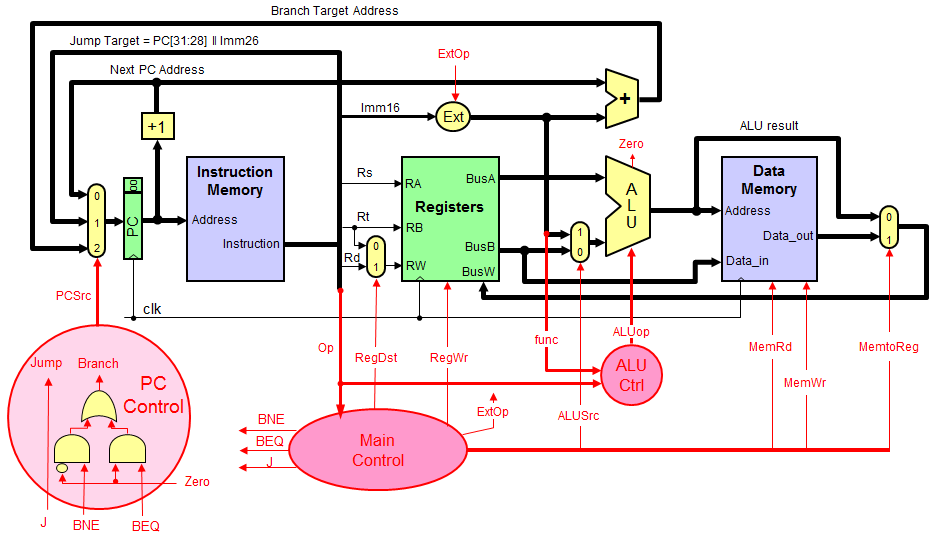
**Q1. Single-Cycle MIPS Processor**

We wish to add the instruction **jalr** (jump and link register) to the single-cycle datapath. The jump and link register instruction is described below:

**jalr rd, rs # rd = pc + 4 , pc = rs**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **op6 = 0** | **rs5** | **0** | **rd5** | **0** | **Funct6 = 0x9** |

1. Add any necessary datapath and control signals and draw the result datapath. 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!



1. Show the values of the control signals to control the execution of the **jalr** instruction. 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 |
|  |  |  |  |  |  |  |  |  |  |  |

**Q2. Processor Performance**

Suppose we add the multiply and divide instructions. The operation times are as follows:

Instruction memory access time = 190 ps, Data memory access time = 190 ps,

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

ALU delay for basic instructions = 190 ps, ALU delay for multiply or divide = 550 ps

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

Assume the following instruction mix: 30% ALU, 15% multiply & divide, 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.** Consider the following MIPS code sequence:

**a: add $t0, $s0, $s1  
b: sub $t1, $s2, $t0  
c: xor $t0, $s0, $s1  
d: or  $t2, $t1, $t0**

1. Identify all the RAW dependencies between pairs of instructions.
2. Identify all the WAR dependencies between pairs of instructions
3. Identify all the WAW dependencies between pairs of instructions

**Q4.** Use the following MIPS code fragment:

**I1: ADDI $3, $0, 100 # $3 = 100**

**I2: ADD $4, $0, $0 # $4 = 0**

**Loop:**

**I3: LW $5, 0($1) # $5 = MEM[$1]**

**I4: ADD $4, $4, $5 # $4 = $4 + $5**

**I5: LW $6, 0($2) # $6 = MEM[$2]**

**I6: SUB $4, $4, $6 # $4 = $4 – $6**

**I7: ADDI $1, $1, 4 # $1 = $1 + 4**

**I8: ADDI $2, $2, 4 # $2 = $2 + 4**

**I9: ADDI $3, $3, -1 # $3 = $3 – 1**

**I10: BNE $3, $0, Loop # if ($3 != 0) goto Loop**

1. (10 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 will stall the pipeline for 1 clock cycle only. Ignore the "startup cost" of the pipeline.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** |
| **I1: ADDI** | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I2: ADD** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I3: LW** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I4: ADD** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I5: LW** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I6: SUB** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I7: ADDI** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I8: ADDI** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I9: ADDI** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I10: BNE** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I3: LW** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I4: ADD** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. According to the timing diagram of part (a), compute the number of clock cycles and the average CPI to execute ALL the iterations of the above loop.
2. Reorder the instructions of the above loop to fill the load-delay and the branch delay slots, without changing the computation. Write the code of the modified loop.
3. Compute the number of cycles and the average CPI to execute ALL the iteration of the modified loop. What is the speedup factor?