Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Project Report

On

"Design of a Basic Computer: A2R-PC"

[Code No: COMP 315]

Submitted by:

Aakriti Banjara(05)

Arun Bhandari(07)

Regal Adhikari(64)

Submitted to:

Dhiraj Shrestha

Department of Computer Science and Engineering

Submission Date: 14th Jan, 2024

Table of Contents

List of Figure

Chapter 1: Introduction

The report briefs the architecture of a Basic Computer and its operations. It includes the register transfer language that is required to specify the operation given by the user to the computer. The organization of the computer has a set of registers, control structures and set of instructions that it uses. This organization uses a sequence of microoperations that it performs on the data stored in its registers in order to carry out the whole instruction. This repeatedly executes the whole program.

Every register used in the computer has its unique purpose. Besides the registers, the computer has a RAM with numerous memory words. These words store instructions, operands and data. Manipulating these data according to the instruction provided by the user is the basic objective of the computer, whose design and architecture the project focuses on.

Von Neumann architecture is based on the stored-program computer concept, where instruction data and program data are stored in the same memory. It has four main systems to it: a memory, some way to do input/output, an arithmetic/logic unit, and a control unit.

The basic computer designed in this project has the following properties:

1. The RAM (memory) contains 2^14 = 262,144 words.
2. Each word contains 18 bits.
3. Seven memory reference operations can be performed by the computer.
4. Seven computer registers for storing and processing data and addresses.

Chapter 2: Design Considerations

2.1 Instruction code

The instruction code contains two parts, Operation code (op-code) and the Address. The op-code specifies the operation of the instruction code and the address specifies the address of the operand.

I	Opcode	Address

These instruction codes are written in the memory of the computer and are nothing more than a set of binary numbers in the memory word. But the instruction code plays an important role in the computer as it describes the operation the computer needs to conduct between the given registers.

There are basically three types of instruction code in the computer.

1. Memory Reference Instruction (MRI)
2. Register Reference Instruction (RRI)
3. Input / Output Instruction (IOI)

The instruction format further differs for the different types of instruction code, as described below.

2.1.1 Memory Reference Instruction

The bits 0 to 13 give the address of the operand. The bits 14 to 16 give the binary code of the operation to be performed (op-code) and bit 17 gives the addressing mode. It is further specified that the instruction code is an MRI only if the op- code has the decimal value from 1 to 7, i.e., (001 to 111).

17 16 15 14 13 0

I	Opcode	Address

Symbol	**I = 0**	**I = 1**	**Description**
LDA	1xxx	9xxx	Load memory word to the accumulator
STA	2xxx	Axxx	Store the accumulator content to memory
ADD	3xxx	Bxxx	Add memory word to the accumulator
AND	4xxx	Cxxx	AND memory word to the accumulator
BUN	5xxx	Dxxx	Branch unconditionally
BSA	6xxx	Exxx	Branch and save return address
NAND	7xxx	Fxxx	NAND memory word to the accumulator.

Note: x denotes the hexadecimal equivalent of the address, neglecting the bits 12 and 13 of the instruction register.

2.1.2 Register Reference Instruction

The instruction code is an RRI if the bits 14 to 16 have the decimal equivalent of the value 0 (i.e., 000) and the value of bit 17 is set to 1. In this case, bits 12 and 13 are neglected, and the remaining bits 0 to 11 give the register operation code.

17 16 15 14 13 0

I	Opcode	RRI Operation

Symbol	**Instruction Code**	**Description**
CLA	8xxx (add :11)	Clear AC

CMA	8xxx (add :10)	Complement AC
INC	8xxx (add :9)	Increment AC
CIR	8xxx (add :8)	Circulate right AC and E
CIL	8xxx (add :7)	Circulate left AC and E
SPA	8xxx (add :6)	Skip next instruction if AC is positive
SNA	8xxx (add :5)	Skip next instruction if AC is negative
SZA	8xxx (add :4)	Skip next instruction if AC is zero
CLE	8xxx (add :3)	Clear E
CME	8xxx (add :2)	Complement E
SZE	8xxx (add :1)	Skip next instruction if E is positive
HLT	8xxx (add:0)	Halt computer

Note: The instruction code denotes the hexadecimal code of the RRI neglecting bits 12 and 13 of the instruction register. ('add' represents the address in the operand also shown by xxx in the Instruction Code.)

2.1.3 Input/Output Instruction

The instruction code is an IOI only if the bits 14 to 16 have the decimal equivalent of the value 0 (i.e., 000) and the value of bit 17 is set to 0. In this case, bits 12 and 13 are neglected, and the remaining bits 0 to 11 give the input / output operation codes.

17 16 15 14 13 0

I	Opcode	I/O Operation

Symbol	Instruction code	Description
INP	0xxx (add :11)	Input character to AC

OUT	0xxx (add :10)	Output character from AC
SKI	0xxx (add :9)	Skip on input flag
SKO	0xxx (add :8)	Skip on output flag
ION	0xxx (add :7)	Interrupt on
IOF	0xxx (add :6)	Interrupt off

Note: The instruction code denotes the hexadecimal code of the IOI neglecting bits 12 and 13 of the instruction register. ('add' represents the address in the operand also shown by xxx in the Instruction Code.)

2.2 Computer Resistors

The computer consists of 8 resistors. A brief description of all of them is given below.

- **Accumulator (AC)**:The accumulator is the main register of this computer. It is of 18-bits and stores the result of any operation after its completion. Input of the accumulator comes from the output of the arithmetic and logic unit . Output of the AC goes to the common bus system as well as to the ALU as the first operand, second being the Data register.
- **Data Register (DR)**: Data register is of 18-bit and is used to store the operand read from the memory that is to be processed. Output of the data register is interfaced with the ALU. Any value that is to be operated must at first be transferred to the Data register.
- **Instruction Register (IR):** Instruction Register stores the 18-bit instruction code read from the memory. Both the input and output of the instruction register come from and go to the central bus system.
- **Address Register (AR):** The address register holds the address of the operand in the memory. This address lets the specific operand come out from the memory and go to the memory. It is of 14 bits as only 14 bits are required to specify a particular address in a memory having 2^{14} registers.

- **Program Counter (PC):** Program counter contains the address of the memory location where the next instruction is located. During each execution of an instruction the value of the program counter is incremented by one pointing at the next instruction that needs to be executed. Like address register, program counter is of 14 bits.
- **Temporary Register (TR):** Temporary registers store temporary data during the working of the computer. It is 18 bits.
- **Input Register (INR):** Input register is an 8-bit register that holds the input character feed from an input device (e.g. a keyboard). Its output is connected to ALU as the value entered by the user needs to be processed with the accumulator.
- **Output Register (OUTR):** Output register is of 8-bit and holds the output character that needs to be displayed to the user. Its input is connected to the central bus system.

