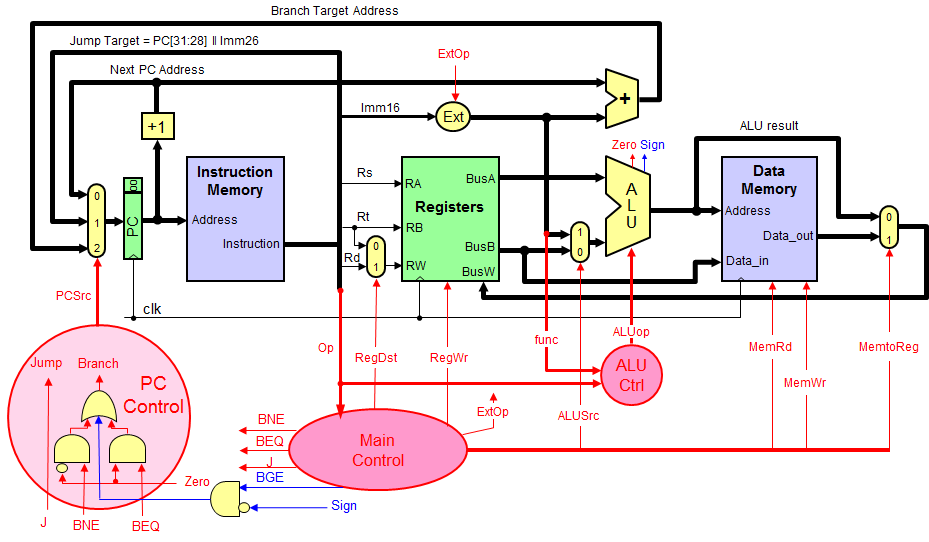
**Q1. (15 pts) Single-Cycle MIPS DataPath**

The basic single-cycle MIPS implementation discussed in class can only implement a subset of instructions. New instructions can be added to the existing ISA. We want to implement a new instruction **bge $t0, $t1, label**, which compares **$t0** and **$t1** and branches only if **$t0 >= $t1**. Assume that the ALU generates another output “sign” which is 1 when the ALU output is negative.

1. Make changes to the single-cycle datapath and control shown below to support this instruction. You should only add wires, gates, muxes to the datapath; do not modify the main functional units (the memory, register file, and ALU) themselves. Try to keep your diagram neat! (added design is represented in blue)



1. Complete the table given below by writing the values (0/1/X) next to each control signal to show the values that would exist in the circuit after all the “work” for executing this instruction has completed, right before the moment that the rising edge of the next clock cycle occurs. **X stands for don’t care**. If you need add a new control signal, please add it along with its value to the table below. Use the following table for ALUCtrl.

|  |  |
| --- | --- |
| **ALU function** | **4-bit ALU Control** |
| AND | 0001 |
| OR | 0010 |
| XOR | 0011 |
| ADD | 0100 |
| SUB | 0101 |
| SLT | 0110 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| RegDst | RegWrite | ExtOp | ALUSrc | MemRead | MemWrite | MemtoReg | ALUCtrl | J | Beq | Bne | **Bge** |
| X | 0 | X | 0 | 0 | 0 | X | 0101 | 0 | 0 | 0 | 1 |

**Q2. (15 pts)** Suppose we add the multiplification/Division instructions. The operation times are as follows:

Instruction memory access time = 200 ps, Data memory access time = 200 ps,

Register file read access time = 150 ps, Register file write access = 150 ps

ALU delay for basic instructions = 220 ps, ALU delay for mul/Div operation = 600 ps

Ignore the other delays in the multiplexers, control unit, sign-extension, etc.

Assume the following instruction mix: 35% ALU, 10% vector, 20% load, 10% store, 15% branch, and 10% jump.

1. What is the total delay for each instruction class and the clock cycle for the single-cycle CPU design?
2. Assume we fix the clock cycle to 200 ps for a multi-cycle CPU, what is the CPI for each instruction class and the speedup over a fixed-length clock cycle?

