Design and Synthesis of a 32-bit RISC Processor

CS39001: Computer Organization and Architecture Laboratory

Group 37: Dharnesh Bala (23CS10008), Rohit Ranjeet Satpute (23CS10060)

10th September 2025

Contents

1 Instruction Format and Encoding

1.1 General Instruction Format

Our 32-bit RISC processor uses a unified instruction format with 6-bit opcode and 6-bit opcode extension, providing 4096 unique instruction encodings.

Opcode	Opcode Extension	Operand Fields
6 bits	6 bits	20 bits

Table 1: General 32-bit Instruction Format

1.2 Register-Register (R-Type) Instructions

Opcode	Opcode Ext	Rd	Rs	Rt
6 bits	6 bits	4 bits	4 bits	4 bits

Table 2: R-type Instruction Format (remaining 8 bits unused)

Instruction	Opcode	Opcode Ext	ALU Control	Usage
ADD	000000	000000	FnClass=011, add_sub=0	ADD rd,rs,rt
SUB	000001	000000	FnClass=011, add_sub=1	SUB rd,rs,rt
AND	000011	000000	FnClass=100, LogicFn=00	AND rd,rs,rt
OR	000100	000000	FnClass=100, LogicFn=01	OR rd,rs,rt
XOR	000101	000000	FnClass=100, LogicFn=10	XOR rd,rs,rt
NOR	000110	000000	FnClass=100, LogicFn=11	NOR rd,rs,rt
SLL	001000	000000	FnClass=101, ShiftFn=00	SLL rd,rs,rt
SRL	001001	000000	FnClass=101, ShiftFn=01	SRL rd,rs,rt
SRA	001010	000000	FnClass=101, ShiftFn=10	SRA rd,rs,rt
SLT	001011	000000	FnClass=001	SLT rd,rs,rt
SGT	001100	000000	FnClass=010	SGT rd,rs,rt
HAM	001110	000000	FnClass=110	HAM rd,rs
MOVE	010110	000000	—	MOVE rd,rs
CMOV	010111	000000	—	CMOV rd,rs,rt

Table 3: R-type Instructions with Opcodes and ALU Control Signals

1.3 Immediate (I-Type) Instructions

Opcode	Opcode Ext	Rd/Rs	Rs/Rt	Immediate
6 bits	6 bits	4 bits	4 bits	12 bits

Table 4: I-type Instruction Format

Instruction	Opcode	Opcode Ext	ALU Control	Usage
ADDI	000000	000001	FnClass=011, add_sub=0	ADDI rd,rs,#imm
SUBI	000001	000001	FnClass=011, add_sub=1	SUBI rd,rs,#imm
INC	000010	000000	FnClass=011, add_sub=0	INC rd,rs
DEC	000010	000001	FnClass=011, add_sub=1	DEC rd,rs
ANDI	000011	010000	FnClass=100, LogicFn=00	ANDI rd,rs,#imm
ORI	000100	010000	FnClass=100, LogicFn=01	ORI rd,rs,#imm
XORI	000101	010000	FnClass=100, LogicFn=10	XORI rd,rs,#imm
NOT	000111	000000	FnClass=100, LogicFn=11	NOT rd,rs
SLLI	001000	010000	FnClass=101, ShiftFn=00	SLLI rd,rs,#amt
SRLI	001001	010000	FnClass=101, ShiftFn=01	SRLI rd,rs,#amt
SRAI	001010	010000	FnClass=101, ShiftFn=10	SRAI rd,rs,#amt
LUI	001101	000000	FnClass=000	LUI rd,#imm

Table 5: I-type Instructions with Opcodes and ALU Control Signals

1.4 Memory Instructions

Opcode	Opcode Ext	Reg	Base Reg	Displacement
6 bits	6 bits	4 bits	4 bits	12 bits

Table 6: Memory Instruction Format

Instruction	Opcode	Opcode Ext	Usage
LD	010000	000000	LD rd,disp(rs)
ST	010001	000000	ST rs,disp(rt)