2.3 Instruction Cycle

The instructions in the memory need to be processed and executed. This whole process cycles until the instruction is to stop the process. This cycle is called the instruction cycle. It is divided into three parts, Fetch, Decode and Execute.

1. **Fetch:** In this cycle the content of PC is transferred to AR. Then, the content of the memory that the AR points to is transferred to IR. The value of the PC is incremented so that it points to the next instruction. The RTL for this cycle is given below.

   $R'T_0$: AR $\leftarrow$ PC
   $R'T_1$: IR $\leftarrow$ M[AR], PC $\leftarrow$ PC+1

2. **Decode:** During the decode cycle, the bit 17 is the IR is transferred to a flip flop I. The bits 15 to 17 are decoded, and the bits 0-13 are transferred to the

AR. The RTL for the microoperation is given below.

$R'T_2$: I $\leftarrow$ IR(17),

D_0-D_7 $\leftarrow$ Decode IR(14-16),

AR $\leftarrow$ IR(0-13)

$R'T_3I$: AR $\leftarrow$ M[AR]

3. **Execute:** This cycle differs for different types of instructions. In this cycle the instruction is executed accordingly. The complete set of RTL and the control functions for different types of instructions will be listed later in the report.

2.4 Interrupt cycle

During each of the above described cycles, the computer keeps checking for an input to be given by the user. For this the computer needs to check for a set in the input flag (FGI). When it finds one it leaves the task it is performing and initiates information transfer. The difference in information flow rate between the computer and that of the input-output devices makes the type of transfer inefficient. As an alternative, we allow the external device to inform the computer when it is ready for the transfer. This allows the computer to do some other work in the meantime.

The interrupt enable flip-flop IEN is used to allow a programmer to choose whether to allow an interrupt during the program or not. During the interrupt cycle, the return address is saved in a specific memory address. The program then branches to the input/output program in the memory and after its execution returns back to the program where it was before interruption. Until the interrupt cycle is executed no further interrupts are allowed to occur.

2.5 Common Bus System

It is an 18 bit common bus system that connects all the registers to the memory. The input of the bus is controlled by three selection lines. The following figure is a block diagram of the common bus system of the computer.

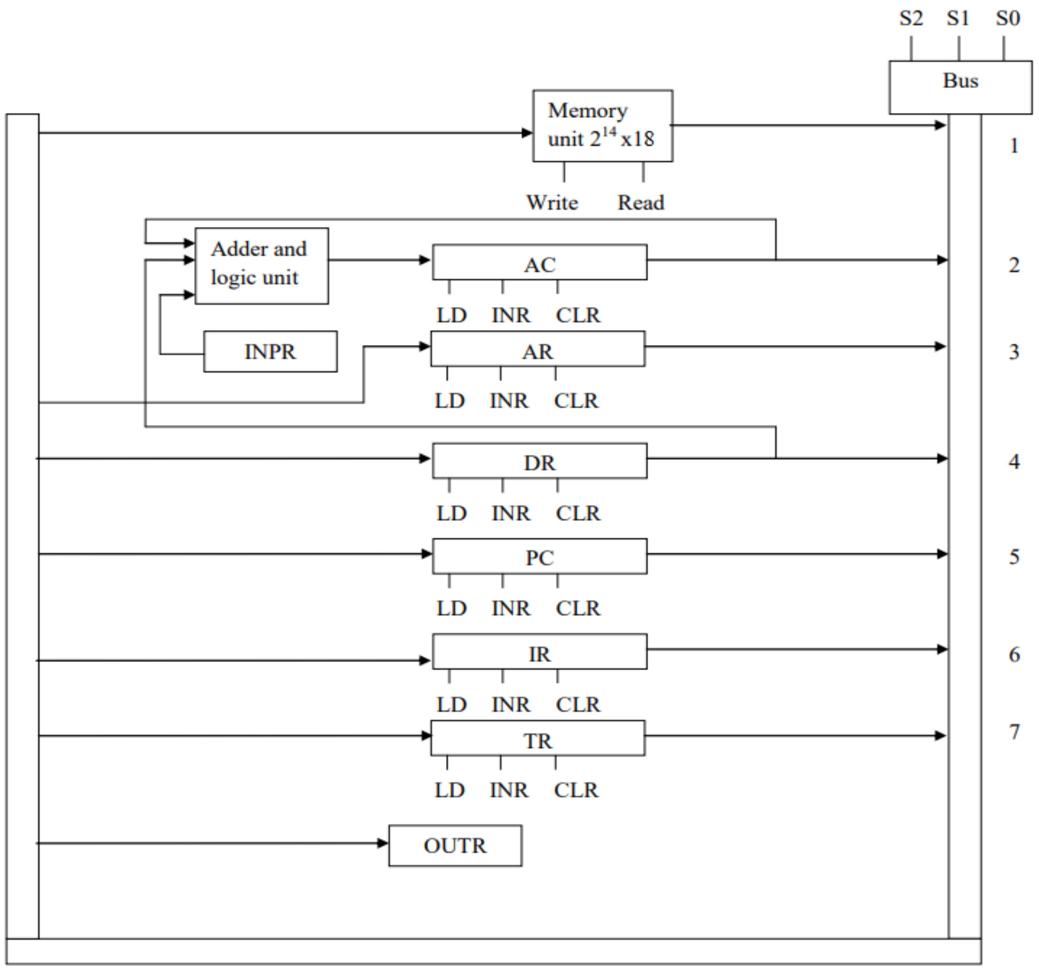

Figure 1:Block diagram of the Common Bus System

2.6 RTL and Control Signals

The complete set of RTLs and their control signals are given below.

Fetch:

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

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

Decode:

$R'T_2$: $I \leftarrow IR(17)$, $D0\text{-}D7 \leftarrow$ Decode $IR(14\text{-}16)$, $AR \leftarrow IR(0\text{-}13)$

$D_0'T_3I$: $AR \leftarrow M[AR]$

Interrupt:

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

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

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

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

Memory Reference Instruction:

LDA: D_1T_4: $DR \leftarrow M[AR]$

D_1T_5: $AC \leftarrow DR$, $SC \leftarrow 0$

STA: D_2T_4: $M[AR] \leftarrow AC$, $SC \leftarrow 0$

ADD: D_3T_4: $DR \leftarrow M[AR]$

D_3T_5: $AC \leftarrow AC+DR$, $E \leftarrow C_{out}$, $SC \leftarrow 0$

AND: D_4T_4: $DR \leftarrow M[AR]$

D_4T_5: $AC \leftarrow AC \wedge DR$, $SC \leftarrow 0$

BUN: D_5T_4: $PC \leftarrow AR$, $SC \leftarrow 0$

BSA: D_6T_4: $M[AR] \leftarrow PC$, $AR \leftarrow AR+1$

D_6T_5: $PC \leftarrow AR$, $SC \leftarrow 0$

NAND: D_7T_4: DR ← M[AR]

D_7T_5: AC ← AC^DR

D_7T_6: AC ← AC', SC ← 0

Register Reference Instruction:

$D_0IT_3 = r$

IR(i)=B(i) (i=0,1,2,…,11)

r:SC ← 0

CLA: rB_{11}: AC ← 0

CMA: rB_{10}: AC ← AC'

INC: rB_9: AC ← AC+1

CIR: rB_8: AC ← shr AC, AC(15) ← E, E ← AC(0)

CIL: rB_7: AC ← shl AC, AC(0) ← E, E ← AC(15)

SPA: rB_6: if(AC(15)=0) then (PC ← PC+1)

SNA: rB_5: if(AC(15)=1) then (PC ← PC+1)

SZA: rB_4: if(AC=0) then (PC ← PC+1)

CLE: rB_3: E ← 0

CME: rB_2: E ← E'

SZE: rB_1: if(E=0) then (PC ← PC+1)

HLT: rB_0: S ← 0

Input/ Output Instruction:

$D_0I'T_3 = p$

IR(i) = B(i) (i=6,7,8,9,10,11,)

p:SC ← 0

INP: pB_{11}: AC(0-7) ← INPR, FGI ← 0

OUTR: pB_{10}:OUTR ← AC(0-7), FGO ← 0

SKI: pB_9: if(FGI=1) then (PC ← PC+1)

SKO: pB_8: if(FGO=0) then (PC ← PC+1)

ION: pB_7:IEN ← 1

IOF: pB_6:IEN $\leftarrow$ 0

The following figure is the flow chart of a program including the interrupt cycle.

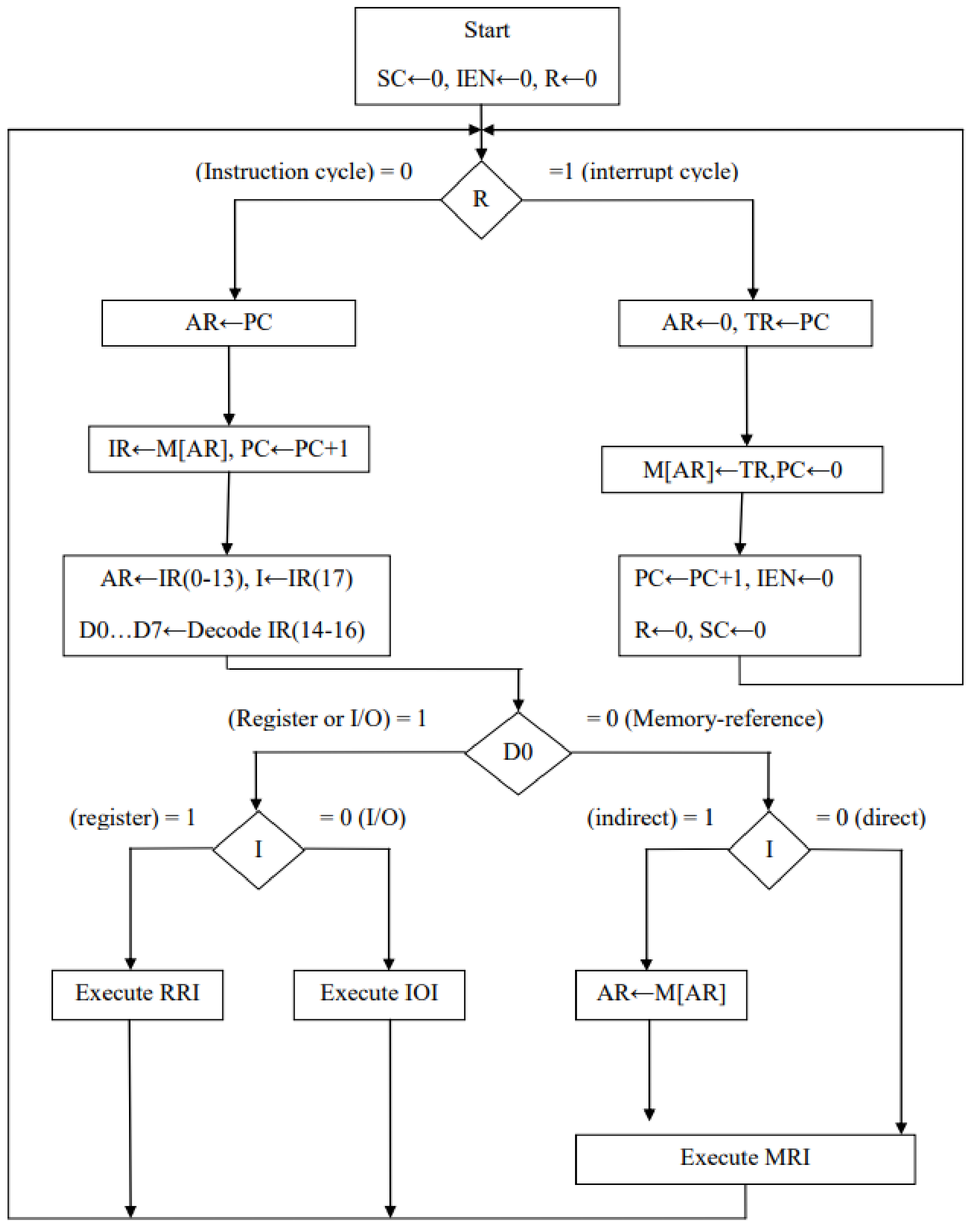

Figure 2: Program Flow Chart

2.7 Control Unit:

The control unit consists of several control gates that control the register transfers in order to carry out the specified operation in the processing unit. The following figure shows the block diagram of the control unit along with its inputs from the Instruction Register and the sequence counter.

