KATHMANDU UNIVERSITY

Dhulikhel, Kavre

A Mini Project Report

on

"Simulation of JAPBK Computer"

COMP 315

(For the partial fulfillment of 3rd year / 1st semester in Computer Engineering)

Submitted to:

Mr. Pankaj R. Dawadi

Department of Computer Science and Engineering (DoCSE)

Submitted by:

Jason Maharjan(26)

Anup Raj Niroula(30)

Prabhat Pandey (31)

Bijay Shrestha (47)

Karun Shrestha (49)

Date: 2019/12/27

Table of Contents

1. INTRODUCTION 3
2. DESIGN 5
2.1 Instructions 5
2.1.1 Memory Reference Instructions 5
2.1.2 Register Reference Instructions 8
2.1.3 Input Output Instructions 10
2.2 Common Bus System 12
2.3 Components Used 14
2.3.1 Registers 14
2.3.3 Control Signals 23
3. FlipFlops 31
4. Number of Components Used 32
5. Explanation of instruction 33
5.1. Fetch 33
5.2. Decode 34
5.3. Execution 35
5.3.l. Memory reference instructions 35
5.3.2. Register reference instructions 38
5.3.3. Input/output 39
DESIGN OF INDIVIDUAL COMPONENT 41
6. CONCLUSION 47
BIBLIOGRAPHY 48

1. INTRODUCTION

A 32K X 20 computer has a very simple architecture compared to today's complex computers. But understanding the way it functions helps understand many complex concepts which are the pillars to the development of the modern computers.

The simulation for our computer was done in software called "Logisim". The program is a really simple and does have its limitations, but for the study of a CPU, it is enough. Our designed computer has a very small instruction set. It supports basic arithmetic and logic operations such as addition, subtraction and, or, etc. Accumulator is used to store the output of any ALU operation. Data register is used to store data from memory before performing any operations on them. Address register is used to specify memory address with the help of common address bus. Data is transferred between registers using common address bus. Program counter stores the next instruction to be executed and the computer interfaces with the user using input and output registers. All the instructions are executed by using the signals generated by the control unit consisting of a sequence counter and a large collection of logic gates. Our computer has a 4 bit opcode, 15 bit address length and 1 bit Memory Reference type Flag (I).

Following section describes the design process and the internal architecture of our computer as well as instruction set supported by this computer. We have also provided an explanation on how each instruction is executed.

Our computer consists of the following hardware components:

1. A memory of 32K*20.
2. Six registers: AR, PC, DR, AC, IR and TR.
3. Three flip-flops: S, ENS, E.
4. Two Decoders: a 4*16 operation decoder and a 4*16 timing decoder are reused in different components.
5. Priority encoder: five 8*3 encoders and two 2*4 encoder.

6. A 20-bit common bus.

7. Control logic gates.

8. Adder, Subtractor and logic circuit connected to the input of AC.

2. DESIGN

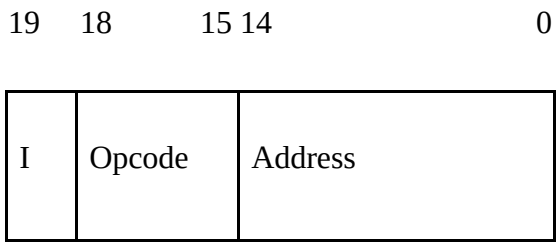
2.1 Instructions

Instruction is a binary code that specifies a sequence of microoperations for the computer. The binary instruction code is a group of bits that instruct the computer to perform a specific operations. It is divided into 2 parts, Addressing mode, Operation code and Address. The number of bits required for the operation code of an instruction depends on the total number of operations available in the computer.

We had designed 32K * 20 bit computer, 32K indicates the memory unit with 32,768 words.

$32K = 2^5 K = 2^5 . 2^{10} = 2^{15}$

So, each instruction code is 20 bit and we need 15 bits to specify an address, 1 bit for the addressing modes, 4-bit for an operation code.

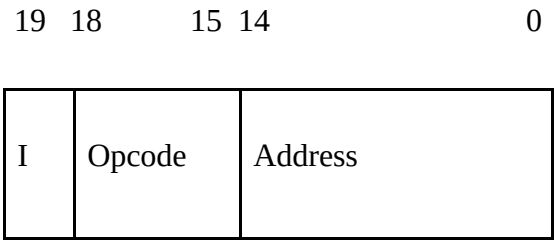


2.1.1 Memory Reference Instructions

A memory reference instruction is defined by the operation code 0000 through 1110 with

15-bit to specify an address and 1-bit for addressing mode I. An opcode is the first byte of an instruction in machine language which tells the hardware what operation needs to be performed with this instruction. Every processor/controller has its own set of opcodes defined in its architecture.

The instruction code format for the MRI is:



Here, I = 0 direct addressing and I = 1 indirect addressing.

In our computer design, the memory reference instructions are:

Symbol	**Opcode**		**Description**
	I = 0 **Indirect**	**I = 1** **Direct**	
AND	0 0000X	1 0000X	AND the AC with memory word
OR	0 0001X	1 0001X	OR the AC with memory word
XOR	0 0010X	1 0010X	XOR the AC with the memory word
NAND	0 0011X	1 0011X	NAND the AC with the memory word
NOR	0 0100X	1 0100X	NOR the AC with the memory word

ADD	0 0101X	1 0101X	ADD the memory word with AC and store in AC
ADC	0 0110X	1 0110X	ADD the memory word with AC, store in AC and carry in E
SUB	0 0111X	1 0111X	Subtract the memory word with AC and store in AC
LDA	0 1001X	1 1001X	Load accumulator with memory word
STA	0 1010X	1 1010X	Store value of AC to memory word
BUN	0 1011X	1 1011X	Branch Unconditionally
BSA	0 1100X	1 1100X	Branch and save return address

Table 2.1.1: Memory Reference Instructions.

Note: Here, ’X’ is a 15-bit memory address.

2.1.2 Register Reference Instructions

In a register reference instruction, the operation code contains 0000 through 1110 with

15-bit to specify an address and the value 0 in addressing mode I. Register reference instruction indicates the registers instead of memory address.

The instruction code format for the RRI is:

19 18 15 14 0

0	1111	Address

In our computer design, the register reference instructions are:

Symbols	**Instructions**	**Descriptions**
CLA	0 1111 100000000000000	Clear AC
CLE	0 1111 010000000000000	Clear E
CMA	0 1111 001000000000000	Complement AC
CME	0 1111 000100000000000	Complement E
CIR	0 1111 000010000000000	Circulate right AC and E

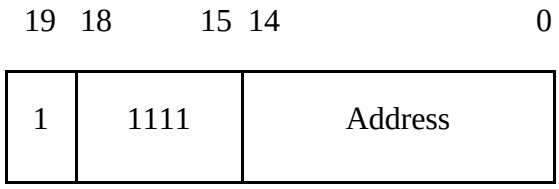
CIL	0 1111 000001000000000	Circulate left AC and E
INC	0 1111 000000100000000	Increment AC
DEC	0 1111 000000010000000	Decrement AC
HLT	0 1111 000000000000100	Halt computer
SHR	0 1111 000000000000010	Shift the AC by one bit right
SHL	0 1111 000000000000001	Shift the AC by one bit left

Table 2: Register Reference Instructions

2.1.3 Input Output Instructions

An input output instruction is defined by the operation code 1111 with 1 value in addressing mode I. Input Output Instructions indicate the transfer of data between peripheral devices and main memory.

The instruction code format for the IOI is:



In our computer design, the memory reference instructions are:

Symbols	**Instructions**	**Descriptions**
INP	1 1111 100000000000000	Input character to AC
OUT	1 1111 010000000000000	Output character from AC
SKI	1 1111 001000000000000	Skip on input flag
SKO	1 1111 000100000000000	Skip on output flag
ION	1 1111 000010000000000	Interrupt on
IOF	1 1111 000001000000000	Interrupt off

Table 3: Input Output Instructions

2.2 Common Bus System

The computer here designed has 8 registers, memory unit and control unit. Path must be provided for the communication between two registers in the computer. Use of wire from one register to other registers may be excessive. So, the more efficient system is used for transferring the date between many registers called common bus system. Common bus system is an efficient scheme for transferring the information in system which has many registers. If common bus is not in use, then every register has to be connected to individual bus which may create a complex and impossible situation. So, it must require a scheme in which bus is connected to the memory as well as the registers.