**Q3.** (10 pts) Consider the following MIPS code sequence:

**a: lw $5, 100($2)**

**b: add $2, $3, $5**

**c: sub $5, $5, $2**

**d: sw $5, 100($2)**

1. (5 pts) Identify all the RAW dependencies between pairs of instructions.

($5, b, a)

($5, c, a)

($2, c, b)

($2, d, b)

($5, d, c)

1. (3 pts) Identify all the WAR dependencies between pairs of instructions

($2, b, a)

($5, c, b)

1. (2 pts) Identify all the WAW dependencies between pairs of instructions

($5, c, a)

**Q4.** (25 pts) Use the following MIPS code fragment:

**I1: ADD $t0, $0, $a0 # $a0 = & array[0]**

**I2: ADDI $t1, $0, 100**

**Loop:**

**I3: LW $t2, 0($t0)**

**I4: SLL $t2, $t2, 2**

**I5: SW $t2, 0($t0)**

**I6: ADDI $t0, $t0, 4**

**I7: ADDI $t1, $t1, -1**

**I8: BNE $t1, $0, Loop**

1. (5 pts) list all the RAW data dependences in the code above within one loop iteration. Record the register, source instruction, and destination instruction. ($t2, I4, I3), ($t2, I5, I4), ($t1, I8, I7)
2. (5 pts) Show the timing of one loop iteration on the 5-stage MIPS pipeline **without forwarding hardware**. Complete the timing table, showing all the stall cycles. Assume that the register write is in the first half of the clock cycle and the register read is in the second half. Also assume that the branch is handled by predicting it as **NOT TAKEN**. If the branch outcome is **TAKEN**, it will stall the pipeline for 1 clock cycle only.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** |
| **I1:** | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I2:** |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I3:** |  |  | IF | Stall | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I4:** |  |  |  |  | IF | Stall | Stall | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I5:** |  |  |  |  |  |  |  | IF | Stall | Stall | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |
| **I6:** |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |
| **I7** |  |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |
| **I8:BNE** |  |  |  |  |  |  |  |  |  |  |  |  | IF | Stall | Stall | ID | EX | MEM | WB |  |  |  |  |  |
| **I3:** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | IF | IF | ID | EX | MEM | WB |  |  |  |
| **I4:** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | IF | Stall | Stall | ID | EX | MEM | WB |
| I3: Clock cycle 5 is the earliest clock cycle to read $t0 since I1 writes to $t0, so, ID has to be in cc5 and a stall cycle has to be inserted in cc4.  I4: read $t2, which is written by I3, so cc8 is the earliest clock cycle to read $t2 and ID has to be in cc8 and 2 stalls are inserted in cc6 and cc7.  I5: read $t2, which is written by I4, so cc11 is the earliest clock cycle to read $t2 and ID has to be in cc11 and 2 stalls are inserted in cc9 and cc10.  I8: read $t1, which is written by I7, so cc16 is the earliest clock cycle to read $t1 and ID has to be in cc16 and 2 stalls are inserted in cc14 and cc15.  I3 after BNE: BNE predicts as “NOT TAKEN” so, in cc16, next instruction (I9) is fetched into the pipeline, while actually I3 this is a taken branch, so the correct instruction I3 has to be fetched again. | | | | | | | | | | | | | | | | | | | | | | | | | |

1. (5 pts) According to the timing diagram of part (b), compute the number of clock cycles and the average CPI to execute ALL the iterations of the above loop.

**There are 100 iterations and each iteration requires 13 cycles = 6 cycles to start the 6 instructions in loop body + 6 stall cycles + 1 wrong fetch (cc3 to cc16, cc4 stall is only needed for the first iteration)**

**There are 2 additional cycles to start the first 2 instructions before the loop.**

**Therefore, total cycles = 100 \* 13 + 2 + 1 (cc4 in first iteration) + 4 (to complete the last instruction ID, EX, MEM, WB) = 1307 cycles**

