NORTH SOUTH UNIVERSITY

Project Report
On
19 bits MIPS Architecture

Course Code : CSE 332

Course: Computer Organization and Architecture

Section: 02

Submitted by

MD.FATIN HABIB NIHAL
ID: 1911350642

Submitted to

Ms. Tanjila Farah
Senior Lecturer
NORTH SOUTH UNIVERSITY

Introduction:

In this project our main objective is to design a 19 bits MIPS architecture. This architecture is able to do some specific operations like arithmetic and logic operations, branching, loops and jump operations. To design this architecture we are going to use three operands and the operands are s, t, and d. In this architecture to do the arithmetic operations we need register based operands and to do the data transfer operations from memory to register we need memory based operands. Here, in this architecture we took the opcode as 4 bits so that here we can set only 2^4 = 16 operations.

List of Components:

Here to build this architecture we use different types of components like Registers, Program Counter, ROM,RAM ALU, Control unit, Multiplexers, Extenders, Adders, Shifter etc.

Registers: Mainly a register file is not a disk file, it is a small set of high speed storage cells inside the CPU. There are some register which has some special purpose like IR and PC. There are also some general purpose register which are mainly stores operands of instruction such as add, subtraction, multiplication etc. Generally a register stores bits of data as well as mapping locations in a CPU of a computer. Here for this project we use a 32 bit register file. As in this 19 bit architecture I use 5 bits for rs, rt and rd that's why I designed 32 single registers inside the register file.

