Department of Electrical And Computer Engineering

North South University

Project Report

Submitted By

Name: Afsara Waziha

Student ID: 2021235642

Course	**: Computer Organization and Architecture**
Semester	**: Summer 2022**
Section	**: 03**

Faculty Advisor

Tanjila Farah(tnf)

Objective: My main goal for this project is to build an 18-bit ISA that can perform some specific problems for some instructions. For Example- Arithmetic and logic function, branching, and others.

Operands: I will be using three operands and will be addressing them as d, t, and s.

Types of operands: Here, to implement arithmetic instruction we need register operands and for data transfer instruction from memory to register we need memory operands. Therefore, we need two types of operands.

1. Register Based
2. Memory Based

Operations: In my designed Architecture, there will be five types of operations. They are:

1. Arithmetic
2. Logic
3. Data Transfer
4. Unconditional Jump
5. Conditional Branch

Formats: I am using three types of formats in my architecture. They are

1. Register type / R- Type
2. Immediate Type/ I- Type
3. Jump Type/ J- Type

Format of R-Type instruction:

Opcode	Rs	Rt	Rd
6 Bits	4 Bits	4 Bits	4 Bits

Format of I-Type instruction:

Opcode	Rs	Rt	Immediate
6	4	4	4

Format of J-Type instruction:

Opcode	Target
6	12

Types of Operation:

Opcode	Name	Type	Category	Operation
000000	beq	I	Conditional	Check equality
000001	add	R	Logical	Sum two number
000010	and	R	Logical	Bitwise and
000011	sw	I	Data Transfer	Store value
000100	addi	I	I type	Add with immediate
000101	nop			No operation
000110	sub	R	Arithmetic	Subtraction
000111	slt	R	Conditional	Compare less than
001000	jmp	J	Unconditional	Jump to address
001001	lw	I	Data Transfer	Load value
001010	sll	R	Logical	Shift left logic

Description of Each Format:

Beq: It checks whether the values of two registers are the same or not. If it's the same it performs the operation located in the address at Immediate value.

For example: beq r1 r1 3

000000	0001	0001	0011

Add: It sums the value in rs and rt then store in rd.

For example: add r0 r1 r2

Meaning: r2=r0+r1

000001	0000	0001	0010

And: It AND's two register values and stores the result in the destination register.

For example: and r3 r4 r5

Meaning: r5=r3 && r4

000010	0011	0100	0101

Sw: Store a value from the register to memory.

For example: sw r4 r2 1

000011	0100	0010	0001

Addi: It adds a value from the register with a value and stores the result in destination register.
For example: addi r0 r1 3
Meaning: r1=r0+3

000100	0000	0001	0011

Nop: No operation

Sub: It subtracts two registers and stores the result in the destination register.
For example: sub r3 r0 r2
Meaning: r2=r3-r0

000110	0011	0000	0010

Slt: If rs is less than rt, rd is set to 1. It gets 0 otherwise.
For example: slt r5 r6 r7
Meaning: if r5<r6 then r7=1, else r7=0

000111	0101	0110	0111

Jmp: Jump to a specific address.
For example: j 7

001000	000000000111

Lw: It loads a required memory value and writes it back into the register. It adds rs and immediate value to get the address of data memory.
For example: lw r1 r3 4

001001	0001	0011	0100

Sll: It shifts bits to the left and fills the empty bits with zeros. The shift amount is depended on the rt.
For example: sll r2 r1 r3
Meaning: r3=r2<<r1

001010	0010	0001	0011

Control Unit and ALU control Table:

Opcode	Operations	ALU op	msb	lsb	Binv	Cin	WC	Mem2Reg	ALUsrc	RegDst	Str	Ld	Branch	MEMen	Jump
000000	beq	xx	x	x	0	0	0	0	0	0	0	0	1	0	0
000001	add	11	1	1	0	0	1	0	0	0	0	0	0	0	0
000010	and	01	0	1	0	0	1	0	0	0	0	0	0	0	0
000011	sw	11	1	1	0	0	0	0	1	0	1	0	0	1	0
000100	addi	11	1	1	0	0	1	0	1	1	0	0	0	0	0
000101	nop	xx	x	x	0	0	0	0	0	0	0	0	0	0	0
000110	sub	11	1	1	1	1	1	0	0	0	0	0	0	0	0
000111	slt	10	1	0	0	0	1	0	0	0	0	0	0	0	0
001000	jmp	xx	x	x	0	0	0	0	0	0	0	0	0	0	1
001001	lw	11	1	1	0	0	1	1	1	1	0	1	0	1	0
001010	sll	00	0	0	0	0	1	0	0	0	0	0	0	0	0

