Computer Architecture

Milestone III

Lobna ElHawary 900160270

Marian Ramsis 900163444

Shahd El Ashmawy 900161393

1. **Items updated from single cycle implementation**

* **Modified Control unit:**
  + **Removed Jal select**: we thought we’d use it in the single cycle implementation to select between Pc and rs1 for Jal which was incorrect.
  + **Added “AUIPCselect” to control unit**: This acts as a select line for “ alusrca\_mux” (which is the first input to the ALU) and this selects between PC and the forwarded result from Forwarding MUX A. In case of AUIPC, the MUX chooses the PC value to add it the immediate value selected from forwarding MUX B.
* **Memory:**
  + Implemented as one memory instead of two separate memories for instructions and data. This was done by creating a clock **halfclk** that was half the speed of the original clock. **Halfclk** is assigned as the output of the **clockdivider** module. The memory module takes the normal full speed clock as input, however, inside the module itself, reading is done continuously with each cycle of the normal clock. Writing to the memory is only done if **memwrite** is 1 and the clock is 0, and **data\_out** from the memory (whether instruction or data) is only obtained when the clock is 1.
  + In the **RISCV** module, all the modules that require clocks will take the risc**\_**clk as normal ( which is the full speed clock), except **IF/ID** register and **PC** , which will take the **negation** of the “**halfclk**” and the **MEM**/**WB** register and the **MUX** that determines whether which address is to be taken, the one from the PC or that forwarded from the ALU result from the MEM stage, both of which receive the **halfclk**.
* **Forwarding unit:**
  + Modified the forwarding unit so it will only forward to the Forwarding MUXs from the values in the registers (**ID\_EX\_Reg1 and ID\_EX\_Reg2)** and from **data\_out¸** which is the output from the **WB** stage.

1. **Items Added**

* **Forwarding Mux A and Forwarding Mux B:**
  + **Forwarding MUX A:** Chooses between **ID\_EX\_Reg1**, which is the value inside readdata1 as propagated by the **ID\_EX register**, the **ID\_EX\_readdata1\_ouput** which is the output of the multiplexer in the **WB** stage, and **EX\_MEM\_aluresult\_output** which is the alu result which is being forwarded from the EX stage. Forwarding MUX A results in a result **alusrc A**.
  + The same can be said for **Forwarding MUX B**, which takes the same inputs, and results in **alusrc B**.
* **Forwarding Unit:** 
  + Responsible for determining the values of the select lines of the Forwarding MUX A and B respectively.

|  |  |
| --- | --- |
| **Value of select line for**  **Forward A and Forward B** | **Output of the MUX** |
| 00 | ID\_EX\_readdata1\_ouput |
| 01 | writedata |
| 10 | EX\_MEM\_aluresult\_output |
| 11 | 32 bit 0 |

* **Hazard and Flushing:**
  + Responsible for outputting a stall flag which then does two things:
    - Determines the value of **load** (which is the inverse of the stall flag), which then feeds into the **PC** and the **IF/ID** register and forces a stall.
  + The Stall flag ORed with the **PCsrc** is also fed into the **MUX** immediately following the control unit, as a select line which determines whether to allow for the propagation of the control bits to the **ID/EX** register or whether to proceed with a flush (by choosing 0 as the value outputted from the MUX) if a branch is taken and we choose to flush the previous instructions.
  + Multiplexers were also added after instruction memory and after the **ID\_EX** register which are also responsible for the flushing mechanism in their respective stages. They use **PCsrc** (which results from the ANDing of the comparators result received from **EX/MEM** register and the value of the branch signal from the propagated control signals In the **EX/MEM** register.
* **Implemented Registers for the different stages:**
  + Responsible for storing the values from the previous stage as the instruction traverses the pipeline and each stage is updated with each clock cycle.

Below is the Table of Values for the Control unit for each Instruction type.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Control unit | | | | | | | | | | |
| Instruction | Inst [6-2] | Branch | MemRead | MemtoReg | ALUOp | MemWrite | ALUSrc | RegWrite | PC\_Select | AUIPCselect |
| R-Format | 01100 | 0 | 0 | 00 | 10 | 0 | 0 | 1 | 0 | 0 |
| LW | 00000 | 0 | 1 | 01 | 00 | 0 | 1 | 1 | 0 | 0 |
| SW | 01000 | 0 | 0 | xx | 00 | 1 | 1 | 0 | 0 | 0 |
| BEQ | 11000 | 1 | 0 | xx | 01 | 0 | 0 | 0 | 0 | 0 |
| JAL | 11011 | 0 | 0 | 11 | 10 | 0 | 1 | 1 | 1 | 0 |
| JALR | 11001 | 0 | 0 | 11 | 11 | 0 | 1 | 1 | 1 | 0 |
| AUIPC | 00101 | 0 | 0 | 00 | xx | 0 | 1 | 1 | 0 | 1 |
| LUI | 01101 | 0 | 0 | 00 | xx | 0 | 1 | 1 | 0 | 0 |
| Immediates | 00100 | 0 | 0 | 00 | 11 | 0 | 1 | 1 | 0 | 0 |

All figures referenced in this report can also be found in the Excel sheet in the Zip file.