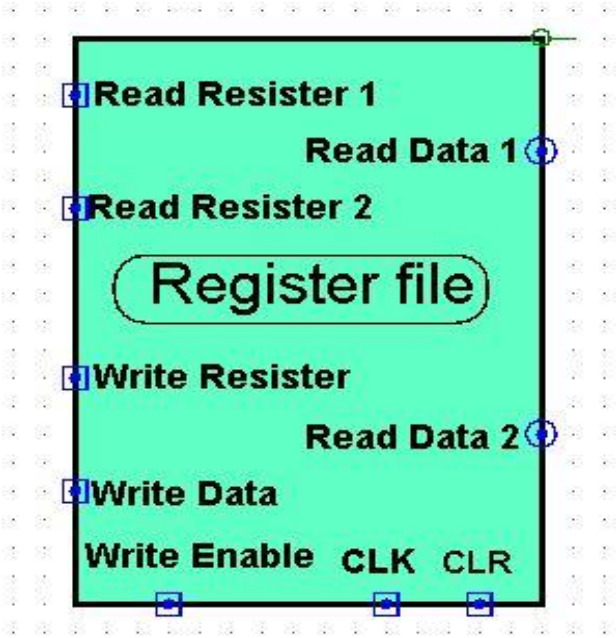

Figure-1: 32 Bit Register File IC

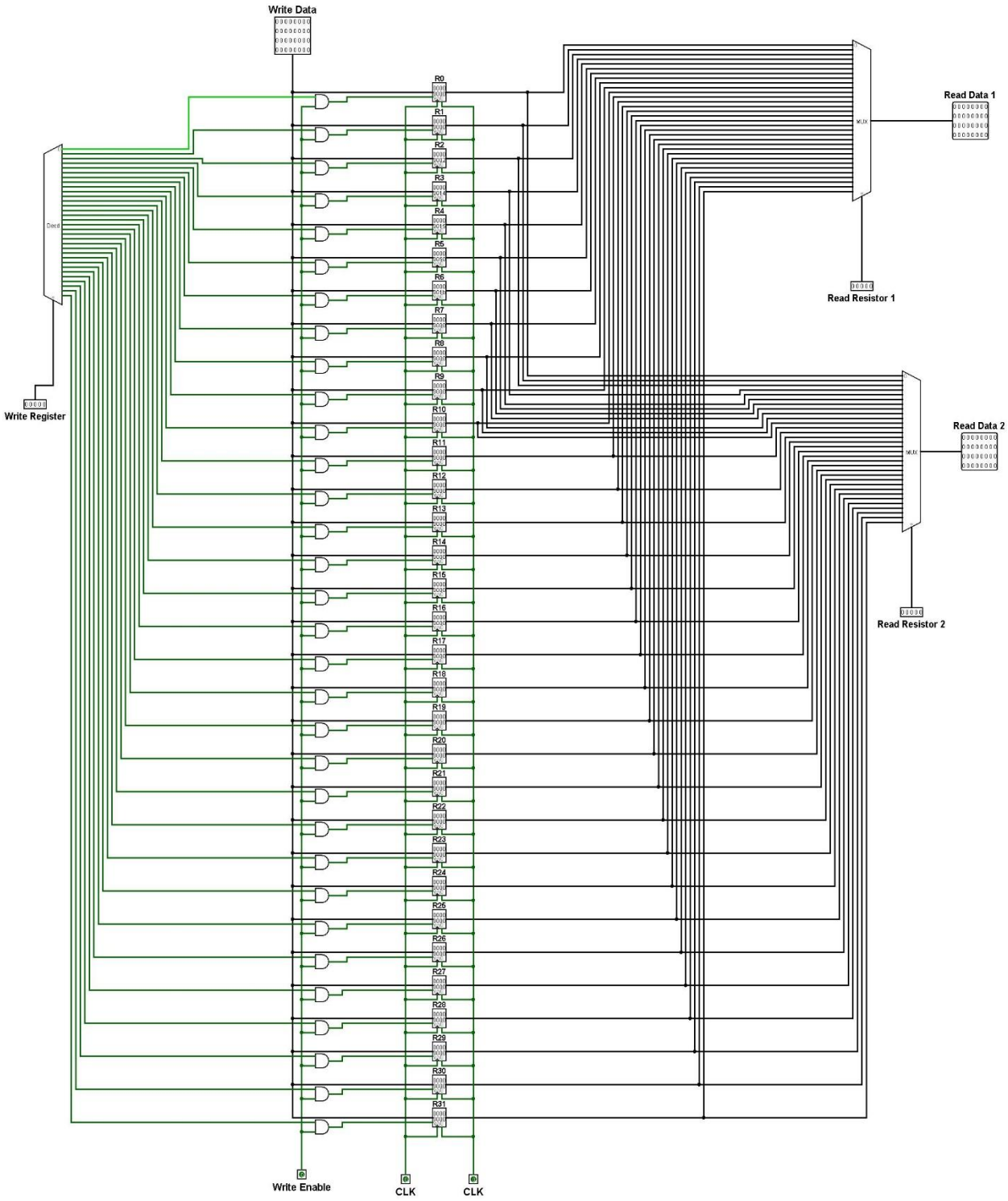

Figure-2: 32 bit Register File.

Program Counter: Here we use a program counter of 16 bit and it is used give the instruction towards ROM.

ALU: Basically ALU stands for the Arithmetic logic unit. It is a combinational circuit which is capable of doing all kinds of arithmetic and logical operation of a computer. It also performs the micro operations. Basically ALU performs all the arithmetic operations like addition, subtraction or the logical operations like AND and OR. Generally an ALU performs an operation and then transfer the result into a destination register.

