**LAB 8-11 (September 25 – October 4)**

**Objective**

1. Implement basic datapath components

**Logistics**

* No demonstration required for the datapath components
* There are two parts
* You should plan for completing part-1 by October 2 and part-2 by October 9 (Lab 12)
* Part-2 has a pre-lab component to be turned in by October 2nd.
* Lab 12 will give the specifications for your datapath and will be released on October 6

**Part 1. Datapath Component Design and Simulation (No submission, no demo)**

* Below is a list of datapath components needed in your project. Templates are in the “DatapathComponents” folder.
  + RegisterFile
  + ALU32Bit
  + DataMemory
  + SignExtension
  + Mux32Bit2To1
* Follow the comments given in the source codes
* Synthesize and conduct functional verification with post-routing simulation for each component
* Exhaustively test your components

It is your responsibility to make sure that these data path components are working properly. We will not be testing the functionality of these components individually

**Part 2: Objectives:**

* + - Design and develop an **ALU** that supports all operations required by the given MIPS ISA.

**Table 1.** Required MIPS Operations for the 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** |  |  |  |

**ALU Design**

* **Pre-Lab:** (**Due October 2**) **(20 pts)**
  + Study the given instruction list and identify all operations needed. Refer to the MIPS ISA Reference.
    - **List** all the arithmetic operations.
  + **Draw** the block diagram of the ALU module with inputs and outputs properly labeled with their bit-width.
    - You need to revise the ALU outputs of the ALU32Bit.v from “Part 1”. You need to accommodate potential 64-bit outputs from certain instructions, as well as deal with interpreting the outcomes of arithmetic operations needed by different types of branch instructions. Overflow detection is not needed.
  + TAs will collect your papers at the beginning of your lab session on October 2nd. **This activity is worth 20 pts.**
* **Method:**
  + Revise ALU32Bit.v and ALU32Bit\_tb.v implemented in Part 1.
  + Test your ALU in post-routing simulation for all the operations. There is no demonstration requirement for this task.