DOCUMENTATION

Design and Synthesis of a 32-Bit RISC-Like Processor

Instruction Set Architecture

And

Control Unit Design Report

Group 7

22CS10049 – Nived Roshan Shah

22CS30016 – Omkar Bhandare

INSTRUCTION SET:

The following table depicts the given instruction set architecture in the problem.

Instruction	Addressing Type	Instruction Type	Usage	Description
ADD	Register	R	add rd, rs, rt	*rd <- (rs) op (rt)*
SUB	Register	R	sub rd, rs, rt	
AND	Register	R	and rd, rs, rt	
OR	Register	R	or rd, rs, rt	
XOR	Register	R	xor rd, rs, rt	
NOR	Register	R	nor rd, rs, rt	
NOT	Register	R	not rd, rs, rt	
SL	Register	R	sl rd, rs, rt	*rd <- (rs) left-shift by (rt)*
SRL	Register	R	srl rd, rs, rt	*rd <- (rs) logical right-shift by (rt)*
SRA	Register	R	sra rd, rs, rt	*rd <- (rs) arithm right-shift by (rt)*
INC	Register	R	inc rd, rs, #4	*rd <- (rs) + (4)*
DEC	Register	R	dec rd, rs, #4	*rd <- (rs) - (4)*
SLT	Register	R	slt rd, rs, rt	*rd <- if (rs < rt) 1, else 0*
SGT	Register	R	sgt rd, rs, rt	*rd <- if (rs > rt) 1, else 0*
LUI	Register	I	lui rt, #imm	*rt <- $(Imm)_{16} \mathbin{\Vert} (0)_{16}$*
HAM	Register	R	ham rd, rs	*rd <- no. of ones in (rs)*
ADDI	Immediate	I	addi rs, #imm	*rs <- (rs) op (#imm)*
SUBI	Immediate	I	subi rs, #imm	
ANDI	Immediate	I	andi rs, #imm	
ORI	Immediate	I	ori rs, #imm	
XORI	Immediate	I	xori rs, #imm	
NORI	Immediate	I	nori rs, #imm	
SLI	Immediate	I	sli rs, #imm	
SRLI	Immediate	I	srli rs, #imm	
SRAI	Immediate	I	srai rs, #imm	
SLTI	Immediate	I	slti rs, #imm	
SGTI	Immediate	I	sgti rs, #imm	
LD	Register Indexed	I	ld rt, rs	*rt <- MEM[(rs) + imm]*
ST	Register Indexed	I	st rt, rs	MEM[rt + imm] <- (rs)
BR	PC Relative	J	br #offset	PC <- PC + offset
BMI	PC Relative	I	bmi rs, #imm	PC <- PC + offset, if (rs < 0)
BPL	PC Relative	I	bpl rs, #imm	PC <- PC + offset, if (rs > 0)
BZ	PC Relative	I	bz rs, #imm	PC <- PC + offset, if (rs = 0)
MOVE	Register	R	move rd, rs	rd = (rs)
CMOV	Register	R	cmov rd, rs, rt	rd = (rs < rt) ? rs : rt
HALT	Interrupt	J	halt	*Halts*
NOP	Dummy	J	nop	*No operation*

R-Type Instructions:

OPCODE	SOURCE REG 1	SOURCE REG 2	DESTINATION REG	DON'T CARE	FUNCT
6 Bits	5 Bits	5 Bits	5 Bits	6 Bits	5 Bits

Instruction	Opcode	Source Reg 1	Source Reg 2	Destination Reg	Don't Care	Function Code
ADD	000000	RS	RT	RD	XXXXXX	00000
SUB	000000	RS	RT	RD	XXXXXX	00001
AND	000000	RS	RT	RD	XXXXXX	00010
OR	000000	RS	RT	RD	XXXXXX	00011
XOR	000000	RS	RT	RD	XXXXXX	00100
NOR	000000	RS	RT	RD	XXXXXX	00101
NOT	000000	RS	RT	RD	XXXXXX	00111
SL	000000	RS	RT	RD	XXXXXX	01000
SRL	000000	RS	RT	RD	XXXXXX	01001
SRA	000000	RS	RT	RD	XXXXXX	01010
INC	000000	RS	00100	RD	XXXXXX	01011
DEC	000000	RS	00100	RD	XXXXXX	01100
SLT	000000	RS	RT	RD	XXXXXX	01101
SGT	000000	RS	RT	RD	XXXXXX	01110
HAM	000000	RS	XXXXX	RD	XXXXXX	01111
MOVE	000100	RS	XXXXX	RD	XXXXXX	XXXXX
CMOV	001000	RS	RT	RD	XXXXXX	XXXXX

I-Type Instructions:

OPCODE	SOURCE REG 1	SOURCE REG 2	IMMEDIATE VALUE
6 Bits	5 Bits	5 Bits	16 Bits

Instruction	Opcode	Source Reg 1	Source Reg 2	Immediate
ADDI	000001	RS	RS	#imm
SUBI	000011	RS	RS	#imm
ANDI	000101	RS	RS	#imm
ORI	000111	RS	RS	#imm
XORI	001001	RS	RS	#imm
NORI	001011	RS	RS	#imm
SLI	010001	RS	RS	#imm
SRLI	010011	RS	RS	#imm
SRAI	010101	RS	RS	#imm
SLTI	011011	RS	RS	#imm
SGTI	011101	RS	RS	#imm
LUI	001101	RS	RS	#imm
LD	011101	RS	RT	#imm
ST	011111	RS	RT	#imm
BMI	100001	RS	RS	#imm
BPL	100011	RS	RS	#imm
BZ	100101	RS	RS	#imm

J-Type Instructions:

OPCODE	OFFSET VALUE
6 Bits	26 Bits

Instruction	Opcode	Offset Value
BR	100110	#offset
HALT	000110	#offset
NOP	001010	#offset

Register Encoding:

Register	Function	Code
\$R0	Hardwired to 0	00000
\$R1-\$R15	General Purpose Registers	00001 – 01111
\$RET	Store return value of function	10000
\$PC	Program Counter	10001

Design of Data Path:

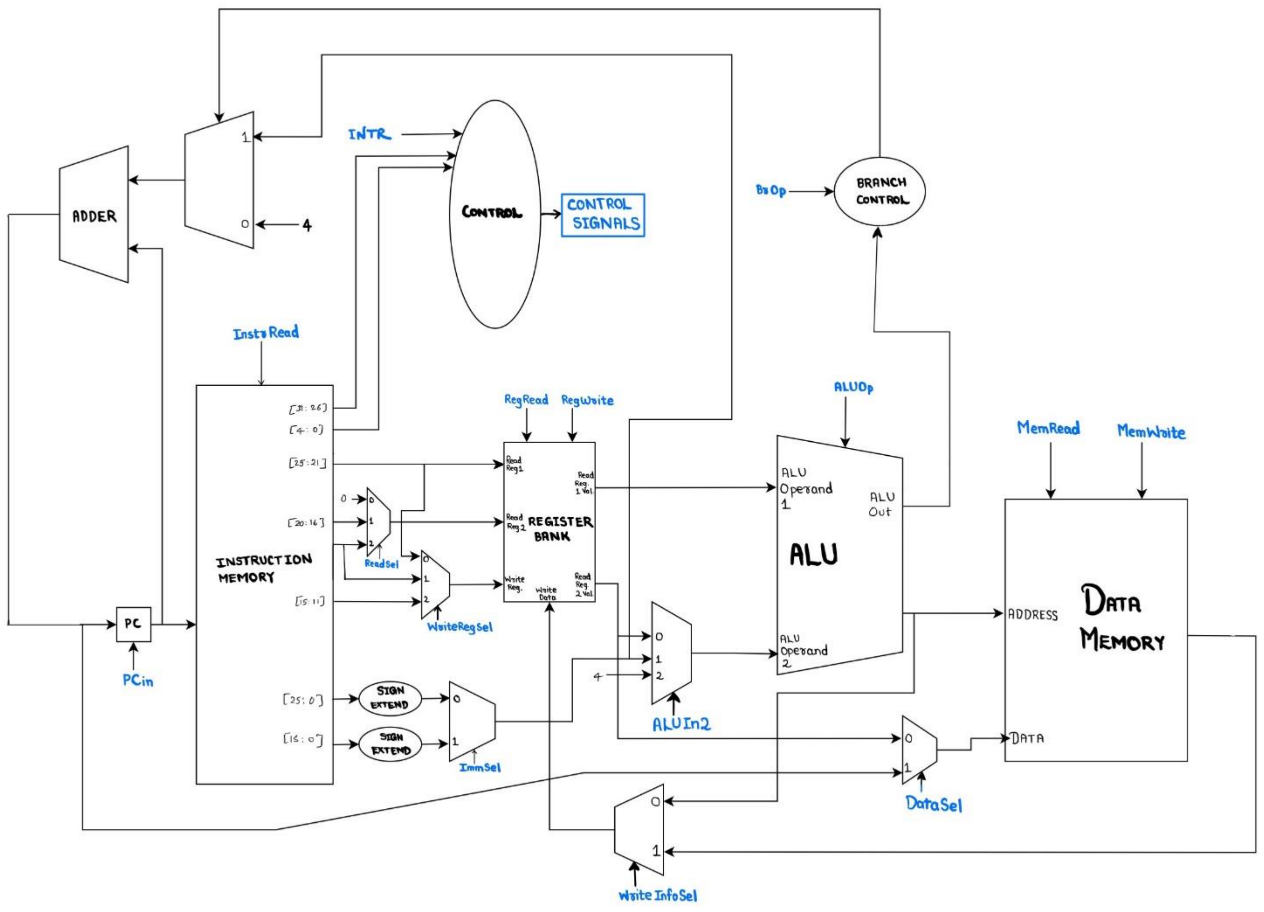

Table for Control Signals

Opcode	ACTION	Pcin	InstRead	ReadSel	WriteRegSel	WriteInfoSel	RegRead	RegWrite	ImmSel	ALUin2	ALUOp	BrOp	DataSel	MemRead	MemWrite
000000	**ALU**	1	1	1	2	0	1	1	x	0	Func	100	x	x	x
001101	lui	1	1	x	1	0	1	1	1	1	xxxx	100	x	x	x
000100	move	1	1	0	2	0	1	1	x	0	0000	100	x	x	x
001000	cmov	1	1	1	2	0	1	1	x	0	xxxx	100	x	x	x
000001	addi	1	1	x	1	0	1	1	1	1	0000	100	x	x	x
000011	subi	1	1	x	1	0	1	1	1	1	0001	100	x	x	x
000101	andi	1	1	x	1	0	1	1	1	1	0010	100	x	x	x
000111	ori	1	1	x	1	0	1	1	1	1	0011	100	x	x	x
001001	xori	1	1	x	1	0	1	1	1	1	0100	100	x	x	x
001011	nori	1	1	x	1	0	1	1	1	1	0101	100	x	x	x
001111	sli	1	1	x	1	0	1	1	1	1	0111	100	x	x	x
010001	srli	1	1	x	1	0	1	1	1	1	1000	100	x	x	x
010011	srai	1	1	x	1	0	1	1	1	1	1001	100	x	x	x
011001	slti	1	1	x	1	0	1	1	1	1	1100	100	x	x	x
011011	sgti	1	1	x	1	0	1	1	1	1	1101	100	x	x	x
011101	ld	1	1	x	1	1	1	1	1	1	0000	100	x	1	0
011111	st	1	1	2	x	x	1	0	1	1	0000	100	0	0	1
100001	bmi	1	1	0	x	x	1	0	1	0	0001	000	x	0	0
100011	bpl	1	1	0	x	x	1	0	1	0	0000	001	x	0	0
100101	bz	1	1	0	x	x	1	0	1	0	0100	010	x	0	0
100110	br	1	1	x	x	x	0	0	0	x	1111	011	x	0	0
000110	halt	0	1	x	x	x	0	0	x	x	xxxx	100	x	0	0
001010	nop	1	1	x	x	x	0	0	x	x	xxxx	100	x	0	0

Descriptions of Control Signals:

PCin - Determines if adjusted PC value is {1} loaded into PC register or {0} ignored

InstRead - Determines if {1} instruction at pointer is fetched or {0} skipped

ReadSel - Selects second read register source: {0} zero constant, or registers 0-15 from {1} bits [20:16] or {2} bits [4:0]

WriteRegSel - Determines destination register: {0} same as Read Register 1, {1} registers 1-15 from second operand, or {2} registers 1-15 from third operand

WriteInfoSel - Selects data to write to register: {0} ALU result or {1} memory-read data

RegRead - Enables {1} or disables {0} register reading operations

RegWrite - Enables {1} or disables {0} register writing operations

ImmSel - Selects immediate value format: {0} 26-bit or {1} 16-bit

ALUin2 - Selects second ALU operand: {0} Read Register 2 value, {1} immediate value, or {2} constant 4

ALUOp - Specifies ALU function: {0000} Add, {00001} Subtract, {00010} AND, {00011} OR, {00100} XOR, {00101} NOT, {00110} Left Arithmetic Shift, {00111} Right Arithmetic Shift, {01000} Right Logical Shift, {11111} No operation

BrOp - Determines PC Adder's second operand: {1} flag-controlled value or {0} constant 4

DataSel - Selects between {0} Read Register 2 value or {1} modified PC value

MemRead - Enables {1} or disables {0} memory read operations

MemWrite - Enables {1} or disables {0} memory write operations

HALTCtrl - Determines if execution {1} stops or {0} continues. It is only 1 for the HALT instruction, for the rest it is "don't care"

Assumptions and Considerations:

- Harvard-Architecture has been taken for consideration in the given processor design (that is, instruction and data memory are separate)
- The immediate type registers are used for ALU and Load/Store instructions, however ALU operations are performed on the source registers themselves (according to the problem statement). This makes the RT bits of ALU Immediate operations sort of redundant, but cohering to the MIPS architecture (for provision-based development) we have not added a different instruction type for the ALU-Imm Type instructions and kept them I-Type.