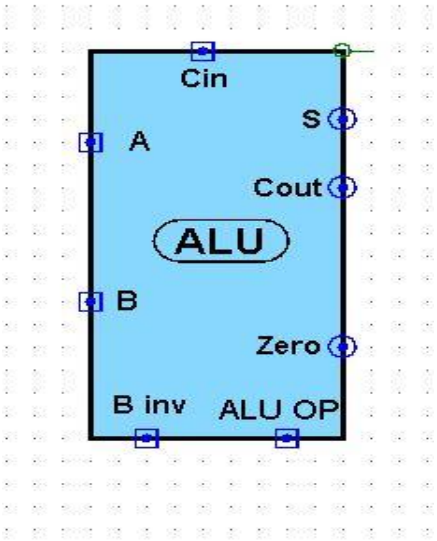

Figure-3: 32 bit ALU IC

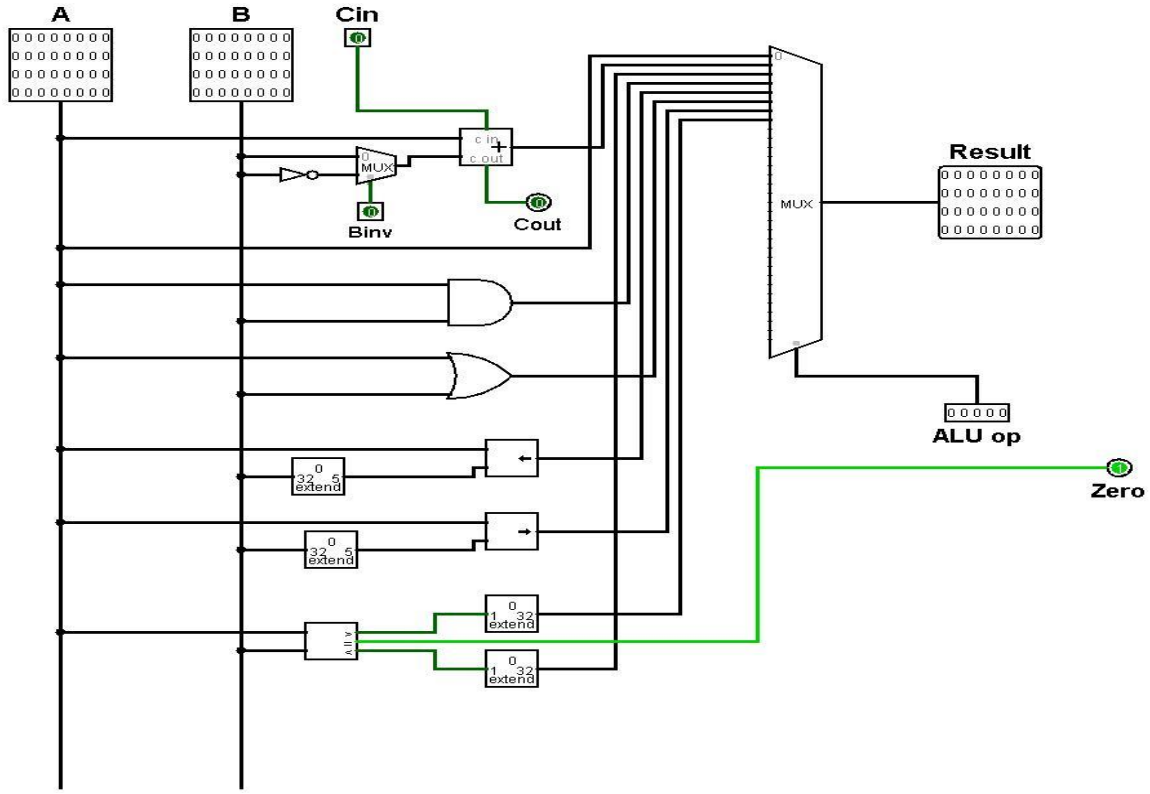

Figure-2: 32 bit ALU

Control Unit: It is a part of datapath and this circuit has been used for giving instruction and activate pins of other components. Here in this 19 bit mips architecture I use a decoder with select bit 4 and some or gates to build the control unit. Here in the control unit I set the opcode value of the instruction bit as input and in every output pins I set different instructions like Alu op, write resistor, B inv, Cin, Reg dest, ALU Src, Mem2Reg, Str(Write Memory), Ld(Read memory), MemSel, Branch, and Jump.

