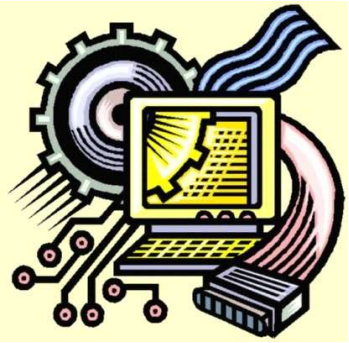
Computer Organization

Project 2 Report



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 datapath sets the control signals properly, and then registers are read, ALU output is generated. 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	Reg Dst	ALUSrc	Extop	MemTo Reg	Reg Write	Mem Read	Mem Write	ALUOp	JM	BLTZ AL	BRZ	BALN	BRANCH
andi	00	1	0	00	1	0	0	100	0	0	0	0	0

bltzal instruction(I type):

100010(opcode)	rs	rt	Label
6 bits	5 bits	5 bits	16 bits

Syntax: bltzal $rs, Label

Instruction is fetched from memory. The datapath sets the control signals properly. If the value of $rs is less than zero, branch to PC-relative address and link address is saved in register 31. New PC ← PC+4+(Label<<2).

OP	Reg Dst	ALUSrc	Extop	MemTo Reg	Reg Write	Mem Read	Mem Write	ALUOp	JM	BLTZ AL	BRZ	BALN	BRANCH
bltzal	10	0	1	10	1	0	0	011	0	1	0	0	0

brz instruction (R type):

000000(opcode)	rs	rt	rd	shamt	010100(function code)
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Syntax: brz $rs

Instruction is fetched from memory. The datapath sets the control signals properly. If the previous instruction's status is zero, (Stat[z]==1) branch to address found in register $rs. PC ← R[rs].

OP	Reg Dst	ALUSrc	Extop	MemTo Reg	Reg Write	Mem Read	Mem Write	ALUOp	JM	BLTZ AL	BRZ	BALN	BRANCH
brz	X	X	1	X	0	0	0	010	0	0	1	0	0

baln instruction (J type):

011011(opcode)	Target
6 bits	26 bits

Syntax: baln Target

Instruction is fetched from memory.The datapath sets the control signals properly. If the previous instruction's status is negative (Status[N]==1), branch

to pseudo- direct address and link address is stored in register 31($ra). R[31] ←PC+4. If the condition is not satisfied, PC ← PC+4.

OP	Reg Dst	ALUSrc	Extop	MemTo Reg	Reg Write	Mem Read	Mem Write	ALUOp	JM	BLTZ AL	BRZ	BALN	BRANCH
baln	10	X	1	10	1	0	0	X	0	0	0	1	0

jm instruction (I type):

010010(opcode)	rs	rt	imm16
6 bits	5 bits	5 bits	16 bits

Syntax: jm imm16($rs)

Instruction is fetched from memory. The datapath sets the control signals properly and then registers are read, ALU output is generated. ALU makes logical ADD operation between register $rs and the sign extended immediate.The output is the address on Data Memory.. The value on this address indicates the address to jump (indirect jump). PC←M

OP	Reg Dst	ALUSrc	Extop	MemTo Reg	Reg Write	Mem Read	Mem Write	ALUOp	JM	BLTZ AL	BRZ	BALN	BRANCH
jm	X	1	1	01	0	1	0	101	1	0	0	0	0

srlv instruction (R type):

000000(opcode)	rs	rt	rd	shamt	000110(function code)
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Syntax: srlv $rd, $rt, $rs

Instruction is fetched from memory. The datapath sets the control signals properly. and then registers are read, ALU output is generated. ALU makes shift operation. Register $rt is shifted to right by the value in register $rs, and store the result in register $rd. PC←PC+4

OP	Reg Dst	ALUSrc	Extop	MemTo Reg	Reg Write	Mem Read	Mem Write	ALUOp	JM	BLTZ AL	BRZ	BALN	BRANCH
srlv	01	0	1	00	1	0	0	010	0	0	0	0	0

