Department of Computer Science & Engineering
Microprocessor & Computer Architecture
UNIT 3

Memory Hierarchy

1)

If a direct mapped cache has a hit rate of 95%, a hit time of 4 ns, and a miss penalty of 100 ns, what is the AMAT? If an L2 cache is added with a hit time of 20 ns and a hit rate of 50%, what is the new AMAT?

AMAT = Hit time + Miss rate x Miss penalty = 4 + 0.05 x 100 = 9 ns

AMAT = Hit TimeL1 + Miss RateL1 x (Hit TimeL2 + Miss RateL2 x Miss PenaltyL2)= 4 + 0.05 x (20 + 0.5x100) = 7.5 ns

2)
In a two-level cache system, the access times of L_1 and L_2 1 and 8 clock cycles, respectively. The miss penalty from the L_2 cache to main memory is 18 clock cycles. The miss rate of L_1 cache is twice that of L_2. The average memory access time(AMAT) of this cache system is 2 cycles. What is the miss rates of L_1 and L_2 .

Explanation:

- Access time of L1 = 1
- Access Time of L2=8
- miss penalty L_1cache (2*L_2) = 18*2 = 2*a
- miss penalty L_2 cache say a = 18
- AMAT (average memory access time) =2

AMAT = Access time of L1 + (MissRate L1 *miss penalty L_1)where miss penalty L1 = Access time of L2 + (MissRate L2 * miss penalty L2)

2 = 1+ 2*a *(8 + a* 18)

Solving the equation,

a=0.111

3)

Identify the dependency

Sequence of instructions:

1. lw $s2, 0($s1)
2. lw $s1, 40($s6)
3. sub $s6, $s1, $s2
4. add $s6, $s2, $s2
5. or $s3, $s6, $zero
6. sw $s6, 50($s1)

Dependencies:
3 depends on 1 ($s2)
3 depends on 2 ($s1)
4 depends on 1 ($s2)
5 depends on 4 ($s6)
6 depends on 2 ($s1)
6 depends on 4 ($s6)

4)

Assume the 5-stage MIPS pipeline with no forwarding, and each stage takes 1 cycle. Instead of inserting nops, you let the processor stall on hazards. How many times does the processor stall? How long is each stall (in cycles)? What is the execution time (in cycles) for the whole program?

Ignoring the stalls for a moment, the program takes 10 cycles to execute { not 6, because the first 4 cycles, it does not commit (finish) an instruction { those 4 cycles, the pipeline is still filling up. It is also not 30, because when the first instruction commits, the 2nd instruction is nearly done, and will commit in the next cycle. Remember, pipelines allow multiple instructions to be executing at the same time.
With the stalls, there are only two stalls { after the 2nd load, and after the add { both are because the next instruction needs the value being produced. Without forwarding, this means the next instruction is going to be stuck in the fetch stage until the previous instruction writes back. These are 2 cycle stalls (when in doubt, draw diagrams like the ones on the Implementing MIPS slides, slide 63). So to answer the question, 2 stalls, 2 cycles each, and the total is 10 + 2 _ 2 = 14 cycles to execute the program.

5)

Assume the 5-stage MIPS pipeline with full forwarding. Write the program with nops to eliminate the hazards. (Hint: time travel is not possible!)

Draw a diagram { in the second stall, the result of the add is available at the register at the end of the execute stage when the next instruction wants to move to the execute stage, so you can forward the value of $s6 as the input to the execution stage (as the argument for the or) { this removes the second 2-cycle stall. However, the loaded value is not ready until the end of the memory stage, so you cannot using forwarding to remove both cycles { you still need to wait 1 cycle. The solution is to place a NOP after the second load. (Note: this is also the reason why MIPS has load delay slots).
Speaking of delay slots, the question was ambiguous in whether delay slots were used or not. If they are, all the answers are slightly different:

Part 1: You have one fewer dependence - 3 does not depend on 2, because it is in the delay slot.
Part 2: The first stall is only 1 cycle, so the program executes in 13 cycles.
Part 3: You do not require any nops, because of the delay slot.