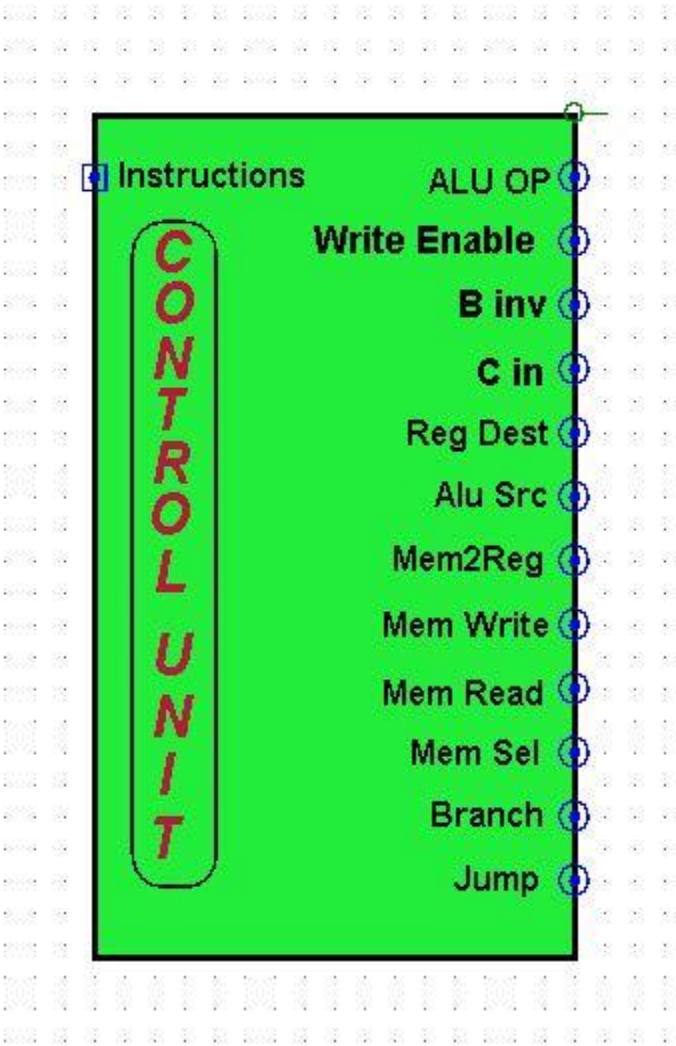

Figure: Control Unit IC

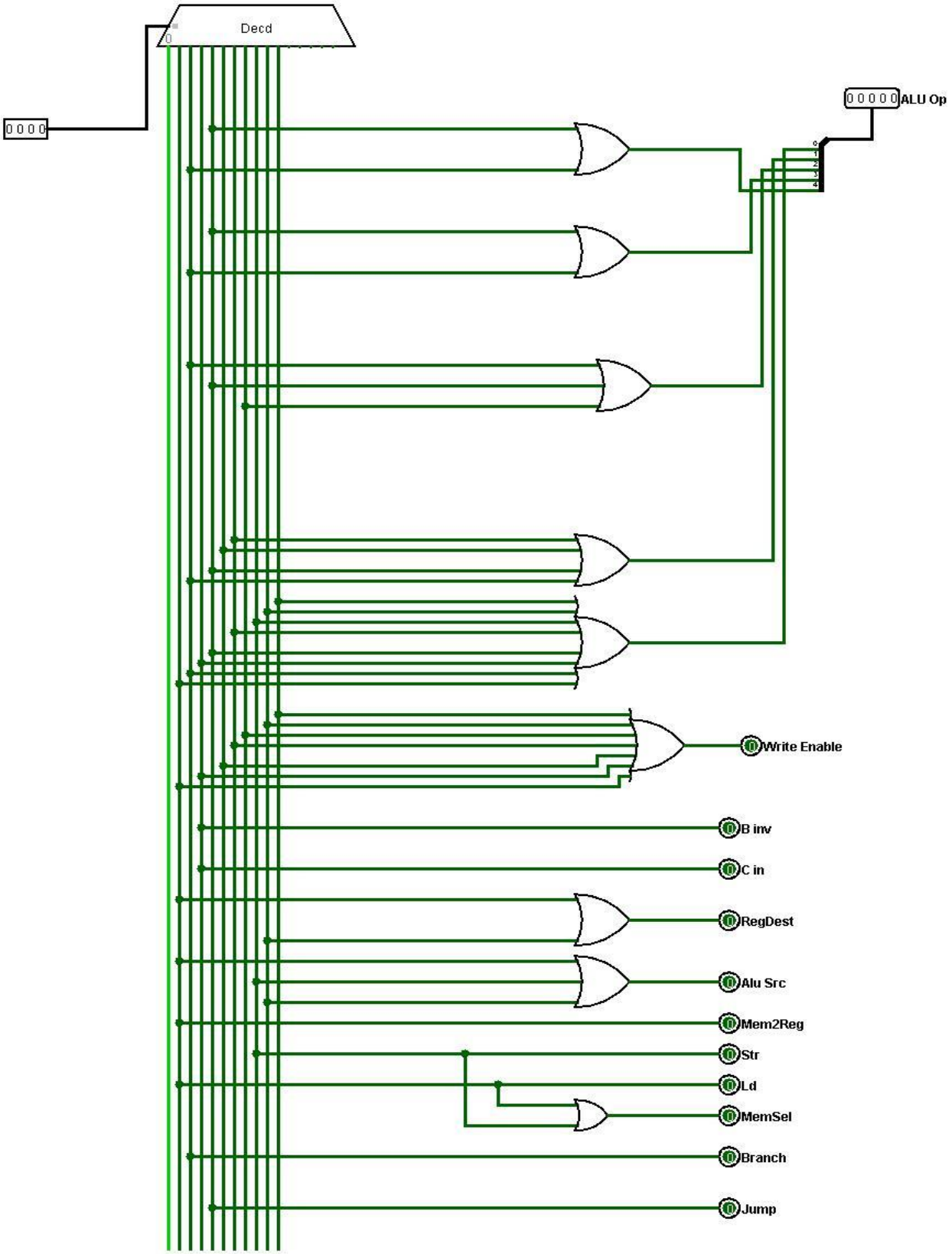

Figure-6: Control Unit

ALU Op: it's value determines what operation should perform the ALU.

Write Enable: It is used to enable the write enable pin of the register file.

B inv: it's value can be 0 or 1. It will be 1 when the ALU going to perform the subtraction operation and for the other operations the value of Binv is zero.

Cin: it is a pin of the adder. It's value can be 0 or 1. It will be 1 when the ALU going to perform the substraction operation and for the other operations the value of Cin is zero.

Reg dest: it is use to enable the pin of the multiplexer which hass used for the I type instructions. The pin will be enabled when the Lw and Addi operations ae executed.

Alu Src: it is used to enable the pin of the multiplexer before the Alu to choose a value between rt and the immediat type. This pin will be enabled when the Lw, Sw and Addi operations are executed.

Mem2Reg: This pin has been used to select value between the Alu output and the memory output from a multiplexer.

Str: This pin has been used to write a value into the memory.

Ld: This pin has been used to read a value from the memory .