**Total instruction executed = 2 + 6 \* 100 = 602 instructions (counting first two)**

**Average CPI = 1307 / 602 = 2.17**

1. (5 pts) Redo part (b) to show the timing of one loop iteration with **full forwarding** hardware. If forwarding happens, please show how the data is forwarded with an arrow.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** |
| **I1:** | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I2:** |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I3:** |  |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I4:** |  |  |  | IF | Stall | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I5:** |  |  |  |  |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I6:** |  |  |  |  |  |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |
| **I7** |  |  |  |  |  |  |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |
| **I8:BNE** |  |  |  |  |  |  |  |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |
| **I3:** |  |  |  |  |  |  |  |  |  | IF | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |
| **I4:** |  |  |  |  |  |  |  |  |  |  |  | IF | Stall | ID | EX | MEM | WB |  |  |  |  |  |  |
| **I3: Read $t0, $rs = $rd3, so forwarding from MEM to ID.**  **I4: Read $t2, $rs = $rd2 and EX.MemRd == 1, so insert a stall cycle and in cc6 and forwarding from MEM to ID.**  **I5: Read $t2, $rt = $rd2, so forwarding from EX to ID.**  **I8: Read $t1, $rs = $rd2, so forwarding from EX to ID.** | | | | | | | | | | | | | | | | | | | | | | | |

1. (5 pts) According to the timing diagram of part (d), compute the number of clock cycles and the average CPI to execute ALL the iterations of the above loop with full forwarding hardware. What is the speedup factor?

**There are 100 iterations and each iteration requires 8 cycles = 6 cycles to start the 6 instructions in loop body + 1 stall cycle + 1 wrong fetch (cc3 to cc10, cc4 stall is only needed for the first iteration)**

**There are 2 additional cycles to start the first 2 instructions before the loop.**

**Therefore, total cycles = 100 \* 8 + 2 + 1 (cc4 in first iteration) + 4 (to complete the last instruction ID, EX, MEM, WB) = 807 cycles**

**Total instruction executed = 2 + 6 \* 100 = 602 instructions (counting first two)**

**Average CPI = 807 / 602 = 1.34**

1. (Bonus 5 pts) Assuming delayed branching, rewrite the above code to take advantage of the branch delay slot. Show the timing diagram for one loop iteration of the reordered code with full forwarding hardware.

**Hints: insert 1 instruction between LW and SLL instructions to remove the load delay stall and select one independent instruction for the branch delay slot.**

**I1: ADD $t0, $0, $a0 # $a0 = & array[0]**

**I2: ADDI $t1, $0, 100**

**Loop:**

**I3: LW $t2, 0($t0)**

**I4: ADDI $t0, $t0, 4**

**I5: SLL $t2, $t2, 2**

**I6: SW $t2, 0($t0)**

**I7: BNE $t1, $0, Loop**

**I8: ADDI $t1, $t1, -1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** |
| **I1:** | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I2:** |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I3:** |  |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I4:** |  |  |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I5:** |  |  |  |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I6:** |  |  |  |  |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I7:BNE** |  |  |  |  |  |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |
| **I8:** |  |  |  |  |  |  |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |
| **I3:** |  |  |  |  |  |  |  |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |
| **I4:** |  |  |  |  |  |  |  |  |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |
| Forwarding from I2 to I4 is only exist for the first iteration. | | | | | | | | | | | | | | | | | | | | | | | |

1. (Bonus 5 pts) Compute the number of cycles and the average CPI to execute ALL the iteration of the modified loop in part (f) with full forwarding hardware and delayed branching. What is the speedup factor comparing without forwarding and the branch delay slot.

**There are 100 iterations and each iteration requires 6 cycles = 6 cycles to start the 6 instructions in loop body**

**There are 2 additional cycles to start the first 2 instructions before the loop.**

**Therefore, total cycles = 100 \* 6 + 2 + 4 (to complete the last instruction ID, EX, MEM, WB) = 606 cycles**

**Total instruction executed = 2 + 6 \* 100 = 602 instructions (counting first two)**

**Average CPI = 606 / 602 = 1.01**

**Speedup Factor = CPIpart-c/CPIpart-g = 2.17/1.01 = 2.15**