE R2, R6, Loop
```

Assume that the first register specified in the instruction is the destination register. Also assume the following delays:

Load followed by ALU operation using the result = 1 cycle
MUL followed by ADD = 0 cycles
ADD followed by SD = 2 cycles
ALU followed by the branch on the result value = 1 cycle
Branch penalty of 1 cycle

a) (6 points) How many cycles does it take for one loop iteration (part that repeats) to execute without any optimizations?

b) (7 points) Rearrange loop instructions (but do not unroll the loop) to minimize number of stalls. You may schedule one instruction in branch delay slot to eliminate branch penalty. How long does it take now to execute one loop iteration?

c) (8 points) Unroll the loop from b) only as many times as necessary to eliminate all stalls (but not more!) and schedule it. How many cycles does one (original, not unrolled) iteration take now?

d) (4 points) Explain the difference between loop unrolling and software pipelining.