Branch: This pin will activate when the giving two input values are equal.

Jump: This pin use to jump into a certain line .

ROM-RAM: The most important part of a computer system is the memory. Because, without a memory a computer cannot perform the a simple task. RAM and ROM are the primary memory of a computer system. Random Access Memory is a primary volatile memory and Read Only memory is a primary non volatile Memory. RAM is used for the read and write values. But the ROM has been used for the reading instructions only. ROM stores the operations which are going to use construct the system. We took the address bit 16 of the ROM and the data bits is 19. For the RAM we converted the input pin by an extender from 32 to 24 and the data bits of the RAM is 19 bits.

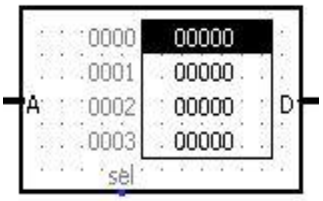

Figure-7: ROM

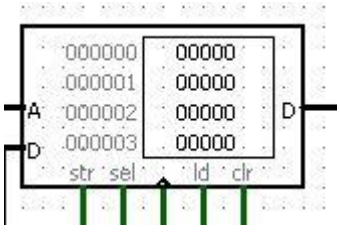

Figure-8: RAM

Datapath: The data path is an important part of a processor, since it implements the fetch-decode-execute cycle. A very simple data path components include memory that stores the current instructions program counter or PC that stores the address of current instruction and ALU that execute the current instruction. The interconnection of this components creates a data path.