Circuits

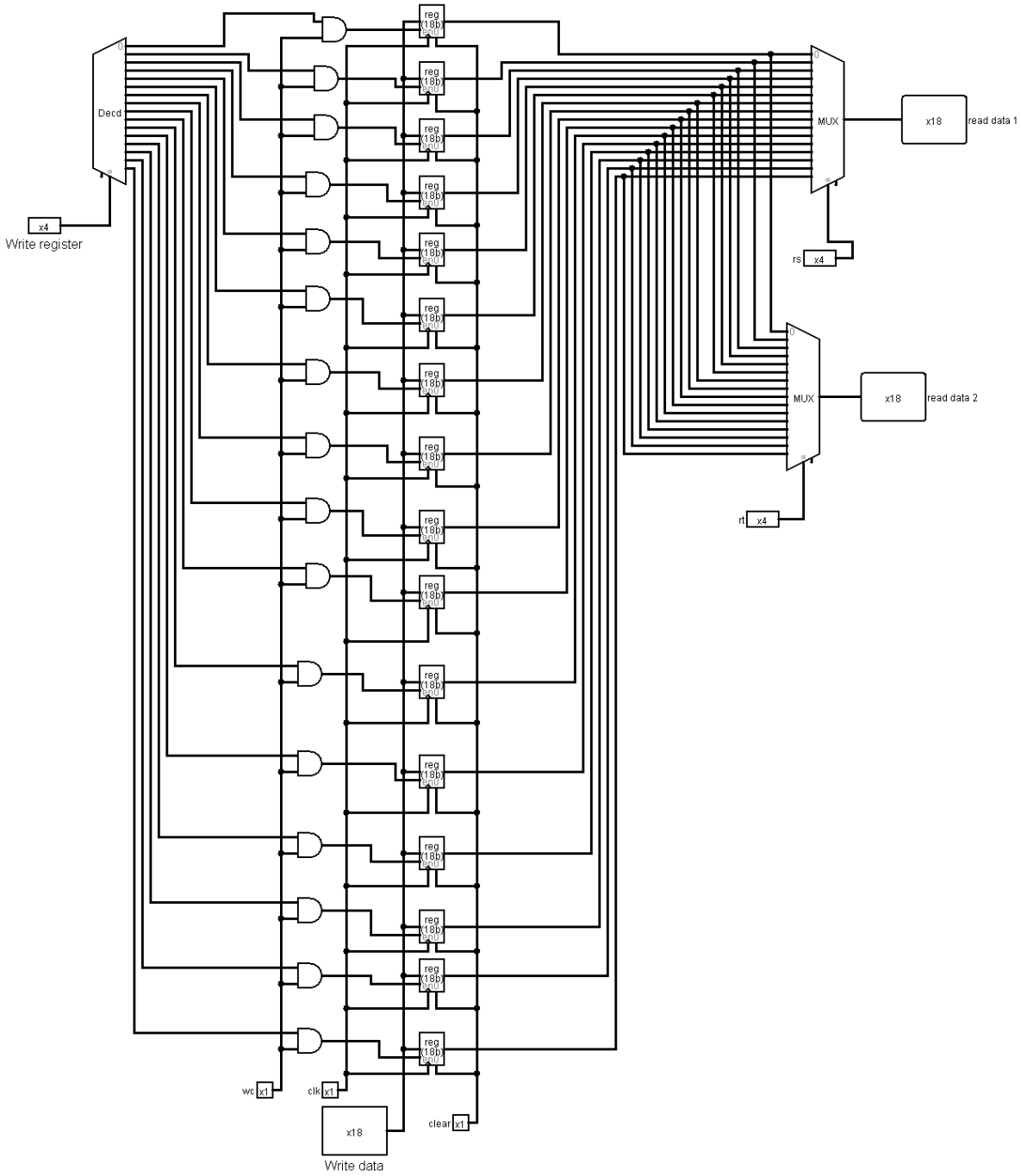
Registers File:

Read Register number 1: Rs (0-3)

Read register number 2: Rt (4-7)

Write register: Rd (8-11)

No of Register: 16



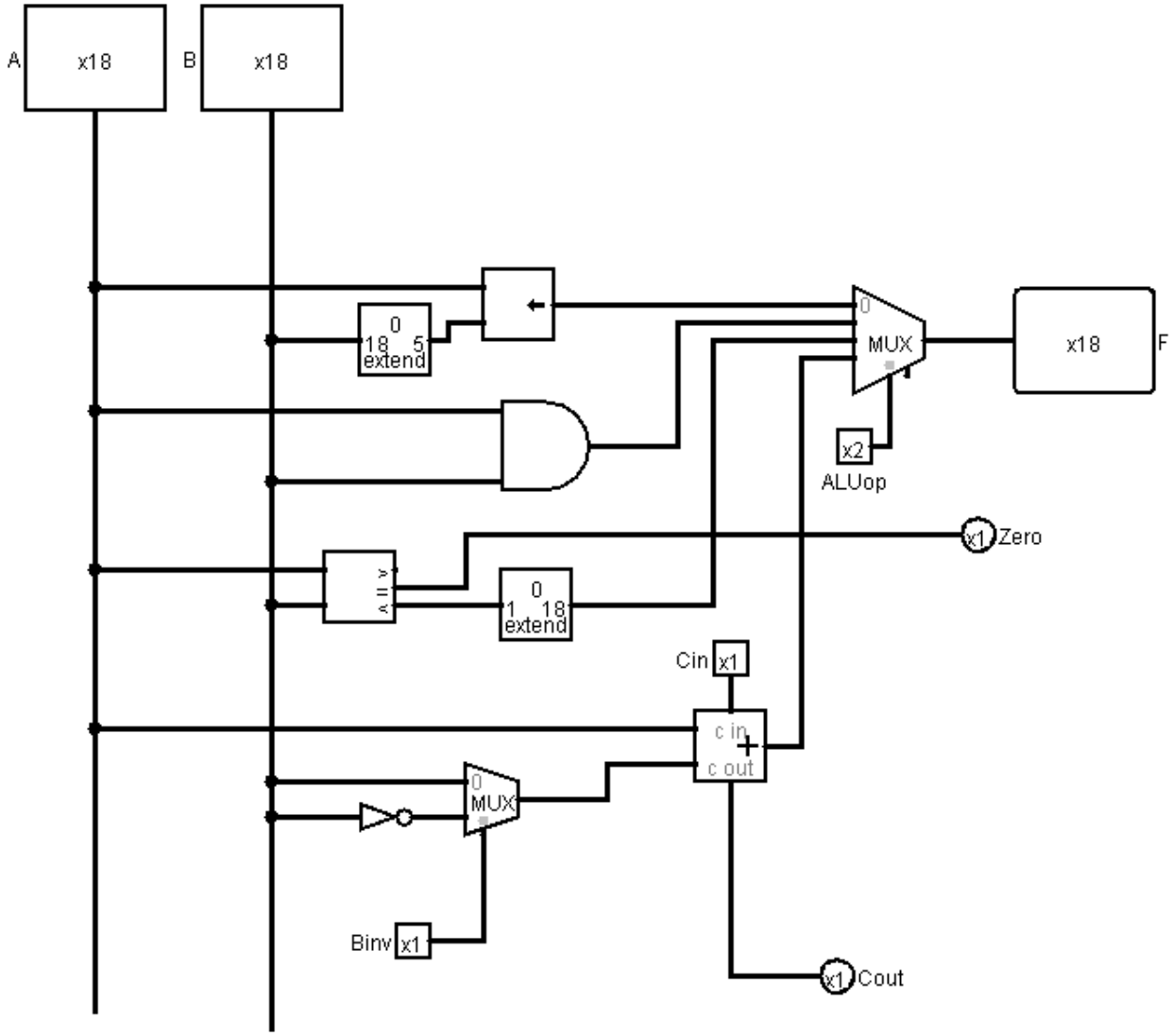
ALU:

A=18 Bit

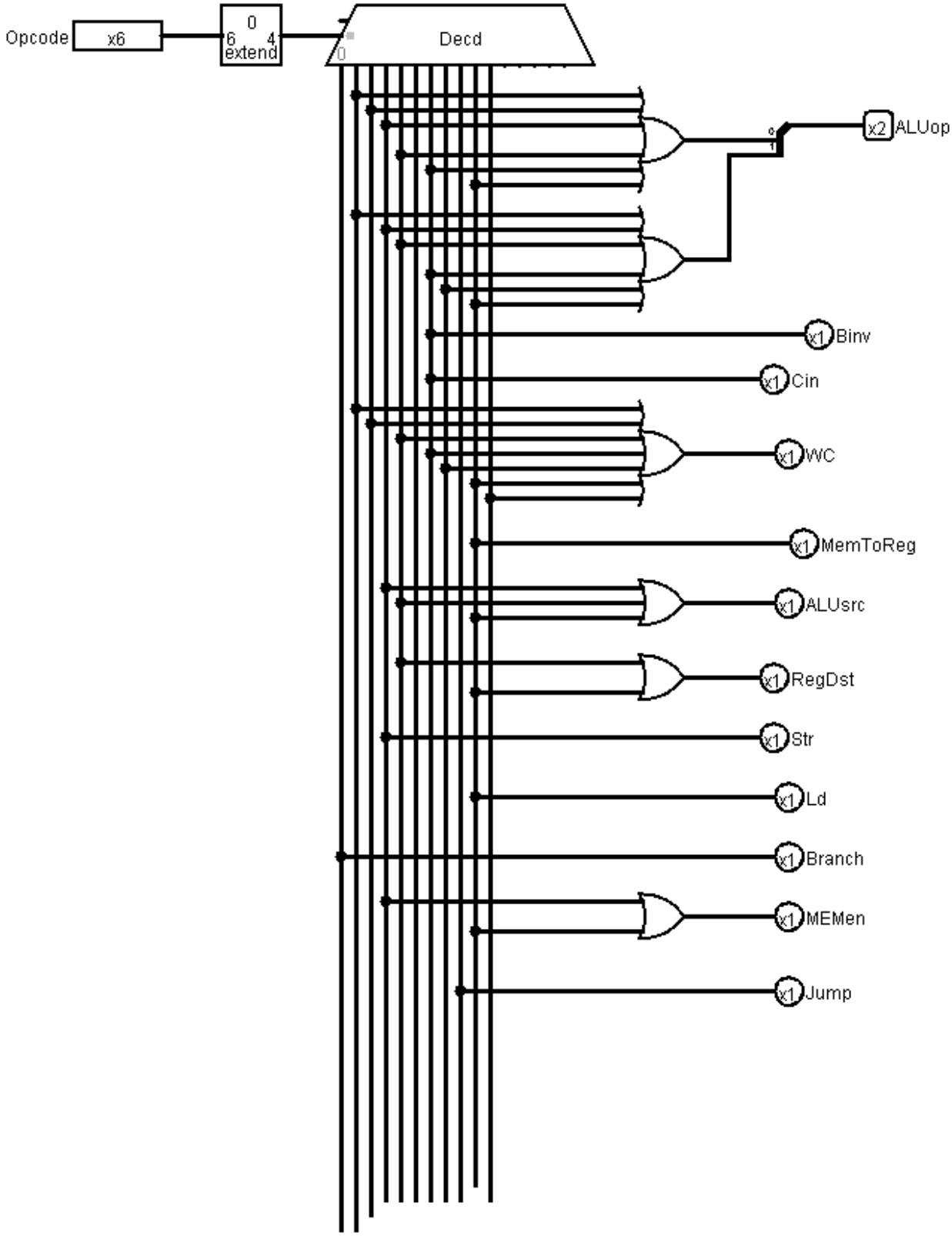
B= 18 Bit

Binv = 1 Bit (For substruction)