Table 7: Memory Instructions

1.5 Branch Instructions

Opcode	Opcode Ext	Test Reg	PC-Relative Offset
6 bits	6 bits	4 bits	16 bits

Table 8: Branch Instruction Format

Instruction	Opcode	Opcode Ext	Condition	Usage
BR	010010	000000	Always	BR #offset
BMI	010011	000000	Rs $<$ 0	BMI rs,#offset
BPL	010100	000000	Rs $>$ 0	BPL rs,#offset
BZ	010101	000000	Rs $=$ 0	BZ rs,#offset

Table 9: Branch Instructions

1.6 Control Instructions

Opcode	Opcode Ext	Unused
6 bits	6 bits	20 bits

Table 10: Control Instruction Format

Instruction	Opcode	Opcode Ext	Description
HALT	011000	000000	Halt until interrupt
NOP	011001	000000	No operation

Table 11: Control Instructions

2 ALU Control Signals

Our custom ALU uses the following control signals to determine the operation:

FnClass	Operation Category
000	Load Upper Immediate (LUI)
001	Set on Less Than (SLT)
010	Set on Greater Than (SGT)
011	Arithmetic (ADD/SUB/INC/DEC)
100	Logic (AND/OR/XOR/NOR/NOT)
101	Shift (SLL/SRL/SRA)
110	Hamming Weight (HAM)
111	Reserved

Table 12: ALU Function Class Encoding

LogicFn	Logic Operation
00	AND
01	OR
10	XOR
11	NOR/NOT

Table 13: Logic Function Encoding (when FnClass=100)

ShiftFn	Shift Operation
00	Shift Left Logical (SLL)
01	Shift Right Logical (SRL)
10	Shift Right Arithmetic (SRA)
11	Reserved

Table 14: Shift Function Encoding (when FnClass=101)

3 Register Usage Convention

Register	Function	Address
R0	Zero Register (always 0)	0000
R1-R15	General Purpose Registers	0001-1111

Table 15: 16 32-bit General Purpose Registers

Register	Function
PC	Program Counter (32-bit)
SP	Stack Pointer (32-bit)

Table 16: Special Purpose Registers

4 Memory Organization

- Memory is byte-addressable with 32-bit addresses
- All data operations are on 32-bit words
- Load and store operations occur from addresses that are multiples of 8
- Memory is implemented as a linear array of registers
- Base addressing mode: address = base_register + displacement

5 Addressing Modes

Addressing Mode	Example	Description
Register	ADD R1,R2,R3	Operands from registers
Immediate	ADDI R1,R2,#25	One operand is immediate
Base	LD R1,10(R2)	Memory address = R2 + 10
PC-Relative	BMI R1,#-10	Branch target = PC + offset
Indirect	(Future extension)	Address from register

Table 17: Supported Addressing Modes

6 Design Justifications

6.1 Instruction Encoding

- **6+6 bit encoding**: Provides 4096 unique opcodes with clear separation between primary and secondary operation codes
- **Uniform format**: Simplifies instruction decoding logic
- **ALU integration**: Direct mapping to ALU control signals reduces control complexity

6.2 ALU Design

- **Function classes**: Hierarchical organization reduces control signal complexity
- **Structural arithmetic**: Full-adder implementation for ADD/SUB as required
- **Behavioral logic**: Simplified implementation for logic operations
- **Shift operations**: Support for both register and immediate shift amounts

6.3 Register Organization

- **16 GPRs**: Balance between instruction encoding space and register availability
- **R0 hardwired to 0**: Simplifies constant loading and comparisons
- **4-bit register addresses**: Efficient encoding within instruction format

7 Implementation Notes

- All instructions are 32 bits wide
- Memory operations must be aligned to 8-byte boundaries
- Branch targets are calculated as PC + sign-extended offset
- Overflow detection is implemented for arithmetic operations
- Control unit uses hardwired logic (not microcode)