Question# 1 (12 points)

Suppose we build an optimizing compiler that discards 50% of the ALU instructions (but cannot reduce other instructions). Assume that the original total instruction count is $3 \times 10^9$ and that the original ALU instruction count is $2 \times 10^9$. Let the clock rate be 1-GHz, let each ALU instruction take 1 clock cycle, and let each non-ALU instruction take 2 clock cycles.

a) Calculate the original MIPS rate and execution time.

b) Calculate the new MIPS rate and execution time when we use the optimizing compiler.

c) Discuss the results in (a) and (b). Are there any contradictions?

Ans.

a. Original MIPS rate and execution time:

Execution Time = Instruction Count * Cycles per instructions * Clock cycle time
= $(2 \times 10^9 \text{ instructions} \times 1 \text{ cycle/instruction} + 1 \times 10^9 \text{ instructions} \times 2 \text{ cycles/instruction}) / 10^9 \text{ cycle/s}$
= 4s

MIPS rate = $\frac{\text{Instruction Count} / 10^6}{\text{Execution Time}} = \frac{3000}{4} = 750 \text{ MIPS}$

b. New MIPS rate and Execution time when using optimizing compiler

Execution Time = Instruction Count * Cycles per instructions * Clock cycle time
= $(1 \times 10^9 \text{ instructions} \times 1 \text{ cycle/instruction} + 1 \times 10^9 \text{ instructions} \times 2 \text{ cycles/instruction}) / 10^9 \text{ cycle/s}$
= 3s

MIPS rate = $\frac{\text{Instruction Count} / 10^6}{\text{Execution Time}} = \frac{2000}{3} = 666 \text{ MIPS}$

C. Discuss results of a and b:

At first, it would seem that a decrease in MIPS rate would lead to an increase in execution time, which is clearly not the case here. However, one notes the optimizing compiler eliminated the most lightweight instructions and so the MIPS rate will decrease as the instructions that take multiple instructions become more dominant. This reinforces the idea that MIPS rate is not a reliable indicator of performance in applications.
**Question #2 (12 points)**
Consider the simple 5-stage integer pipeline.

a) What is data forwarding?

b) Write a MIPS assembly code to explain how data forwarding may eliminate the data hazard stall cycle.

c) Write a MIPS assembly code to show an example where data forwarding cannot completely eliminate the data hazard stall cycles.

**Ans.**

a) What is data forwarding?

Forwarding is making the data available to subsequent instructions as soon as the computation is complete and allowing instructions to receive this data in the beginning of the EX stage instead of retrieving it in ID. Thus, the results of the ALU and MEM register are given as possible source operands to the ALU.

b) Write a MIPS assembly code to explain how data forwarding may eliminate the data hazard stall cycle.

```
ADD R1, R1, 8
ADD R2, R3, R1
```

Without forwarding, the result of R1 is not written until clock cycle five and thus the second ADD does not enter EX until cycle 6. With forwarding, the R1 result is piped back to the input and the ADD is able to enter EX at cycle 4 with no stalls.

c) Write a MIPS assembly code to show an example where forwarding cannot completely eliminate the data hazard stall cycles.

```
LD R1, 0(R2)
ADD R1, R1, R1
```

Here the data is not available until after the MEM stage in cycle 4, and so this cannot be forwarded to the ADD’s EX stage in cycle 4, this must stall and the EX of the ADD can proceed in cycle 5. Data cannot be forwarded backwards in time.
Question #3 (12 points)

a) A pipelined processor uses the delayed branch technique. You are asked to recommend one of two possibilities for the design of this processor. In the first possibility, the processor has a 4-stage pipeline and one delay slot, and in the second possibility, it has a 6-stage pipeline with two delay slots. Compare the performance of these two alternatives, taking only the branch penalty into account. Assume that 20 percent of the instructions are branch instructions and that an optimizing compiler has an 80 percent success rate in filling the single delay slot. For the second alternative, the compiler is able to fill the second slot 25 percent of the time.

b) Suppose that we make the following modification to the register file of the MIPS R3000: The number of general purpose registers is increased to 64 registers each of 16-bits width instead of 32-registers each of 32-bits width. Write a VHDL code that describes this modified register file, taking into account the following: The register file read operations should be asynchronously performed while the write operation should be synchronously performed if and only if the Write Enable (WE) is active.

**Ans.**

a) 

Givens: 20% of inst. are branches, compiler can fill 80% of 1st delay slot and 25% of 2nd delay slot.

