Logo

Description automatically generated

**CMPN301 – Computer Architecture**

**Project Phase 1 Report**

|  |  |
| --- | --- |
| **Team Members** |  |
| **Farah Mohamed Abdelfattah** | 1190176 |
| **Abdelrahman Mohamed Ezzat** | 1190158 |
| **Ziad Ahmed Hamed** | 4200002 |
| **Ali Mohamed Hashish** | 1190223 |
| **Tutorial** | Thursday 8 am |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **OPCODES FOR EACH INSTRUCTION** | | | | | | |
| **NOT** | 1 | 1 | 0 | 0 | 0 |  |
| **INC** | 1 | 1 | 0 | 0 | 1 |  |
| **DEC** | 1 | 1 | 0 | 1 | 0 |  |
| **MOV** | 1 | 1 | 0 | 1 | 1 |  |
| **ADD** | 1 | 1 | 1 | 0 | 0 |  |
| **IADD** | 1 | 1 | 1 | 0 | 0 | 1 |
| **SUB** | 1 | 1 | 1 | 0 | 1 |  |
| **AND** | 1 | 1 | 1 | 1 | 0 |  |
| **OR** | 1 | 1 | 1 | 1 | 1 |  |
| **PUSH** | 0 | 1 | 0 | 0 | 0 |  |
| **POP** | 0 | 1 | 0 | 0 | 1 |  |
| **LDM** | 0 | 1 | 0 | 1 | 0 | 1 |
| **LDD** | 0 | 1 | 0 | 1 | 1 |  |
| **STD** | 0 | 1 | 1 | 0 | 0 |  |
| **JZ** | 1 | 0 | 0 | 0 | 0 |  |
| **JC** | 1 | 0 | 0 | 0 | 1 |  |
| **JMP** | 1 | 0 | 0 | 1 | 0 |  |
| **CALL** | 1 | 0 | 0 | 1 | 1 |  |
| **RET** | 1 | 0 | 1 | 0 | 0 |  |
| **RTI** | 1 | 0 | 1 | 0 | 1 |  |
| **NOP** | 0 | 0 | 0 | 0 | 0 |  |
| **SETC** | 0 | 0 | 0 | 0 | 1 |  |
| **CLRC** | 0 | 0 | 0 | 1 | 0 |  |
| **IN** | 0 | 0 | 0 | 1 | 1 |  |
| **OUT** | 0 | 0 | 1 | 0 | 0 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **INSTRUCTION FORMAT** | | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| OPCODE | | | | | Immediate bit | R destination | | | R source 1 | | | R source 2 | | | unused | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CONTROL SIGNALS** | | | | | | | | | |
|  | **RegWrite** | **PCSrc** | **MemRead (used by memory)** | **MemWrite**  **(used by memory)** | **MemToReg**  **(used by WB)** | **IN**  **(used by WB)** | **OUT**  **(used by WB)** | **SP+**  **(used by memory)** | **SP-**  **(used by memory)** |
| NOT | 1 |  |  |  |  |  |  |  |  |
| INC |  |  |  |  |  |  |  |  |
| DEC |  |  |  |  |  |  |  |  |
| MOV |  |  |  |  |  |  |  |  |
| ADD |  |  |  |  |  |  |  |  |
| IADD |  |  |  |  |  |  |  |  |
| SUB |  |  |  |  |  |  |  |  |
| AND |  |  |  |  |  |  |  |  |
| OR |  |  |  |  |  |  |  |  |
| PUSH |  |  |  | 1 |  |  |  |  | 1 |
| POP | 1 |  | 1 |  | 1 |  |  | 1 |  |
| LDM | 1 |  |  |  |  |  |  |  |  |
| LDD | 1 |  | 1 |  | 1 |  |  |  |  |
| STD |  |  |  | 1 |  |  |  |  |  |
| JZ |  | 1 |  |  |  |  |  |  |  |
| JC |  |  |  |  |  |  |  |  |
| JMP |  |  |  |  |  |  |  |  |
| CALL |  |  | 1 |  |  |  |  | 1 |
| RET |  | 1 |  |  |  |  | 1 |  |
| RTI |  | 1 |  |  |  |  | 1 |  |
| NOP |  |  |  |  |  |  |  |  |  |
| SETC |  |  |  |  |  |  |  |  |  |
| CLRC |  |  |  |  |  |  |  |  |  |
| IN | 1 |  |  |  |  | 1 |  |  |  |
| OUT |  |  |  |  |  |  | 1 |  |  |

