Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Mini Project Report

on

LRN Computer Design

[Code No: COMP 315]

(For partial fulfilment of 3rd Year/1st Semester in Computer Engineering)

Submitted By:

Nirmal Dahal (11)

Ram Koirala (29)

Lokesh Sapkota (47)

Submitted To:

Pankaj Raj Dawadi

Department of Computer Science and Engineering

Submission Date:2022/11/09

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Mr. Pankaj Raj Dawadi, our instructor for encouraging, guiding, supervising, and supporting us to construct our computer design using knowledge of a "Basic Computer." We would like to express our gratitude to everyone who gave us the chance to finish this project. We gained a better understanding of the fundamentals of a computer's memory, registers, and flip-flops thanks to this project.

We would like to express our gratitude to the entire university as well as the Department of Computer Science and Engineering (DOCSE) for allowing us to work on the project.

ABSTRACT

LRN Computer is a simple basic computer of size 16K*20 containing 14-bit address, 4-bit opcode, and 2-bit addressing mode. The design of the LRN Computer can handle a variety of instructions based on the Basic Computer instruction set and some additional instructions. A common bus was used to connect memory, registers, and flip-flops in the computer's design. Logisim was used to simulate the computer's fundamental operation which demonstrates how LRN computer-designed instructions are carried out. For Memory Reference Instructions (MRI), it has immediate addressing mode alongside Direct and Indirect addressing modes to control the location specified in the instructions.

ABBREVIATIONS

• CPU	Central Processing Unit
• RAM	Random Access Memory
• AC	Accumulator
• PC	Program Counter
• TR	Temporary Register
• AR	Address Register
• DR	Data Register
• IR	Instruction Register
• OUTR	Output Register
• INPR	Input Register
• SC	Sequence Counter
• MRI	Memory Reference Instructions
• RRI	Register Reference Instructions
• IOI	Input Output Instructions
• SEQ	Skip if Equal
• LDA	Load Accumulator
• STA	Store Accumulator
• BUN	Branch Unconditionally
• BSA	Branch and Save Return Address
• XCHG	Exchange
• ISZ	Increment and skip if zero
• DSZ	Decrement and skip if zero
• CLA	Clear Accumulator
• CMA	Complement Accumulator
• CLE	Clear E
• CME	Complement E
• CIR	Circular shift right
• CIL	Circular shift Left
• INC	Increment
• DEC	Decrement

- SPA Skip if Positive in Accumulator
- SNA Skip if Negative in Accumulator
- SZA Skip if Zero in Accumulator
- SZE Skip if Zero in E
- HLT Halt
- INP Input AC
- OUT Output AC
- SIE Skip if Input flag is enabled
- SID Skip if Input flag is Disabled
- SOE Skip if Output flag is Enable
- SOD Skip if Output flag is Disabled
- ION Interrupt enable ON
- IOF Interrupt enable OFF

TABLE OF CONTENTS

Contents **Page No**

LIST OF FIGURES

LIST OF TABLES

Chapter 1: INTRODUCTION

A computer is an electronic device that receives user input, processes it according to a set of instructions (called a program), outputs the results (called output), and saves the results for later use. It can handle both numerical and non-numerical (logic and arithmetic) calculations.

The CPU is the computer's central processing unit, which can also be referred to as a processor, central processor, or microprocessor. The central processing unit (CPU) of a computer handles all hardware and software instructions.

A computer's internal registers, timing and control structures, and instruction set define its organizational structure. To bring all of the concepts of computer system architecture together, the process of designing a CPU is both difficult and necessary.

1.1 Stored Program Concept

The storage of instructions in computer memory to enable it to perform a variety of tasks sequentially or intermittently is referred to as the Stored Program Control Concept.

John von Neumann came up with the idea where he suggested that a program be electronically stored in a memory device in the binary-number format so that the computer could modify the instructions based on intermediate computational results.

Both the data that the instructions use and the instructions themselves are stored in the computer's memory under the stored program concept. Instructions and data were regarded as completely distinct entities and were consequently stored separately before this concept's introduction.

Thus, the processor can read from and write to the memory data and instructions.

The processor then addresses the memory, reads the appropriate instructions, executes them, and processes (reads and writes) data in accordance with the executed instruction.

The Von Neumann architecture is said to be the foundation of computers that store both data and instructions in the same memory. The stored program concept is still the foundation of modern desktop computers.

1.2 Von Neumann Architecture

The von Neumann architecture is a design model for stored-program digital computers that makes use of a processing unit and a single, distinct storage structure to store both data and instructions. Five components make up a computer designed using this architecture: a system bus that serves as a data path between these components, an arithmetic-logic unit, a control unit, a memory, and some kind of input/output.

The primary memory M of a computer is responsible for storing programs and data as they are being processed by the CPU. M is RAM. Using load or store instructions, RAM lets the CPU read or change its contents. M is supported by hard disk secondary memory. One works on programs and data in RAM, a fast-access, volatile storage medium, while they are stored on a slow-to-access storage medium like a hard disk.

Chapter 2: DESIGN CONSIDERATIONS

The "LRN Computer" is a computer design that can handle a basic computer's instruction set and some additional instructions. It is a processor with a 20-word length and 16K of memory. It has an address line made of 14 bits and a common bus with 20 bits for data transfer. It has 4 registers with 20 bits, two 14-bit address-related registers (AR and PC), the 8-bit input and output registers, and a 4-bit Sequence Counter, two decoders are included: one 4x16 opcode decoder for the instructions and one 4x16 decoder for the timing signals.

For Memory Reference Instructions (MRI), it has an immediate addressing mode alongside Direct and Indirect addressing modes to control the location specified in the instructions. It has 15 MRIs, 8 Input/Output Instructions (IOIs), and 13 Register Reference Instructions (RRIs).

Thus, the LRN computer consists of the following hardware components:

1. A memory unit with 16384 words (16K) where each word is of 20 bits.
2. Nine Registers: DR, AR, TR, IR, AC, PC, INPR. OUTR, and SC.
3. Nine Flip-Flops: I_1, I_2, S, E, R, lEN, FGI, and FGO.
4. Two Decoders: a 4*16 operation decoder and a 4*16 timing decoder.
5. A 20-bit common bus.
6. Common logic gates.
7. Adder and logic circuit connected to the input of AC.

2.1. Registers

LRN Computer has nine registers. All the registers of the computer are listed in the table below with their size and with a brief description of their Function.

Register Symbol	Number of bits	Register Name	Function
DR	20	Data Register	Holds Memory Operand
AR	14	Address Register	Holds Address of memory
TR	20	Temporary Register	Holds Temporary Data
IR	20	Instruction Register	Holds Instruction Code
AC	20	Accumulator	Processor Register
PC	14	Program Counter	Holds Address of Instruction
INPR	8	Input Register	Holds Input Character
OUTR	8	Output Register	Hold Output Character
SC	4	Sequence Counter	

Table 2.1 List of Registers in 'LRN computer'

2. 2. Flip-Flop

There are Nine Flip-Flops in our basic computer:

I_1, I_2, S, E, R, lEN, FGI, and FGO flip flop

2.3. Common Bus System

The 20-bit data bus is connected to the registers and memory of the LRN Computer. The common bus is constructed by using multiplexers and encoders. The control signals are fed to 8 * 3 priority encoders. The three-bit selector line is used with a multiplexer then a register or memory is selected.

X1	X2	X3	X4	X5	X6	X7	S0	S1	S2	Selected Register
0	0	0	0	0	0	0	0	0	0	None
1	0	0	0	0	0	0	0	0	1	AR
0	1	0	0	0	0	0	0	1	0	PC
0	0	1	0	0	0	0	0	1	1	DR
0	0	0	1	0	0	0	1	0	0	AC
0	0	0	0	1	0	0	1	0	1	IR
0	0	0	0	0	1	0	1	1	0	TR
0	0	0	0	0	0	1	1	1	1	Memory

Table 2.3 Encoder for bus selection circuit

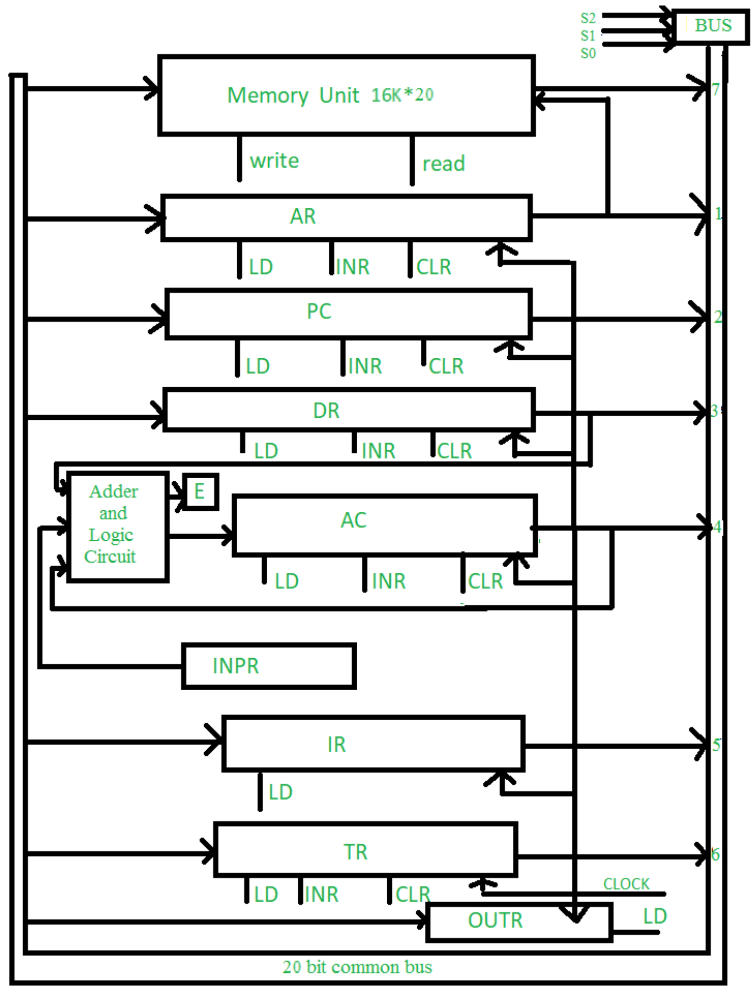

Fig 2.3 Common Bus for the LRN Computer

2.4. Control Unit

The Control Unit is a part of the Central Processing Unit(CPU) that translates from machine instructions to the control signals for the microoperations that implement them. The Control Unit is passed with the opcode and timing signals to create the control signals so that the CPU can work accordingly. The Control Unit of the LRN computer is shown below:

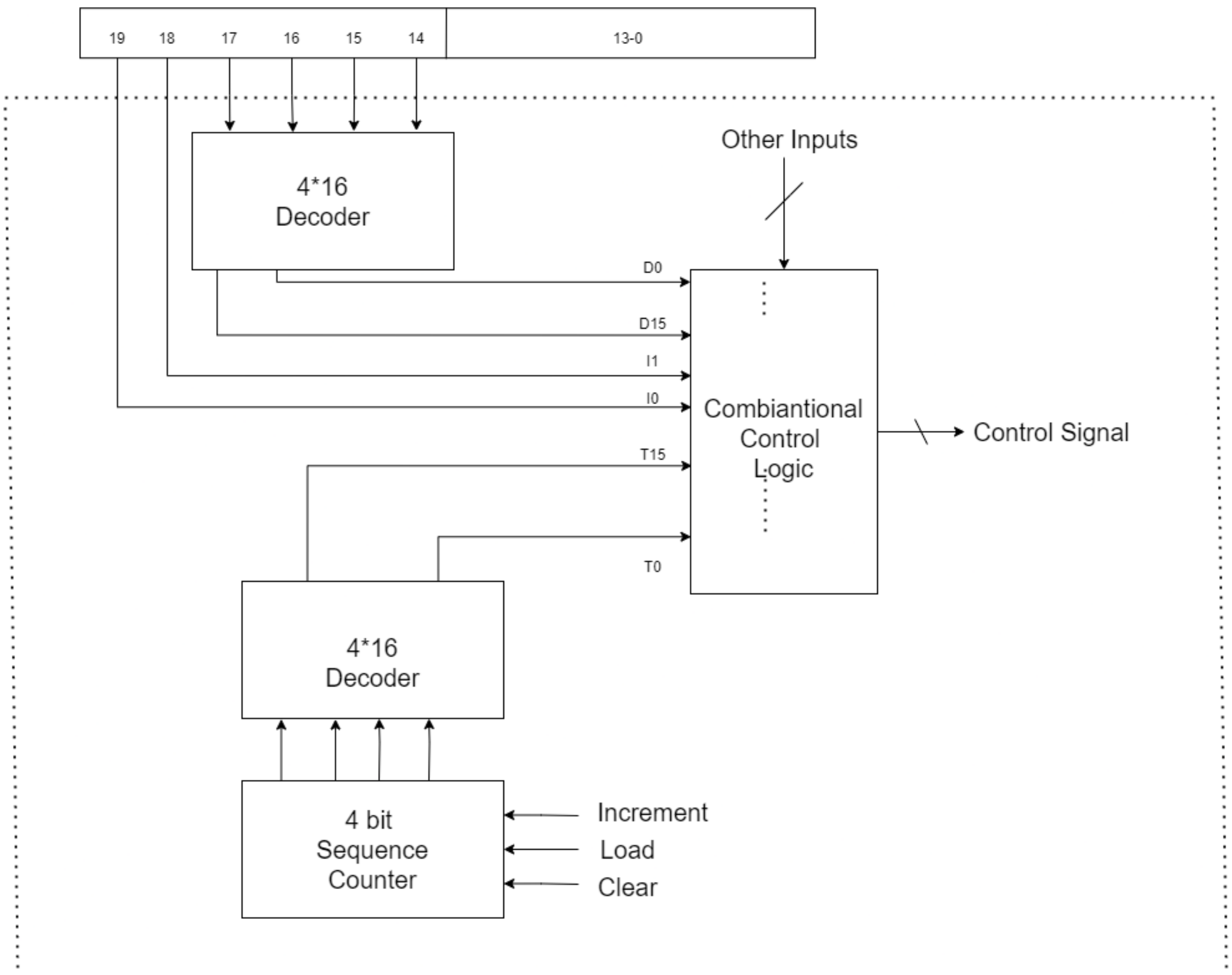

Fig: Control Unit of LRN Computer

2.5. Decoder and Encoder

Two decoders(4*16 timing decoder and 4*16 operation decoders) are used in the LRN basic computer along with one bus encoder.

2.5.1. Opcode Decoder

A 4*16 opcode decoder used in the LRN Basic Computer decodes the 4-bit opcode to provide the necessary instruction to the computer. The 4 bits opcode are given by the 14-17th bit of the instruction register. The outputs from the decoder are taken as symbols D0 to D15.

2.5.2. Timing Decoder

The 4*16 timing decoder is used to decode the 4-bit sequence counter to provide the necessary timing signals required for the different instruction cycles.

2.5.3. Encoder

A bus encoder is used to transfer the values from the different registers to the bus. It encodes the value provided from seven registers(AR, PC, DR, AC, IR, TR, Memory) and provide the control signals to determine which register is selected by bus during a particular register transfer. The truth table for the 3*8 encoder is given below:

X0	X1	X2	X3	X4	X5	X6	X7	S0	S1	S2	Selected Register
1	0	0	0	0	0	0	0	0	0	0	none
0	1	0	0	0	0	0	0	0	0	1	AR
0	0	1	0	0	0	0	0	0	1	0	PC
0	0	0	1	0	0	0	0	0	1	1	DR
0	0	0	0	1	0	0	0	1	0	0	AC
0	0	0	0	0	1	0	0	1	0	1	IR
0	0	0	0	0	0	1	0	1	1	0	TR
0	0	0	0	0	0	0	1	1	1	1	Memory

Table 2.5.3 Encoder for bus selection circuit

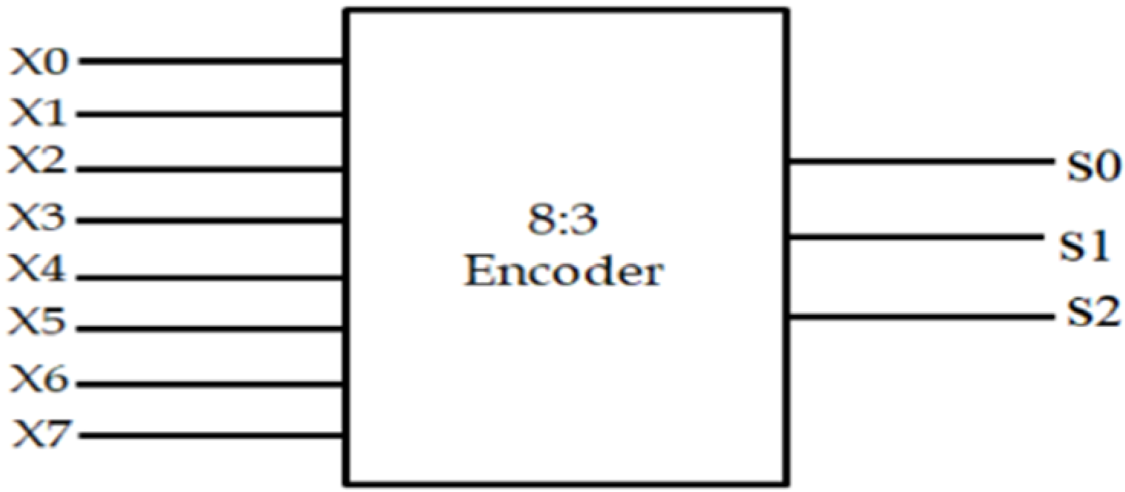

Fig 2.5.3 Encoder for Bus Selection Input

2.6. Instruction Set Architecture

There are three-instruction formats in the LRN Basic Computer i.e., Memory-Reference Instruction, Register-Reference Instruction, and Input-Output Instruction.