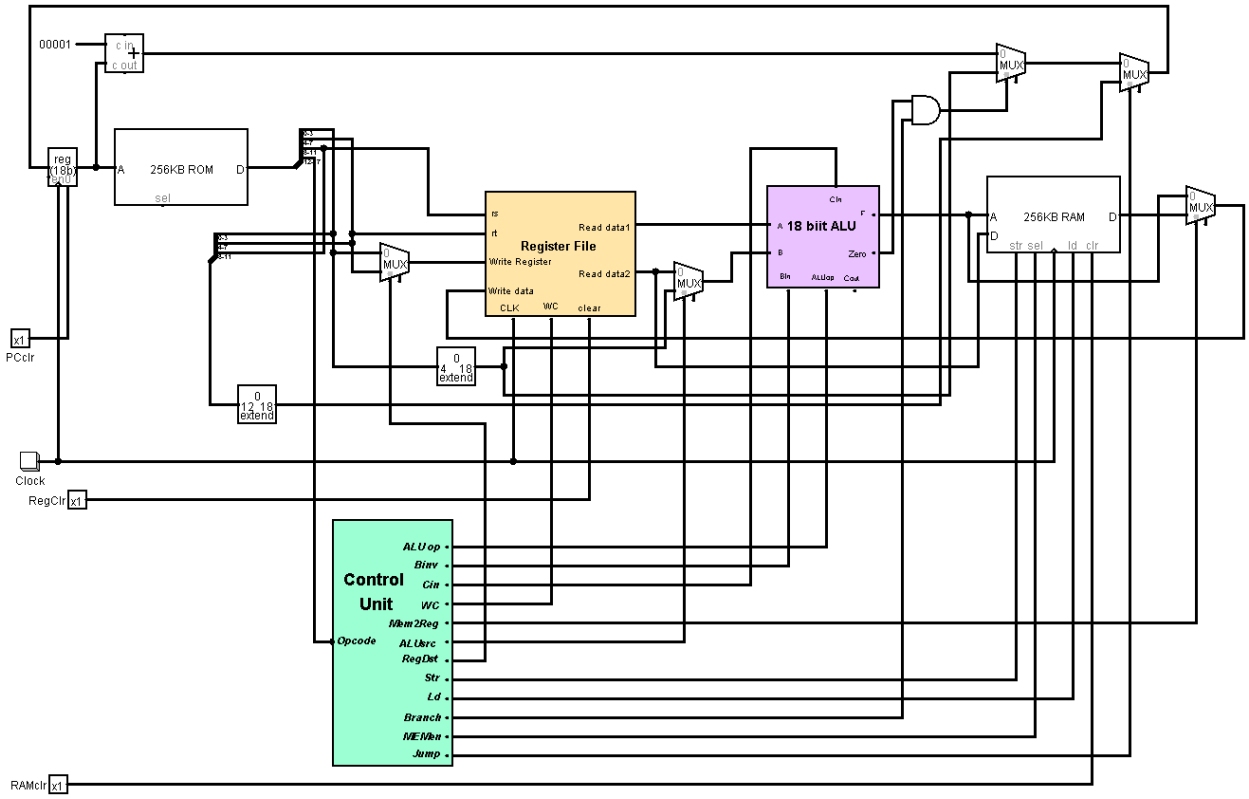
Cin = 1 (For substruction)



Control Unit:



DataPath:


00001
c in
c out
reg
(18)
256KB ROM
A
D
sel
MUX
Register File
rs
rt
Write Register
Write data
CLK
WC
clear
Read data1
Read data2
18 biit ALU
Cin
A
B
Bin
ALUop
F
Zero
Cout
256KB RAM
D
str sel
ld clr
extend
PCclr
Clock
RegClr
RAMclr
Control Unit
Opcode
ALUop
Binv
Cin
WC
Mem2Reg
ALUsrc
RegDst
Str
Ld
Branch
MEMen
Jump