SINGLE CYCLE DATAPATH

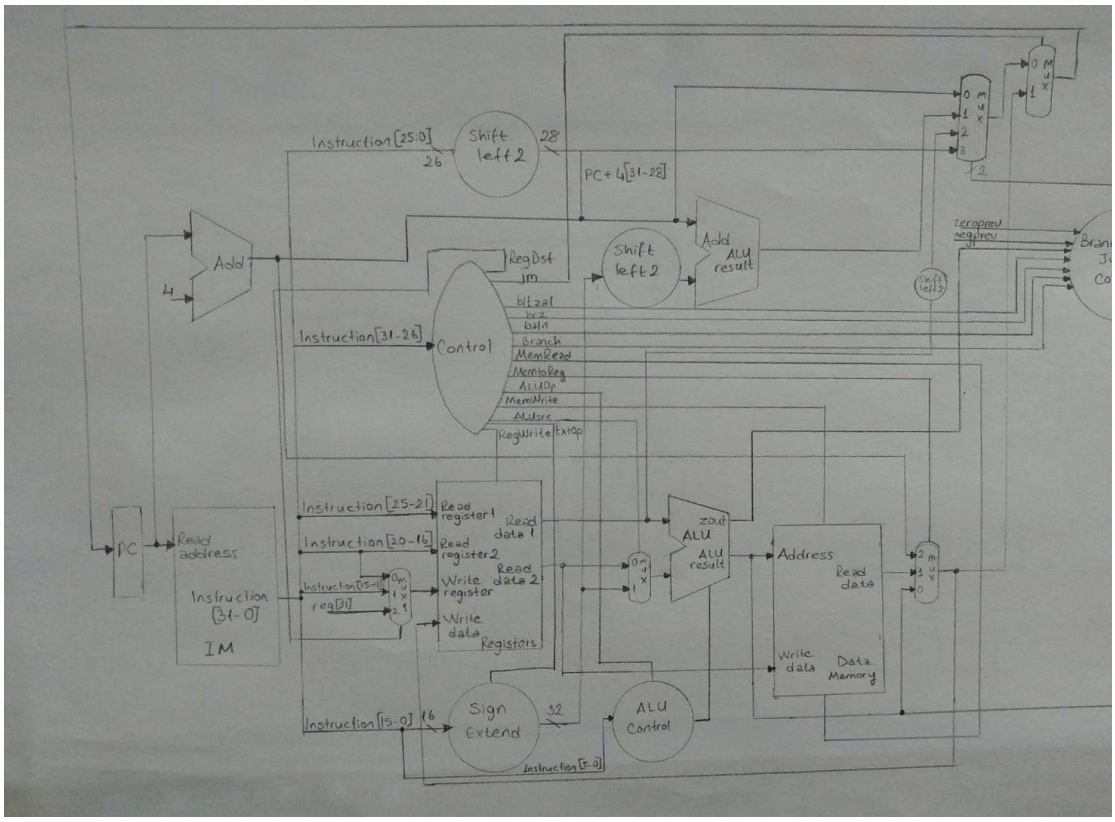


Figure 1

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' requirements. The modified datapath can be seen on Figure 1.

The additions to the existing single cycle datapath contains below:

- Three 4 to 1 Multiplexers
- One 2 to 1 Multiplexer
- Zero extension part in Sign extend module
- Branch Control Unit
- Status registers (zeroprev, negprev)
- Shit left 2 unit for instruction which has pseudo-direct address and another shifter (left 2) for read data 1.
- Reg[31] as multiplexer input for link address

First 4x1 multiplexer is used to decide write register. If the instruction is in R-type, select bits will be 2'b01 (Instruc[15:11]), if it is lw or andi, select bits will be 2'b00 (Instruc[20:16]). If bltzal or baln instructions are fetched, reg[31] input is chosen.

2nd 4x1 multiplexer is used after data memory. If memToReg is '2'b00, ALU result is taken, if it is 2'b01, the mux takes read data value. If it is 2'b10, mux takes PC+4 for link address storage.

3rd 4x1 multiplexer is used to decide the value which is sent to the PC. Output signal of Branch control unit is select bit of this mux. If select bit is 2'b00, the chosen value is PC+4; if it is 2'b01, branch target address; if it is 2'b10, read data 1 value comes after shifted; if it is 2'b11, the value is pseudo-direct address. We put a shifter for read data 1, because the value inside this related register may not be multiple of 4. Thus, we add a shift left 2 unit to prevent any problem about PC addressing.

2x1 multiplexer is used for decision of value comes from either read data 2 or sign extend unit.

We add a zero extension inside of Sign extend module. It is decided by ExtOp signal comes from control unit. If our instruction is andi, extop equals ~andi, and it makes extension by zeros. Otherwise, it put zero or one according to most significant bit of data.

Status registers keep the previous instruction's state: zero for zeroprev or negative for negprev.

Shit left 2 unit for instruction which has pseudo-direct address as branch address and another shifter for read data 1 is put. It is used for brz instruction.