For Alt. A

\[ S = \frac{\text{pipelined depth}}{1 + \text{pipeline stall per inst.}} \]

\[ S_a = \frac{4}{1 + 0.2 \times 1 - 0.2 \times 0.8 \times 1} = \frac{4}{1.04} \]

= 3.846

In case of dependant delay slots (the compiler should first succeed to fill slot 1 before attempting slot 2)

\[ S_b = \frac{6}{1 + 0.2 \times 2 - 0.2 \times 0.8 \times 1 - 0.20 \times 0.8 \times 0.25 \times 2} = \frac{6}{1.16} \]

= 5.172

In case of independent delay slots (the compiler is free to fill any of the two slots)

\[ S_b = \frac{6}{1 + 0.2 \times 2 - 0.2 \times 0.8 \times 1 - 0.20 \times 0.25 \times 2} = \frac{6}{1.16} \]

= 5.263

In all cases, Alt. B has higher speedup factor than alt. A, thus we select alternative B

b) Suppose that we make the following modification to the register file of the MIPS R3000: The number of general purpose registers is increased to 64 registers each of 16-bits width instead of 32-registers each of 32-bits width. Write a VHDL code that describes this modified register file, taking into account the following: The register file read operations should be asynchronously performed while the write operation should be synchronously performed if and only if the Write Enable (WE) is active.

**Ans.**

a) 

Givens: 20% of inst. are branches, compiler can fill 80% of 1st delay slot and 25% of 2nd delay slot.

For Alt. A

\[ S = \frac{\text{pipelined depth}}{1 + \text{pipeline stall per inst.}} \]

\[ S_a = \frac{4}{1 + 0.2 \times 1 - 0.2 \times 0.8 \times 1} = \frac{4}{1.04} \]

= 3.846

In case of dependant delay slots (the compiler should first succeed to fill slot 1 before attempting slot 2)

\[ S_b = \frac{6}{1 + 0.2 \times 2 - 0.2 \times 0.8 \times 1 - 0.20 \times 0.8 \times 0.25 \times 2} = \frac{6}{1.16} \]

= 5.172

In case of independent delay slots (the compiler is free to fill any of the two slots)

\[ S_b = \frac{6}{1 + 0.2 \times 2 - 0.2 \times 0.8 \times 1 - 0.20 \times 0.25 \times 2} = \frac{6}{1.16} \]

= 5.263

In all cases, Alt. B has higher speedup factor than alt. A, thus we select alternative B
b)
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity REG is
    port(CLK: in std_logic;
         RegW: in std_logic;
         DR, SR1, SR2: in unsigned(5 downto 0);
         Reg_In: in unsigned(15 downto 0);
         ReadReg1, ReadReg2: out unsigned(15 downto 0));
end REG;

architecture Behavioral of REG is
    type RAM is array (0 to 63) of unsigned(15 downto 0);
    signal Regs: RAM := (others => (others => '1'));  -- set all reg bits to '1'
begin
    process(clk)
    begin
        if CLK = '1' and CLK'event then
            if RegW = '1' then
                Regs(to_integer(DR)) <= Reg_In;   -- synchronous write
            end if;
        end if;
    end process;
    ReadReg1 <= Regs(to_integer(SR1));    --asynchronous read
    ReadReg2 <= Regs(to_integer(SR2));    --asynchronous read
end Behavioral;
Question #4 (12 points)

Consider a byte-addressable computer with 24-bit addresses, a cache capable of storing a total of 64K bytes of data and blocks of 32 bytes. Show the format of a 24-bit memory address for:

a) direct mapped
b) Fully associative
c) 4-way set associative

Ans.
a) Direct Mapped

Cache size 64K = $2^6 \cdot 2^{10} = 2^{16}$;

# of Cache blocks = Cache size / Block size = $2^{16} / 2^5 = 2^{11}$ blocks.

Hence, 11 bits are needed for the block field and 5 bits are needed for the word field, leaving 8 for the tag.

b) Fully Associative

Again, 5 bits are needed for the word field, leaving 19 for the tag.

c) 4-Way Set Associative

There are $2^{11} / 2^2 = 2^9$ sets in cache, so 9 bits are needed for the set field. We still need 5 bits for the word field, leaving 10 for the tag field.
**Question #5 (12 points)**

A 2-way set-associative cache consists of four sets. Main memory contains 2K blocks of eight words each.

a) Show the main memory address format that allows us to map addresses from main memory to cache. Be sure to include the fields as well as their sizes.

b) Compute the hit ratio for a program that loops 3 times from locations 8 to 51 in main memory. You may leave the hit ratio in terms of a fraction.

**Ans.**

a. \(2K \times 2^3 = 2 \times 2^{10} \times 2^3 = 2^{14}\), so we have 14-bit addresses with 9 bits in the tag field, 2 bits in the set field (since we have four sets), and 3 in the word field.

b. First iteration of the loop: Address 8 is a miss, and then entire block brought into Set 1. Hence, 9-15 are then hits. 16 is a miss, entire block brought into Set 2, 17-23 are hits. 24 is a miss, entire block brought into Set 3, 25-31 are hits. 32 is a miss, entire block brought into Set 0, 33-39 are then hits. 40 is a miss, entire block brought into Set 1 (note we do NOT have to throw out the block with address 8 as this is 2-way set associative), 41-47 are hits. 48 is a miss, entire block brought into Set 2, 49-51 are hits.

For the first iteration of the loop, we have 6 misses, and 5*7 + 3 hits, or 38 hits.

On the remaining iterations, we have 5*8+4 hits, or 44 hits each, for 88 more hits. Therefore, we have 6 misses and 126 hits, for a hit ratio of 126/132, or 95.45%.

**Good Luck!**

Best wishes,

Dr. Mostafa Elsayed