Our Architecture is based on a RISC ISA. It is load/store architecture; except for load/store instructions used for accessing the memory, all instructions operate on registers.

Registers:

Here we have sixteen 16-bit general purpose registers (GPR) .

Instructions' format:

Instructions are divided into 3 types →R, I and J

Every instruction starts with a 4-bit opcode. In addition to the opcode, R-type instructions specify three registers, I-type instructions specify two registers and a 4-bit immediate value; J-type instructions follow the opcode with a 12-bit jump target.

R-type

4-bit opcode	4-bit (rd)	4-bit (rs)	4-bit(rt)

I-type

4-bit opcode	4-bit (rd)	4-bit (rs)	Immediate value(4-bit)

J-type588

4-bit opcode	12-bit address

JR-type

4-bit opcode	4-bit (rs)	8-bit address

JZ-type

4-bit opcode	4-bit (rs)	4-bit(rt)	offset

JG-type

4-bit opcode	4-bit (rs)	4-bit(rt)	offset

JAL-type

4-bit opcode	12-bit (address)

JS-type

4-bit opcode	4-bit (rs)	4-bit(rt)	4-bit (address)

Our Instruction Set:

- ✓ Arithmetic instructions -->

instruction	Example	Comments
Addr	Addr *r1,*r1,*r2	
*addi	Addi *r0,*r1,7	It also performs subtraction
Mul	Mul *r0,*r1,*r2	
Mod	Mod *r3,*r1,*r2	
Fact	Fact *r3,*r2	

✓ Jumps instructions -->

J	J [label]	Unconditional jump
Jr	Jr *r2	Jump register
Jz	Jz *r3,*r1,[label]	Jz if two registers are equal
JG	JG *r3,*r1,[label]	JG if *r3> *r1
JS	JS [label]	JS if after adding 2 registers the result is a signed number
Jal	Jal [subroutine]	Jal to a subroutine

✓ Registers -->

Zero Register	*0
Output registers	*R0,*R1
Input registers	*I0,*I1,*I2,*I3
Temp registers	*T0,*T1,*T2,*T3
Saved registers	*S0,*S1
Special purpose Registers	*RA , *SP ,*PC

✓ The logical & Load/Store Instructions—>

AND	AND*r0,*r1,*r2
OR	OR*r0,*r1,*r2
NOT	NOT*r0,*r1,*r2
L	L *r2 , offset [address]

S	S *r1,offset[address]

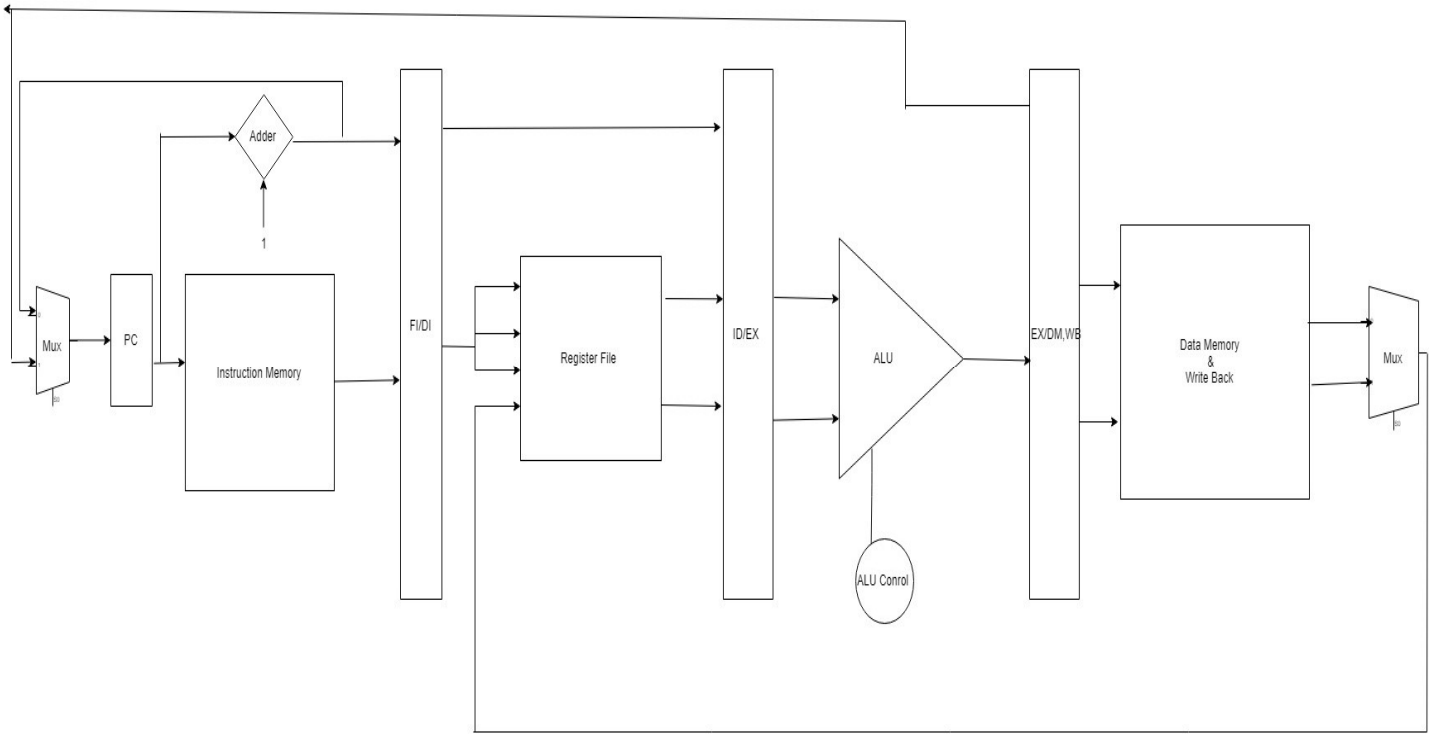
✓ Instructions Opcodes —>

Instruction	Opcode
Addr	0000
*addi	0001
AND	0101
OR	0110
NOR	0111
L	1110
S	1111
Mul	0010
Mod	0011
Fact	0100
J	1010
Jr	1100
Jz	1000
JG	1001
JS	1101
Jal	1011

✓ Registers ‘ Binary representations —>

Registers Names	Their Binary Representations
*0	0000
*r0	0001
*r1	0010
*I0	0011
*I1	0100
*I2	0101
*I3	0110
*T0	0111
*T1	1000
*T2	1001
*S0	1010
*S1	1011
*S2	1100
*ra	1101
*sp	1110
*pc	1111

✓ Data path —>


Adder
1
Mux
PC
Instruction Memory
FI/DI
Register File
ID/EX
ALU
ALU Conrol
EX/DM,WB
Data Memory
&
Write Back
Mux