SECOND DATAPATH IMPLEMENTATION

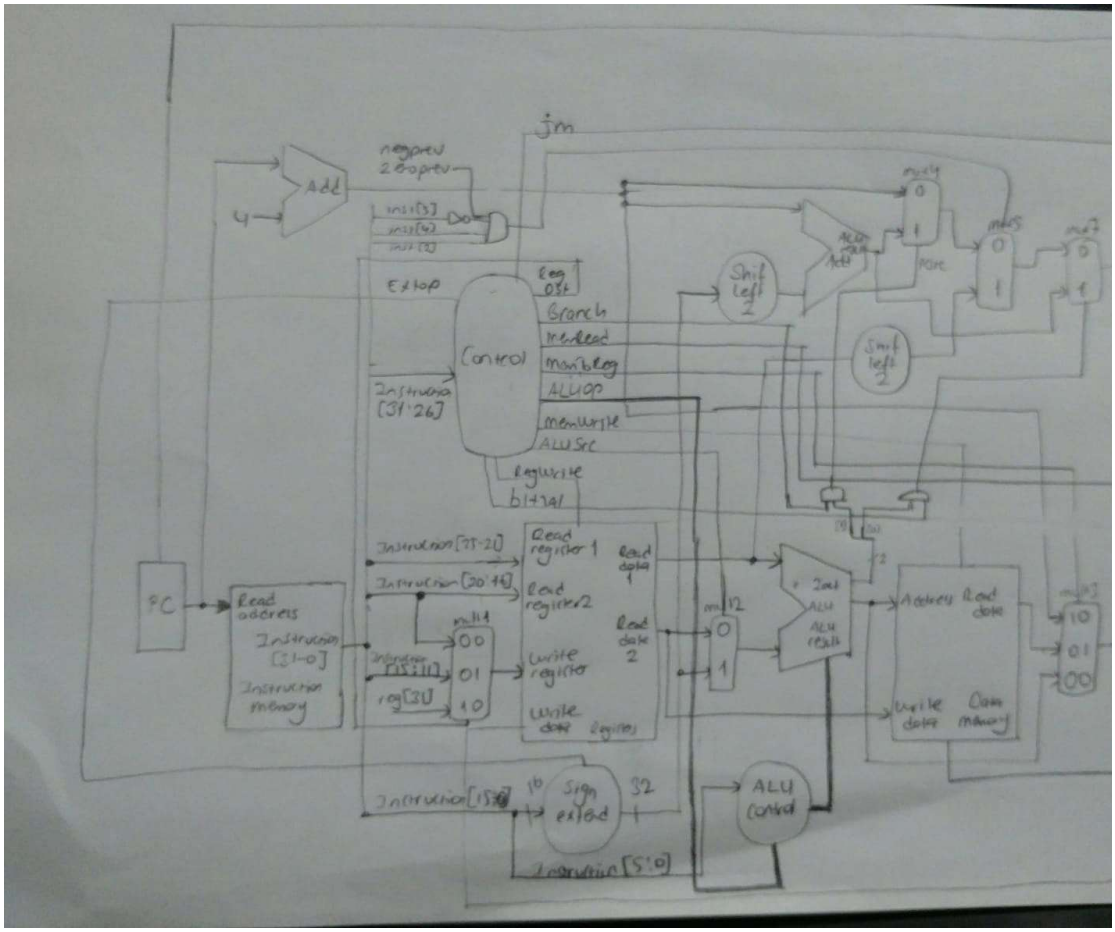


Figure 2

In this design, we work our simulation properly without error. Previous datapath belongs to our basic design but in the simulation we encountered some problems and we couldn't find

bugs and work it correctly. Thus, we also implement the datapant shown in Figure 2.

The additions to the existing single cycle datapath contains below:

- Two 4 to 1 Multiplexers
- Four 2 to 1 Multiplexer
- Zero extension part in Sign extend module
- Status registers (zeroprev, negprev)
- Shit left 2 unit for instruction which has pseudo-direct address and another shifter (left 2) for read data 1.
- Reg[31] as multiplexer input for link address

First 4x1 multiplexer is used to decide write register. If the instruction is in R-type, select bits will be 2'b01 (Instruc[15:11]), if it is lw or andi, select bits will be 2'b00 (Instruc[20:16]). If bltzal or baln instructions are fetched, reg[31] input is chosen.

2nd 4x1 multiplexer is used after data memory. If memToReg is '2'b00, ALU result is taken, if it is 2'b01, the mux takes read data value. If it is 2'b10, mux takes PC+4 for link address storage.

First 2x1 multiplexer (mult2) is used for decision of value comes from either read data 2 or sign extend unit.

Second 2x1 multiplexer (mux4) is 1 when beq instruction satisfies its required conditions. If it is not, mux take PC+4.

Third 2x1 multiplexer (mux5) is 1 when brz instruction satisfies its required conditions.

Fourth 2x1 multiplexer (mux7) is 1 when bltzal instruction satisfies its required conditions.

Fifth 2x1 multiplexer (mux6) is 1 when jm instruction satisfies its

required conditions.

We add a zero extension inside of Sign extend module. It is decided by ExtOp signal comes from control unit. If our instruction is andi, extop equals ~andi, and it makes extension by zeros. Otherwise, it put zero or one according to most significant bit of data.

Status registers keep the previous instruction’s state: zero for zeroprev or negative for negprev.

Shit left 2 unit for instruction which has pseudo-direct address as branch address and another shifter for read data 1 is put. It is used for brz instruction.