2.6.1. Memory Reference Instruction (OP-CODE = 0000 - 1110)

19	18	17	13 0
I_0	I_1	OPCODE	Address

Memory Reference Instructions uses the first 14 bits to specify the address/data, another 4 bits for the opcode, and the remaining two bits to identify the addressing mode. The addressing mode is selected based on the following table:

Addressing mode for Memory Reference Instruction

I_1	I_2	**Addressing mode**
0	0	Direct Addressing mode
0	1	Indirect Addressing mode
1	0	Immediate Addressing mode
1	1	NONE

Table 2.6.1 Addressing mode of LRN Computer

2.6.2. Register-Reference Instruction (OP-CODE = 1111)

19	18	17	13 0
0	0	1 1 1 1	Address

The Register-Reference Instructions are recognized when the opcode is 1111 with 00 in the leftmost bit (bit-18 and 19) of the instruction. The remaining 14 bits are used to determine the type of operation to be performed.

2.6.3. Input-Output Instruction (OP-CODE = 1111)

19	18	17	13 0
1	1	1 1 1 1	Address

Similarly, Input-Output InstructionS are recognized when the opcode is 1111 (17-14)th bits with 11 at (19-18)th bits. The remaining 14 bits are used to determine the type of operation/instruction to be performed.

All the instructions in the LRN Computer are listed below along with the Hex-Code and a short description of their function.

Memory Reference Instruction

Symbol	Hexadecimal Code			Description
Memory-reference instructions				
	I_0I_1 = 00	**I_0I_1 = 01**	**I_0I_1 = 10**	
AND	0(0-3)XXX	4(0-3)XXX	8(0-3)XXX	AND memory word to AC
OR	0(4-7)XXX	4(4-7)XXX	8(4-7)XXX	OR memory word to AC
XOR	0(8-B)XXX	4(8-B)XXX	8(8-B)XXX	XOR memory word to AC
NAND	0(C-F)XXX	4(C-F)XXX	8(C-F)XXX	NAND memory word to AC
NOR	1(0-3)XXX	5(0-3)XXX	9(0-3)XXX	NOR memory word to AC
ADD	1(4-7)XXX	5(4-7)XXX	9(4-7)XXX	ADD memory word to AC
SUB	1(8-B)XXX	5(8-B)XXX	9(8-B)XXX	Subtract memory from AC
SEQ	1(C-F)XXX	5(C-F)XXX	9(C-F)XXX	Skip if equal
LDA	2(0-3)XXX	6(0-3)XXX	A(0-3)XXX	Load memory to AC
STA	2(4-7)XXX	6(4-7)XXX	A(4-7)XXX	Store content of AC in memory
BUN	2(8-C)XXX	6(8-C)XXX	A(8-C)XXX	Branch unconditionally
BSA	2(C-F)XXX	6(C-F)XXX	A(C-F)XXX	Branch and save return address
XCHG	3(0-3)XXX	7(0-3)XXX	B(0-3)XXX	Exchange AC and memory
ISZ	3(4-7)XXX	7(4-7)XXX	B(4-7)XXX	Increment and skip if zero
DSZ	3(8-C)XXX	7(8-C)XXX	B(8-C)XXX	Decrement and skip if zero

Table 2.6.2 Memory Reference Instruction

Register Reference Instructions		
CLA	3E000	Clear the accumulator (AC)
CMA	3D000	Complement the accumulator (AC)
CLE	3C800	Clear the extended accumulator (E)
CME	3C400	Complement the extended accumulator (E)
CIR	3C200	Circulate the accumulator right
CIL	3C100	Circulate the accumulator left
INC	3C080	Increment the accumulator (AC)
DEC	3C040	Decrement the accumulator (AC)
SPA	3C020	Skip if positive in the accumulator (AC)
SNA	3C010	Skip if negative in the accumulator (AC)
SZA	3C008	Skip if zero in the accumulator (AC)
SZE	3C004	Skip if zero in the extended accumulator (E)
HLT	3C002	Halt computer, set S to zero

Table 2.6.3 Register Reference Instructions

Input Output Instructions		
INP	FE000	Store content of INTR to AC (0-7)
OUT	FD000	Store content of AC (0-7) to OUTR
SIE	FC800	Skip if input flag in enabled
SID	FC400	Skip if input flag is disabled
SOE	FC200	Skip if output flag is enabled
SOD	FC100	Skip if output flag is disabled
ION	FC080	Interrupt enable ON
IOF	FC040	Interrupt enable OFF

Table 2.6.3 Input Output Instructions

2.7. Instruction Cycle

In the LRN basic computer, all the machine instruction is executed in the following cycle:

2.7.1. Fetch

In this cycle, the address of the PC is transferred to the AR. Then the instruction present in the Instruction register is fetched from the memory and placed in the IR register and the program counter is incremented by 1 so that it can point to the address of the next instruction in the program.

$R'T_0$: $AR \leftarrow PC$

$R'T_1$: $IR \leftarrow M[AR]$, $PC \leftarrow PC+1$

2.7.2. Decode

As the name suggests, the instruction present in the Instruction Register (IR) is decoded. The I1 and I0 flip flops are set with the 18 and the 19th bit of the IR and the first 14 of the instructions are transferred to the address register. Then, the opcodes present in the 14-17th bit are decoded.

$R'T_2$: $I_0 \leftarrow IR(19)$, $I_1 \leftarrow IR(18)$, $D_0, D_1, \ldots, D_{15} \leftarrow$ Decode $IR(14\text{-}17)$, $AR \leftarrow IR(0\text{-}13)$

2.7.3. Execute the instruction

In this cycle, the actual execution of the instruction takes place.

2.8. Micro Operations

Fetch Cycle:

$R'T_0$: $AR \leftarrow PC$

$R'T_1$: $IR \leftarrow M[AR]$, $PC \leftarrow PC+1$

Decode Cycle:

$R'T_2$: $I_0 \leftarrow IR(19)$, $I_1 \leftarrow IR(18)$, $D_0, D_1, \ldots, D_{15} \leftarrow$ Decode $IR(14\text{-}17)$, $AR \leftarrow IR(0\text{-}13)$

Interrupt Cycle:

$T_0'T_1'T_2'.(IEN).(FGI+FGO)$: $R \leftarrow 1$

RT_0 : $AR \leftarrow 0$, $TR \leftarrow PC$

RT_1 : $M[AR] \leftarrow TR$, $PC \leftarrow 0$

RT_2 : $PC \leftarrow PC+1$, $IEN \leftarrow 0$, $R \leftarrow 0$, $SC \leftarrow 0$

Execution Cycle:

Memory Reference Instruction:

Direct addressing mode: $D_{15}'.T_3.I_0'.I_1'$: $DR \leftarrow M[AR]$

$D_{15}'.T_4.I_0'.I_1'$: Do Nothing

Indirect addressing mode: $D_{15}'.T_3.I_0'.I_1$: $AR \leftarrow M[AR]$

$D_{15}'.T_4.I_0'.I_1$: $DR \leftarrow M[AR]$

Immediate addressing mode: $D_{15}'.T_3.I_0.I_1'$: $DR \leftarrow AR$

$D_{15}'.T_4.I_0.I_1'$: Do nothing

Register Reference Instruction: $D_{15}.T_3.I_0'.I_1'$: Execute an RRI

Input Output Instruction: $D_{15}.T_3.I_0.I_1$: Execute an IOI

Memory Reference Instructions		
AND	D_0T_5	AC←AC^DR , SC←0
OR	D_1T_5	AC←ACvDR , SC←0
XOR	D_2T_5	AC←AC⊕DR , SC←0
NAND	D_3T_5	AC← (AC^DR)' , SC←0
NOR	D_4T_5	AC←(ACvDR)' , SC←0
ADD	D_5T_5	AC←AC+DR,E←Cout,SC←0
SUB	D_6T_5	AC←AC+DR'+1 , SC←0
SEQ	D_7T_5	TR←AC, AC←AC⊕DR,
	D_7T_6	If (AC = 0) then (PC=PC+1), AC←TR, SC←0
LDA	D_8T_5	AC←DR,SC←0
STA	D_9T_5	M[AR]←AC, SC←0
BUN	$D_{10}T_5$	PC←AR, SC←0
BSA	$D_{11}T_5$	M[AR]←PC, AR←AR+1
	$D_{11}T_6$	PC←AR, SC←0
XCHG	$D_{12}T_5$	TR←AC
	$D_{12}T_6$	AC←DR
	$D_{12}T_7$	DR←TR, SC←0
ISZ	$D_{13}T_5$	DR←DR+1
	$D_{13}T_6$	M[AR]←DR (if DR=0 then PC←PC+1), SC←0
DSZ	$D_{14}T_5$	DR←DR-1
	$D_{14}T_6$	M[AR]←DR (if DR = 0 then PC←PC+1), SC←0

Table 2.8.1 List of Micro Operations in Memory Reference Instructions

Register Reference Instruction		
	$D_{15}I_0'I_1'T_3 = r$	(Common to all register-reference instruction)
	$IR(i) = B_i$	(i= 0,1,2,……….,13)
	r	SC←0
CLA	rB13	AC←0
CMA	rB12	AC←AC’
CLE	rB11	E←0
CME	rB10	E←E’
CIR	rB9	AC← shr AC, AC(19) ←E, E←AC(0)
CIL	rB8	AC← shl AC, AC(0) ←E, E←AC(19)
INC	rB7	AC←AC+1
DEC	rB6	AC←AC-1
		AC(19)==0
SPA	rB5	if(AC(19) = 0) then (PC←PC+1)
SNA	rB4	if(AC(19) = 1) then (PC←PC+1)
SZA	rB3	if(AC = 0) then (PC←PC+1)
SZE	rB2	if(E = 0) then (PC←PC+1)
HLT	rB1	S←0

Table 2.8.2 List of Micro Operations in Register Reference Instructions

Input-Output Instruction		
	$D_{15}I_0I_1T_3 = p$	(Common to all input-output instructions)
	$IR(i) = B_i$	(i = 6,7,…….13)
	p	SC←0
INP	pB13	AC (0-9)←INPR, FGI←0
OUT	pB12	OUTR← AC (0-9), FGO←0
SIE	pB11	if (FGI=1) then (PC←PC+1)
SID	pB10	if (FGI=0) then (PC←PC+1)
SOE	pB9	if (FGO=1) then (PC←PC+1)
SOD	pB8	if (FGO=0) then (PC←PC+1)
ION	pB7	IEN←1
IOF	pB6	IEN←0

Table 2.8.3 List of Micro Operations in Input Output Instructions

2.9. Flowchart of Operations

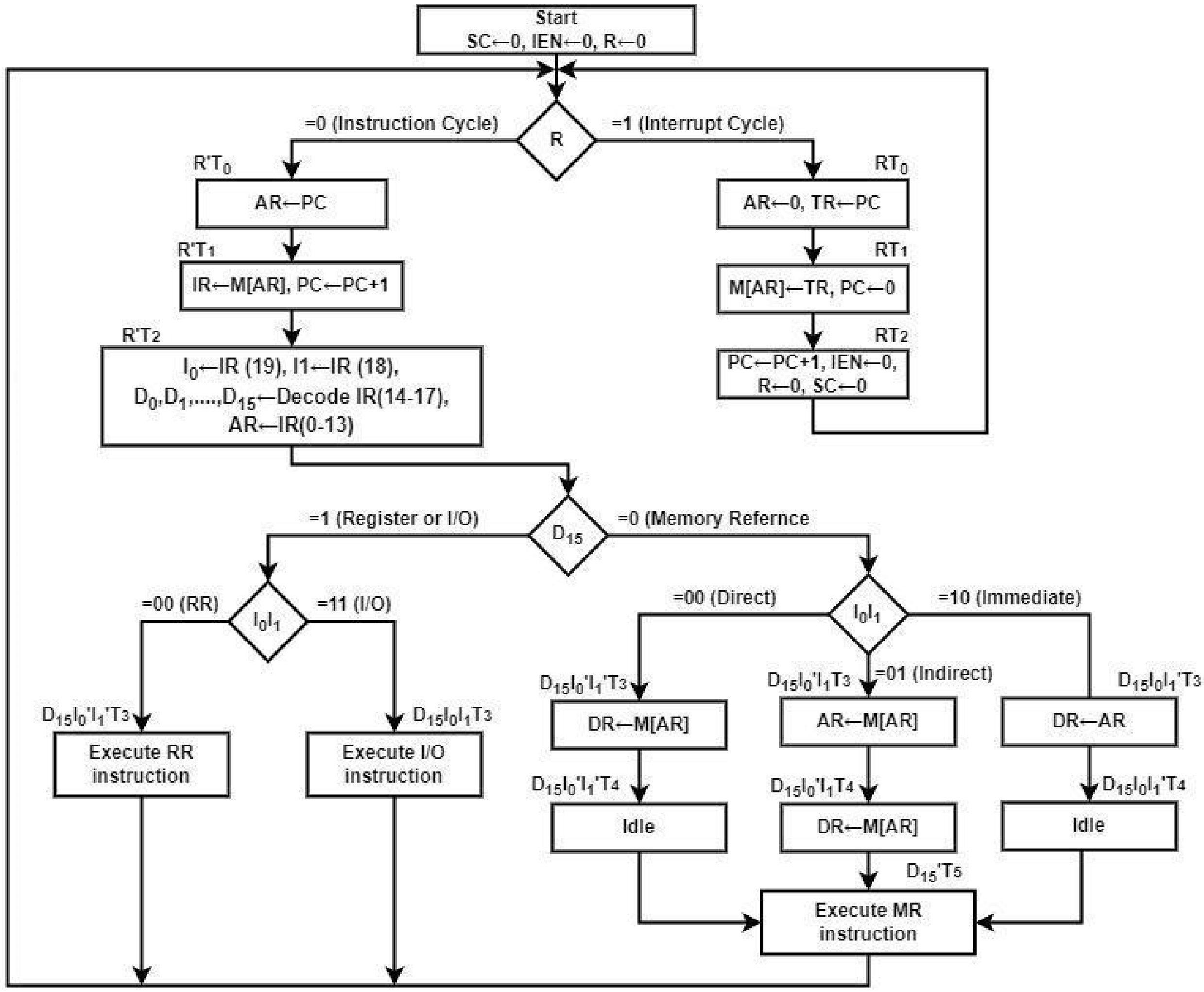

Fig 2.9 Flowchart of operations

Chapter 3: DESIGN OF INDIVIDUAL COMPONENTS

3.1. Design of Registers

3.1.1. Design of AR

The AR stands for address register. The size of the AR is 14 bits. When a memory location is to be selected, the AR stores the address of the location.

$R'T_0$:	$AR \leftarrow PC$	Load
$R'T_2$:	$AR \leftarrow IR(0\text{-}13)$	Load
$D_{15}'.T_3.I_0'.I_1$:	$AR \leftarrow M[AR]$	Load
$D_{11}T_5$:	$AR \leftarrow AR+1$	Increment
RT_0:	$AR \leftarrow 0$	Clear

Load AR(LD) $= R'T_0 + R'T_2 + D_{15}'.T_3.I_0'.I_1$

Increment AR(INCR) $= D_{11}T_5$

Clear AR(CLR) $= RT_0$

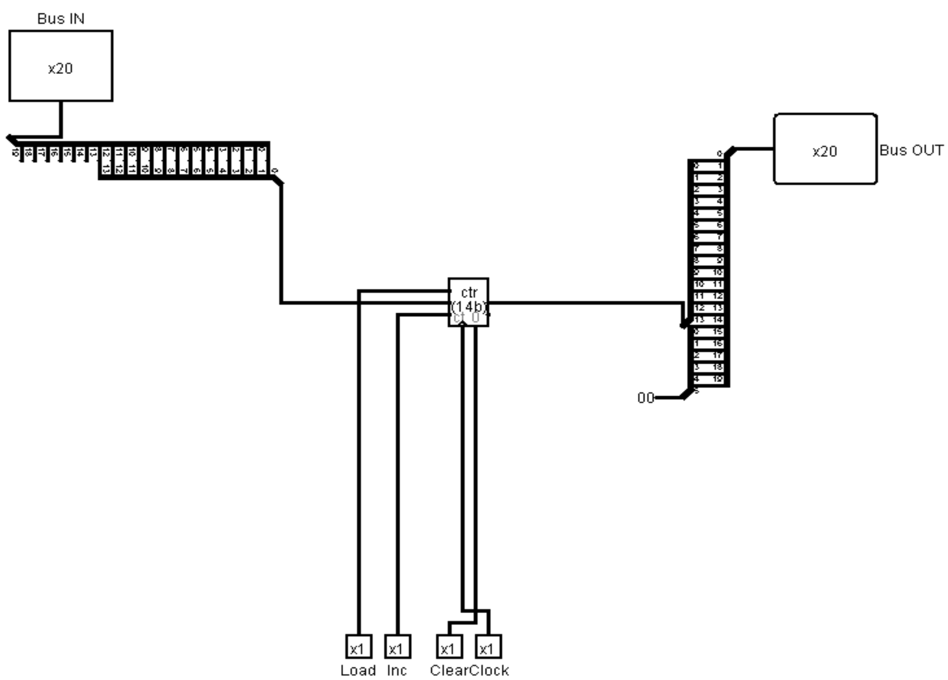

Fig 3.1.1.1 Design of AR

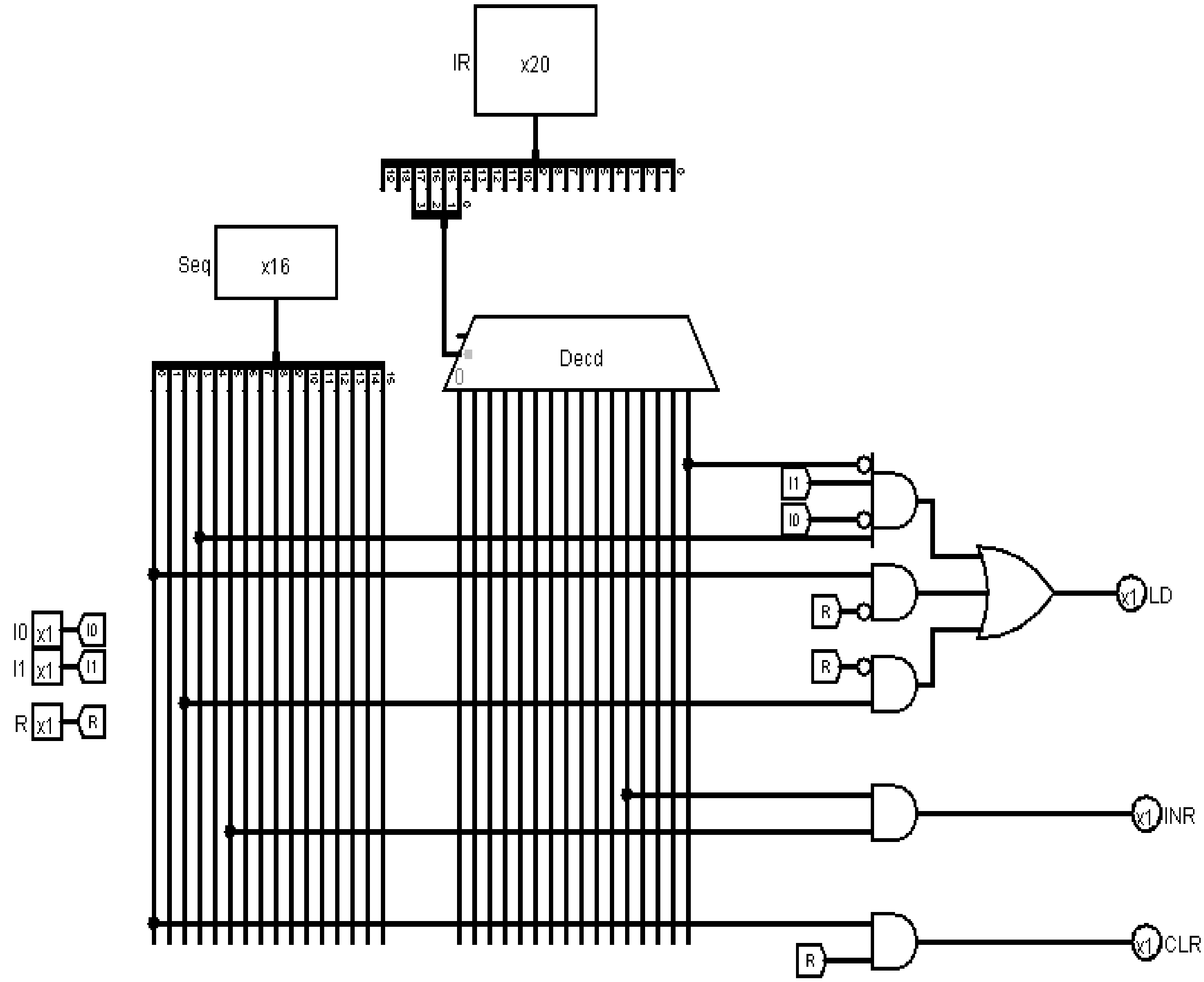

Fig 3.1.1.2 Control of AR

3.1.2. Design of PC

The PC stands for program counter. The size of the PC is 14 bits. The program counter is updated after the completion of certain instructions to maintain program control flow.

$R'T_1$:	$PC \leftarrow PC+1$	Increment
RT_1:	$PC \leftarrow 0$	Clear
RT_2 :	$PC \leftarrow PC+1$	Increment
D_7T_6	If (AC = 0) then (PC=PC+1)	Increment
$D_{10}T_5$	$PC \leftarrow AR$	Load
$D_{11}T_6$	$PC \leftarrow AR$	Load
$D_{13}T_6$	(if DR=0 then $PC \leftarrow PC+1$)	Increment
$D_{14}T_6$	(if DR = 0 then $PC \leftarrow PC+1$)	Increment
rB5	if(AC(15) = 0) then ($PC \leftarrow PC+1$)	Increment
rB4	if(AC(15) = 1) then ($PC \leftarrow PC+1$)	Increment
rB3	if(AC = 0) then ($PC \leftarrow PC+1$)	Increment
rB2	if(E = 0) then ($PC \leftarrow PC+1$)	Increment
pB11	if (FGI=1) then ($PC \leftarrow PC+1$	Increment
pB10	if (FGI=0) then ($PC \leftarrow PC+1$)	Increment
pB9	if (FGO=1) then ($PC \leftarrow PC+1$)	Increment
pB8	if (FGO=0) then ($PC \leftarrow PC+1$)	Increment

Load PC (LD) $= D_{10}T_5 + D_{11}T_6$

Increment PC (INC) $= R'T_1+RT_2+D_7T_6 +D_{13}T_6+D_{14}T_6 +rB_5+rB_4+rB_3+rB_2+pb_{11}+pB_{10}+pB_9+pB_8$

Clear PC (CLR) $= RT_1$

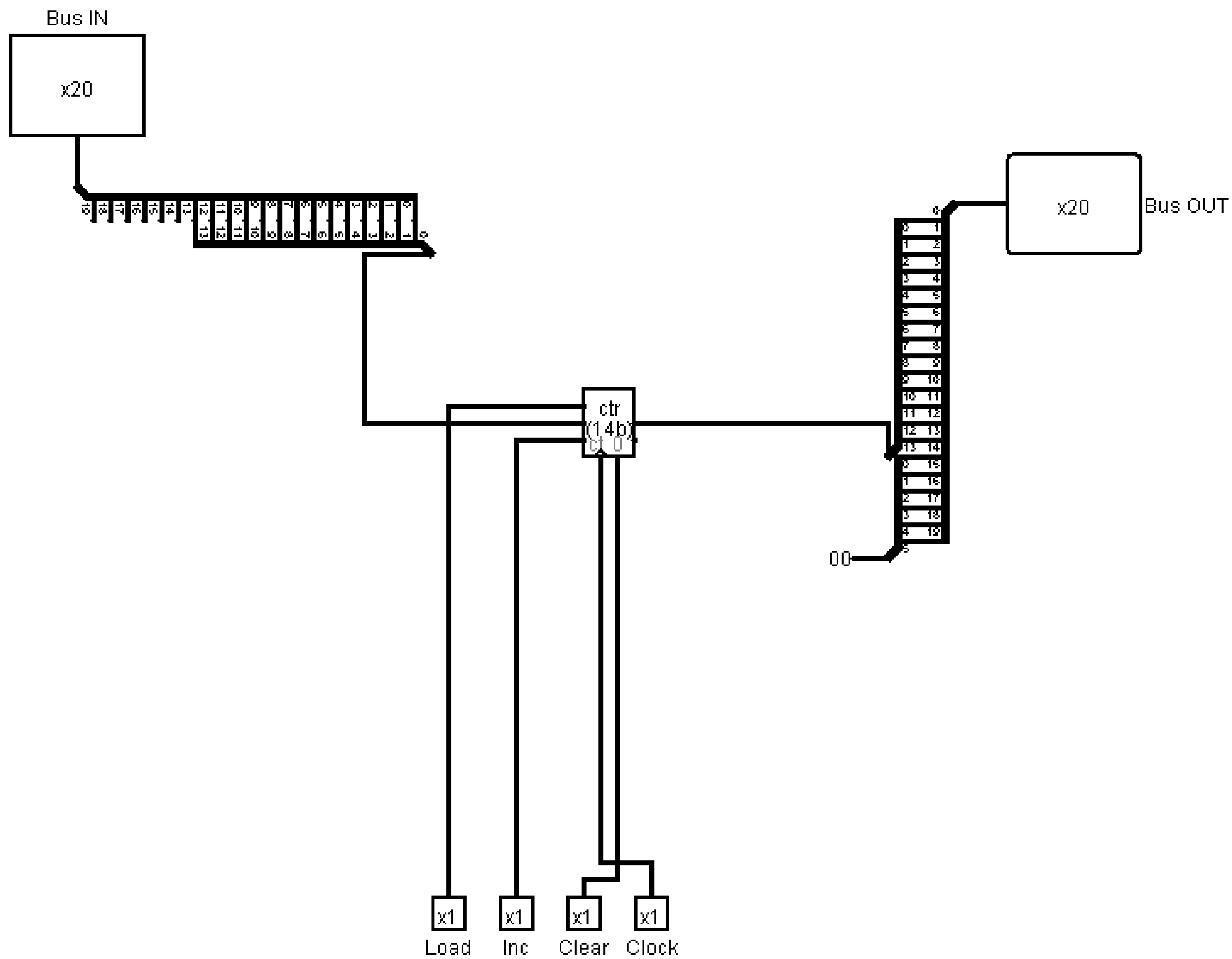

Fig 3.1.2.1 Design of PC

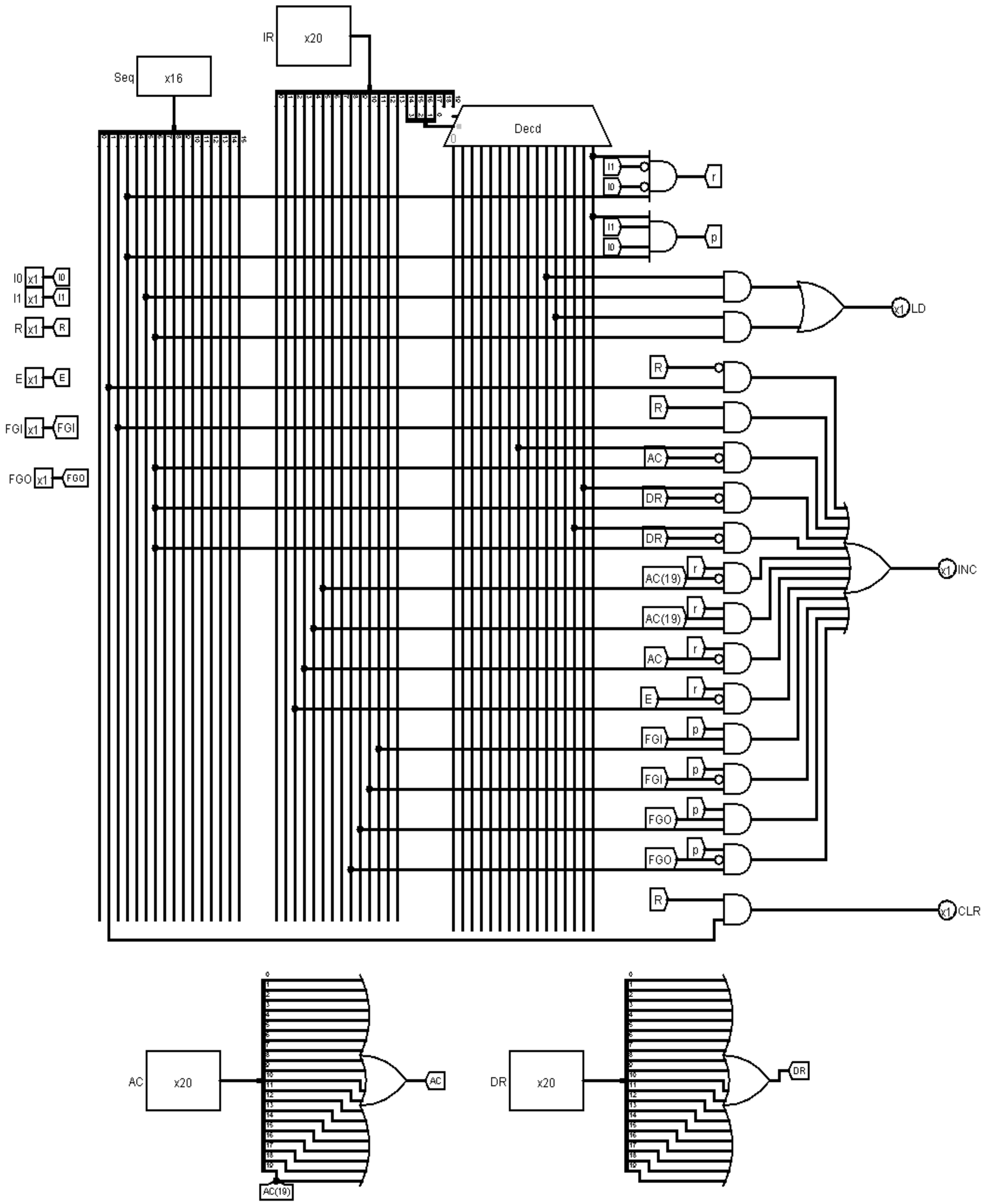

Fig 3.1.2.2 Control of PC

3.1.3. Design of TR

TR stands for temporary register. The size of TR is 20 bits. The temporary register is used while exchanging data present in two memory locations or arithmetic calculations.

RT_0	TR←PC	Load
D_7T_5	TR←AC	Load
$D_{12}T_5$	TR←AC	Load

Load TR (LD) = $RT_0 + D_7T_5 + D_{12}T_5$

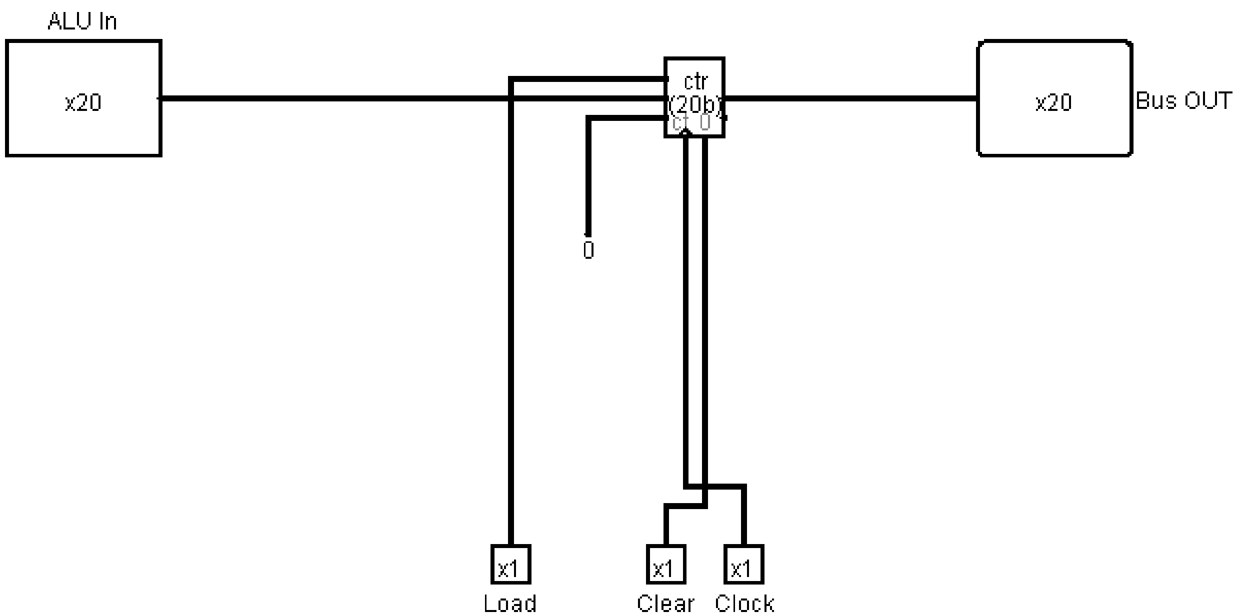

Fig 3.1.3.1 Design of TR

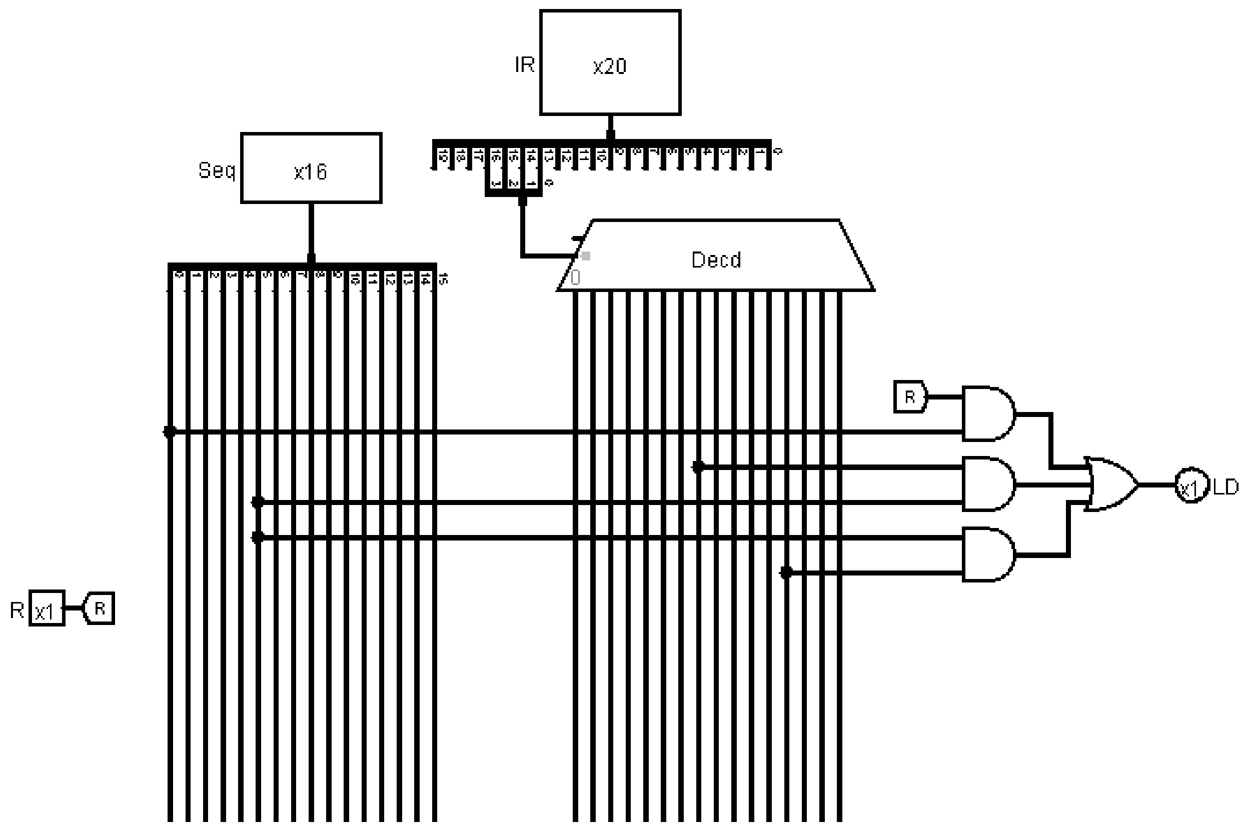

Fig 3.1.3.2 Control of TR

3.1.4. Design of SC

SC stands for sequence counter. The SC is 0 at the beginning of the instruction or interrupt cycle and the SC is set to 0 at the end of each cycle. Otherwise sc is incremented per clock.

RT_2	$SC \leftarrow 0$	Clear
D_0T_5	$SC \leftarrow 0$	Clear
D_1T_5	$SC \leftarrow 0$	Clear
D_2T_5	$SC \leftarrow 0$	Clear
D_3T_5	$SC \leftarrow 0$	Clear
D_4T_5	$SC \leftarrow 0$	Clear
D_5T_5	$SC \leftarrow 0$	Clear
D_6T_5	$SC \leftarrow 0$	Clear
D_7T_6	$SC \leftarrow 0$	Clear
D_8T_5	$SC \leftarrow 0$	Clear
D_9T_5	$SC \leftarrow 0$	Clear
$D_{10}T_5$	$SC \leftarrow 0$	Clear
$D_{11}T_6$	$SC \leftarrow 0$	Clear
$D_{12}T_7$	$SC \leftarrow 0$	Clear
$D_{13}T_6$	$SC \leftarrow 0$	Clear
$D_{14}T_6$	$SC \leftarrow 0$	Clear
r	$SC \leftarrow 0$	Clear
p	$SC \leftarrow 0$	Clear

$$\text{Clear SC (CLR)} = RT_2 + D_0T_5 + D_1T_5 + D_2T_5 + D_3T_5 + D_4T_5 + D_5T_5 + D_6T_5 + D_7T_6 + D_8T_5 + D_9T_5 + D_{10}T_5 + D_{11}T_6 + D_{12}T_7 + D_{13}T_6 + D_{14}T_6 + r + p$$

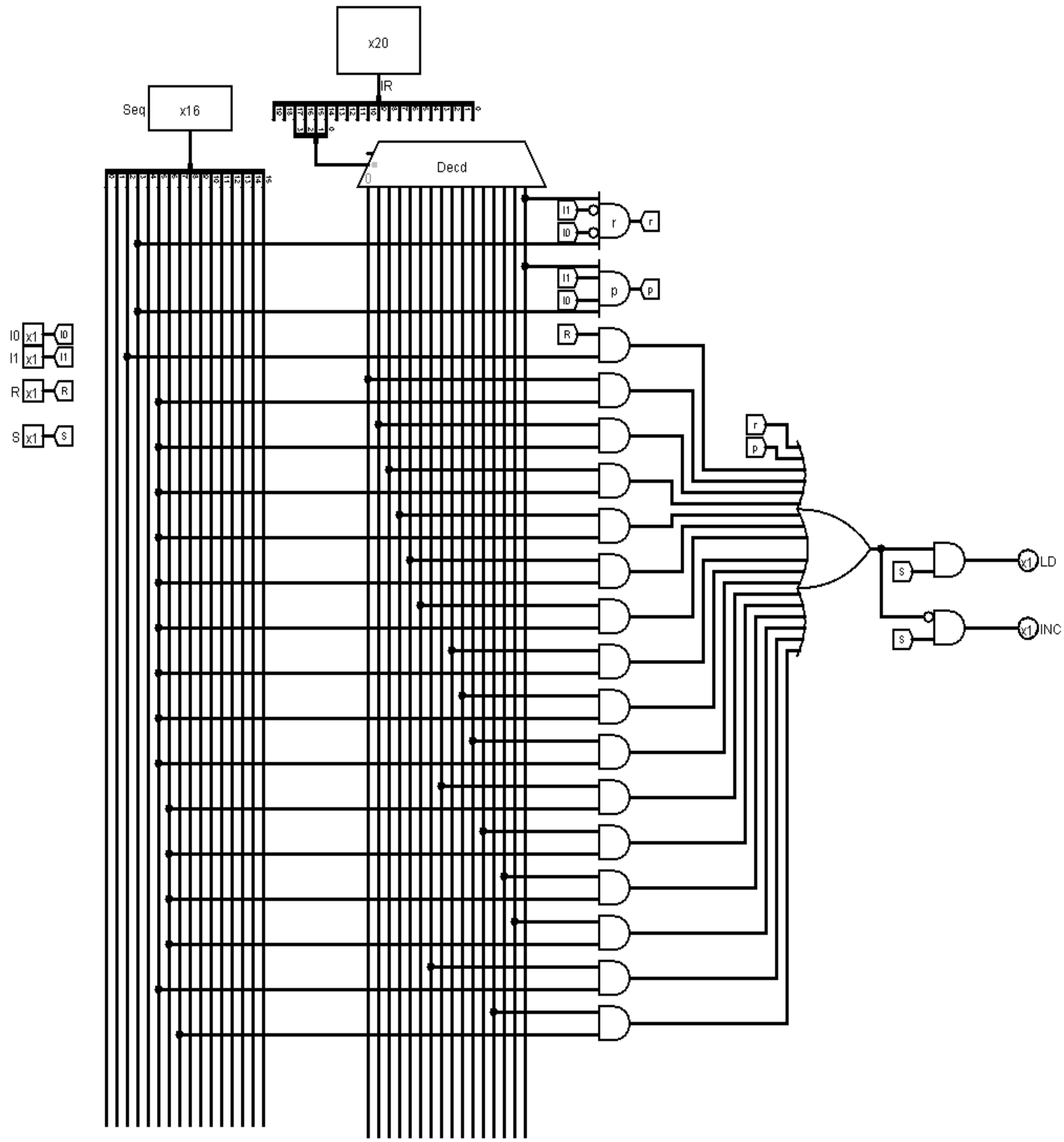

Fig 3.1.4.1 Control of SC

3.1.5. Design of IR

IR stands for Instruction Register. The size of the instruction register is 20 bits. The purpose of IR is to store current instruction.

$R'T_1$: IR←M[AR]	Load

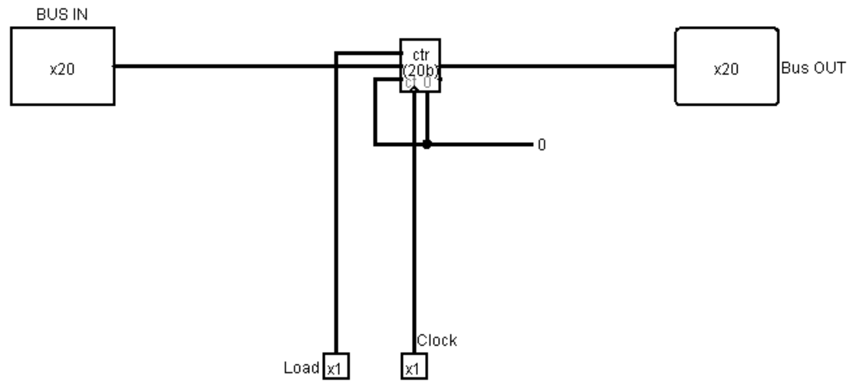

Fig 3.1.5.1 Design of IR

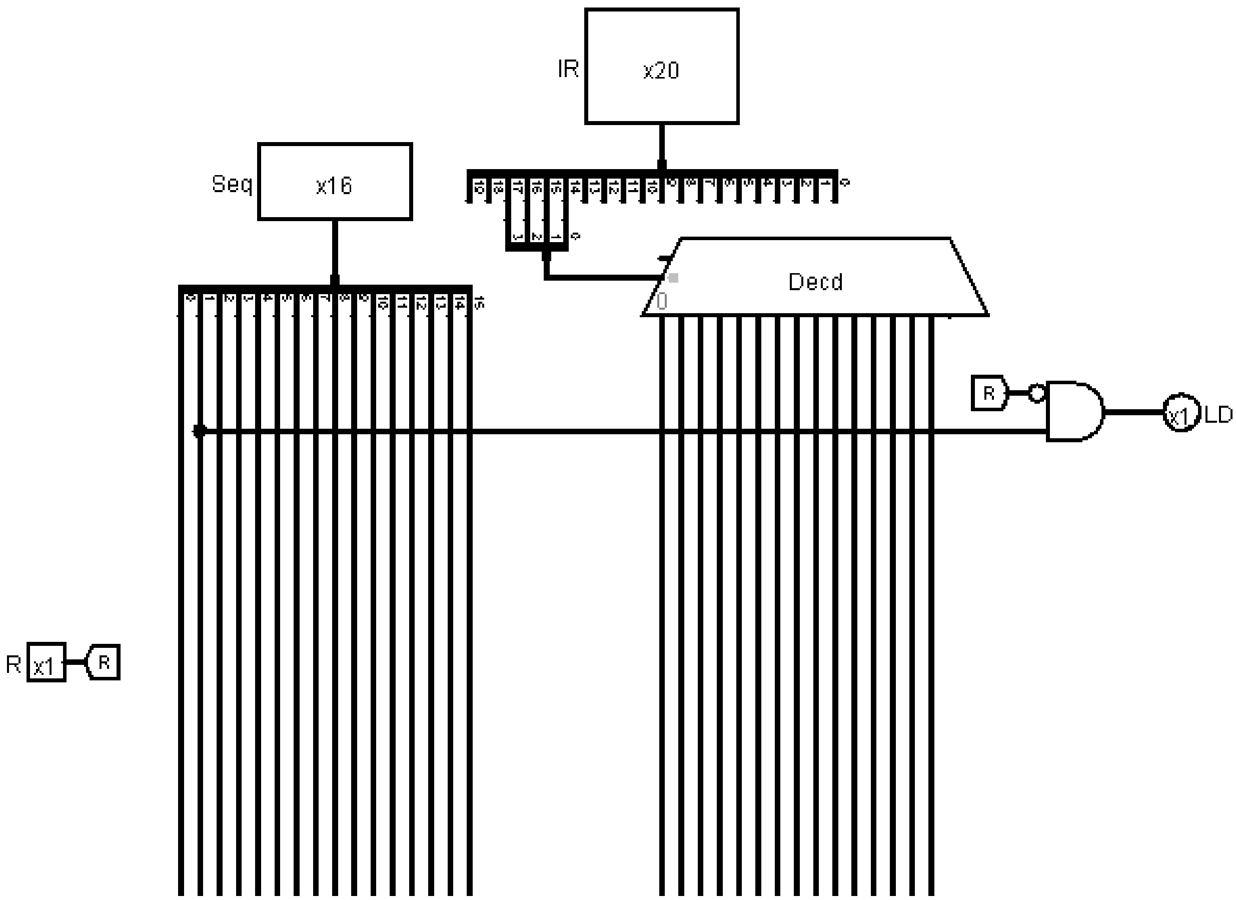

Fig 3.1.5.2 Control of IR

3.1.6. Design of AC

AC stands for Accumulator. It is a register in which intermediate arithmetic logic unit results are stored.

D_0T_5	$AC \leftarrow AC \wedge DR$	Load
D_1T_5	$AC \leftarrow AC \vee DR$	Load
D_2T_5	$AC \leftarrow AC \oplus DR$	Load
D_3T_5	$AC \leftarrow (AC \wedge DR)'$	Load
D_4T_5	$AC \leftarrow (AC \vee DR)'$	Load
D_5T_5	$AC \leftarrow AC+DR$	Load
D_6T_5	$AC \leftarrow AC+DR'+1$	Load
D_7T_5	$AC \leftarrow AC \oplus DR$	Load
D_7T_6	$AC \leftarrow TR$	Load
D_8T_5	$AC \leftarrow DR$	Load
$D_{12}T_6$	$AC \leftarrow DR$	Load
rB12	$AC \leftarrow AC'$	Load
rB9	$AC \leftarrow$ shr AC, $AC(19) \leftarrow E$	Load
rB8	$AC \leftarrow$ shl AC, $AC(0) \leftarrow E$	Load
rB6	$AC \leftarrow AC-1$	Load
pB13	$AC\ (0\text{-}9) \leftarrow INPR$	Load
rB13	$AC \leftarrow 0$	Clear
rB7	$AC \leftarrow AC+1$	Increment

LOAD AC(LD): $D_0T_5 + D_1T_5 + D_2T_5 + D_3T_5 + D_4T_5 + D_5T_5 + D_6T_5 + D_7T_5 + D_7T_6 + D_8T_5 + D_{12}T_6$ + rB12 + rB9 + rB8 + rB6 + pB13

CLEAR AC(CLR): rB13

INCREMENT AC(INC): rB7

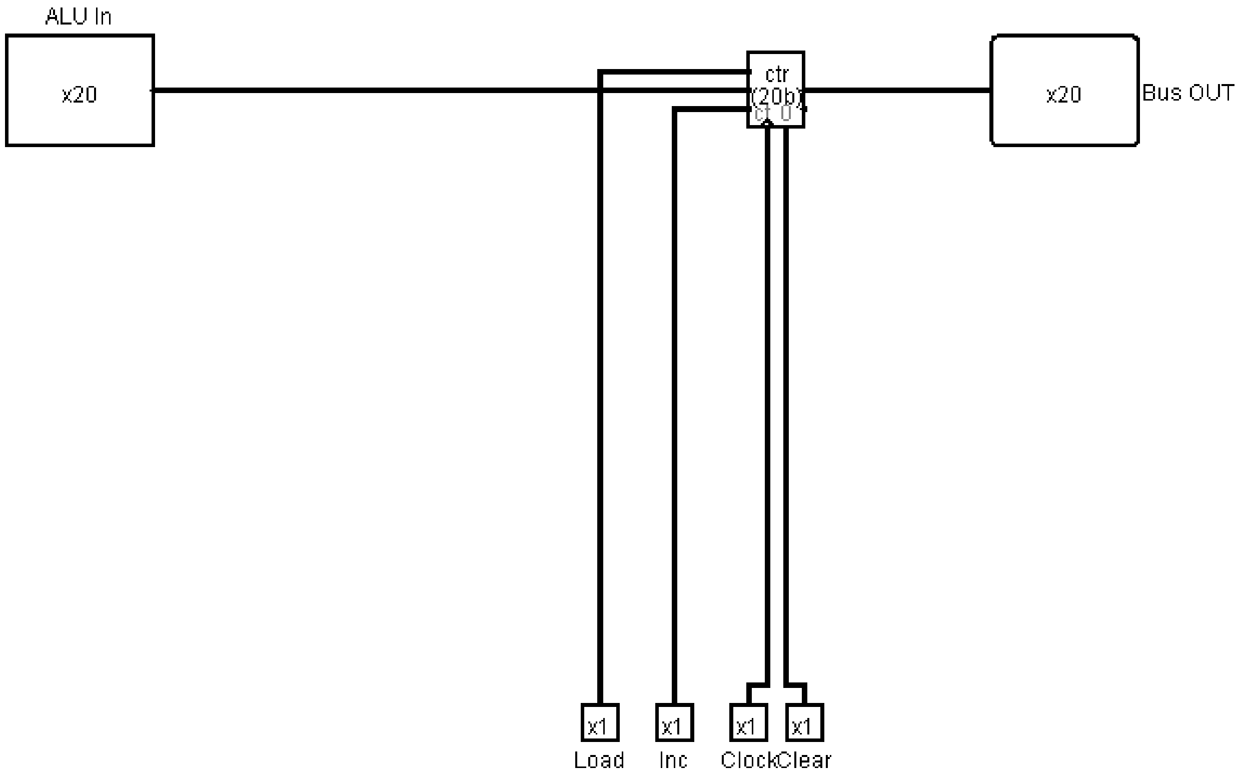

Fig 3.1.6.1 Design of AC

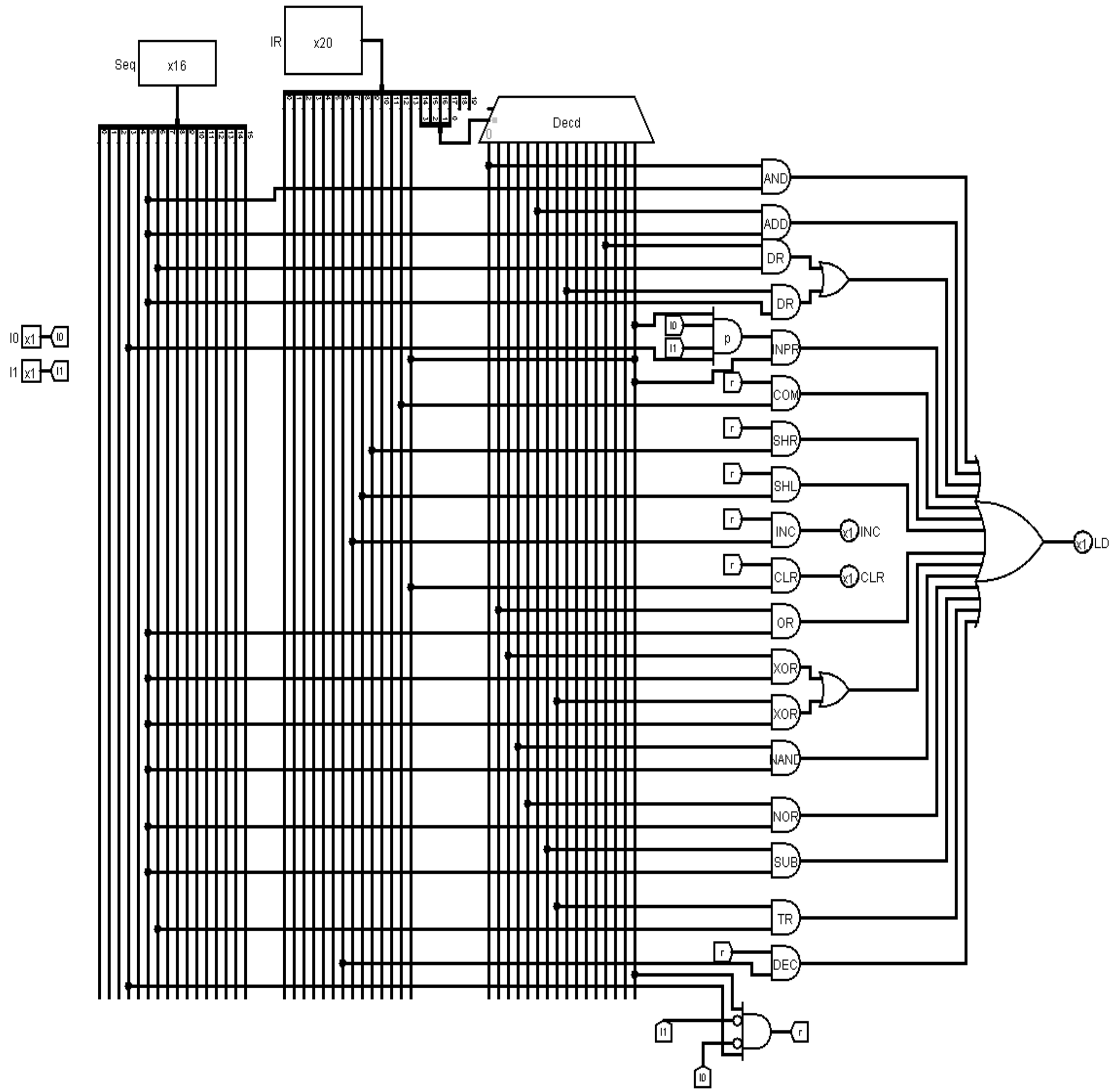

Fig 3.1.6.2 Control of AC

3.1.7. Design of DR

DR stands for data register. The size of DR is 20 bits. The purpose of the DR is to store data.

$D_{12}T_7$	DR←TR	Load
$D_{13}T_5$	DR←DR+1	Increment
$D_{14}T_5$	DR←DR-1	Load
$D_{15}'.T_3.I_0'.I_1'$:	DR←M[AR]	Load
$D_{15}'.T_4.I_0'.I_1$:	DR←M[AR]	Load
$D_{15}'.T_3.I_0.I_1'$:	DR←AR	Load

Load DR (LD) $= D_{12}T_7 + D_{14}T_5 + D_{15}'.T_3.I_0'.I_1' + D_{15}'.T_4.I_0'.I_1 + D_{15}'.T_3.I_0.I_1'$

Increment DR (INC) $= D_{13}T_5$

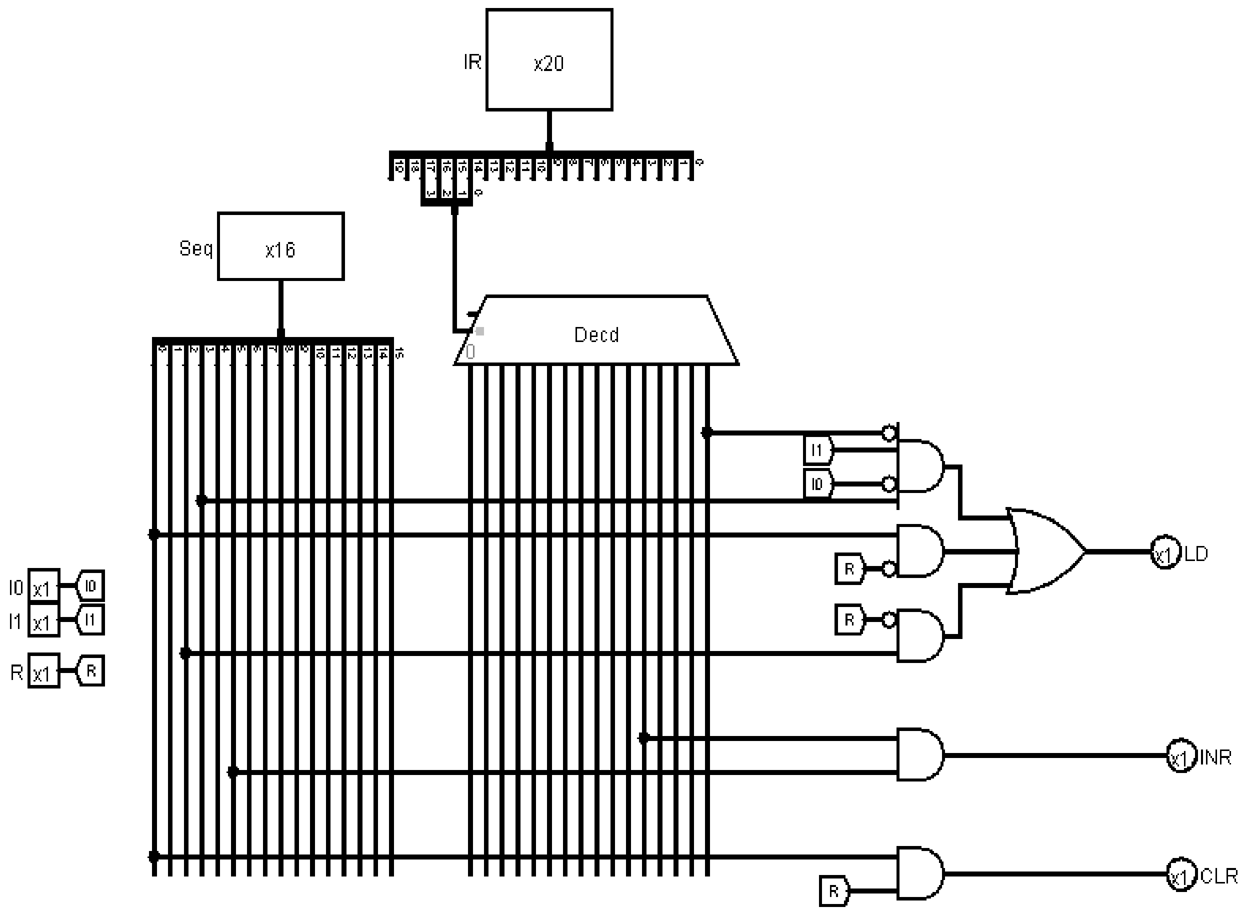

Fig 3.1.7.1 Design of DR

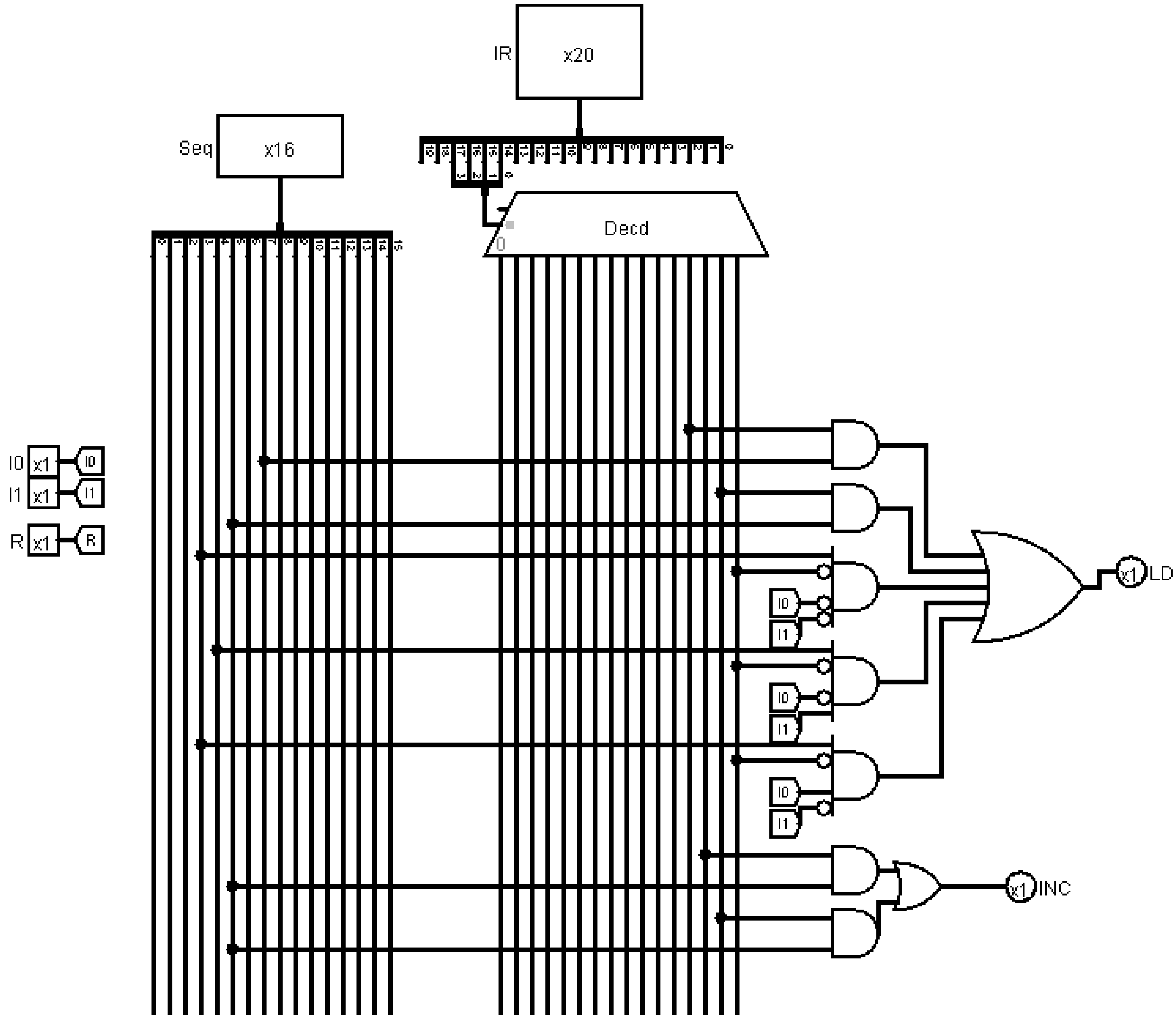

Fig 3.1.7.2 Control of DR

3.1.8. Design of OUTR

OUTR stands for Output Register. It has a size of 8 bits and is used to hold the output data.

pB7	IEN←1

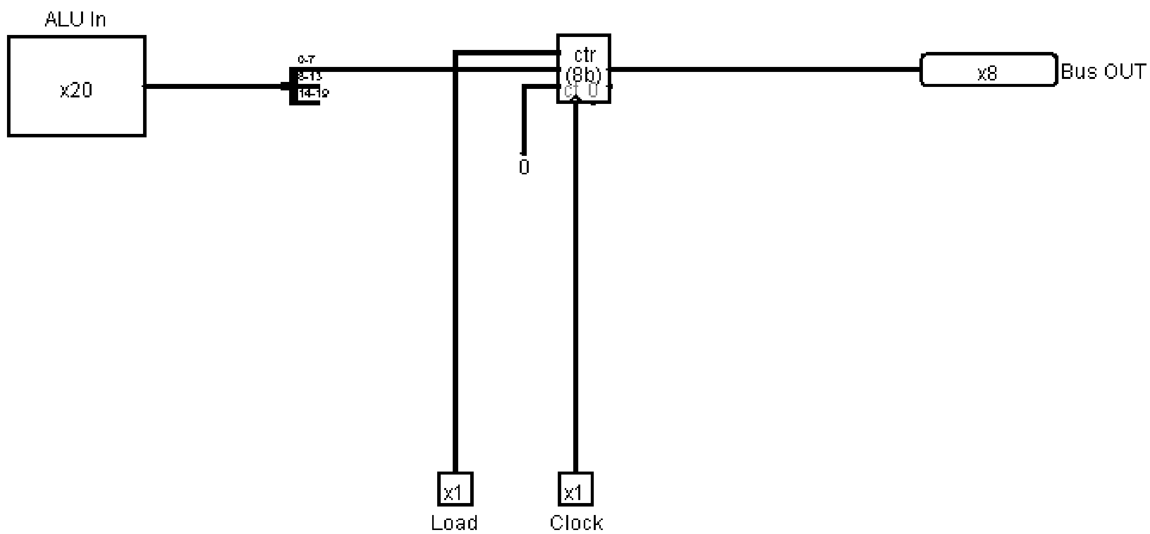

Fig 3.1.8.1 Design of OUTR

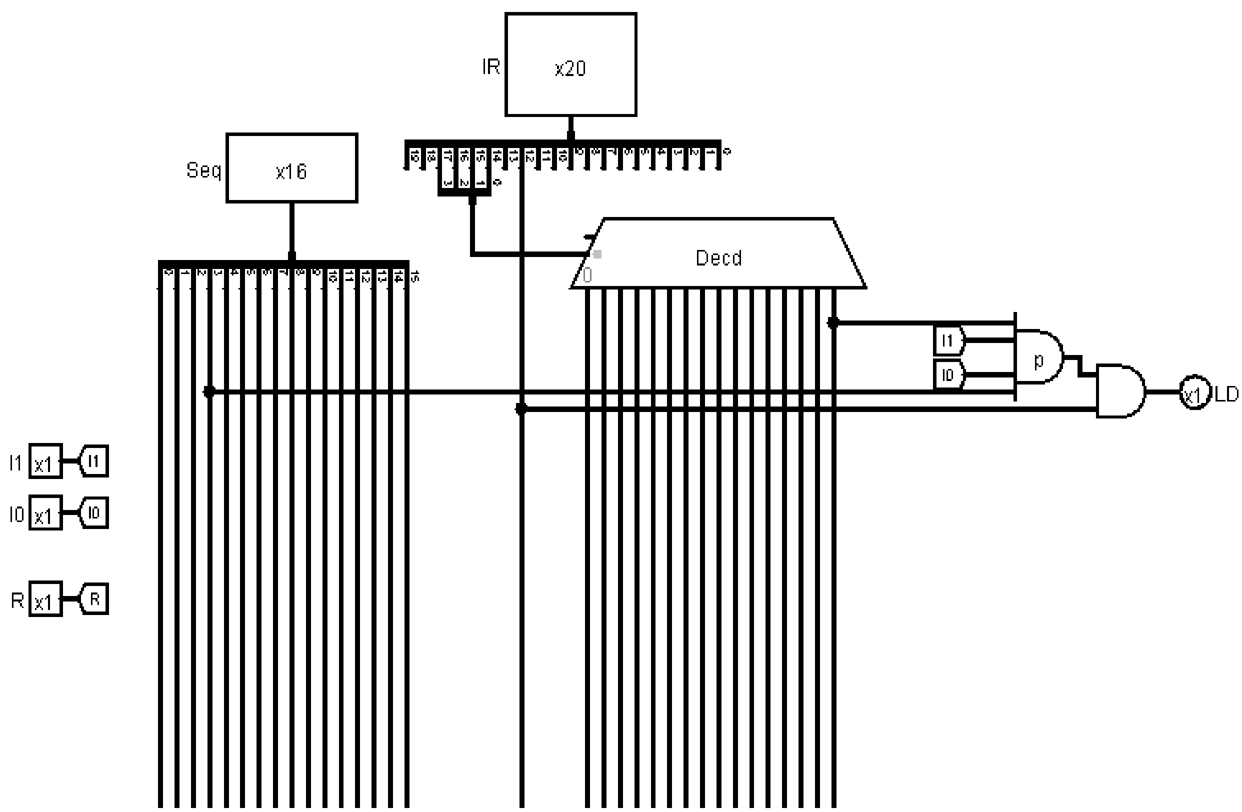

Fig 3.1.8.2 Control of OUTR

3.2 Design of flags

3.2.1. Design of IEN

The IEN stands for Input Enable.

pB7	IEN←1
pB6	IEN←0
RT_2	IEN←0

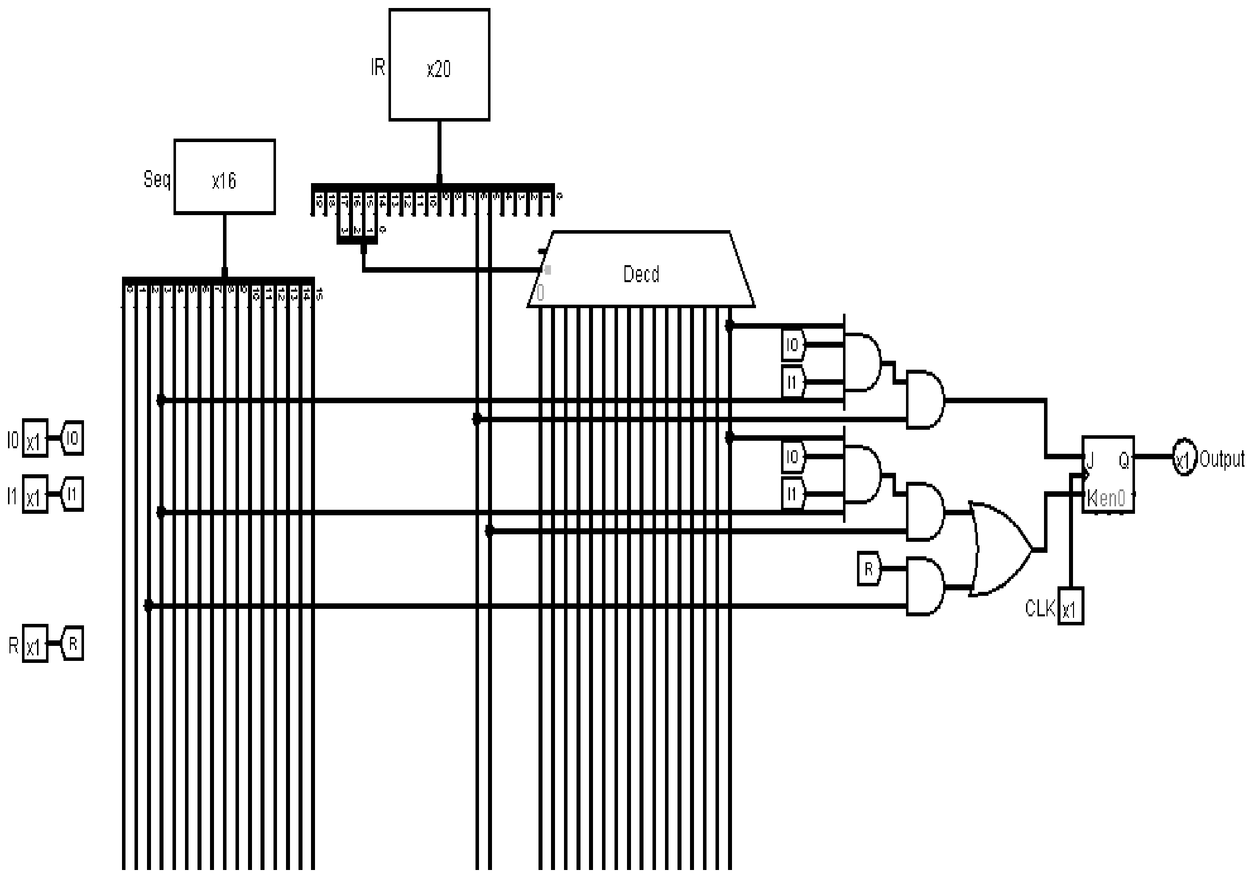

Fig 3.2.1 Design of IEN

3.2.2. Design of FGO

The FGO stands for Flag Output.

pB12	FGO←0

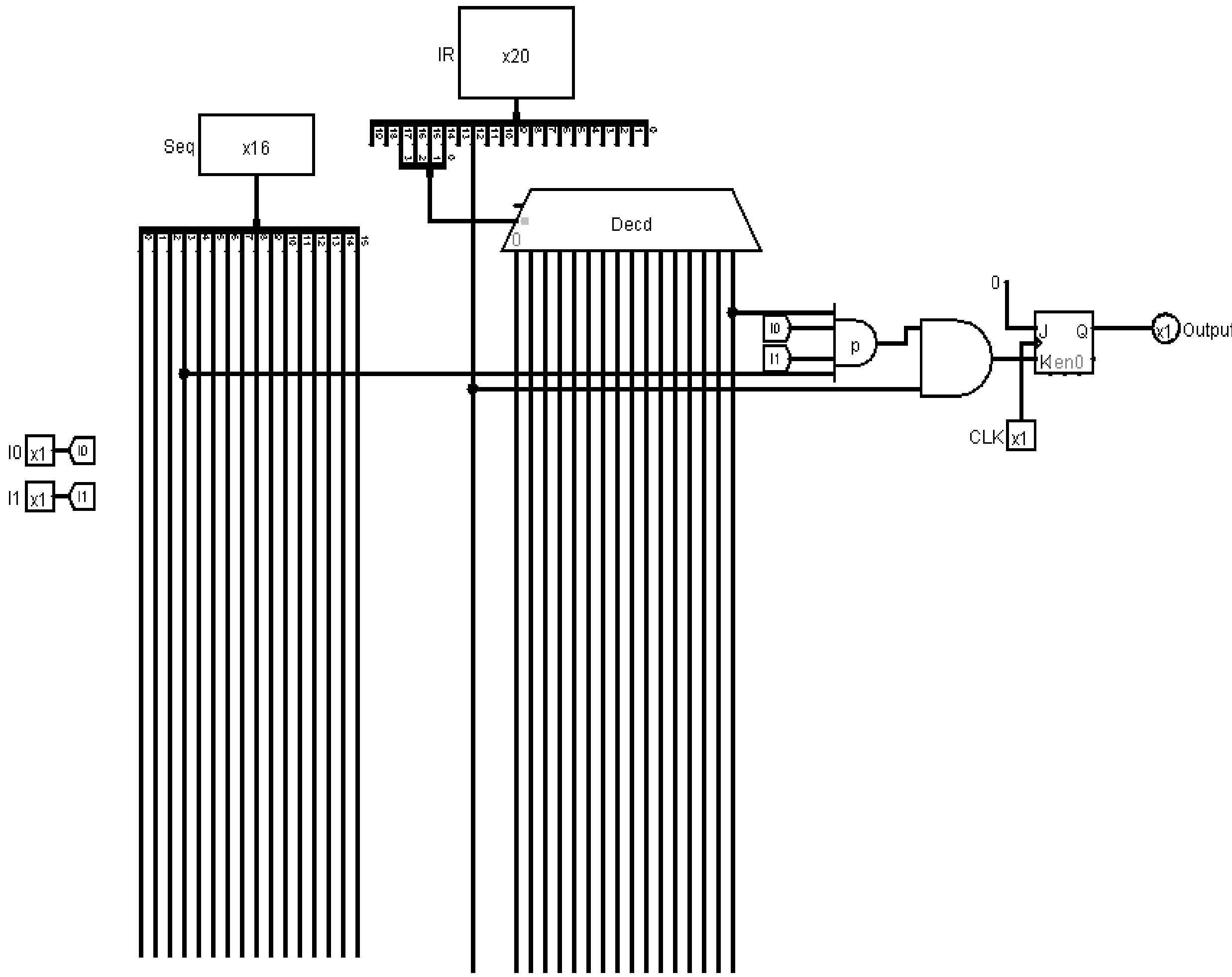

Fig 3.2.2 Design of FGO

3.2.3. Design of FGI

FGI stands for flag input.

pB13	AC (0-10)←INPR, FGI←0

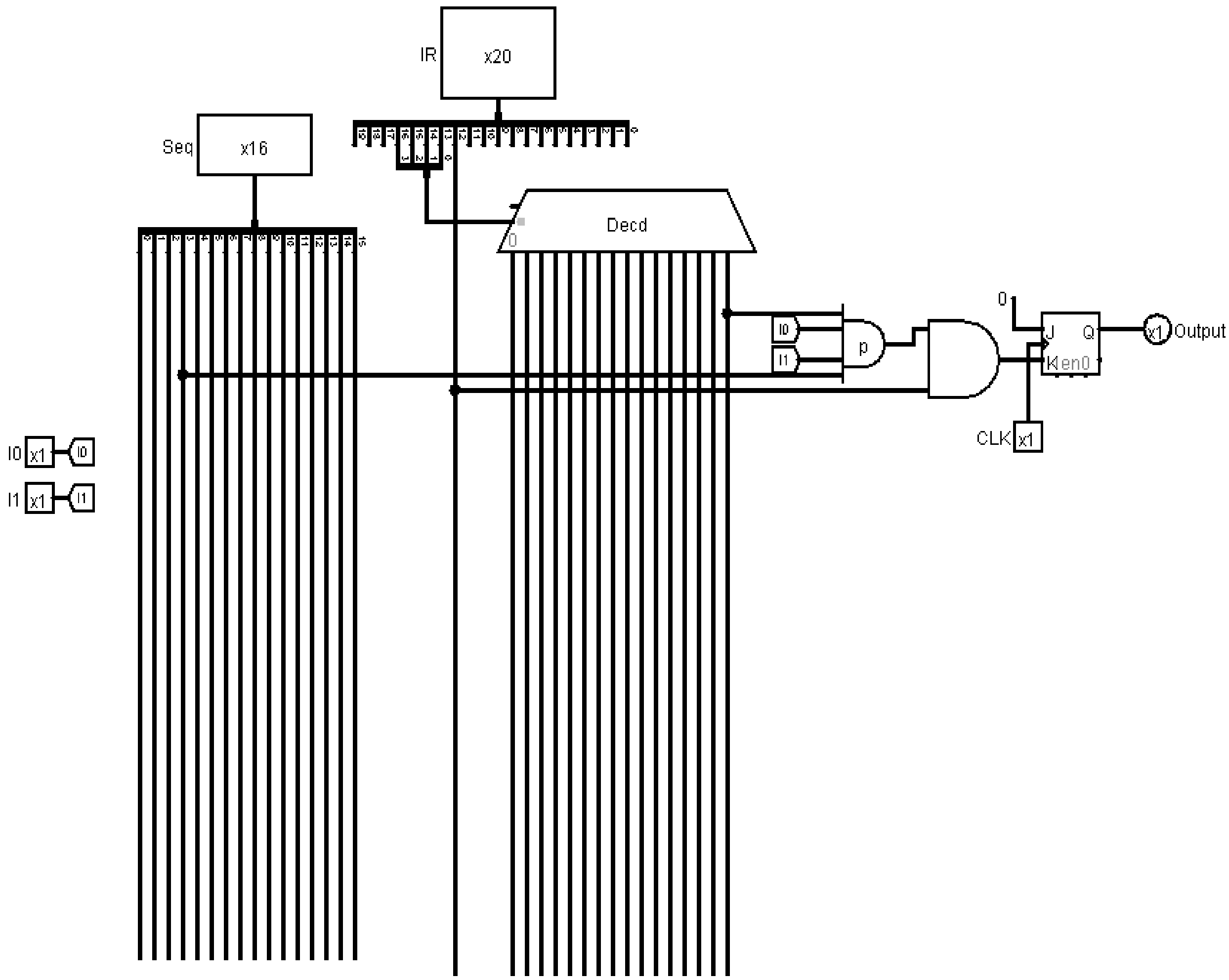

Fig 3.2.3 Design of FGI

3.2.4. Design of R

The R is an interrupt. When R is 0, the instruction cycle is proceeded, otherwise interrupt cycle is processed.

$T_0'T_1'T_2'$.(IEN).(FGI+FGO):	R←1
RT_2	R←0

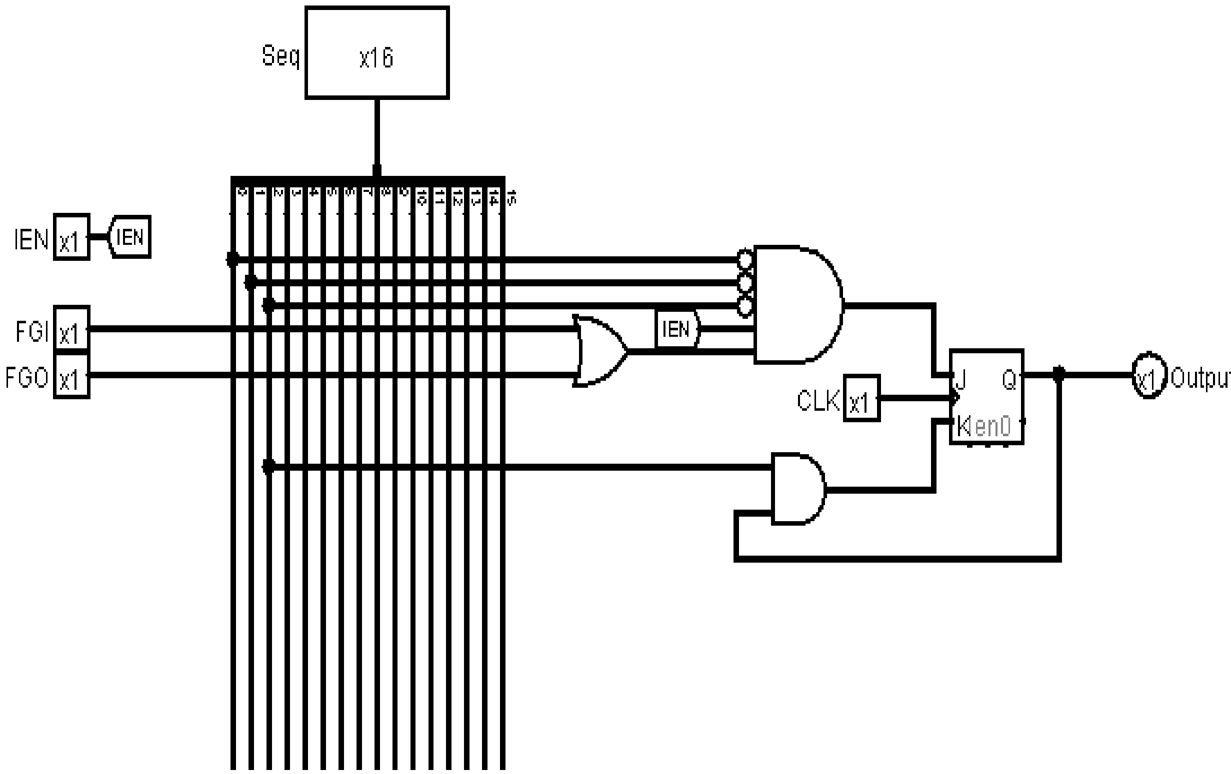

Fig 3.2.4 Design of R

3.2.5. Design of E

rB11	E←0
rB10	E←E’
rB9	E←AC(0)
rB8	E←AC(19)
D_5T_5	E←Cout

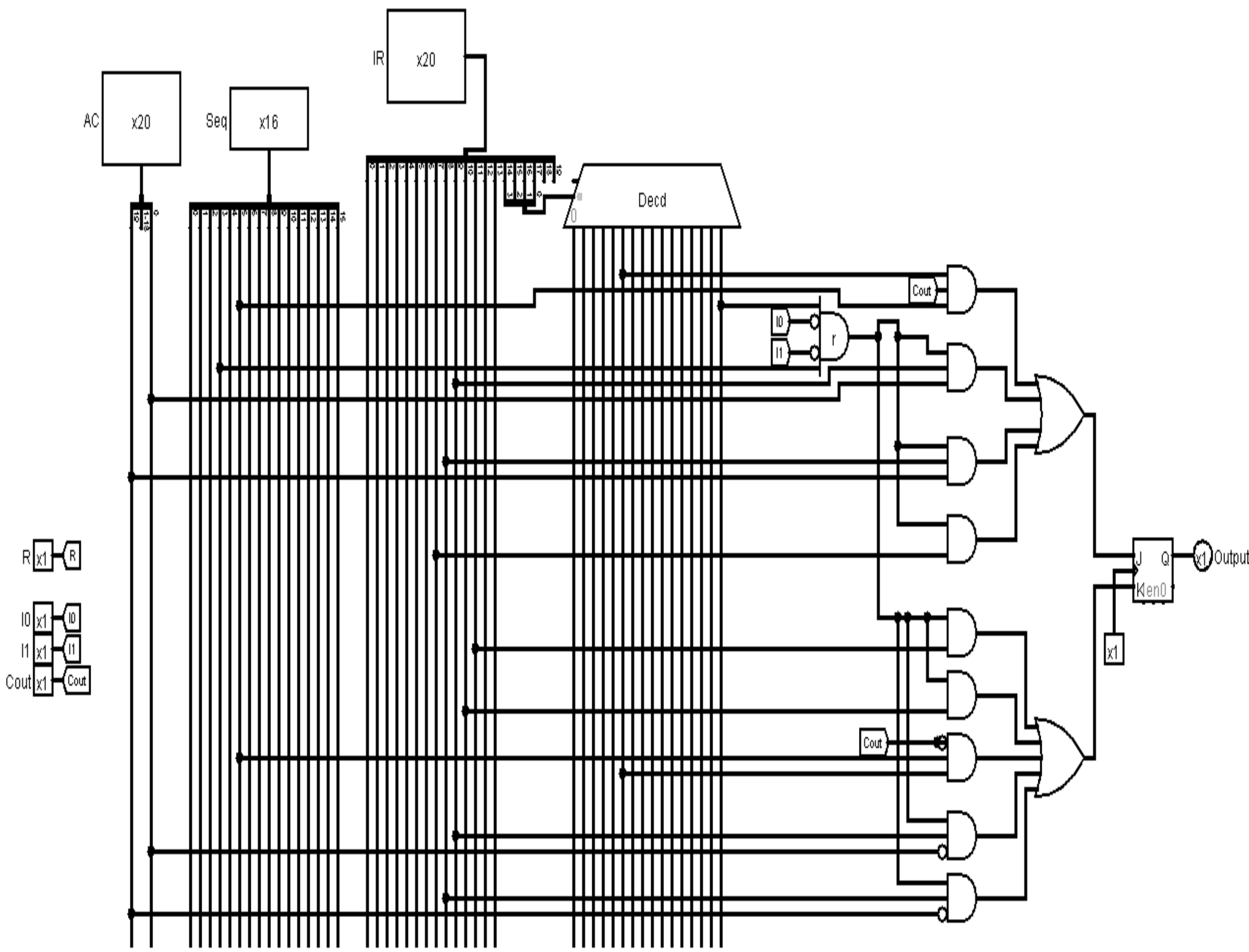

Fig 3.2.5 Design of E

3.2.6. Design of S

rB1	S←0

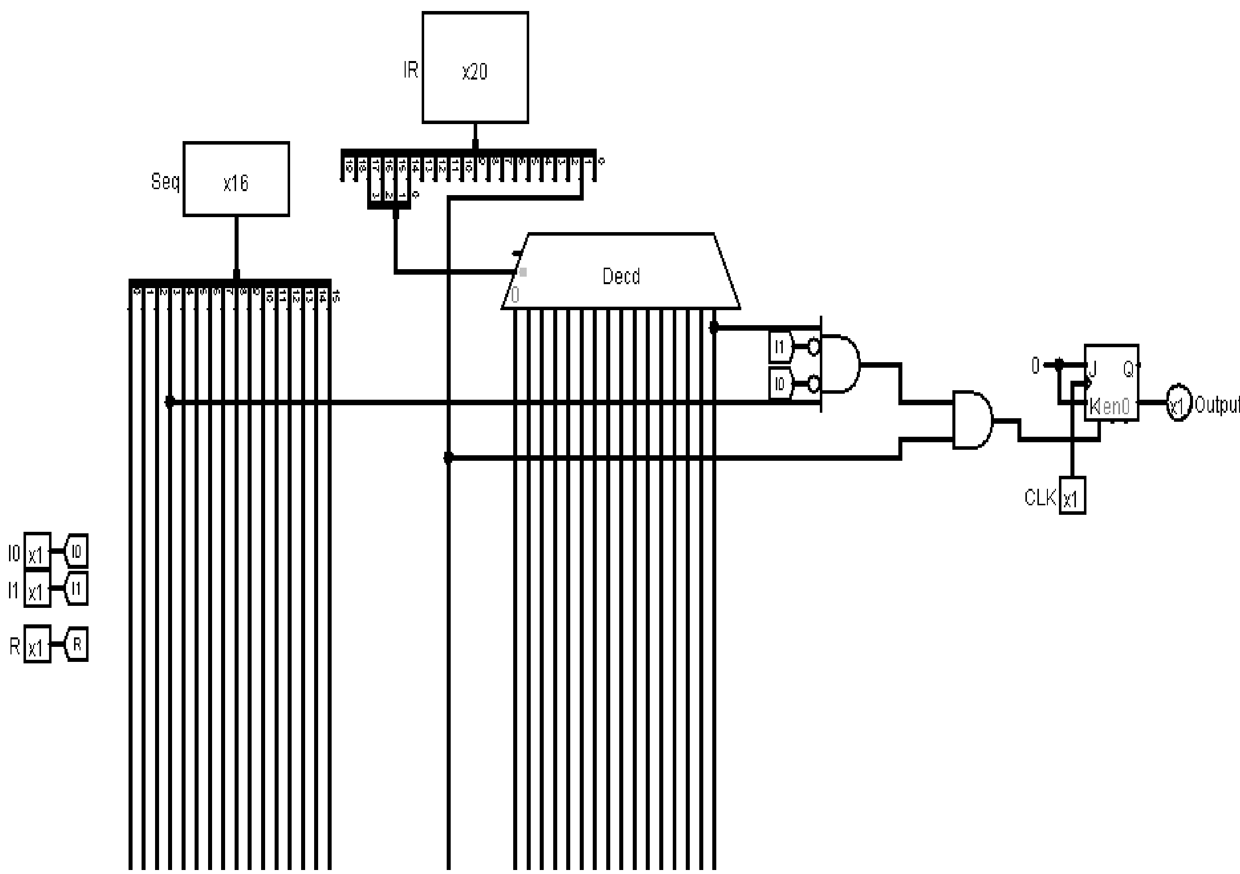

Fig 3.2.6 Design of S

3.2.7. Design of I_0

R'T_2: $I_0 \leftarrow IR(19)$

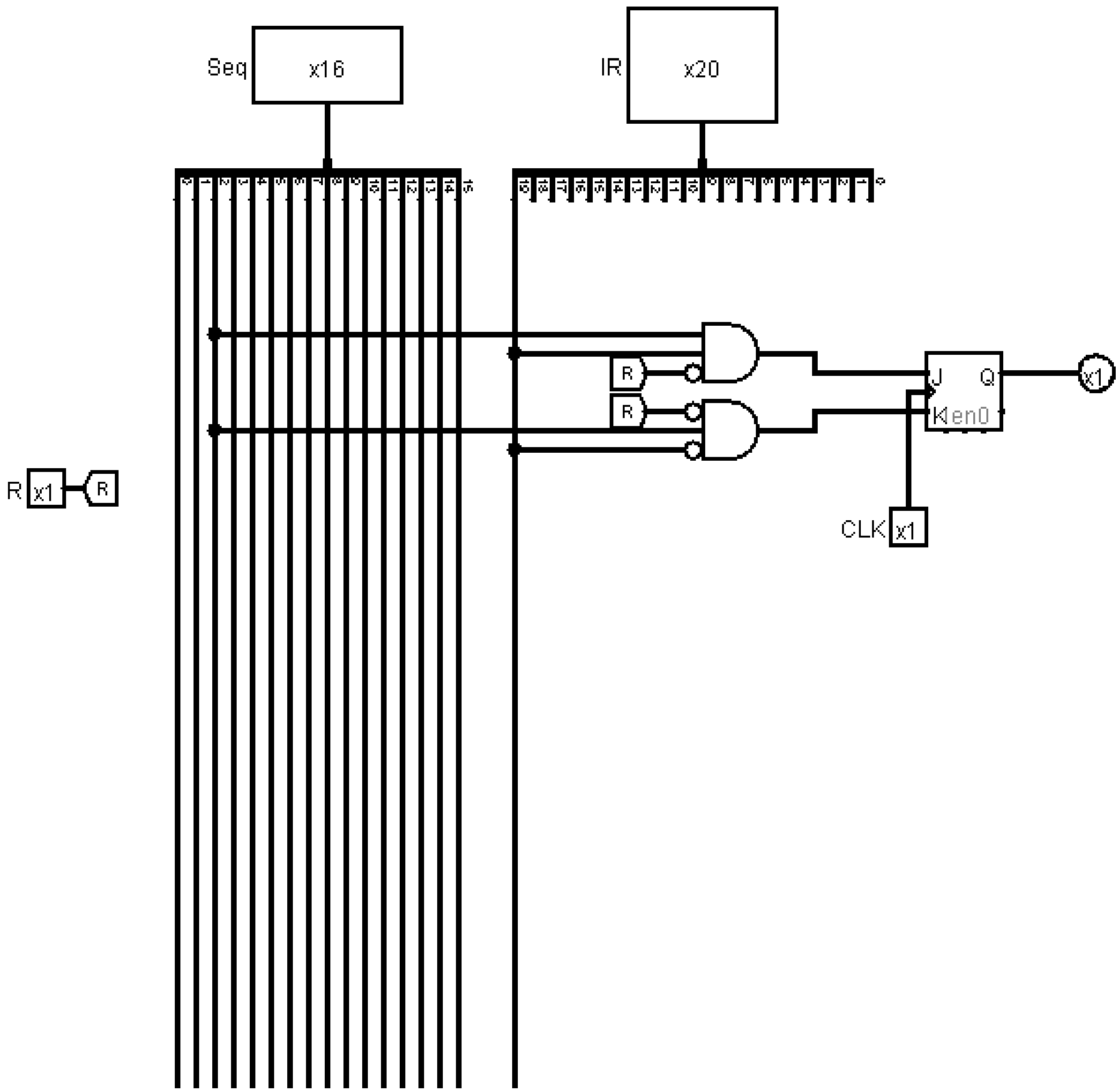

Fig 3.2.7 Design of I_0

3.2.8. Design of I_1

$R'T_2$: $I_1 \leftarrow IR(18)$

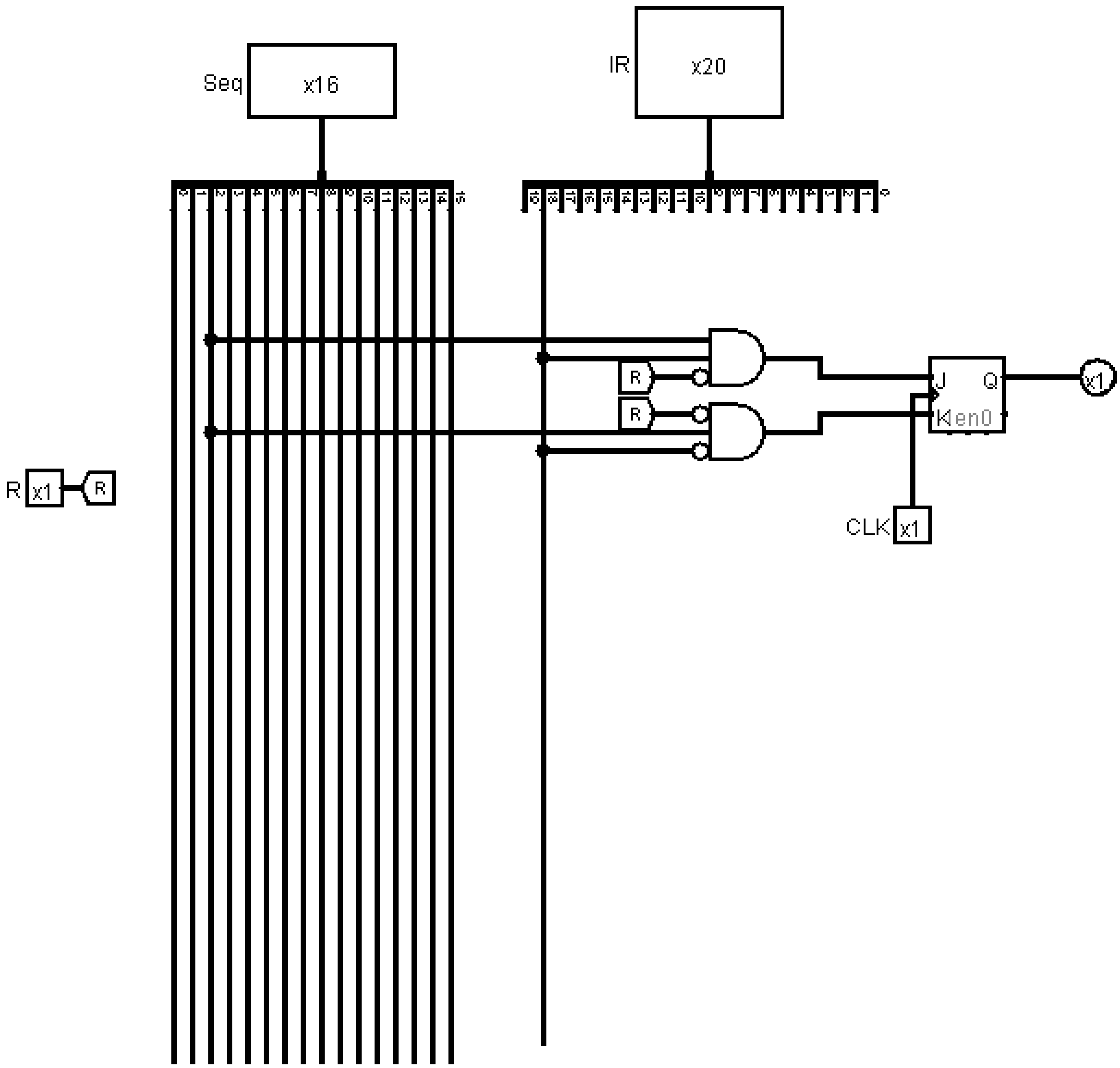

Fig 3.2.8 Design of I_1

3.3. Design of Memory

$R'T_1$:	$IR \leftarrow M[AR]$	Read
$D_{15}'.T_3.I_0'.I_1'$:	$DR \leftarrow M[AR]$	Read
$D_{15}'.T_3.I_0'.I_1$:	$AR \leftarrow M[AR]$	Read
$D_{15}'.T_4.I_0'.I_1$:	$DR \leftarrow M[AR]$	Read
RT_1 :	$M[AR] \leftarrow TR$	Write
D_9T_5	$M[AR] \leftarrow AC$	Write
$D_{11}T_5$	$M[AR] \leftarrow PC$	Write
$D_{13}T_6$	$M[AR] \leftarrow DR$	Write
$D_{14}T_6$	$M[AR] \leftarrow DR$	Write

Read Memory $= R'T_1 + D_{15}'.T_3.I_0'.I_1' + D_{15}'.T_3.I_0'.I_1 + D_{15}'.T_4.I_0'.I_1$

Write Memory $= RT_1 + D_9T_5 + D_{11}T_5 + D_{13}T_6 + D_{14}T_6$

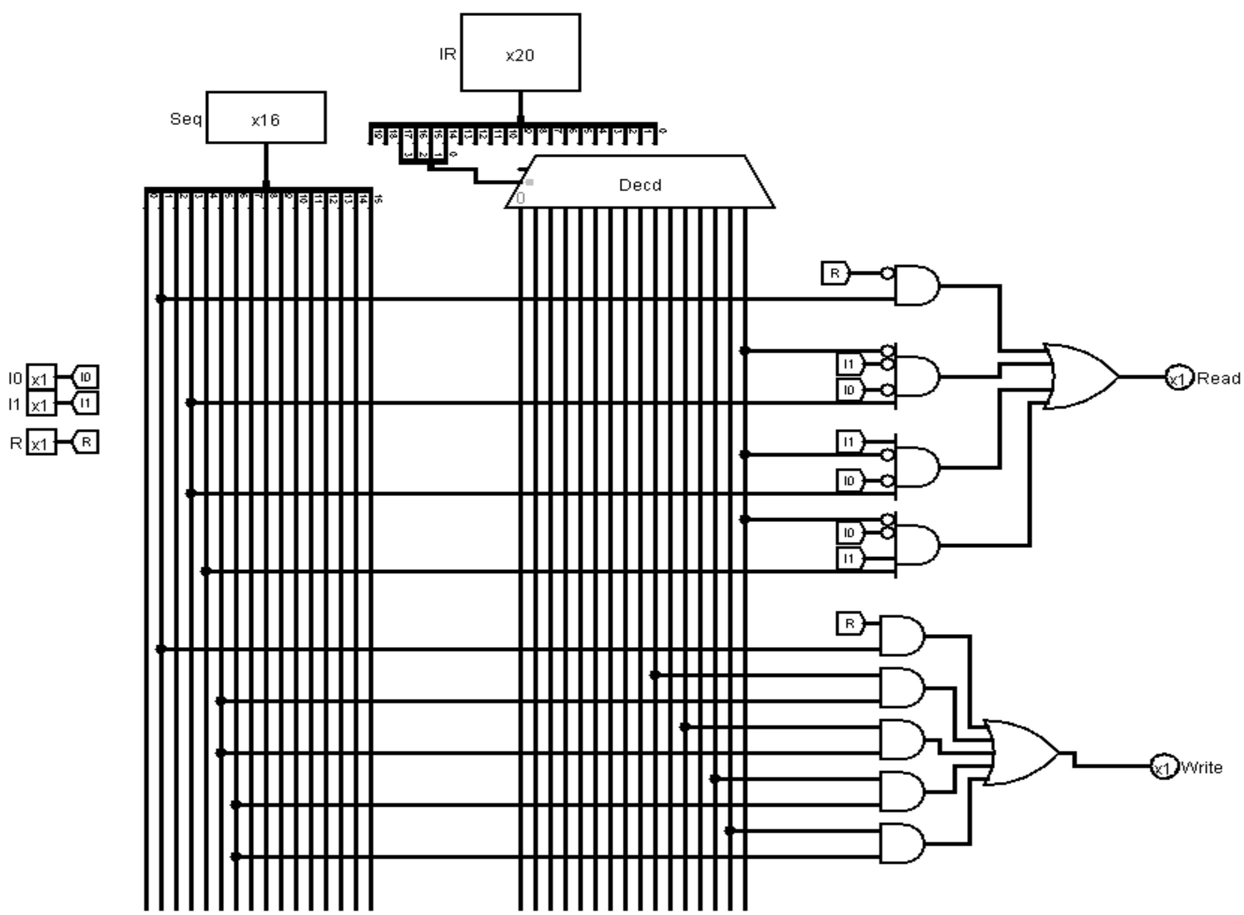

Fig 3.3 Design of Memory

3.4. Design of ALU

ALU stands for Arithmetic and Logic Unit. ALU handles all the arithmetic operations and logical operations. Arithmetic instruction includes ADD, SUB, SHL, SHR, etc and logic instruction includes AND, XOR, etc.

D_0T_5	AC←AC^DR	AND with DR
D_1T_5	AC←AC∨DR	OR with DR
D_2T_5	AC←AC⊕DR	XOR with DR
D_3T_5	AC← (AC^DR)’	NAND with DR
D_4T_5	AC←(AC∨DR)’	NOR with DR
D_5T_5	AC←AC+DR	ADD with DR
D_6T_5	AC←AC+DR’+1	SUB with DR
D_7T_5	AC←AC⊕DR	XOR with DR
D_7T_6	AC←TR	Transfer TR
D_8T_5	AC←DR	Transfer DR
$D_{12}T_6$	AC←DR	Transfer DR
rB13	AC←0	CLEAR AC
rB12	AC←AC’	Complement AC
rB9	AC← shr AC, AC(19) ←E	SHR AC
rB8	AC← shl AC, AC(0) ←E	SHL AC
rB7	AC←AC+1	Increment AC
rB6	AC←AC-1	decrement AC
pB13	AC (0-9)←INPR	INPR transfer