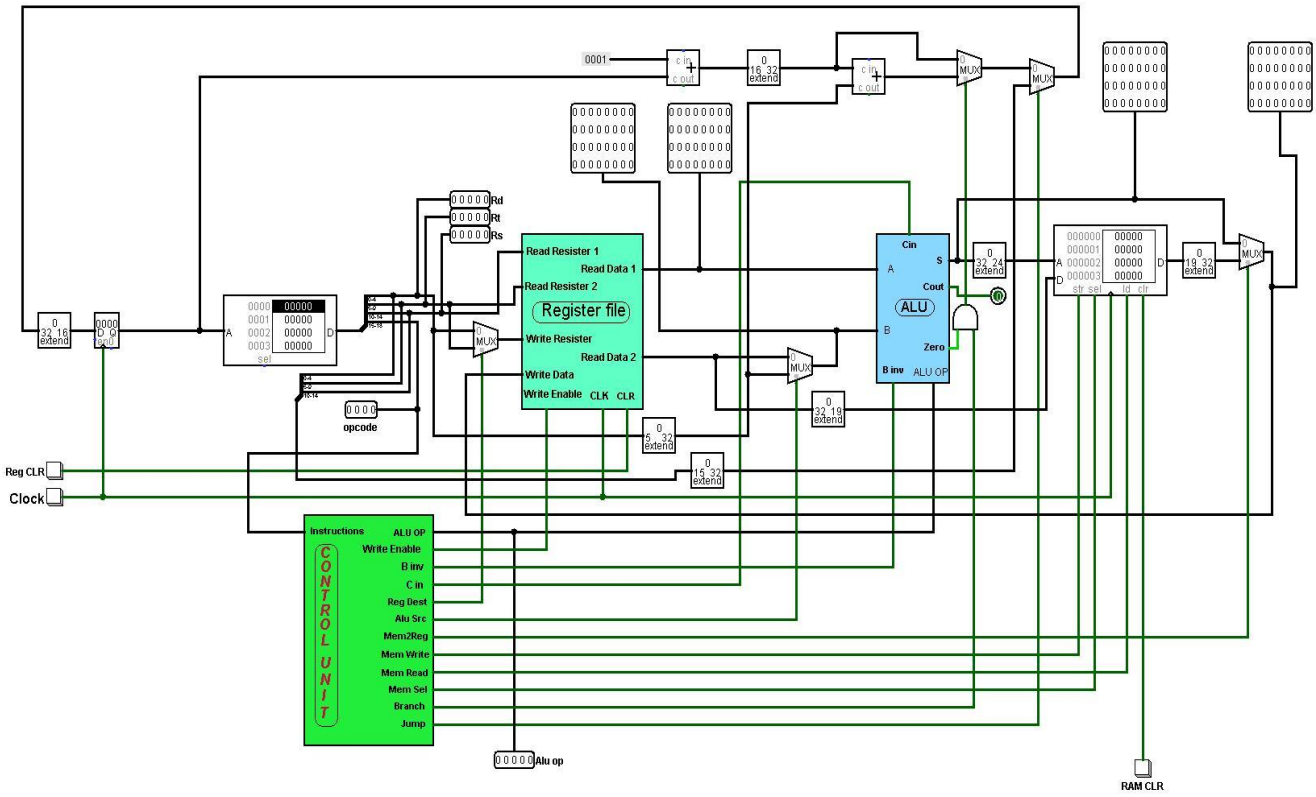

Figure-9: 19 bit MIPS Architecture Datapath

Formats:

Here in this architecture we had introduced three types of formates for our ISA. The types are:

- Register Type – R Type
- Immediate Type – I Type
- Jump Type – J Type.

R-Type ISA Format :

Opcode	Rs	Rt	Rd
4 bits	5 bits	5 bits	5 bits

I-Type ISA Format:

Opcode	**Rs**	**Rd**	**Immediate**
4 bits	5 bits	5 bits	5 bits

J-Type ISA Format:

Opcode	**Address**
4 bits	15 bits

List Of Register:

As we assigned 5 bits into the register so that the number of the register will be $2^5 = 32$. Here the input and output pins of the registers are not accessable by the user.

Number of Resistor	Resistor Names	Usage	Binary Value
0	R0	General Purpose	00000
1	R1	General Purpose	00001
2	R2	General Purpose	00010
3	R3	General Purpose	00011
4	R4	General Purpose	00100
5	R5	General Purpose	00101
6	R6	General Purpose	00110
7	R7	General Purpose	00111
8	R8	General Purpose	01000
9	R9	General Purpose	01001
10	R10	General Purpose	01010
11	R11	General Purpose	01011
12	R12	General Purpose	01100
13	R13	General Purpose	01101
14	R14	General Purpose	01110
15	R15	General Purpose	01111
16	R16	General Purpose	10000
17	R17	General Purpose	10001
18	R18	General Purpose	10010
19	R19	General Purpose	10011
20	R20	General Purpose	10100

21	R21	General Purpose	10101
22	R22	General Purpose	10110
23	R23	General Purpose	10111
24	R24	General Purpose	11000
25	R25	General Purpose	11001
26	R26	General Purpose	11010
27	R27	General Purpose	11011
28	R28	General Purpose	11100
29	R29	General Purpose	11101
30	R30	General Purpose	11110
31	R31	General Purpose	11111