|  |  |
| --- | --- |
| **ALL HAZARDS** | |
| **Hazards** | **Details** |
| **Data Hazards** | **Forwarding unit at ALU:**  Checks for RegWrite signal at all subsequent buffers  Checks for Rd for all subsequent buffers  Checks for Rs1,Rs2 of current execution  And acts accordingly -> checks closer buffers first  Potential instructions to cause hazards [All R-type, POP, IN, LDD, LDM]  WriteBackValue is:  R-Type -> AluOutput  POP and LDD -> MemoryOutput  LDM -> immediate  IN -> in port  **Forwarding unit at Memory:**  Checks for RegWrite signal at MEM2/WB buffer  Checks for Rd at MEM2/WB buffer  Checks for Rs1, Rs2 of current STORE operation  And acts accordingly  Potential instructions to cause hazards [a load (**LDD**) instruction followed by a store (**STD**)] |
| **Hazard Detection unit at Decode:**  Checks for MemRead (only by LDD and POP) signal at ID/EX & EX/MEM1 buffer  Checks for Rd at ID/EX & EX/MEM1 buffer  Checks for Rs1, Rs2 of current fetched instruction (from IF/ID buffer)  And acts accordingly  Potential instructions to cause hazards [a load (**LDD**) instruction or a pop (**POP**)] |
| **Structural Hazards** | **Hazard Detection unit at Decode:**  If current MemRead or MemWrite at decode stage are ‘1’  Checks for MemRead or MemWrite at ID/EX buffer if equal ‘1’  Then stall once |
| **Control Hazards** | **Static Branch Prediction: Predict untaken, If taken:**  **JMP [decode]**  Change PC -> to RD  Flush IF/ID buffer (1 Stall)  **Call [decode]**  Change PC -> to RD  Flush IF/ID buffer  Propagate old PC till Mem stage to save it |
| **JMP conditional [execute] (JZ – JC)**  Change PC -> to RD  Flush IF/ID & ID/EX (2 stalls) |
| **RET/RTI [memory]**  Change PC -> to DataMem[SP]  Flush IF/ID & ID/EX & EX/Mem1 & Mem1/Mem2 (5 stalls) |

|  |  |  |  |
| --- | --- | --- | --- |
| **PIPELINE REGISTERS** | | | |
| **Register** | **Inputs** | **Outputs** | |
| **IF/ID** | * instruction[15..0] * immediate[15..0] * PC[15..0] | * RS1[2..0] * RS2[2..0] * instructionOut[15..0] * immediateOut[15..0] * PCout[15..0] | |
| **ID/EX** | * Clk * inPort * memRead * memToReg * memWrite * outport * pcSrc * regWrite * reset * spDec * spInc * RS1data[15..0] * RS2data[15..0] * instruction[15..0] * immediate[15..0] * PC[15..0] | * inPortOut * isImmediate * memReadOut * memToRegOut * memWriteOut * outPortOut * regWriteOut * spDecOut * spIncOut * RS1dataOut[15..0] * RS2dataOut[15..0] * opcode[4..0] * RD[2..0] * immediateOut[15..0] * PCout[15..0] | |
| **EX/MEM1** | * Clk * inPort * memRead * memToReg * memWrite * outport * regWrite * reset * spDec * spInc * RDreg[2..0] * RS1data[15..0] * RS2data[15..0] * ALUresult[15..0] * PC [15..0] | * inPortOut * memReadOut * memToRegOut * memWriteOut * outPortOut * regWriteOut * spDecOut * spIncOut * RDregOut[2..0] * RS1dataOut[15..0] * RS2dataOut[15..0] * ALUresultOut[15..0] | |
| **MEM1/MEM2** | * dataMemOutput[15..0] * RS1Data[15..0] * ALUoutput[15..0] * RD[2..0] * regWrite\_in * MemToReg\_in * INN\_in * OUTT\_in | * dataMemOutput[15..0] * RS1Data[15..0] * ALUoutput[15..0] * RD[2..0] * regWrite\_in * MemToReg\_in * INN\_in * OUTT\_in | |
| **MEM2/WB** | * dataMemOutput[15..0] * RS1Data[15..0] * ALUoutput[15..0] * RD[2..0] * regWrite\_in * MemToReg\_in * INN\_in * OUTT\_in | * dataMemOutput[15..0] * RS1Data[15..0] * ALUoutput[15..0] * RD[2..0] * regWrite\_in * MemToReg\_in * INN\_in * OUTT\_in |

|  |  |
| --- | --- |
| **BUFFER INPUTS (FROM QUARTUS – WITH SIZES)** | |
| **IF/ID buffer** | **Memory stage - 2 buffers are inside** |
|  |  |
| **ID/EX buffer** | **EX/MEM1 buffer** |
|  |  |

Diagram, schematic

Description automatically generatedDiagram, schematic

Description automatically generated**SCHEMATIC OF THE DESIGN**

**TODOS:**

1. PCSrc mux must be a unit
2. Change MEM[0] “initial PC” & MEM[1] “interrupt” to be in Instruction Cache
3. Jump already fetched then an interrupt happens [code]
4. In data propagated through all pipeline registers [Ziad]
5. In signal stops at ALU & overwrite ALU output [Ziad]
6. Load use case when stalling only once [MEM1/MEM2 buffer]
7. Reading flags from stack -> reading 32 bits
8. Modify increments to PC in design
9. Check LDM -> how to execute and OUT in stalls & forwarding, and MOV in stalls and forwarding, OUT in memory forwarding will probably fail
10. For MOV, we can deal with it like OUT -> so we can forward in memory w khalas

**Tasks:**

1. Modify design
2. Check XXX flags (done)
3. Make sure stall is not XX at beginning
4. Expand instruction cache size [ask about expected size]
5. Data hazards [HDU] [Farah] (done)
6. Structural hazards [HDU] [Hashish]
7. Control hazards [Ziad and Boody]
8. Interrupts [Hashish]
9. Forwarding units [Farah]

<https://app.diagrams.net/#G14Fvc_gAKcYsJDFnu-0WdP1k0BAtX3QD7>