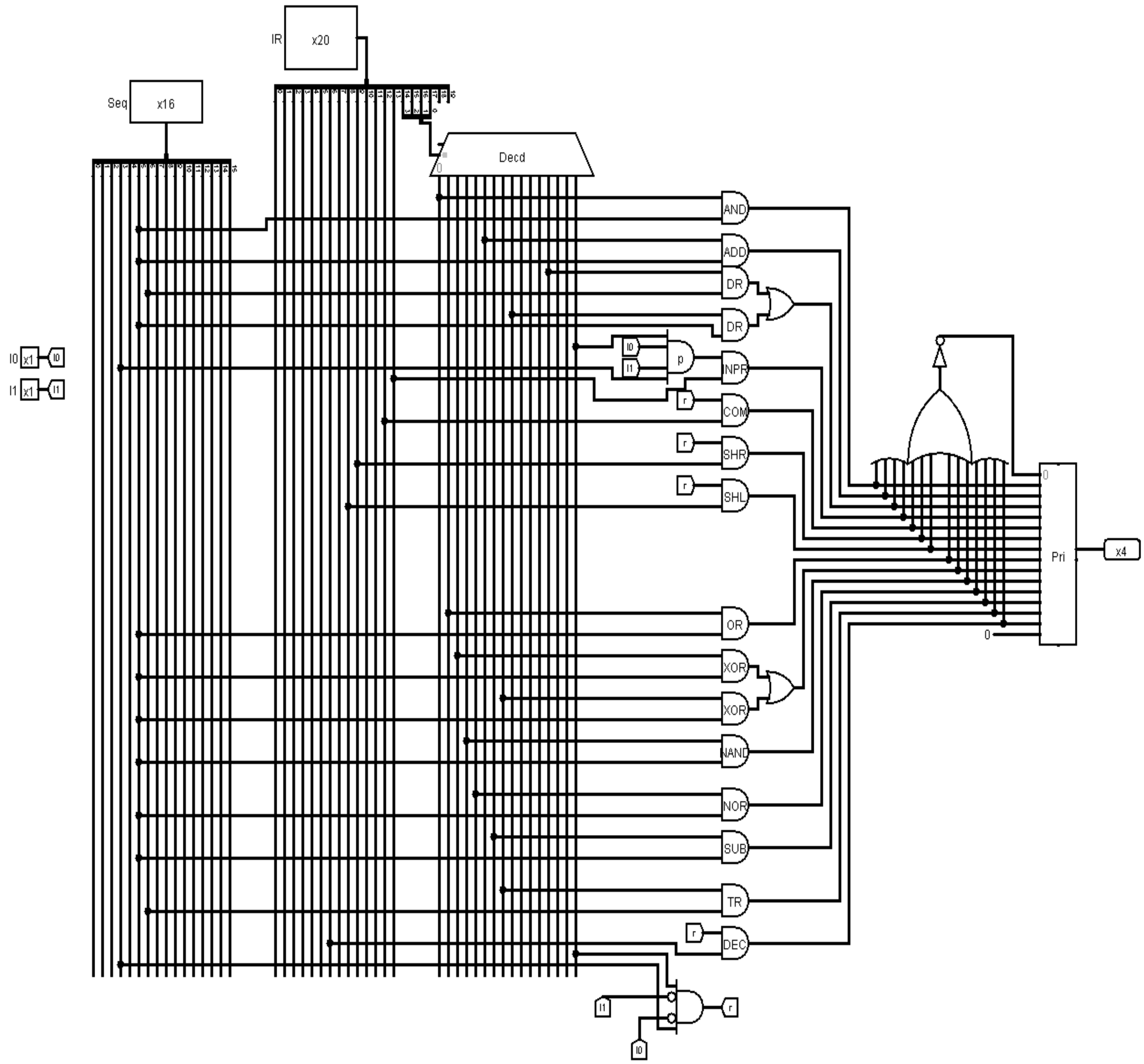

Fig 3.4.1 ALU Controller

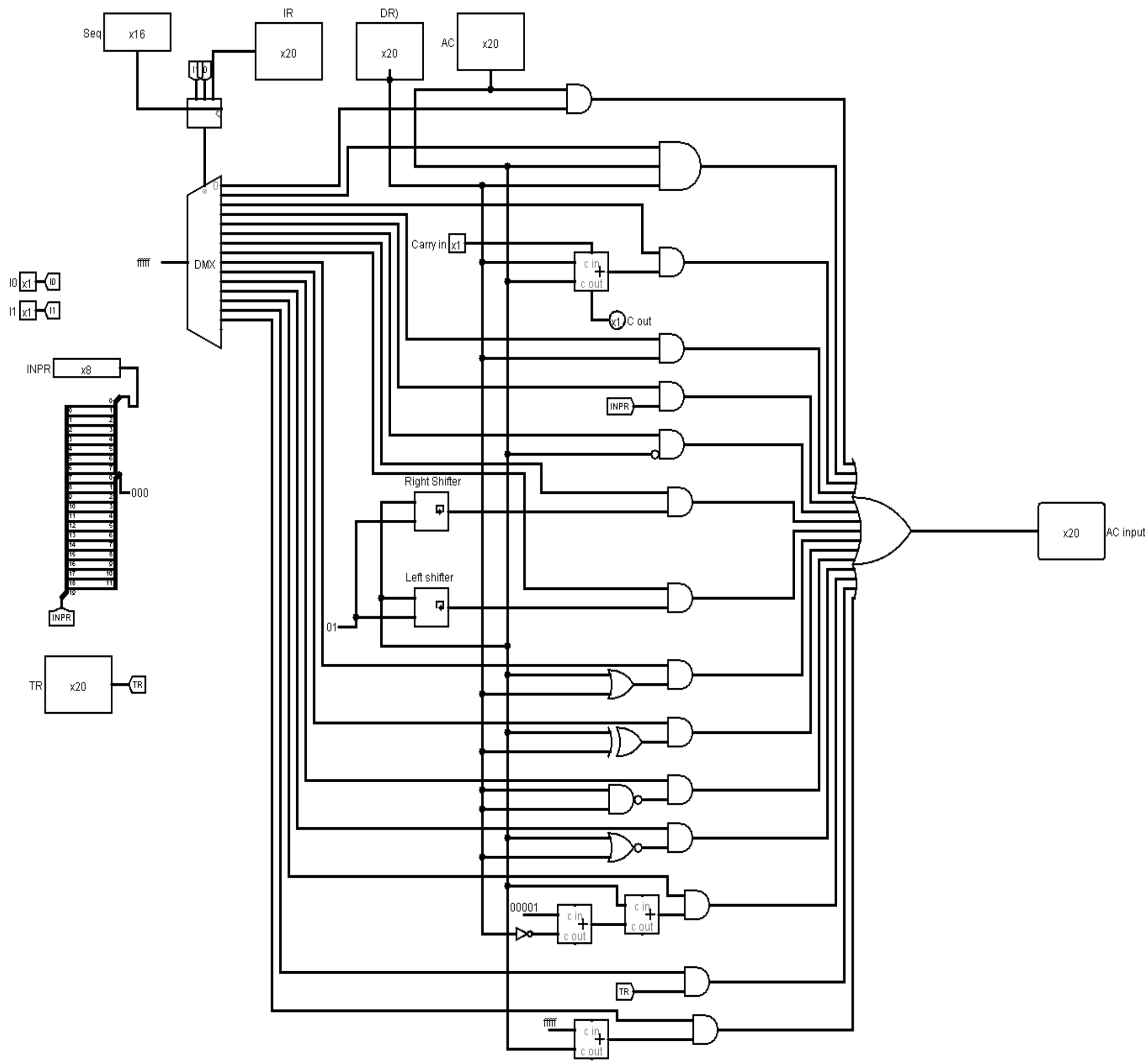

Fig 3.4.2 Design of ALU

3.5. Design of Common Bus Control

AR	$D_{15}'.T_3.I_0.I_1'$: $D_{10}T_5$ $D_{11}T_6$	$DR \leftarrow AR$ $PC \leftarrow AR$ $PC \leftarrow AR$
PC	$R'T_0$: RT_0: $D_{11}T_5$	$AR \leftarrow PC$ $TR \leftarrow PC$ $M[AR] \leftarrow PC$
DR	D_8T_5 $D_{12}T_6$ $D_{13}T_6$ $D_{14}T_6$	$AC \leftarrow DR$ $AC \leftarrow DR$ $M[AR] \leftarrow DR$ $M[AR] \leftarrow DR$
AC	D_7T_5 D_9T_5 $D_{12}T_5$ pB12	$TR \leftarrow AC$ $M[AR] \leftarrow AC$ $TR \leftarrow AC$ $OUTR \leftarrow AC$ (0-9)
IR	$R'T_2$:	$I_0 \leftarrow IR(19)$, $I_1 \leftarrow IR(18)$
TR	RT_1 : D_7T_6 $D_{12}T_7$	$M[AR] \leftarrow TR$ $AC \leftarrow TR$ $DR \leftarrow TR$
Memory	$R'T_1$: $D_{15}'.T_3.I_0'.I_1'$: $D_{15}'.T_3.I_0'.I_1$: $D_{15}'.T_4.I_0'.I_1$:	$IR \leftarrow M[AR]$ $DR \leftarrow M[AR]$ $AR \leftarrow M[AR]$ $DR \leftarrow M[AR]$

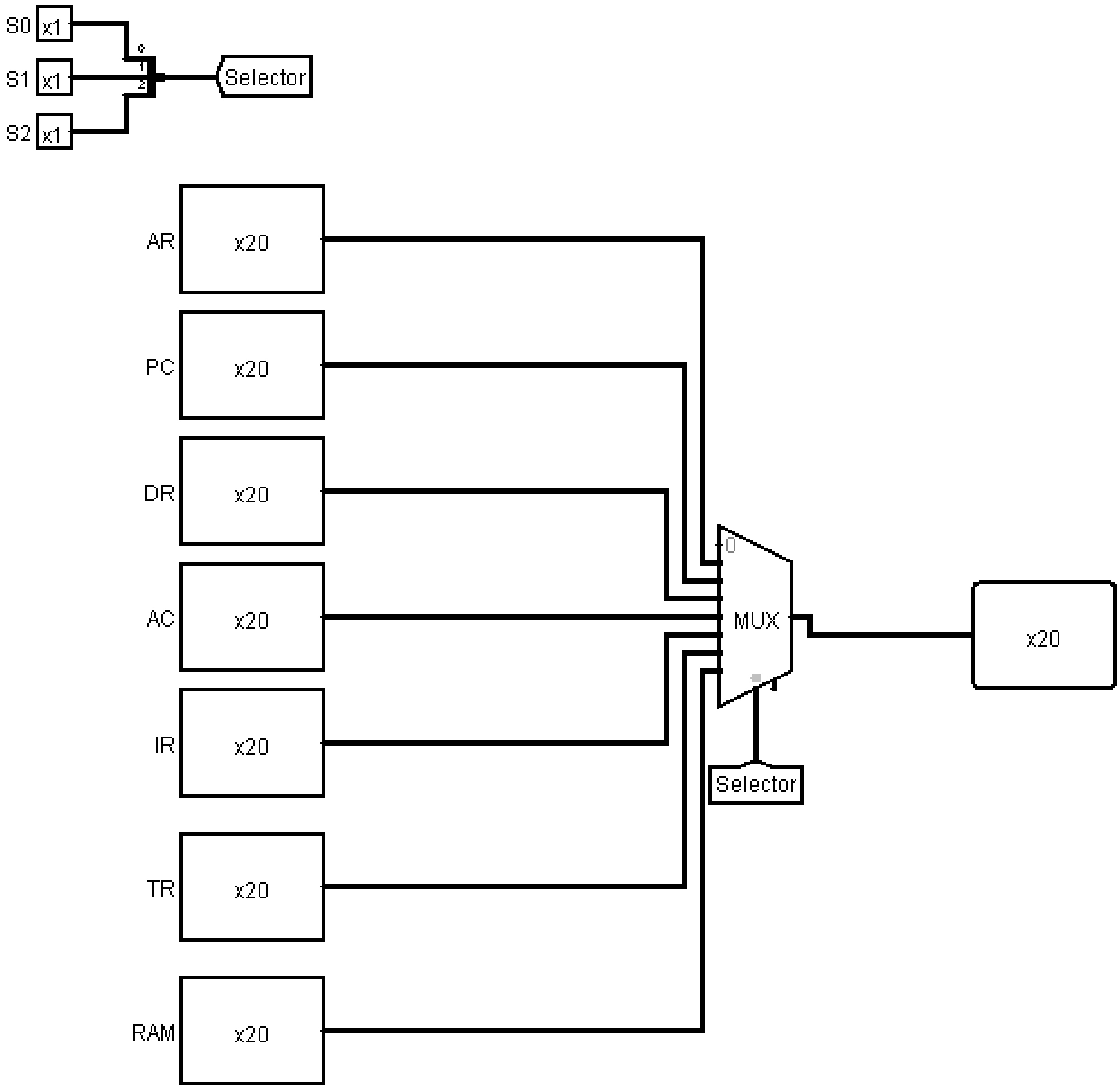

Fig 3.5.1 Design of BUS

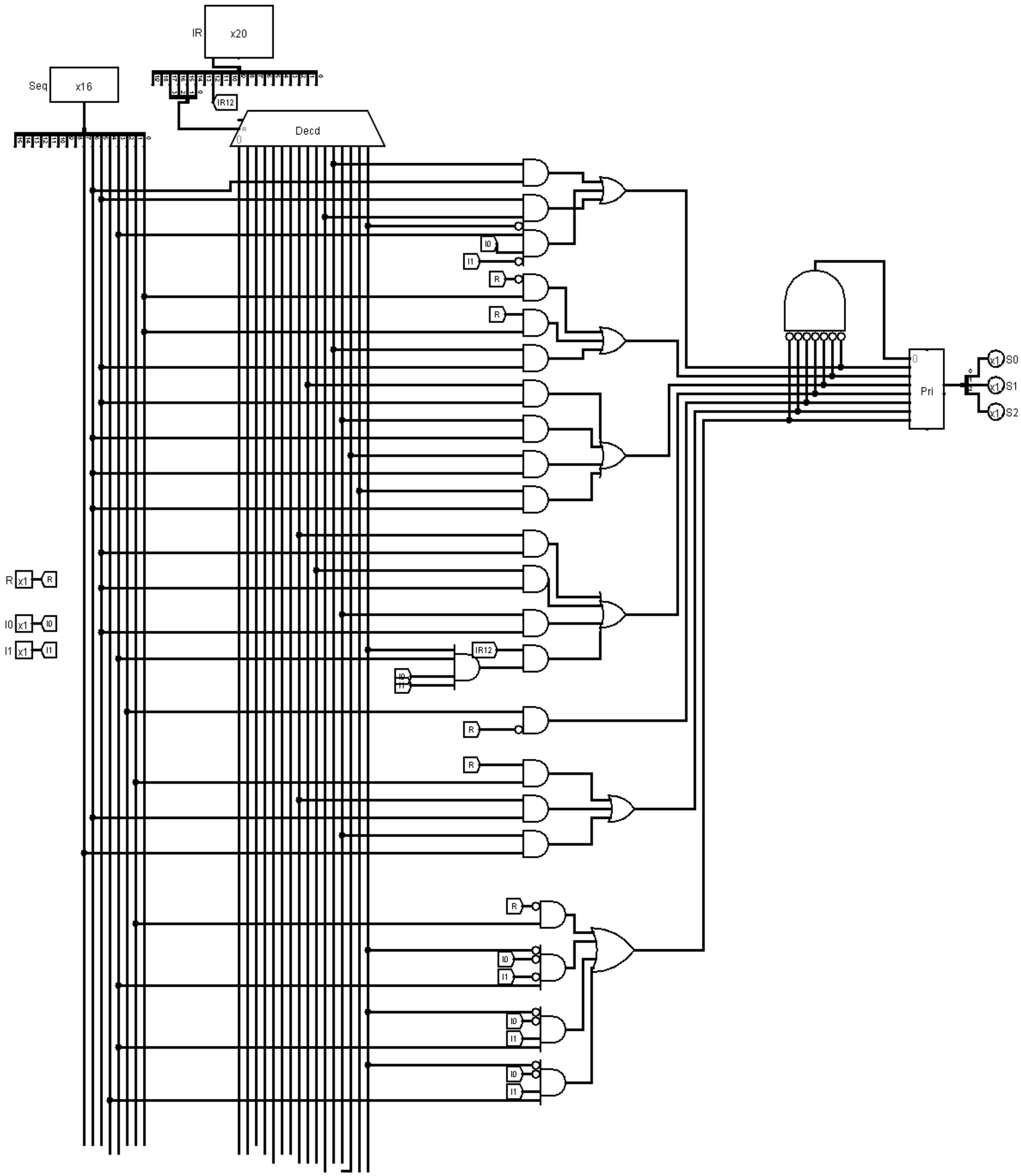

Fig 3.5.2 Control of Bus

3.6 LRN Basic Computer Diagram

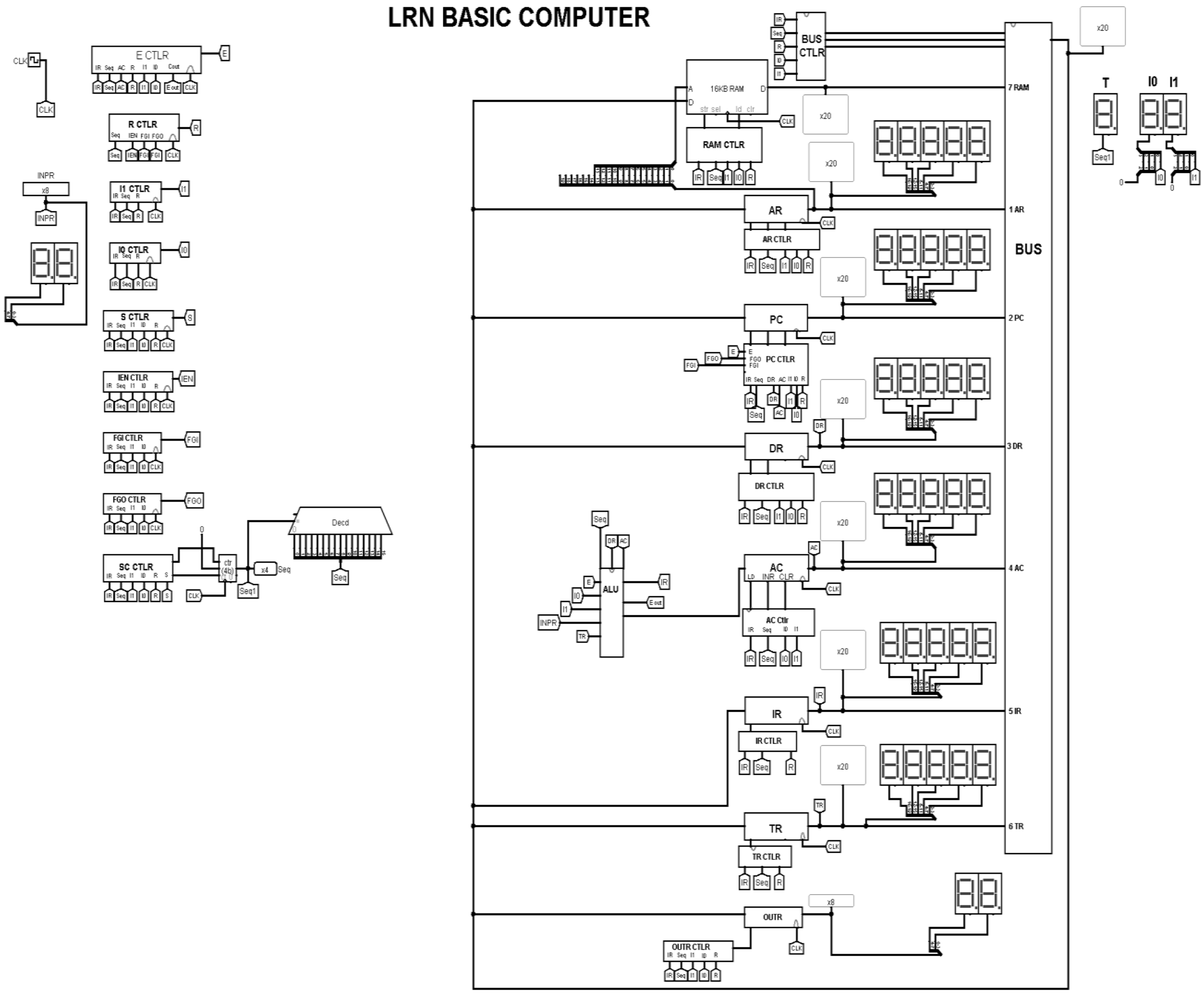

Fig 3.6 LRN Basic Computer

Chapter 4: CONCLUSION

Hence, a 16K * 20-bit LRN Computer was designed and simulated. A computer that can handle a basic computer's instruction set and some additional instructions was successfully designed using the assembly of various registers, flip-flops, and memory devices. This implementation of the computer architecture knowledge and the subsequent simulation of the design on Logisim helped to fully understand basic computer operation.

Chapter 5: REFERENCES

Mano, M. (1982). *Computer system architecture*. Englewood Cliffs, N.J.: Prentice-Hall.