**Project has 2 phases.**

**Phase 1: LABS 16-23 (2000 points)**

* + **Objective:**
    - Expand your MIPS datapath to support 5-stage pipeline with forwarding and hazard detection (Due November 16).

**Table 1.** Required MIPS Operations for the pipelined datapath design

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Type** | **Instruction** | **Code** | **Type** | **Instruction** | **Code** |
| **Arithmetic** | **Add** | **add** | **Logical** | **And** | **and** |
|  | **Add Immediate Unsigned Word** | **addiu** |  | **And immediate** | **andi** |
|  | **Add Unsigned Word** | **addu** |  | **Or** | **or** |
|  | **Add Immediate** | **addi** |  | **Not or** | **nor** |
|  | **Subtract** | **sub** |  | **Exclusive or** | **xor** |
|  | **Multiply** | **mul** |  | **And Immediate** | **andi** |
|  | **Multiply Word** | **mult** |  | **Or immediate** | **ori** |
|  | **Multiply Unsigned Word** | **multu** |  | **Exclusive or Immediate** | **xori** |
|  | **Multiply and add word to Hi,Lo** | **madd** |  | **Sign-extend half word** | **seh** |
|  | **Multiply and subract word to Hi,Lo** | **msub** |  | **Shift left logical** | **sll** |
| **Data** | **Load word** | **lw** |  | **Shift right Logical** | **srl** |
|  | **Store word** | **sw** |  | **Shift Word Left Logical Variable** | **sllv** |
|  | **Store byte** | **sb** |  | **Shift Word Right Logical Variable SRLV** | **srlv** |
|  | **Load half** | **lh** |  | **Set on less than** | **slt** |
|  | **Load byte** | **lb** |  | **set on less than immediate** | **slti** |
|  | **Store half** | **sh** |  | **move conditional on not zero** | **movn** |
|  | **Move to Hi Register** | **mthi** |  | **move conditional on zero** | **movz** |
|  | **Move to Lo Register** | **mtlo** |  | **Rotate Word Right Variable** | **rotrv** |
|  | **Move from Hi Register** | **mfhi** |  | **Rotate word right** | **rotr** |
|  | **Move from Lo Register** | **mflo** |  | **Shift word right arithmetic** | **sra** |
|  | **Load Upper Immediate** | **lui** |  | **Shift Word Right Arithmetic Variable** | **srav** |
| **Branches** | **branch if greater than or equal to zero** | **bgez** |  | **Sign-Extend Byte** | **seb** |
|  | **branch on equal** | **beq** |  | **Set on Less Than Immediate Unsigned** | **sltiu** |
|  | **branch on not equal** | **bne** |  | **Set on Less Than Unsigned SLTU** | **sltu** |
|  | **branch on greater than zero** | **bgtz** |  |  |  |
|  | **branch on les than or equal to zero** | **blez** |  |  |  |
|  | **branch on less than zero** | **bltz** |  |  |  |
|  | **jump** | **j** |  |  |  |
|  | **jump register** | **jr** |  |  |  |
|  | **jump and link** | **jal** |  |  |  |

* **Method:**
  + Add registers between stages first, pass through all the signals as a result of an instruction from previous stage.
    - Conduct post routing simulation for functional verification of basic pipeline no dependencies between instructions (write your own program with independent sequence of instructions)
  + Then handle dependencies between instructions by Forwarding/Hazard Detection Unit
    - Incrementally add hazard detection and forwarding units, show that each case is resolved with forwarding or hazard detection unit. (write a short program for each dependency case and test individually
* **Demonstration:**
  + Functional verification based on post-routing simulation due **November 16** during your lab session.
  + Your instruction memory must read from input file named "Instruction\_memory.txt".
    - Inside initial begin block of Instruction\_memory.v, use following initialization method:
      * $readmemh ("Instruction\_memory.txt", <memory register identifiers>)
  + Public test cases (project\_public\_test\_cases.s) and the Instruction\_memory.txt for those test cases are included in the LABS16-23 folder.
    - File is generated by MIPS Helper and branch offset bug is fixed in this file
    - Note that these public test cases cover only a subset of dependency scenarios.
  + At the beginning of the lab you will be given the instruction memory ("**Instruction\_memory.txt**") with the private test cases.
  + During the lab you will generate the post-routing simulation using this instruction memory
  + Waveform must show the following signals: content of s1,s2,s3, and s4 registers along with "pcresult".
  + TAs will check your waveforms and will let you know about your score for private testcases on the spot.
  + We will use public test-cases only if most of the private test cases fail.
* **Offline Testing:**
  + Functionality of individual instructions (test cases used for Labs 9-15) on the pipelined datapath will be tested offline based on post-routing simulation
  + Follow the submission instructions carefully.
* **Deliverable:**
  + Submit the following files (**deadline: November 16, 5pm**)
    - All verilog files: "\*.v"
    - Data file used for initializing data and instruction memory (\*.txt)
  + Include the following notes under comment section during your submission
    - Number of pipeline stages:
      * We accept both four and five stage based pipelined datapaths
      * Indicate the number of stages (4 or 5) in your design.
    - Branch decision and resolution stage:
      * Similarly also indicate the stage (DE or EX) where you are making branch decision.
    - If you don't include any note, we will **assume** that it is a 5-stage pipeline and branches are resolved during ID stage during the offline testing.
* **Penalty Conditions:**
  + private test cases on the pipelined datapath (2000 points)
    - If majority of the private test cases fail, then you can use public test cases for a maximum score of 800 points
  + Percent effort not reported (250 pt penalty)
  + Late submission (10% per day)
  + Submitting files in a folder or in compressed form (zip/tar). (20% penalty)
  + Changing the file name or extension. (20% penalty)
  + Failing to demonstrate (80% penalty)
  + Design works in behavioral simulation but fails to synthesize (70% penalty)
  + Design works in behavioral simulation, synthesizes with warnings but post-routing simulation fails (60% penalty)
  + Both team members must attend the demonstration

**Phase 2: LAB 24 (500 points)**

* **Objective:**
  + Execute your vbsme\_zigzag.s on the pipelined datapath (due November 21)
* **Eligibility**
  + At least 75% of the instructions are functional on the single cycle datapath
  + Pipelined design passes at least 75% of the private test cases
* **Method**
  + A test case will be given at the beginning of the lab for your data memory
  + Prepare your instruction memory before the lab with your SAD routine
  + Prepare data memory with the new test case
  + Execute the program on the FPGA
  + The (X,Y) coordinates of the block with the current minimum SAD should be displayed on the FPGA
  + Display will start with (0,0) and each time a block with smaller SAD is found new coordinates should be displayed
* **Deliverable**:
  + Submit the following files (**deadline: November 21, 5pm**)
    - All verilog files: "\*.v"
    - Data file used for initializing data and instruction memory (\*.txt)
    - .s (final form of your sad routine)
* **Penalty Conditions:**
  + Percent effort not reported (100 pt penalty)
  + Late submission (10% per day)
  + Submitting files in a folder or in compressed form (zip/tar). (20% penalty)
  + Changing the file name or extension. (20% penalty)
  + Failing to demonstrate (80% penalty)
  + Design works in behavioral simulation but fails to synthesize (70% penalty)
  + Design works in behavioral simulation, synthesizes with warnings but post-routing simulation fails (60% penalty)
  + Design works in post-routing simulation, but FPGA fails to display (25% penalty)
  + Both team members must attend the demonstration