**MARMARA UNIVERSITY**

**FACULTY OF ENGINEERING**

**DEPARTMENT OF COMPUTER ENGINEERING**



**CSE 3038 Computer Organization**

**Project 2 Report**

**Çağla Şen**

**150116001**

**Büşra Gökmen**

**150116027**

# andi instruction(I type):

|  |  |  |  |
| --- | --- | --- | --- |
| 001100(opcode) | rs | rt | imm16 |
| 6 bits | 5 bits | 5 bits | 16 bits |

**Syntax:** andi $rt, $rs, Label

Instruction is fetched from memory. The control signal is assigned as andisignal in control unit. Aluop is 1 for andi. ALU makes logical AND operation between register $rs and the zero-extended immediate. The output is put into register $rt. Then, PC ←PC+4.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **OP** | **RegDst** | **ALUSrc** | **Extop** | **MemToReg** | **RegWrite** | **MemRead** | **MemWrite** | **ALUOp** | **JM** | **BLTZAL** | **BRZ** | **BALN** | **BRANCH** |
| andi | 00 | 1 | 0 | 00 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | X |

# blez instruction(I type):

|  |  |  |  |
| --- | --- | --- | --- |
| 000110(opcode) | rs | rt | Label |
| 6 bits | 5 bits | 5 bits | 16 bits |

**Syntax:** blez $rs, Label

Instruction is fetched from memory. The control signal is assigned as blezsignal in control unit. Aluop is 101 for blez then we control blezsign signal in jumpbranch control unit. If the value of $rs is less than or equal to 0, branch to PC-relative address and link address is saved in register 31. New PC ← PC+4+(4\*Label).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **OP** | **RegDst** | **ALUSrc** | **Extop** | **MemToReg** | **RegWrite** | **MemRead** | **MemWrite** | **ALUOp** | **JM** | **BLTZAL** | **BRZ** | **BALN** | **BRANCH** |
| blez | X | X | 1 | 0 | 0 | 0 | 0 | 101 | 0 | X | X | X | X |

# balrn instruction (R type):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 000000(opcode) | rs | rt | rd | shamt | 010111(function code) |
| 6 bits | 5 bits | 5 bits | 5 bits | 5 bits | 6 bits |

**Syntax:** balrn $rs, $rd

Instruction is fetched from memory. The control signal is assigned as balrnsignal in control unit. Nout and Zout inputs are taken in JumpBranchcontrol unit. Then balrnsign is assign with this inputs. If the previous instruction’s status is one, (Stat[n]==1) branch to address found in register $rs. PC ← R[rs].

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **OP** | **RegDst** | **ALUSrc** | **Extop** | **MemToReg** | **RegWrite** | **MemRead** | **MemWrite** | **ALUOp** | **JM** | **LINK** | **BRZ** | **BALN** | **BRANCH** |
| balrn | X | X | X | X | 0 | X | 0 | X | 0 | 1 | X | X | X |

# jal instruction (J type):

|  |  |
| --- | --- |
| 000011(opcode) | Target |
| 6 bits | 26 bits |

**Syntax: jal** Target

Instruction is fetched from memory. The control signal is assigned as jalsignal in control unit. Then we control jalsignal signal in jumpbranch control unit. Then jump to PC-relative address and link address is saved in register 31. New PC ← PC+4+[31-28].

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **OP** | **RegDst** | **ALUSrc** | **Extop** | **MemToReg** | **RegWrite** | **MemRead** | **MemWrite** | **ALUOp** | **JM** | **LINK** | **BRZ** | **BALN** | **BRANCH** |
| jal | X | X | 1 | 1 | 1 | 1 | 0 | X | 1 | 1 | 0 | 0 | X |

# jmor instruction (R type):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 000000(opcode) | rs | rt | rd | shamt | 100101(function code) |
| 6 bits | 5 bits | 5 bits | 5 bits | 5 bits | 6 bits |

**Syntax: jmor $rs,$rt**

Instruction is fetched from memory. The signal is assign as jmorsignal with function code of jmor in control unit. Then assign jmor sign in ALU control for for ALU or operation. ALU makes logical OR operation between register $rs and $rt. The output is the address on Data Memory. The value is jump to PC-relative address and link address is saved in register 31. PC←PC + 4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **OP** | **RegDst** | **ALUSrc** | **Extop** | **MemToReg** | **RegWrite** | **MemRead** | **MemWrite** | **ALUOp** | **Jump** | **BLTZAL** | **BRZ** | **BALN** | **BRANCH** |
| jmor | X | 0 | 1 | 01 | 1 | 1 | 0 | 10 | X | 0 | 0 | 0 | X |

# sll instruction (R type):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 000000(opcode) | rs | rt | rd | shamt | 000000(function code) |
| 6 bits | 5 bits | 5 bits | 5 bits | 5 bits | 6 bits |

**Syntax:** sll $rd, $rt, $rs

Instruction is fetched from memory. The signal is assign as sllsignal with function code of sll in control unit. ALU makes shift operation with shamt input as sum=b<<shamt. Register $rt is shifted to left by the value in shamt, and store the result in register $rd. PC←PC+4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **OP** | **RegDst** | **ALUSrc** | **Extop** | **MemToReg** | **RegWrite** | **MemRead** | **MemWrite** | **ALUOp** | **JM** | **BLTZAL** | **BRZ** | **BALN** | **BRANCH** |
| sll | 01 | 0 | 1 | 00 | 1 | 0 | 0 | 010 | 0 | 0 | 0 | 0 | X |

# SINGLE CYCLE DATAPATH

# 

Figure 1

In this project, some given MIPS instructions is implemented into a existing code and path which already includes lw, sw, add, sub, and, or, slt, and beq instructions. During implementation both of the code and datapath are modified to meet our new instructions (andi,sll,balrn,blez,jal,jmor) requirements. The modified datapath can be seen on Figure 1.

The additions to the existing single cycle datapath contains below:

* The number of instruction has been increased.
* 4x1 mux is added to Proccessor.
* A new adder(add3) is added to Processor. This adder adds pc+4 to Target.
* 2 bit is added to function code bit number ([5:0] fcode).
* We took instruction signals from the control. We made the necessary signals for blez balrn jal jmor sll andi.
* aluop2 is added to control unit and alucontrol.
* We added a mux (mult5) that we decided whether to do zero extend or sign extend according to the outgoing andi signal (mult5). The result of this mux goes to the signextend or zeroextend component.
* The mult2 mux take signextend or zeroextend and datab. It gives output out2 according to alusrc(that is 1 or not) .
* We added shamt (shift amount) input to ALU for sll instruction.
* We add Target to using jal instruction. The Target is equal to instruc[25:0] .
* We add jmorsig to ALU to took signal from alucontrol .
* jmor output signal is added.
* We took an or signal from function code of jmor instruction.
* Status registers (zeroprev, negprev) are set.
* JumpBranchcontrol unit is added to datapath.
* Reg[31] as multiplexer input for link address

Example Runs:





