Single-cycle Simple RISC Processor Design Report

By

Chappidi Yoga Satwik (19CS30013)

Seemant Guruprasad Achari (19CS10055)

Instruction Format and Encoding

The following architecture has 32-bits long instruction encoding and 32 general-purpose registers with no. 31 being $ra, to store the return address of the callee. $10, $11, $12 and $13 to include the first four arguments and $14 for the return value. $0 is the stack point and $1 is the base pointer.

R-type instructions

OP Code	RS	RT	Function	Don’t Care
6 bits	5 bits	5 bits	8 bits	8 bits

Table I a

Instruction	Usage	OP Code	Function Code
add	add rs, rt	000000	00000001
comp	comp rs, st	000000	00000010
shift left logical variable	shllv rs, rt	000000	00000100

shift right logical variable	shrlv rs, rt	000000	00001000
shift right arithmetic variable	shrav rs, rt	000000	00010000
branch register	br rs	000000	00100000
AND	and rs, rt	000000	01000000
XOR	xor rs, rt	000000	10000000

I-type instructions

OP Code	RS	RT	Immediate
6 bit	5 bit	5 bit	16 bit

Table I b

Instruction	**Usage**	**OP Code**	**Function Code**
Load Word	lw rs, imm(rt)	100000	NA
Store Word	sw rs, imm(rt)	100001	NA

J-type instructions

OP Code	Address
6 bit	26 bit

Table I c

Instruction	**Usage**	**OP Code**	**Function Code**
Unconditional Branch	b L	100101	NA

Branch and Link	bl L	100110	NA
Branch on Carry	bcy L	100111	NA
Branch on No Carry	bcny L	101000	NA

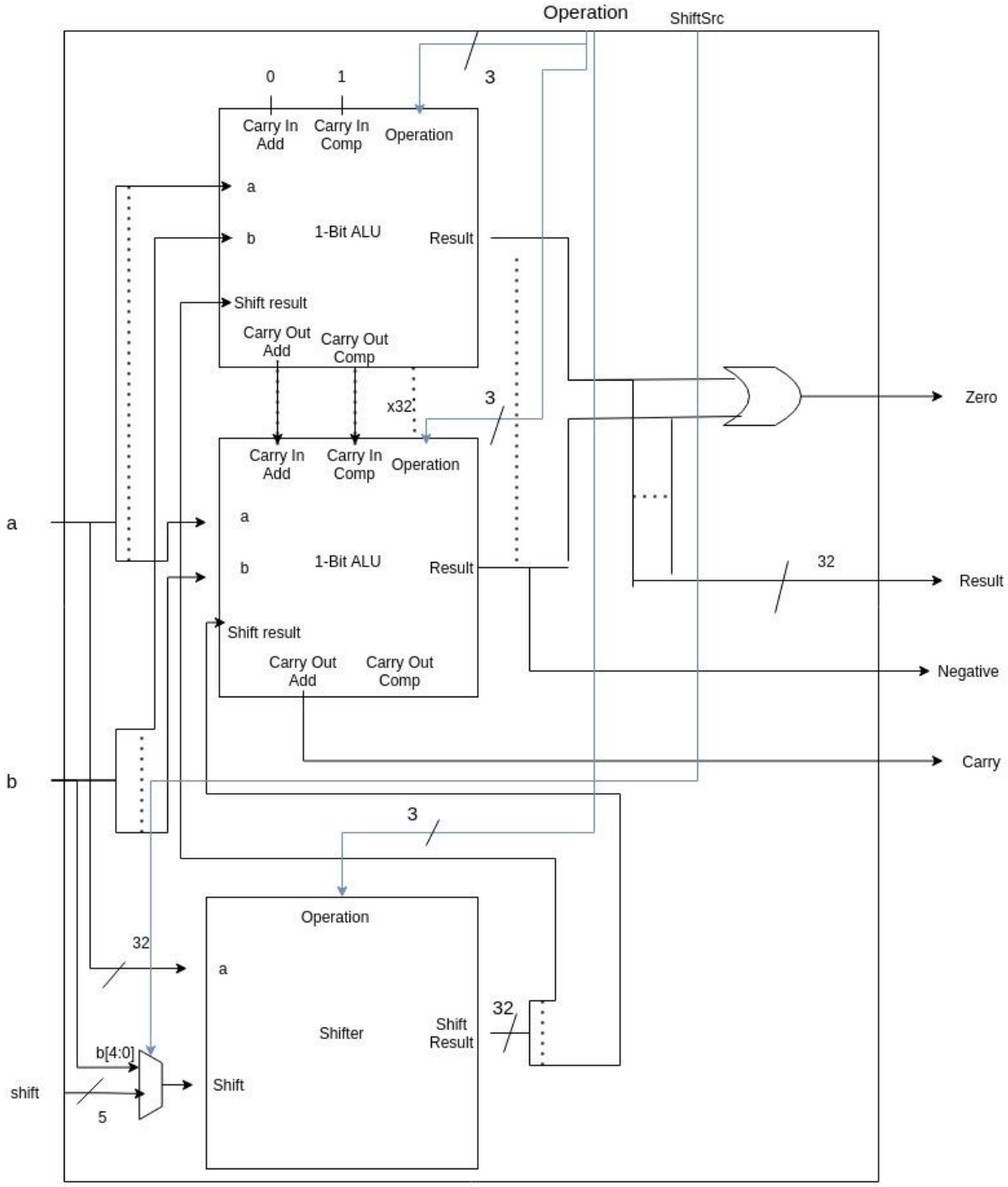
I*-type instructions

OP Code	RS	Shift	Immediate
6 bit	5 bit	5 bit	16 bit

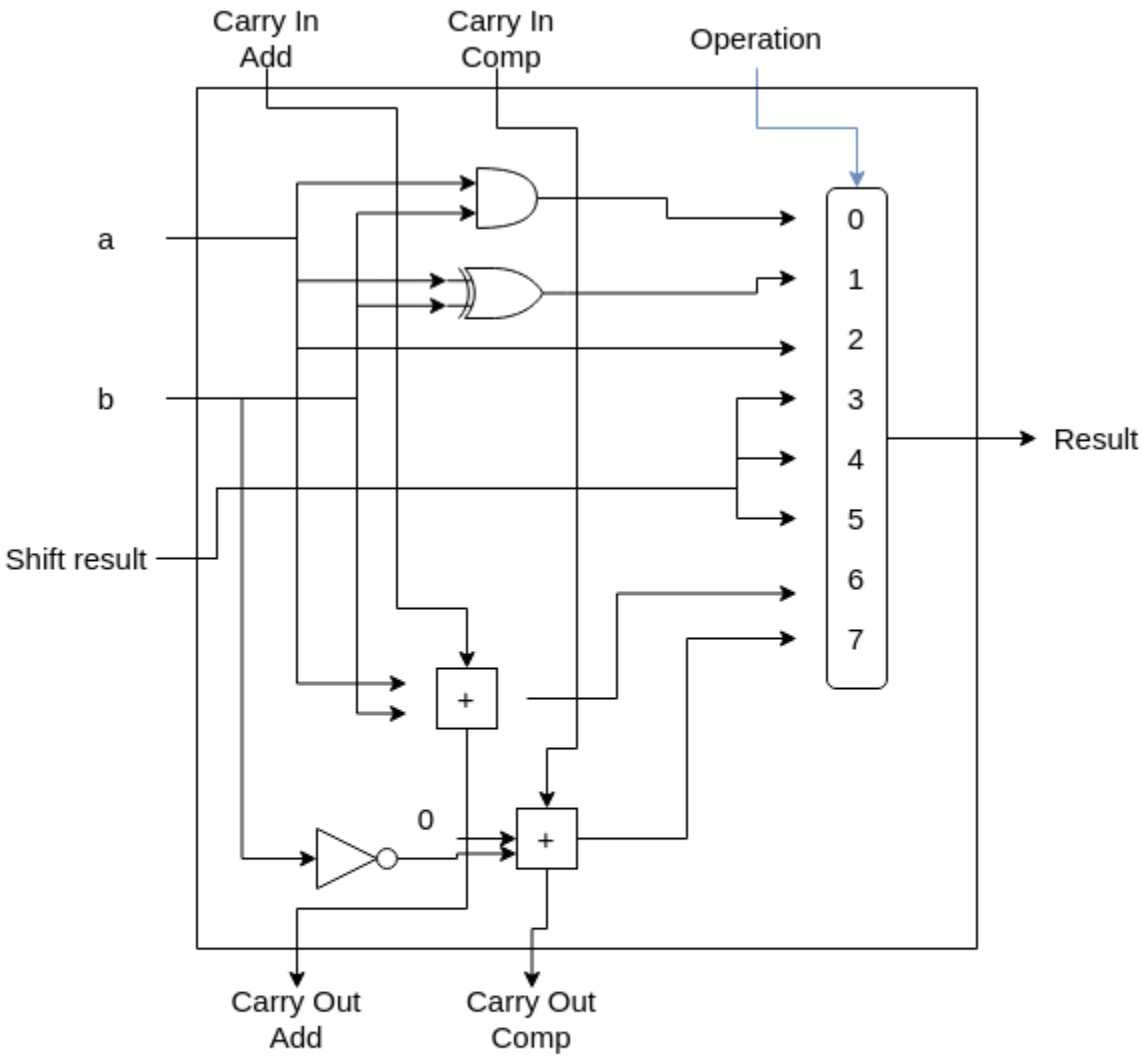
Table I d

Instruction	**Usage**	**OP Code**	**Function Code**
Add Immediate	addi rs, imm	000001	NA
Complement Immediate	compi rs, imm	000010	NA
Shift Left Logical	shll rs, sh	000011	NA
Shift Right Logical	shrl rs, sh	000100	NA
Shift Right Arithmetic	shra rs, sh	000101	NA
Branch on less than 0	bltz rs, L	000110	NA
Branch on flag 0	bz rs, L	000111	NA
Branch on flag not 0	bnz rs, L	001000	NA

ALU Design



ALU



1 - Bit ALU

Control signals are colored blue, Data paths are colored black

Table II

Operation	Action
000	AND of a and b
001	XOR of a and b
010	Pass a, so Negative and Zero flags are computed

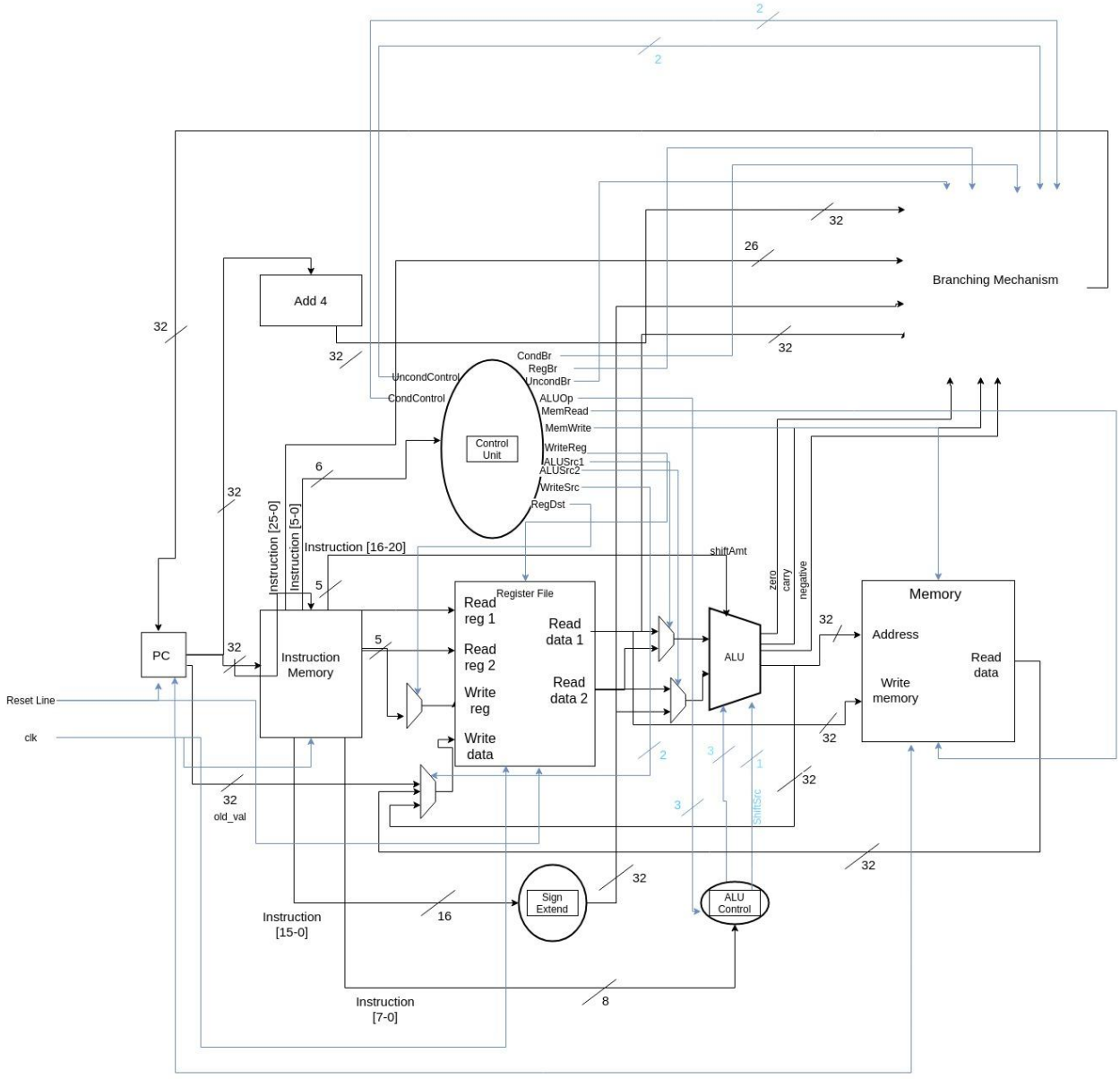
011	Pass Left Shift result from Shifter Module
100	Pass Right Shift result from Shifter Module
101	Pass Arithmetic Shift result from Shifter Module
110	Addition of a and b
111	Complement of b

Truth Table for ALU Control

Table III

Instruction	ALU Op	Function Code	Operation	ShiftSrc
add	000	00000001	110	X
comp	000	00000010	111	X
shllv	000	00000100	011	1
shrlv	000	00001000	100	1
shrav	000	00010000	101	1
and	000	01000000	000	X
xor	000	10000000	001	X
bltz, bz, bnz	001	XXXXXXXX	010	X
shll	010	XXXXXXXX	011	0
shrl	011	XXXXXXXX	100	0
shra	100	XXXXXXXX	101	0
addi, lw, sw	101	XXXXXXXX	110	X
compi	110	XXXXXXXX	111	X

Complete Data Path with Control Signals



Control signals are colored blue, Data paths are colored black

Control Signals

CondBr: Set true for conditional branching, i.e., for bltz, bz, bnz, bcy, bncy
RegBr: Set true for jump to register, i.e. br
UncondBr: Set true for unconditional branching, i.e. b, bl
MemRead: Set true when memory is to be read, i.e. lw
MemWrite: Set true when memory is to be written on, i.e. sw
WriteReg: Set true when we have to write into a register
ALUSrc1: The first input for ALU can either be DataRead 1 or DataRead 2. ALUSrc1 controls from where we input into the ALU's first port.
ALUSrc2: The second input for ALU can either be data read from a register or sign-extended immediate value. ALUSrc2 controls from where we input into the ALU's second port.
WriteSrc: There are three sources from where we can write into the registers, i.e., data read from memory or result from the ALU or the next instruction (for bl instruction). WriteSrc is a 2 bit wide control signal deciding on the source from where to write.
RegDst: For other than bl instruction, the destination register is the first register itself, but for bl it is fixed, i.e., the $ra register (register number 31).
ALUOp: ALUOp is a 3-bit wide control signal to ALU Control. ALUOp and Function Code are used to decide with result should be outputted by the ALU. Please see Table III.
UncondControl: To decide between b, bl, bcy and bncy
CondControl: To decide between bltz, bnz, bz

Truth table for Control Signals

Table IV

Instrc	OpCode	FuncCode	Cond Br	RegBr	Uncond Br	MemRead	MemWrite	WriteReg	ALUSrc1	ALUSrc2	WriteSrc	RegDst	uncondControl	condControl	ALUOp
add	000000	00000001	0	0	0	0	0	1	1	1	10	1	X	X	000
comp	000000	00000010	0	0	0	0	0	1	1	1	10	1	X	X	000
shllv	000000	00000100	0	0	0	0	0	1	1	1	10	1	X	X	000
shrlv	000000	00001000	0	0	0	0	0	1	1	1	10	1	X	X	000
shrav	000000	00010000	0	0	0	0	0	1	1	1	10	1	X	X	000
br	000000	00100000	0	1	0	0	0	0	1	X	X	X	X	X	XXX
and	000000	01000000	0	0	0	0	0	1	1	1	10	1	X	X	000
xor	000000	10000000	0	0	0	0	0	1	1	1	10	1	X	X	000
lw	100000	X	0	0	0	1	0	1	0	0	01	1	X	X	101
sw	100001	X	0	0	0	1	1	0	0	0	X	X	X	X	101
b	100101	X	0	0	1	0	0	0	1	X	X	X	01	X	XXX
bl	100110	X	0	0	1	0	0	1	1	X	00	0	01	X	XXX
bcy	100111	X	0	0	1	0	0	0	1	X	X	X	11	X	XXX
bcny	101000	X	0	0	1	0	0	0	1	X	X	X	10	X	XXX
addi	000001	X	0	0	0	0	0	1	1	0	10	1	X	X	101
compi	000010	X	0	0	0	0	0	1	1	0	10	1	X	X	110

shll	000011	X	0	0	0	0	0	1	1	X	10	1	X	X	010
shrl	000100	X	0	0	0	0	0	1	1	X	10	1	X	X	011
shra	000101	X	0	0	0	0	0	1	1	X	10	1	X	X	100
bltz	00011 0	X	1	0	0	0	0	0	1	X	X	X	X	10	001
bz	000111	X	1	0	0	0	0	0	1	X	X	X	X	11	001
bnz	001000	X	1	0	0	0	0	0	1	X	X	X	X	01	001