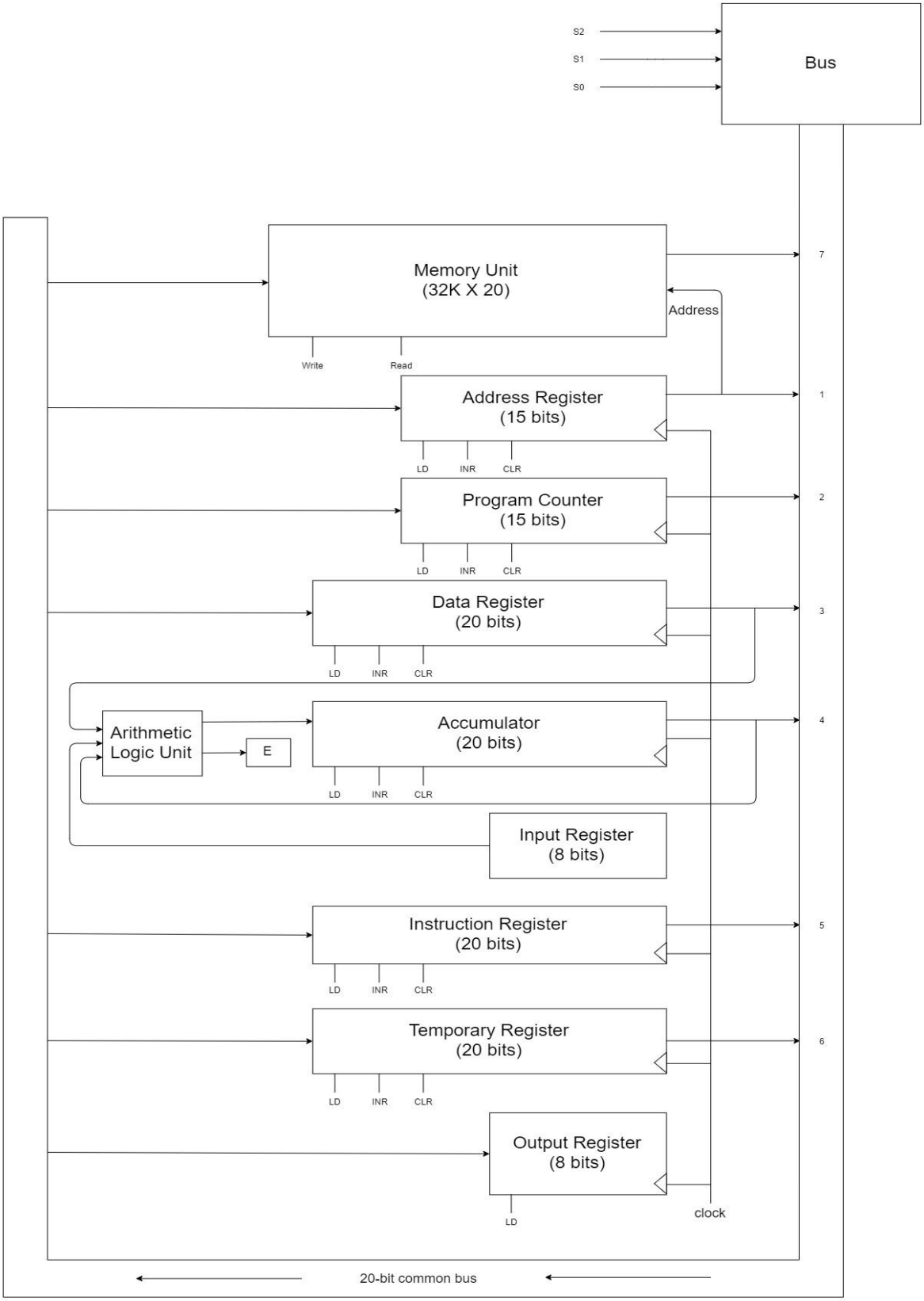


Figure 1 : Block Diagram

2.3 Components Used

2.3.1 Registers

The various components given below comes together to form a simple working computer system. The various components are selected by the Bus as per the output from the encoder which can be seen from the table below.

X1	X2	X3	X4	X5	X6	X7	S2	S1	S0	Selected Registers
0	0	0	0	0	0	0	0	0	0	None
1	0	0	0	0	0	0	0	0	1	Address Register
0	1	0	0	0	0	0	0	1	0	Program Counter
0	0	1	0	0	0	0	0	1	1	Data Register
0	0	0	1	0	0	0	1	0	0	Accumulator
0	0	0	0	1	0	0	1	0	1	Instruction Register
0	0	0	0	0	1	0	1	1	0	Temporary Register
0	0	0	0	0	0	1	1	1	1	Memory

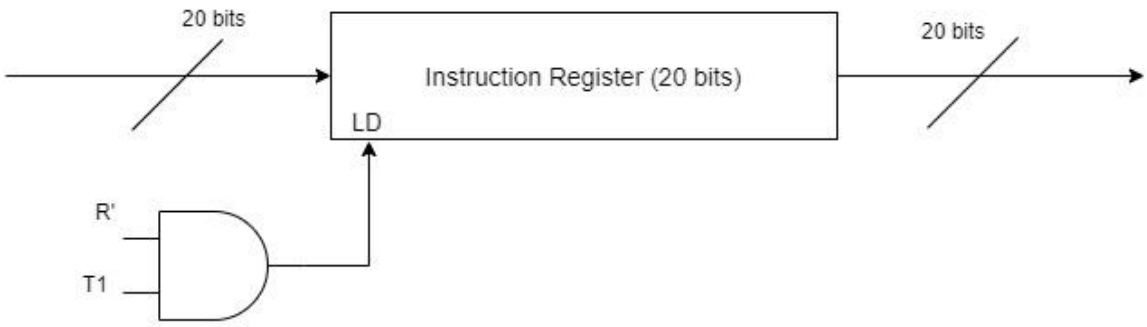
The registers are selected in the common bus system with the help of mux and a priority encoder.

2.3.1.1 Instruction Register

Size: 20 bits

Instruction Register stores the 20-bit instruction read from the memory. At the start of each instruction cycle, the address contained in program counter is transferred to AR, and the content of the memory location pointer by AR is then transferred to the Instruction register through the common bus system (CBS). Once the instruction is loaded in the IR it is then decoded to obtain the decoded lines (D0-D15) and B0-B10 which along with the timing signals is used to provide the necessary control signals for the computer to execute the stated operation. The composition of the component of the Instruction Register is given as:

Load = $R.T_0$



2.3.1.2 Program Counter

Size: 15 bits

Program counter contains the address of the memory location where the next instruction is located. At the beginning of each instruction cycle (fetch), the content of the program counter is transferred to AR and then is incremented by one to point to the next consecutive location.

In the case of subroutines, the content of the program counter (PC) is saved in the memory and is loaded with the location of memory containing the subroutine procedure. Finally at