Types Of Operations:

This architecture is able to do only 5 types of operations. The operation types are:

- Arithmetic
- Logical
- Data Transfer
- Conditional Branch
- Unconditional Jump

Category	Operations	Name	Type	Opcode	Syntax	Comments
No operation	Nothing just transfer the value of A	nop	R	0000	nop R0 R3 R7	Will pass the value of Rs register.
Data Transfer	Load Word	lw	I	0001	lw R0 R3 7	R3=Mem[R0+7]
Conditional	Check Equality	beq	I	0010	beq R1 R2 5	If (R1==R2) then 5
Arithmetic	Subtraction	sub	R	0011	sub R3 R4 R5	R5 = R3 – R4
Unconditional	Jump	jmp	J	0100	J 5	Jump to line 5
Conditional	Compare less than	slt	R	0101	slt R3 R4 R5	If(R3<R4) then R5=1 else R5=0
Logical	Bit-By-Bit And	and	R	0110	and R3 R4 R5	R5 = R3 & R4
Logical	Shift left	sll	R	0111	sll R3 R4 R5	R5 = R3<<R2
Data Transfer	Store Word	sw	I	1000	sw R0 R3 7	Mem[R0+7]=R3

Arithmetic	Add number with an immediate value	addi	I	1001	addi R0 R6 7	R6 = R0+7
Arithmetic	Add two numbers	add	R	1010	add R3 R4 R5	R5 = R3 + R4

Control Unit Table:

Opcode	Operations	ALU OP						B inv	C in	Write register	RegDest	ALU Src	Mem2Reg	Str	Ld	Branch	Jump
0000	nop	00000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0001	lw	00001	0	0	0	0	1	0	0	1	1	1	1	0	1	0	0
0010	beq	xxxxx	1	1	1	1	1	0	0	0	0	0	0	0	0	1	0
0011	sub	00001	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
0100	jmp	xxxxx	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1
0101	slt	00010	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0110	and	00011	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0
0111	sll	00100	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0
1000	sw	00001	0	0	0	0	1	0	0	0	0	1	0	1	0	0	0
1001	addi	00001	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0
1010	add	00001	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0

Assembler:

To convert the assembly instruction into the binary and Hexa value we had built an assembler. We will give instruction in assembly language into the input file of the assembler.Firstly, it will convert the instruction into binary value . Then, it cut the binary value by 4 bits to convert it into the hexadecimal value. We will found the hexadecimal value into the output file.

Limitations:

Here we didn't allocate any bits for the shift amount and function in the R-Type formate. We use the 5 bits of immediate for the shifting purpose.

Discussion:

In this whole project we have constructed a 19 bits MIPS architechture data path. For building this datapath circuit we used a ROM for giving the instruction, Register file, Alu, Program Counter, Control Unit, ROM and RAM. Firstly, we connect the 16 bit program counter to the ROM with address bit 16 and data bits 19. Than the ouput value of the ROM has divided into four parts

opcode,rs,rt,rd. we connect the rs and rt line into the pin of read resistor 1 and read resistor 2. The line of rd will be connected into the write register line directly for the R-type instruction and we use a mux before the write resistor line for the I-type Instructions. The inputs of the multiplexers will come from the Rd and Rt. In I type format we consider the last 5 bits as immediate value. By this 5 bits we also did our shifting works. Than the output lines of the register file will be connected into the input lines of the ALU . The ALU of this data path is 32 bits . For the R-type instruction the output lines of ALU will be connected directly to the write data line of the register file. But for the I type instruction we used a RAM for write and read the value in to memory. After giving an instruction the final result of the instruction will go through multiplexer. The multiplexer will give value according to the select input value of the mux and it is known as Mem2reg. This pins value will be 1 only one time when we we are going to load any value from the memory. Here in this datapath we also constructed the branch and Jump instruction. The branch will work when the giving to inputs are equal and go to the another line. As we have to do 11 operations by this data path so that we cannot keep our opcode bit less than 4. Among these 11 instructions in this datapath we constructed the SLL instruction as a R-type instruction. So that whenever we are going to shift left any value we need to first lw the values in to a resistor than we will do the operations. We also constructed an Assembler to converting our instructions into binary and hexadecimal values. After giving all the connections properly we have checked our datapath and it is working properly.