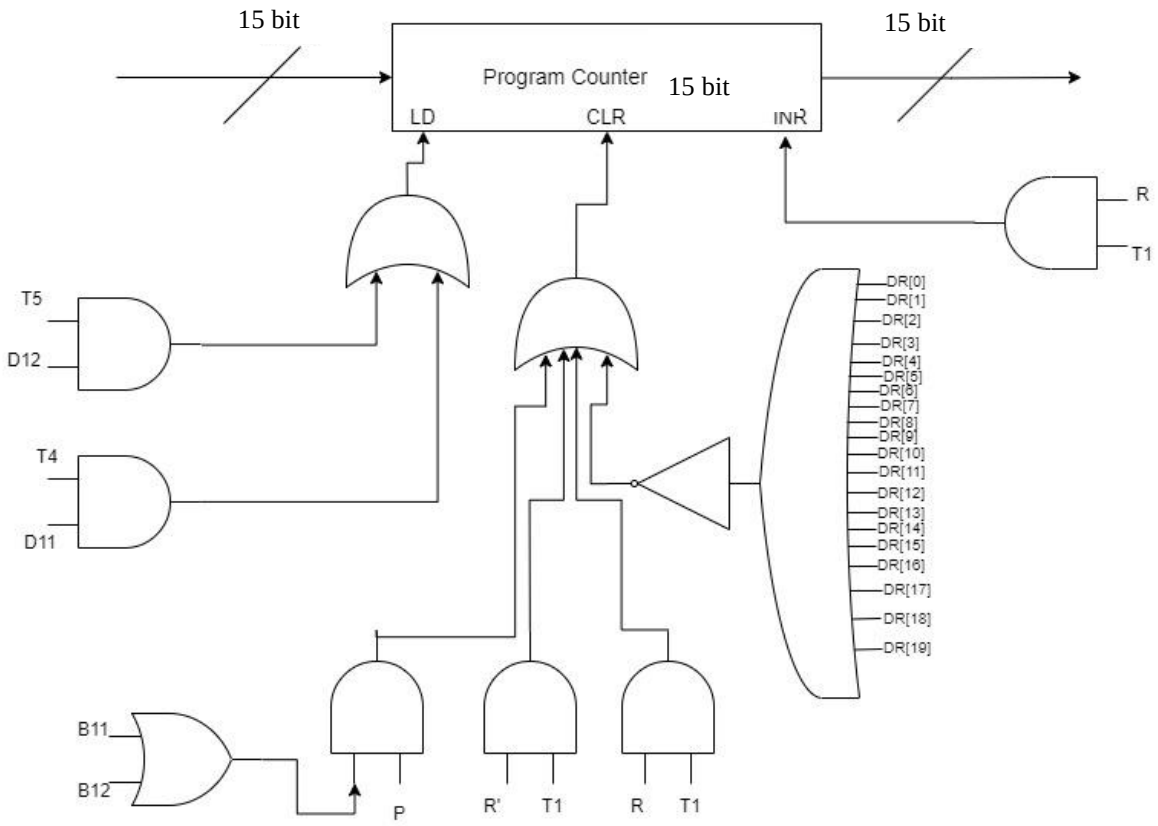
the end of subroutine call PC is loaded with the previously saved address and the execution cycle continues.

The composition of the components of the program counter is given as :

Load = $D_3.T_4$

Increment = $R'.T_1 + R.T_2$

Clear = $R.T_1$



2.3.1.3. Data Register

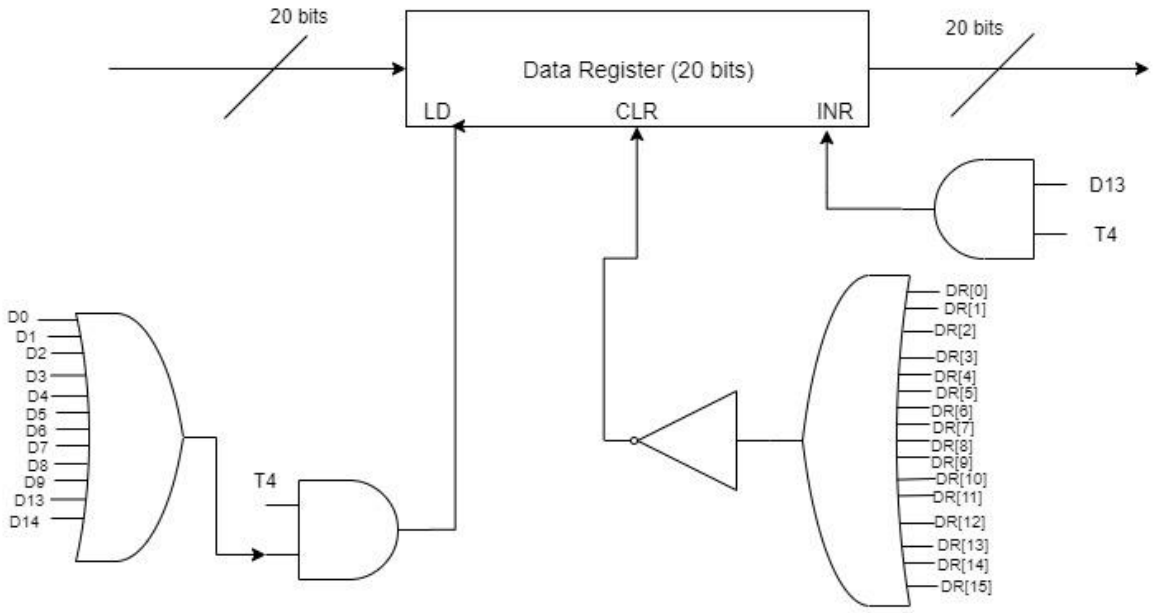
Size: 20 bits

Data register is used to store the operand read from the memory that is to be processed.

Output of data register is an input to the ALU.

The composition of the components of the Data Register is given as :

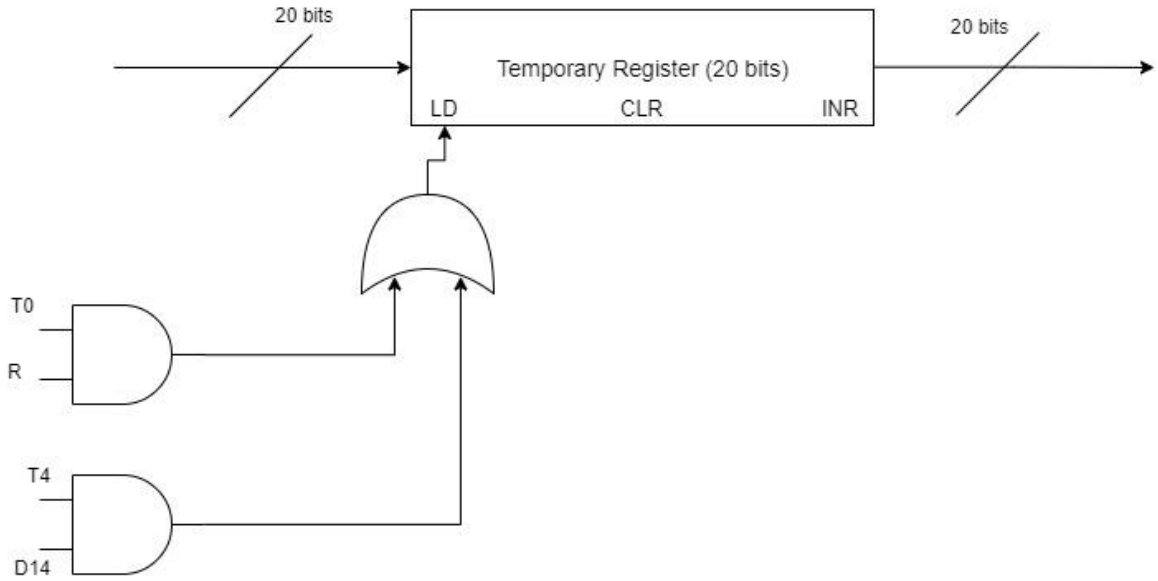
Load = $D_0.T_4 + D_1.T_4 + D_2.T_4 + D_6.T_4$



2.3.1.4. Temporary Register

Temporary Registers are used to store the temporary values during the execution. Its composition is given by :

Load = R'.T_1



2.3.1.5. Accumulator

Size: 20 bits

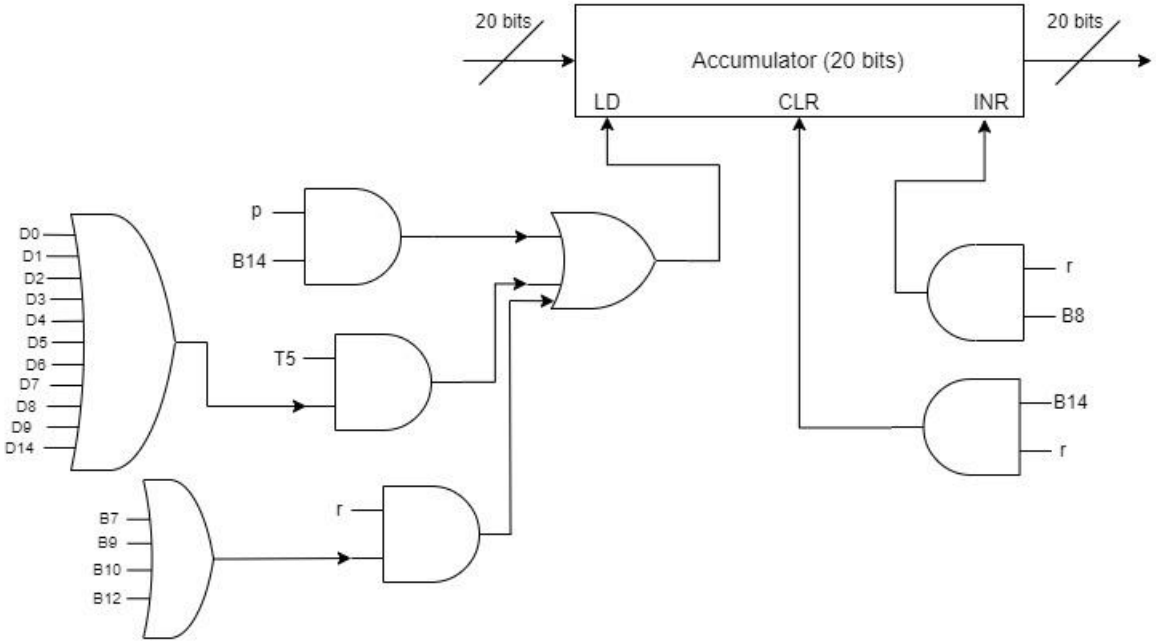
Accumulator is the main register for most of the operations. It stores the result of any operation after its completion. Input of the accumulator comes from the output of the ALU. The input of accumulator is not directly interfaced with the CBS. Output of the AC goes to the common bus system as well as to ALU as the first operand, second being the Data register.

The composition of the components of the program counter is given as :

Load = $D_0.T_5 + D_1.T_5 + D_2.T_5 + D_5.T_5 + D_6.T_5 + p.B_{11} + r.B_3 + r.B_2$

Increment = $r.B_5$

Clear = $r.B_1$



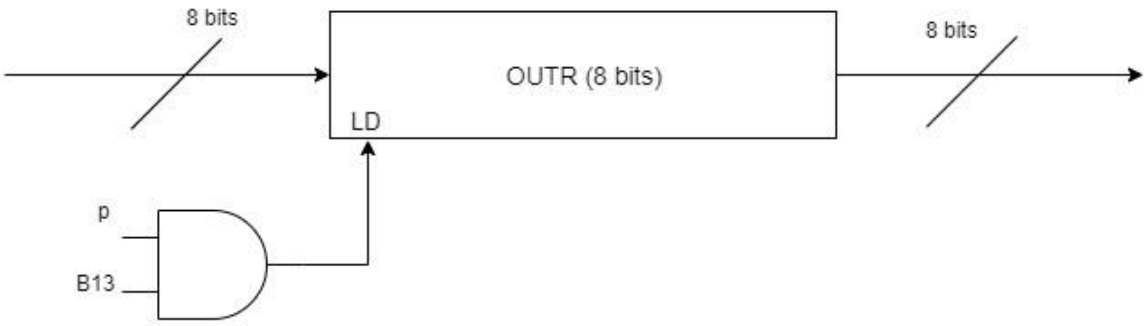
2.3.1.6. Input and Output Registers

Size: 8 bits

There are two registers each of 8 bit for both input and output.

Input register (INPR) is connected to the ALU and the content of the input register is transferred to the ALU on selection of the transfer operation.

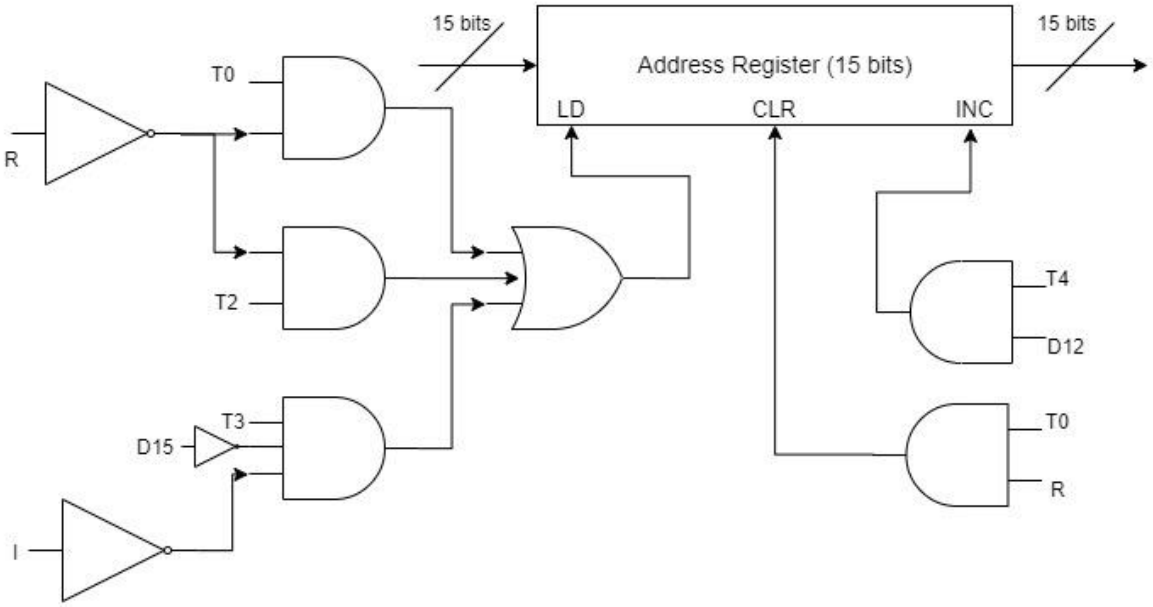
Output register (OUTR) is used by the computer to provide output to the external world. It receives its output from the common bus system which usually comes from the accumulator. This output may then be connected to an external display.



2.3.1.7. Address Register

Memory Address Register (MAR) is the CPU register that either stores the memory address from which data will be fetched from the CPU or the address to which data will be sent and stored. Its component composition is given as: Load = $R'.T_0 + R'.T_2 + D_7'.I.T_3$

Clear = $R.T_0$



2.3.2 Memory

2.3.2.1 Memory Unit

*Size: 32K*20*

Memory stores both the instruction and the data on which processing is to be performed.

We have selected a memory of 2048 words with a word size of 16-bits for our computer. This means that we require 11 bit address line to address all the word of the memory. Memory receives its address from the Address Register (AR), it receives/sends its data from the common bus system, read or write operations is distinguished from the control signal.

The composition for the components of the memory unit is :

Read = D0.T4 + D1.T4 + D2.T4 + D5.T4 + D5.T5 + D6.T4 + D6.T5 + R'.T1 + R'.T2 Write = D4.T4 + R.T1

2.3.2.2 Control Unit

Control unit (CU) generates the necessary control and timing signals to carry out the sequences of micro-operations in order to execute the given instruction.

2.3.2.3 Arithmetic and Logic Unit (ALU)

An arithmetic logic unit (ALU) is a digital circuit used to perform arithmetic and logic operations. It represents the fundamental building block of the central processing unit (CPU) of a computer. The ALU uses operands and code that tells it which operations to perform for input data. After the information has been processed by the ALU, it is sent to the computer's memory. Arithmetic and logic unit in our computer performs 10 main operations: AND, ADD, SUB, XOR, LDA, Complement, Circular Shift Right, Circular Shift Left, Shift Left, Shift Right.

2.3.3 Control Signals

Control signals for the instructions are specified below:

2.3.3.1. Fetch and Decode Cycle

Cycle	Control Signal	Register Transfer Language
Fetch	T0	AR← PC
	T1	IR← M[AR], PC← PC+1
Decode	T2	I←IR(19), D0, D1,….,D14←Decode [IR(15-18)], AR← IR(0-14)

Table 4: Fetch and Decode Cycle

2.3.3.2 Execution of MRI

Symbols	Operation Decoder	RTL	Descriptions
AND	D0	AC ← AC ∧ M[AR]	AND the AC with memory word
OR	D1	AC ← AC ∨ M[AR]	OR the AC with memory word
XOR	D2	AC ← AC ⊕ M[AR]	XOR the AC with the memory word
NAND	D3	AC ← (AC ∧ M[AR])'	NAND the AC with the memory word
NOR	D4	AC ← (AC ∨ M[AR])'	NOR the AC with the memory word
ADD	D5	AC ← AC+ M[AR]	ADD the memory word with AC and store in AC

SUB	D7	AC ← AC -M[AR]	Subtract the memory word with AC and store in AC
LDA	D9	AC ← M[AR]	Load accumulator with memory word
STA	D10	M[AR] ← AC	Store value of AC to memory word
BUN	D11	PC ← AR	Branch Unconditionally
BSA	D12	M[AR] ← PC, PC ← PC + 1	Branch and save return address

Table 6: Execution of MRI Signals

2.3.3.3 Execution of RRI

For RRI, r = D15I'T3

Bi = IR(i), i=0,1,2...,14 Indicate the particular RRI.

RRI	Control Signal	RTL	Description
	r:	SC ← 0	Clear SC
CLA	rB14:	AC ← 0	Clear AC
CLE	rB13:	E ← 0	Clear E
CMA	rB12:	AC ← 0	Complement AC
CME	rB11:	E ← 0	Complement E
CIR	rB10:	E ←AC[0] , AC ← SHR, AC[15] ← E	Circulate right AC and E
CIL	rB9:	E ←AC[15] , AC ← SHL, AC[0] ← E	Circulate left AC and E

INC	rB8:	AC ← AC+1	Increment AC
DEC	rB7:	AC ← AC-1	Decrement AC
HLT	rB2:	SC ← 0, S ← 0	Halt computer
SHR	rB1:	AC ← SHR AC	Shift the AC by one bit right
SHL	rB0:	AC ← SHL AC	Shift the AC by one bit left

Table 5: Execution of RRI signals

2.3.3.4 Input Output operations:

For IOI, p = D15IT3

B(i) = IR(i), i=7,8,...,14 Indicate the particular IOI instruction

RRI	**Control Signal**	**RTL**	**Description**
	p:	SC←0	Clear SC.
INP	pB14:	AC ←INPR,FGI←0	Input a character from AC
OUT	pB13:	OUTR←AC,FGO←0	Output a character from AC
SKI	pB12:	If(FGI==1) then PC←PC+1	Skip on input flag
SKO	pB11:	If(FGO==1) then PC←PC+1	Skip on output flag

ION	pB10:	IEN←1	Sets interrupt flag
IOF	pB9:	IEN←0	Resets interrupt flag

Table 7: III Signals

Sequence Counter

It is the part of the computer that determines whether the given instruction has completed its execution or not. The completion of the instruction is denoted by SC ←0. Its component composition is given as :

$$\text{Clear} = p + r + R.T_2 + D_0.T_5 + D_1.T_5 + D_2.T_5 + D_3.T_4 + D_4.T_4 + D_5.T_5 + D_6.T_5$$

3. FlipFlops

FlipFlop	Description	Instruction
IEN	Start Stop Flip Flop	$pB_7 + pB_6 + RT_2$
I	Interrupt	$R'T_2$
R	Memory Reference type Flag	$T_0'.T_1'.D_2'.IEN.(FGI + FGO)$
E	Carry Flag	$r.B_3 + r.B_4 + r.B_2 + D_2.T_5$
FGI	Input Flag	$p.B_{11}$
FGO	Output Flag	$p.B_{10}$

Table 8: FlipFlops

4. Number of Components Used

S.N.	Component	Number
1	Memory(32K x 20)	1
2	Registers	12
3	FlipFlop	6
4	Full Adder	1
6	3 bit binary counter	1
7	Decoders(4 x 16), (3 x 8) and (2x1)	8
8	Encoder (3*8)	4
9	MUX (2x1)and (4x1)	6

5. Explanation of instruction

5.1. Fetch

During the fetch cycle the instruction is read from the memory and loaded into the instruction register. For this, the address of the next location containing the instruction in the memory is contained in program counter. This address is transferred to Address Register during the first T0 timing signal.

On the next timing signal the content of the memory pointed by the address register is

copied to the instruction register in the meantime, Program counter as well as address

register is incremented by one.

Following RTL denotes the fetch cycle.

When R →0, we use R' in the instruction

$R'T_0$: AR ← PC

$R'T_1$: IR ← M [AR], PC ← PC + 1

When Interrupt occurs R →1, so we use R in the instruction

RT_0: AR ← 0, TR ← PC

RT_1: M [AR] ← TR, PC ← 0

5.2. Decode

In decoding cycle, the instruction stored in the instruction register is decoded to determine the type of instruction. Necessary decoding lines are generated to specify the operation. Following is the RTL for the decoding cycle:

When R →0

$R'.T_2 : D_0, \ldots, D_{14} \leftarrow$ Decode IR(15-18), AR ← IR(0-14), I ← IR[19]

When R →1

$R.T_2$:PC←PC+1,IEN←0,R←0,SC←0

5.3. Execution

5.3.l. Memory reference instructions

5.3.1.1. AND

This instruction copies the operand specified by the address to the data register during the first

cycle of its execution. On the next cycle, this data is then ANDed with content of the accumulator and the result is stored in the accumulator.

The ANDing operation is performed inside the ALU. RTL for this instruction is:

D_0T_4: DR←M [AR]

D_0T_5: AC←AC ^ DR, SC ←0

5.3.1.2. OR

This instruction copies the operand specified by the address to the data register during the first cycle of its execution. On the next cycle, this data is then ORedwith content of the accumulator and the result is stored in the accumulator.

The ORing operation is performed inside the ALU. RTL for this instruction is:

D_1T_4: DR←M [AR]

D_1T_5: AC←AC v DR, SC ←0

5.3.1.3. XOR

This instruction copies the operand specified by the address to the data register during the first cycle of its execution. On the next cycle, this data is then XORed with the contents of the accumulator and the result is stored in the accumulator. The XOR operation is performed inside the ALU. RTL for this instruction is:

D_2T_4: DR←M[AR]

D_2T_5: AC←AC $\oplus$ DR, SC ←0

5.3.1.4. NAND

This instruction copies the operand specified by the address to the data register during the first cycle of its execution. On the next cycle, this data is then NANDed with the contents of the accumulator and the result is stored in the accumulator. The NAND operation is performed inside the ALU. RTL for this instruction is:

D_3T_4: DR←M[AR]

D_3T_5: AC← (AC ∧ DR)', SC ←0

5.3.1.5. NOR

This instruction copies the operand specified by the address to the data register during the first cycle of its execution. On the next cycle, this data is then NORed with the contents of the accumulator and the result is stored in the accumulator. The NORoperation is performed inside the ALU. RTL for this instruction is:

D_4T_4: DR←M[AR]

D_4T_5: AC← (AC V DR)', SC ←0

5.3.1.6. ADD

This instruction is used to add the data from the memory with the data stored in the accumulator and store the resulting data in the accumulator. Its instruction is :

D_5T_4: DR←M [AR]

D_5T_5:AC← AC + DR, SC ← 0

5.3.1.7. SUB

This instruction is used to subtract the data from the memory with the data stored in the accumulator and store the resulting data in the accumulator. Its instruction is :

D_6T_4: DR←M [AR]

D_6T_5: AC←AC - DR, E ← Cout, SC ← 0

5.3.1.8. LDA

This instruction loads the content of the memory location specified by the address. It takes two cycles to execute this instruction, during the first cycle the content of the memory location pointed by the address register is copied to the DR. On the next cycle the content of data register is transferred to AC through the ALU. RTL for this instruction is given below

D_7T_4: DR ←M [AR]

D_7T_5: AC ←DR, SC ←0

5.3.1.9. STA

This instruction stores the content of AC into the memory word specified by the effective address. The RTL for this instruction is:

D_8T_4: M[AR] ←AC, SC ←0

5.3.1.10. BUN

This instruction causes the computer to branch unconditionally to the given address and continue the execution of instruction for the location.

It takes only one step to execute this instruction. To execute this instruction content of the address is simply transferred to the program counter and the sequence counter is reset.

D_9T_4: PC ←AR, SC ←0

5.3.1.11. BSA

This instruction causes the computer to save the return address and branch to the given address and continue the execution of instruction for the location.

$D_{10}T_4$: M[AR]←PC, AR ← AR + 1

$D_{10}T_5$: PC ←AR, SC ←0

5.3.2. Register reference instructions

$r = D_{15}I'T3$

5.3.2.1. CLA

This instruction clears the content of the accumulator. Clear signal of the AC is activated to clear its content.

rB_{14}: AC←0

5.3.2.2. CLE

This instruction clears the carry flagE.

rB_{13} : E←0

5.3.2.3. CMA

This instruction complements the content of the accumulator.

rB_{12}: AC←AC'

5.3.2.4. CME

This instruction complements the content of the flag E.

rB_{12}: E←E'

5.3.2.5. CIR

This is the circular right shift instruction and causes the content of the accumulator to be circularly shifted right without loss of any bit. This operation is carried out in ALU.

rB_{11}: AC←shr AC, AC(15)←E, E←AC(0)

5.3.2.6. CIL

This is the circular left shift instruction and causes the content of the accumulator to be circularly shifted left without loss of any bit. This operation is carried out in ALU.

rB_{10}: AC←shl AC, AC(0)←E, E←AC(15)

5.3.2.7. INC

It increments the content of AC by one. This is carried out by applying the INR signal to the AC.

rB_9: AC←AC+1

5.3.2.8. DEC

It decrements the content of AC by one..

rB_8: AC←AC-1

5.3.2.9. HLT

Stop the execution of the program by resetting the start-stop flipflop. rB_0: S ←0

5.3.3. Input/output

operations $D_{15}IT3=p$

5.3.3.1. INP

This is an input instruction. It causes the computer to accept a $_4$-bit data from the input register and clears the input flag. Clearing the input flag FGI indicates that the data has been accepted so that inputting device can provide with next 4-bits of data.

$pB_{14:}$ AC ←INPR, FGI←0

5.3.3.2. OUT

This is an output instruction. It causes the computer to send $_4$-bit of data stored in the AC to be transferred to the output register and clear the output flag. Clearing the output flag indicated the external reading device that the content is ready in the output register to be copied. Once the date is copied by the

external device from the output register, the output flag is set allowing the computer to provide next $_4$-bits of data.

pB_{13}: OUTR←AC,FGO←0

5.3.3.3. SKI

Skip on input flag causes the computer to skip the next instruction if the input flag is set.

pB_{12}: If(FGI==1) then PC←PC+1

5.3.3.4. SKO

Skip on output flag causes the computer to skip the next instruction if the output flag is set.

pB_{11}: If(FGO==1) then PC←PC+1

5.3.3.5. ION

Sets the interrupt flip flop pB_{10}: IEN ← 1

5.3.3.6. IOF

Resets the interrupt flip flop pB_9: IEN ← 0

DESIGN OF INDIVIDUAL COMPONENT

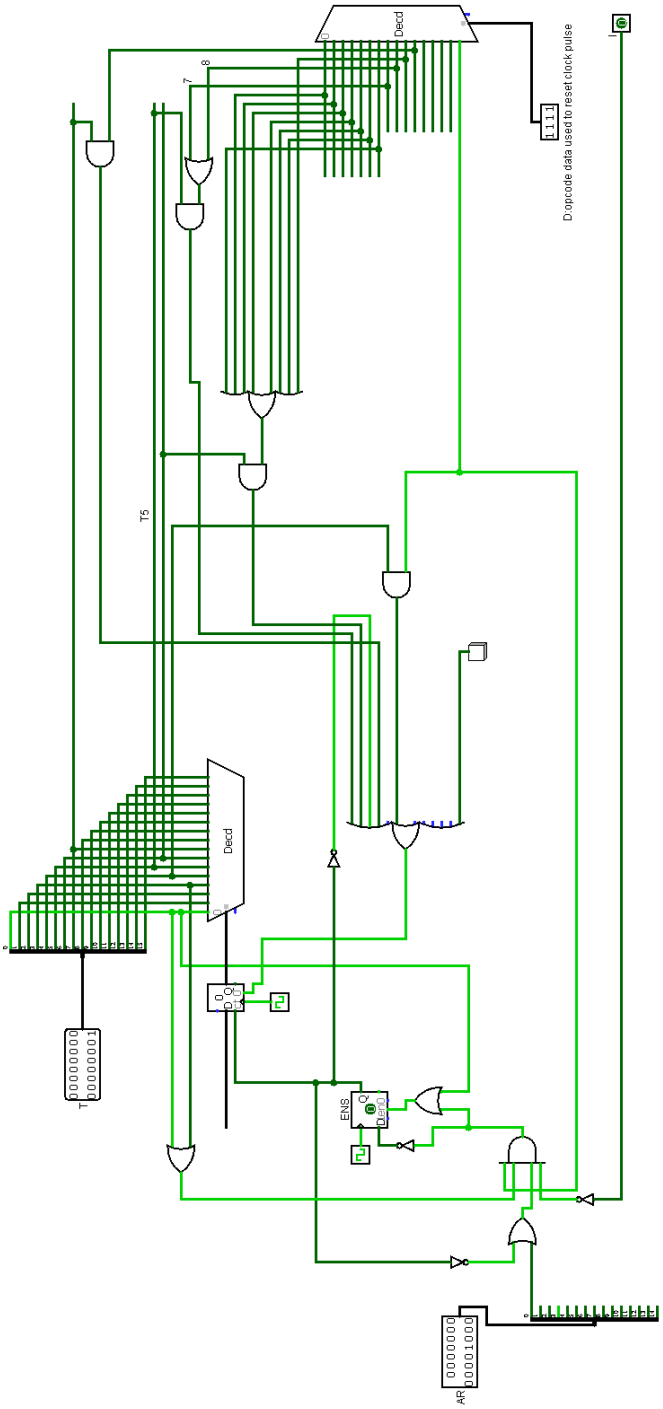


Figure 1 Sequence Counter (SC)

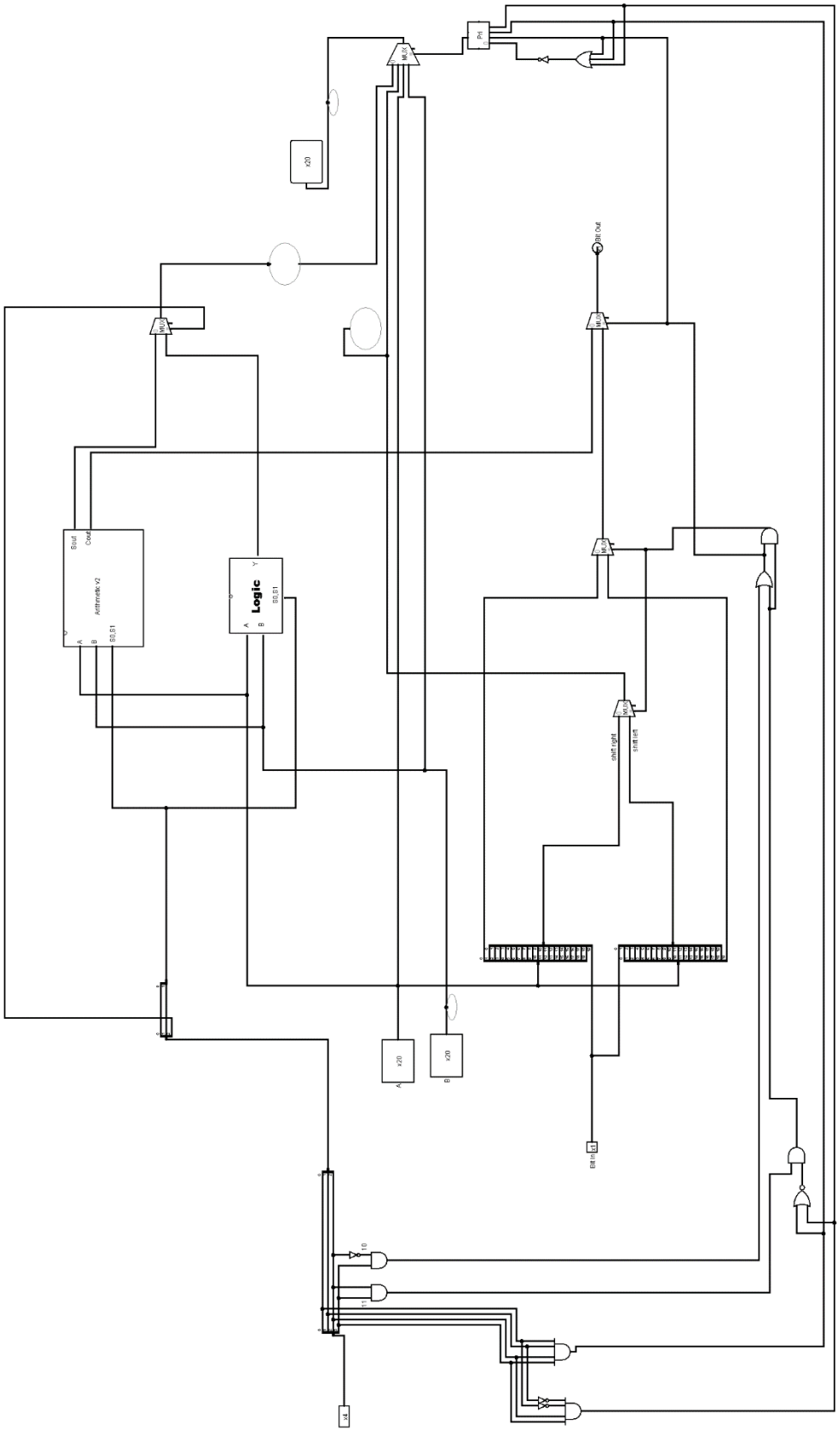


Figure 2 Arithmetic and Logic Unit (ALU)

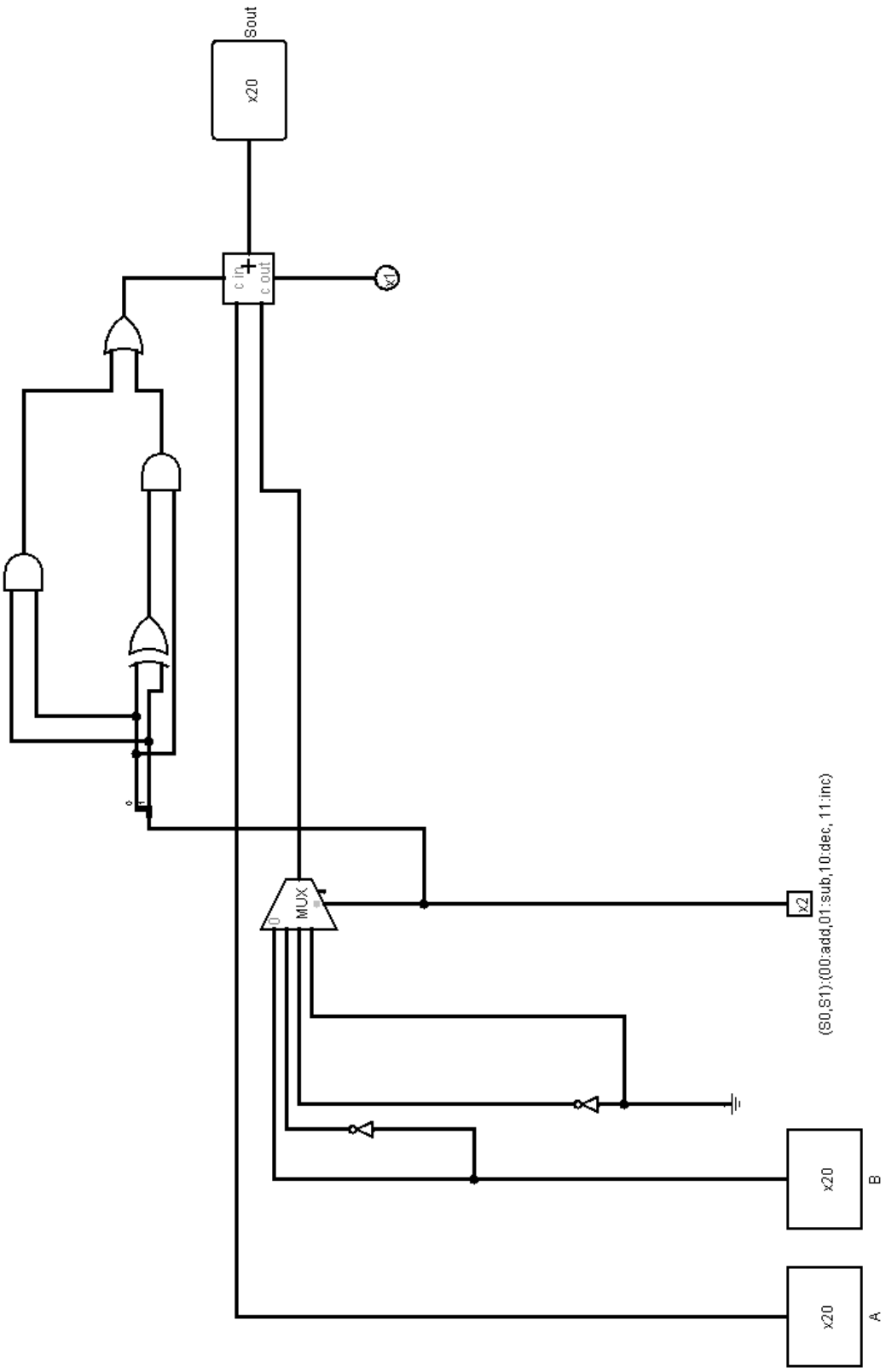


Figure 3 Arithmetic Circuit

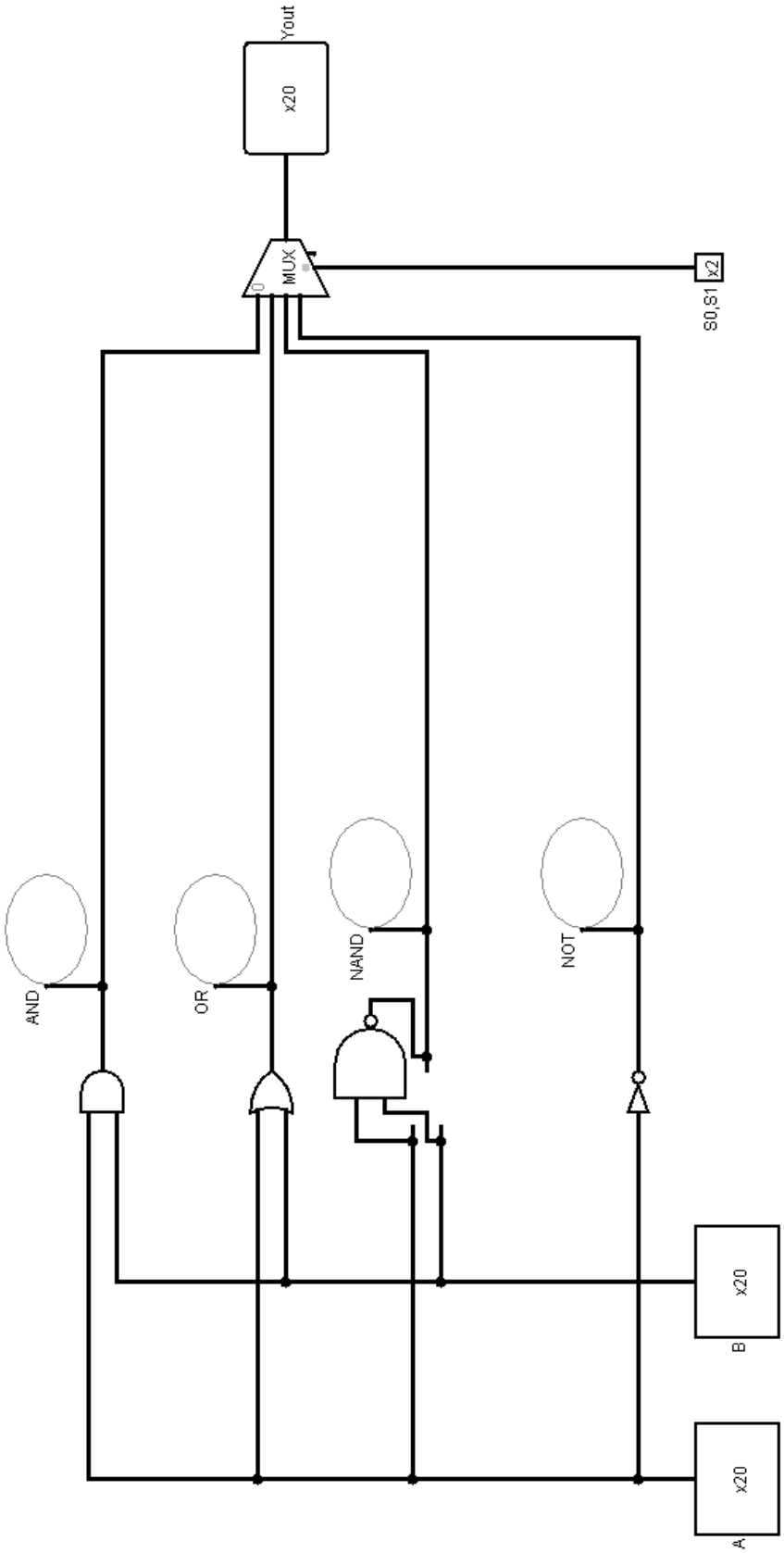


Figure 4 Logic Circuit

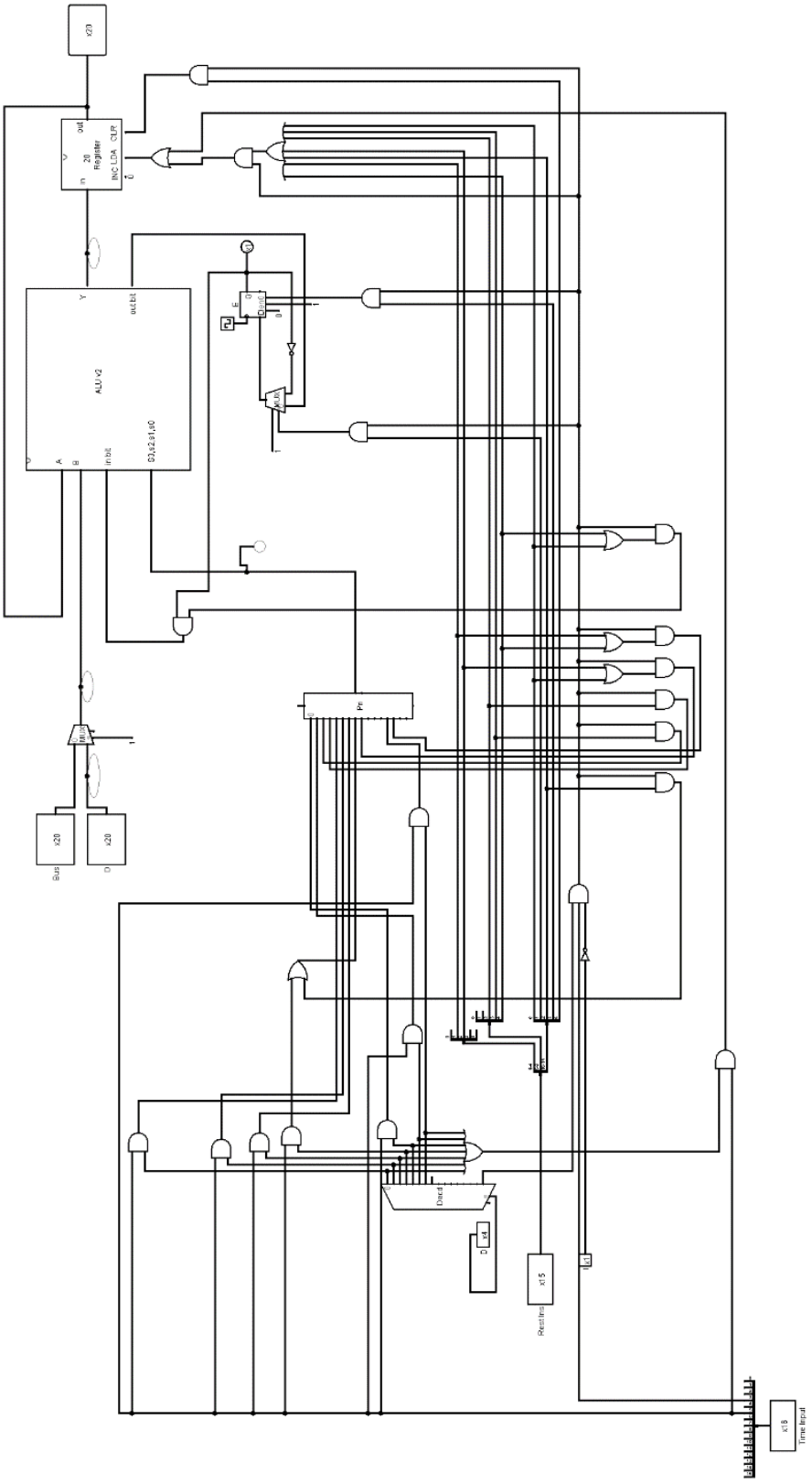


Figure 5 Accumulator (AC)

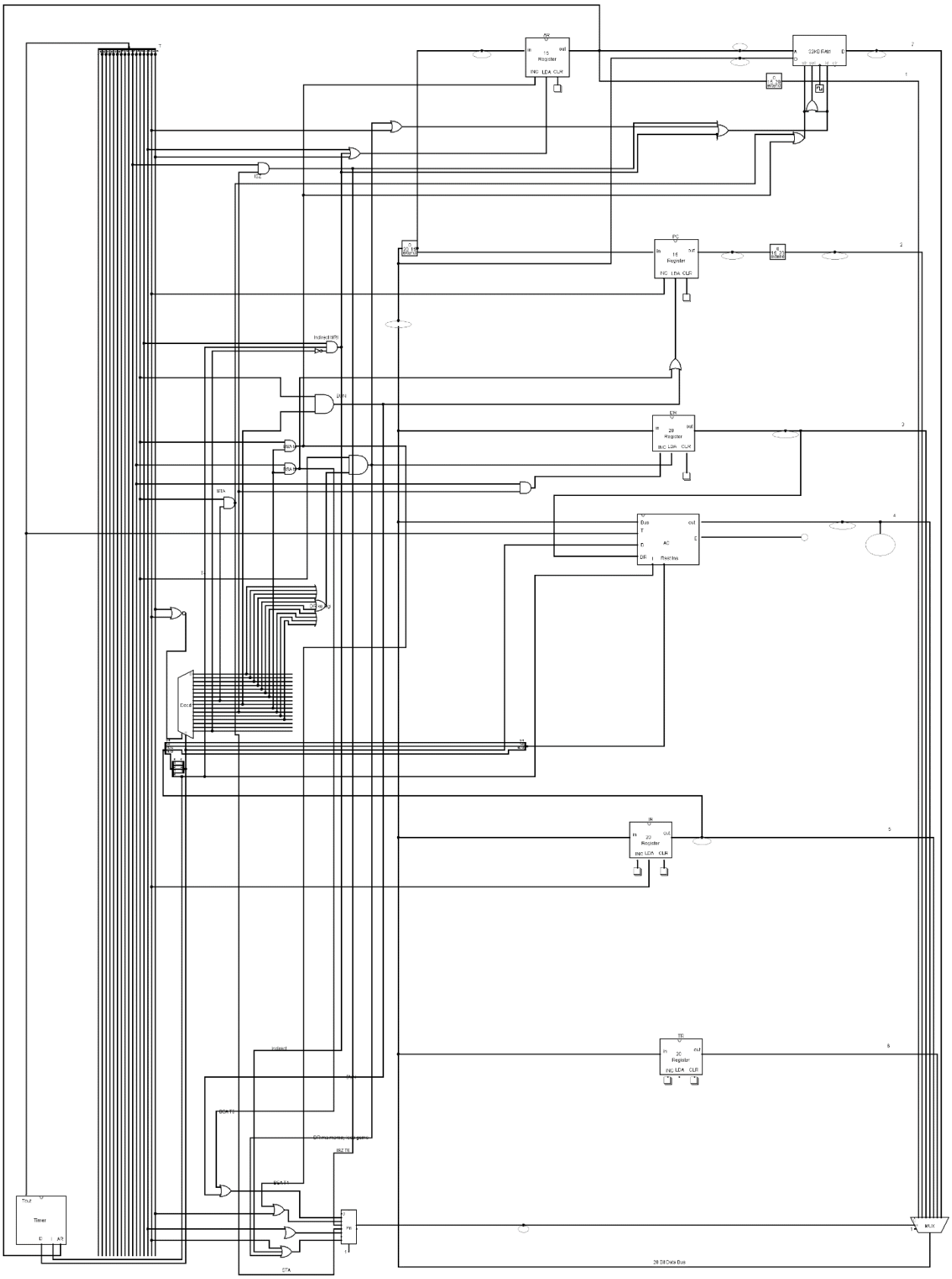


Figure 6 CPU MAIN BUS Circuit

6. CONCLUSION

The objective of designing this computer was not to build a computer that can match the modern computer in any aspect of computing. But the goal was to simulate a simple model to describe how computers function. The 32K X 20 computer explains how an ALU works, what CU is, how registers work within the CPU and how each assembly level instruction is handled by the CPU. In this way, our CPU is being simulated in Logisim.

BIBLIOGRAPHY

1. Mano, M. Morris. Computer system architecture. 3rd ed. Englewood Cliffs, N.J.: Prentice Hall, 1993. Print.

2. Hennessy, John L., and David A. Patterson. Computer architecture: a quantitative approach. 5th ed. Waltham, MA: Morgan Kaufmann, 2012. Print.