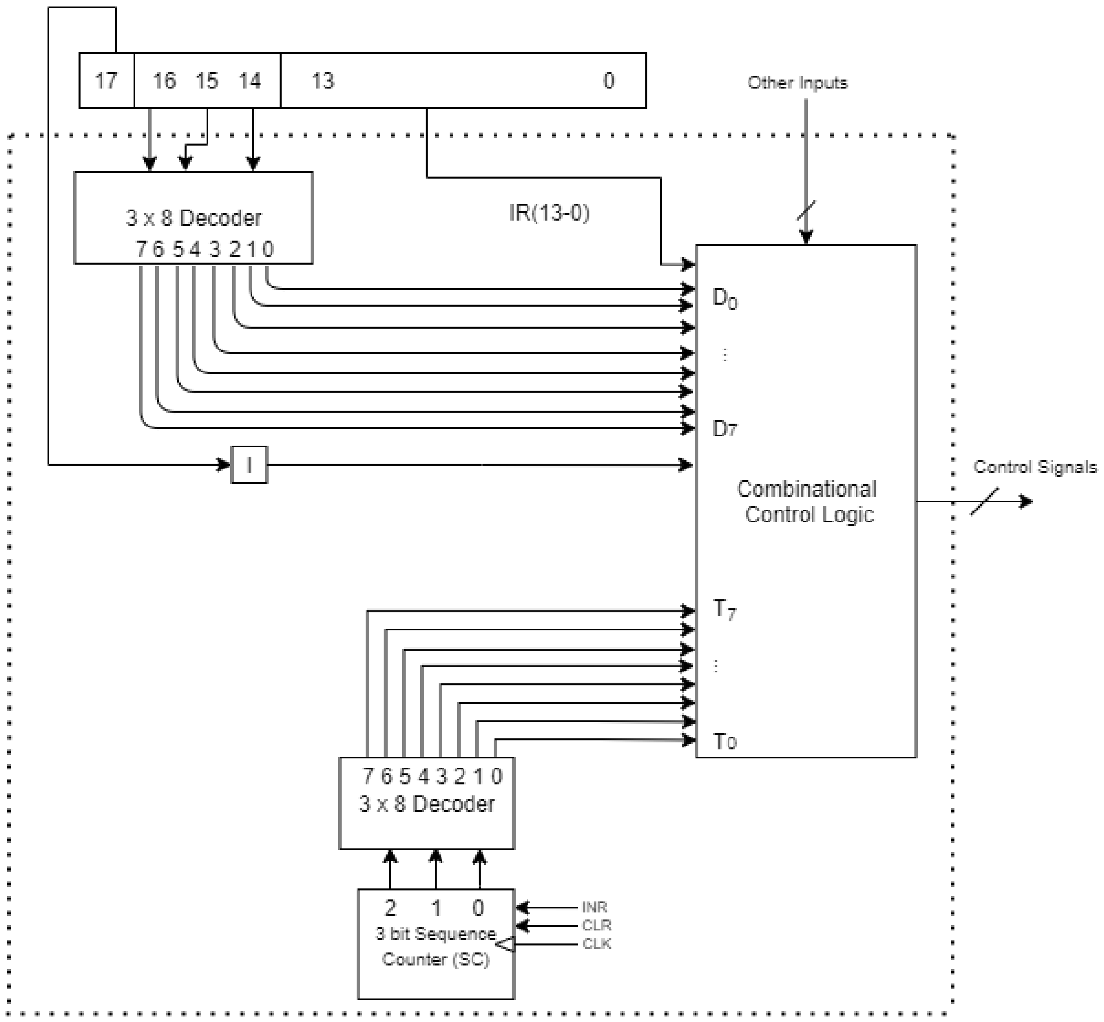

Figure 3: Block Diagram of the Control Unit

Chapter 3: Design of Individual Unit

3.1 Control Logic Gates:

An individual control gate unit is associated with an individual pin of a register or a flag. The control gate is built from the control functions and is the hardware implementation of the above given functions.

AR:

LD(AR) = $R'T_0 + R'T_2 + D_0'T_3I$

INR(AR) = D_6T_4

CLR(AR) = RT_0

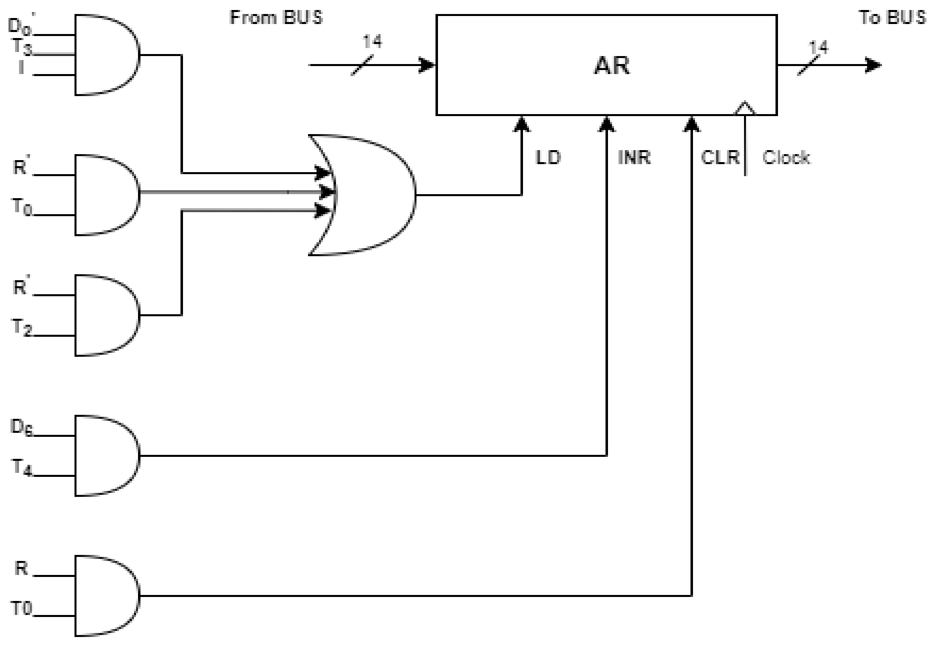

Figure 4: Control Circuit for AR

Memory:

Read = $R'T_1+D_0'T_3I+D_1T_4+D_3T_4+D_4T_4+D_7T_4$

Write = $RT_1+ D_2T_4+ D_6T_4$

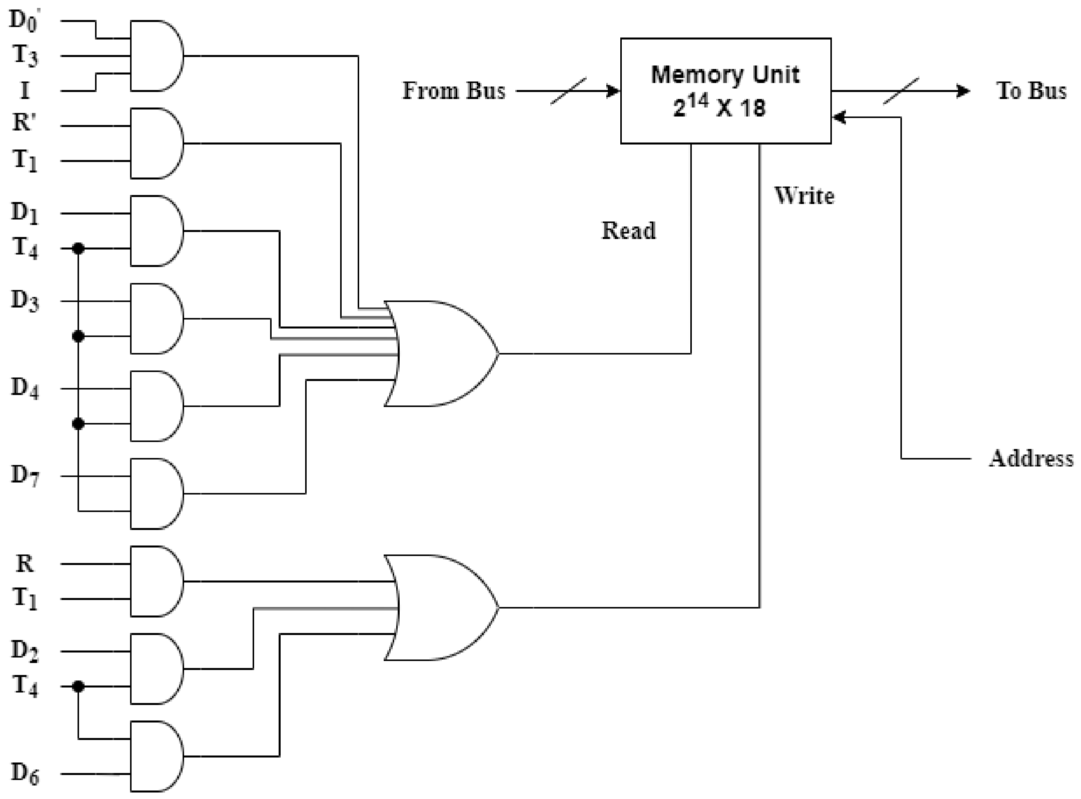

Figure 5: Control Circuit for Memory

AC:

LD= $D_1T_5 + D_3T_5 + D_4T_5 + D_7T_5 + D_7T_6 + rB_{10} + rB_8 + rB_7 + pB_{11}$

INR= rB_9

CLR= rB_{11}

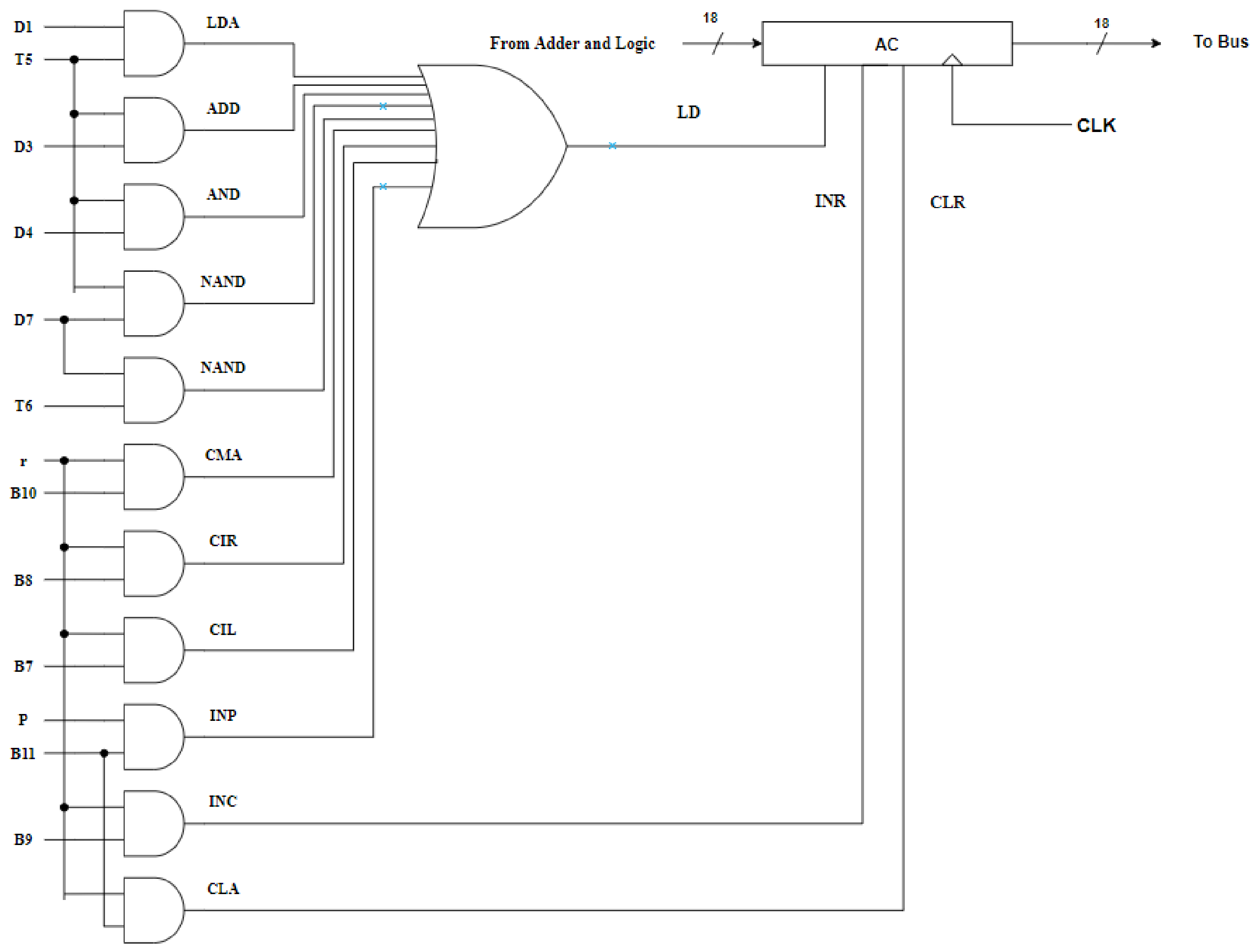

Figure 6: Control Circuit for AC

DR:

$LD = D_1T_4 + D_3T_4 + D_4T_4 + D_7T_4$

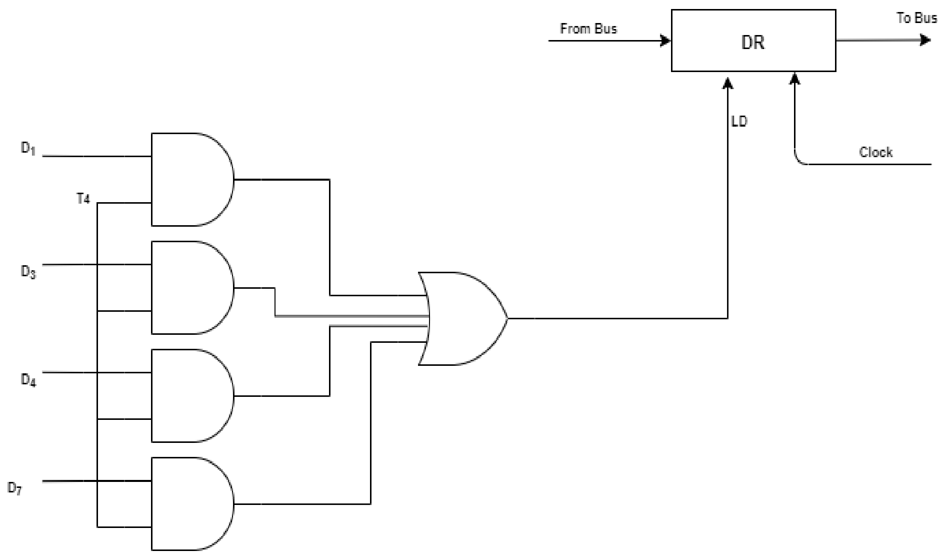

Figure 7: Control Circuit for DR

IR:

$LD = R'T_1$

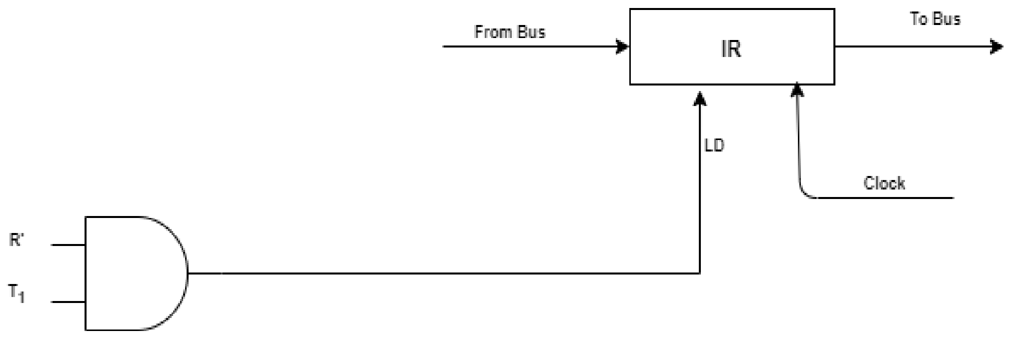

Figure 8: Control Circuit for IR

TR:

LD = RT_0

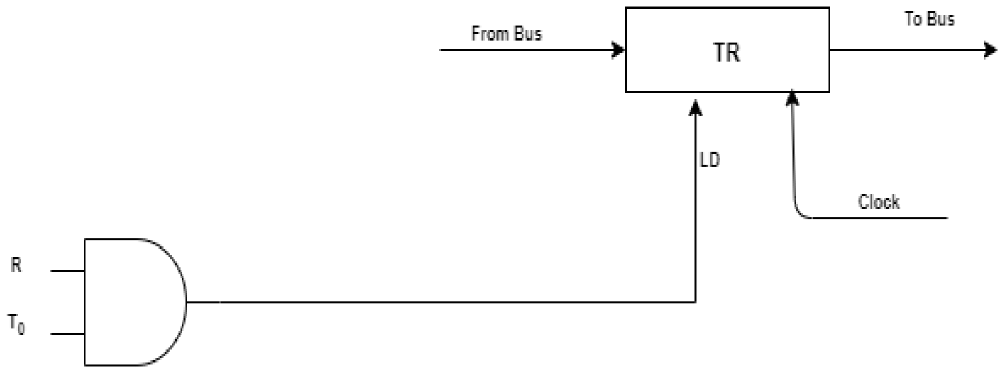

Figure 9: Control Circuit for TR

PC:

$LD = D_5T_4 + D_6T_5$

$INR = R'T_1+RT_2+rB_6AC(15)+rB_5AC(15)+rB_4AC(0)+rB_1E$

$CLR = RT_1$

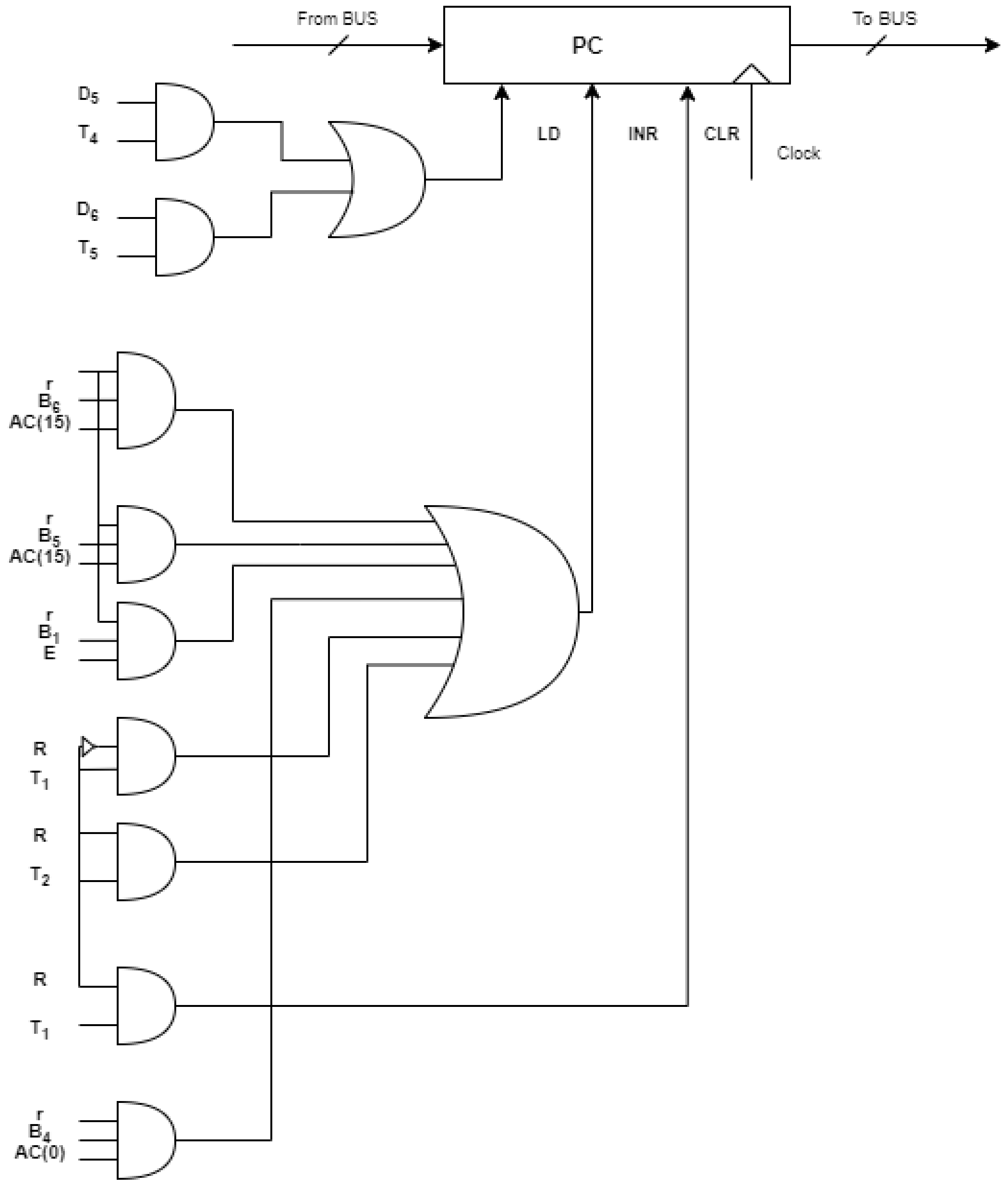

Figure 10: Control Circuit for PC

OUTR:

LD= pB_{10}

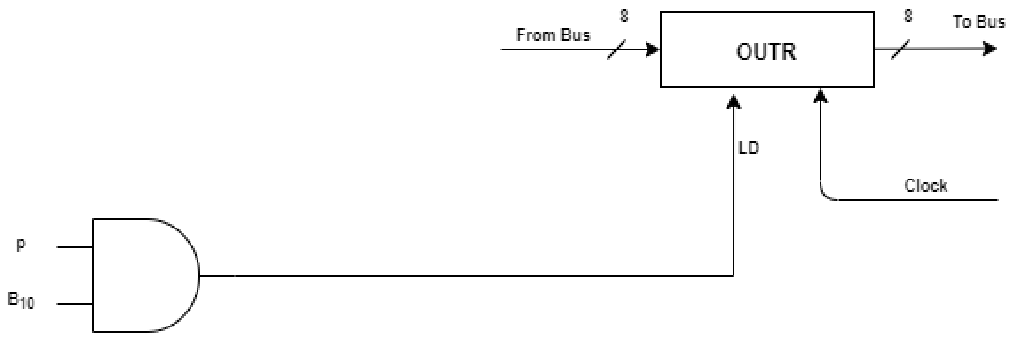

Figure 11: Control Circuit for OUTR

SC:

CLR= $RT_2+ D_1T_5+ D_2T_4+ D_3T_5+D_4T_5+D_5T_4+ D_6T_5+ D_7T_6+p+r$

INR = CLR’

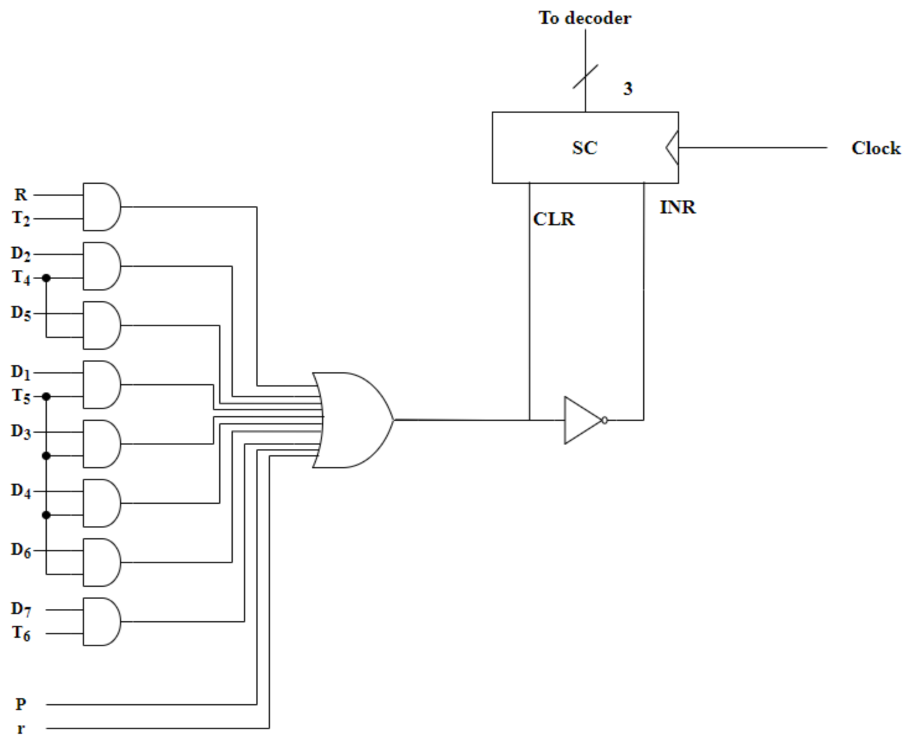

Figure 12: Control Circuit for SC

IEN:

RESET: $RT_2 + pB_6$

SET: pB_7

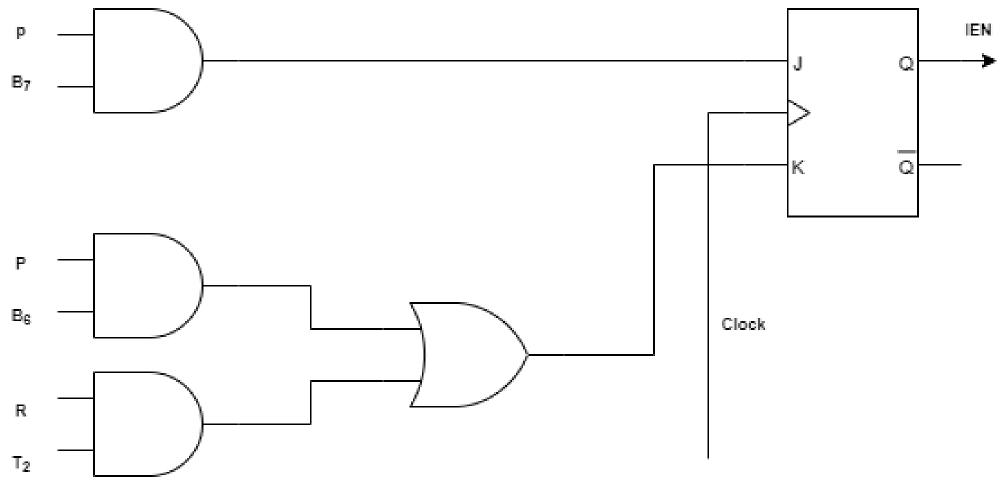

Figure 13: Control Circuit for IEN

R:

SET = $T_0'T_1'T_2'$(IEN)(FGO+FGI)

RESET = RT_2

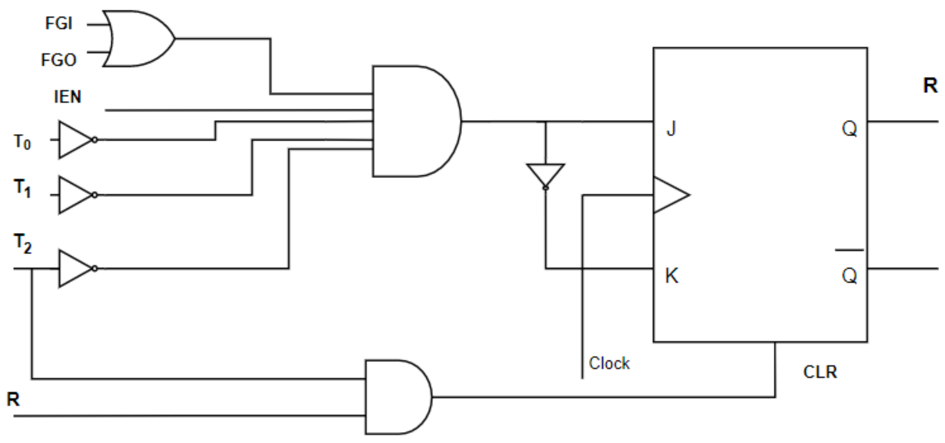

Figure 14:Control Circuit for R

FGO:

RESET = pB_{10}

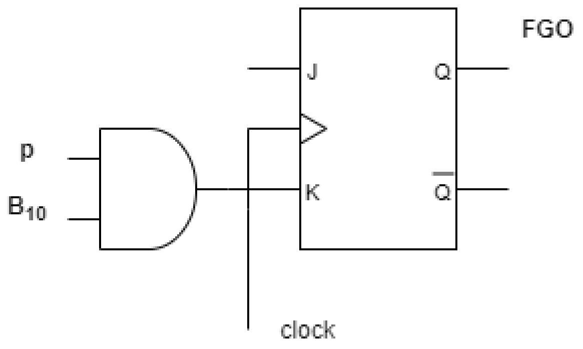

Figure 15: Control Circuit for FGO

FGI:

RESET = pB_{11}

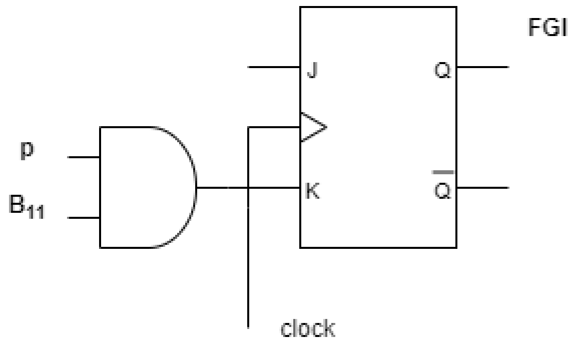

Figure 16: Control Circuit for FGI

3.2 Decoder Design for the Operation Code (3x8):

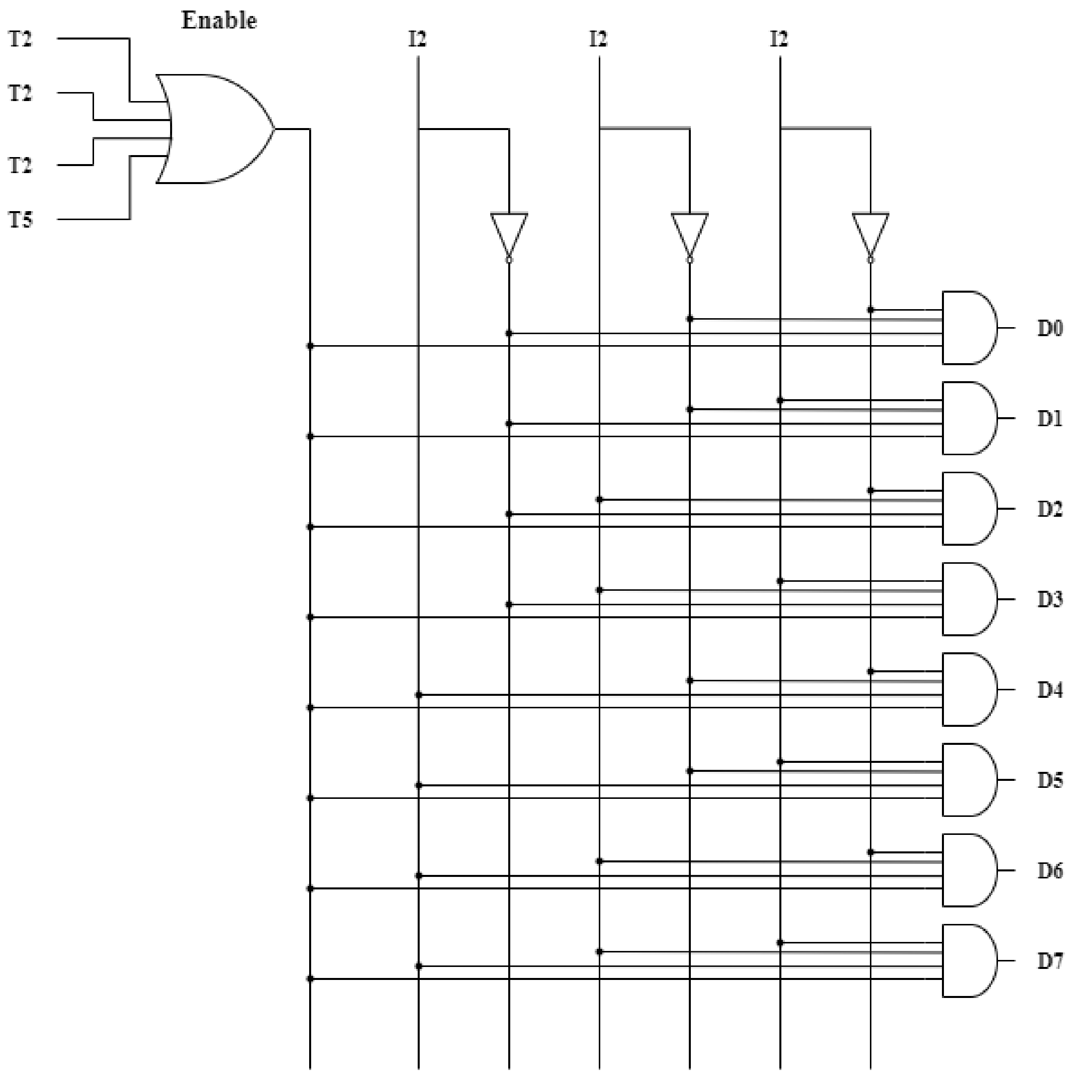

Figure 17: Decoder Circuit

3.3 Control of Common Bus

The bus line is controlled by the multiplexer selection lines that come from the output of an 8*3 encoder. The selection line selects amongst the register or the memory for the input. The outputs of the encoder depend on its 8 input lines. The following table specifies the binary number for which a particular register is selected.

Inputs							**Outputs**			**Register for selected Bus**
X_1	X_2	X_3	X_4	X_5	X_6	X_7	S_1	S_2	S_3	
0	0	0	0	0	0	0	0	0	0	None
1	0	0	0	0	0	0	0	0	1	Memory
0	1	0	0	0	0	0	0	1	0	AC
0	0	1	0	0	0	0	0	1	1	AR
0	0	0	1	0	0	0	1	0	0	DR
0	0	0	0	1	0	0	1	0	1	PC
0	0	0	0	0	1	0	1	1	0	IR
0	0	0	0	0	0	1	1	1	1	TR

The control functions for the individual inputs of the encoder are given below:

$X_1 = R'T_1+D_0'T_3I+D_1T_4+D_3T_4+D_4T_4+D_7T_4$

$X_2 = D_2T_4$

$X_3 = D_5T_4+D_6T_5$

$X_4 = D_1T_5$

$X_5 = R'T_0+RT_0+D_6T_4$

$X_6 = R'T_2$

$X_7 = RT_1$

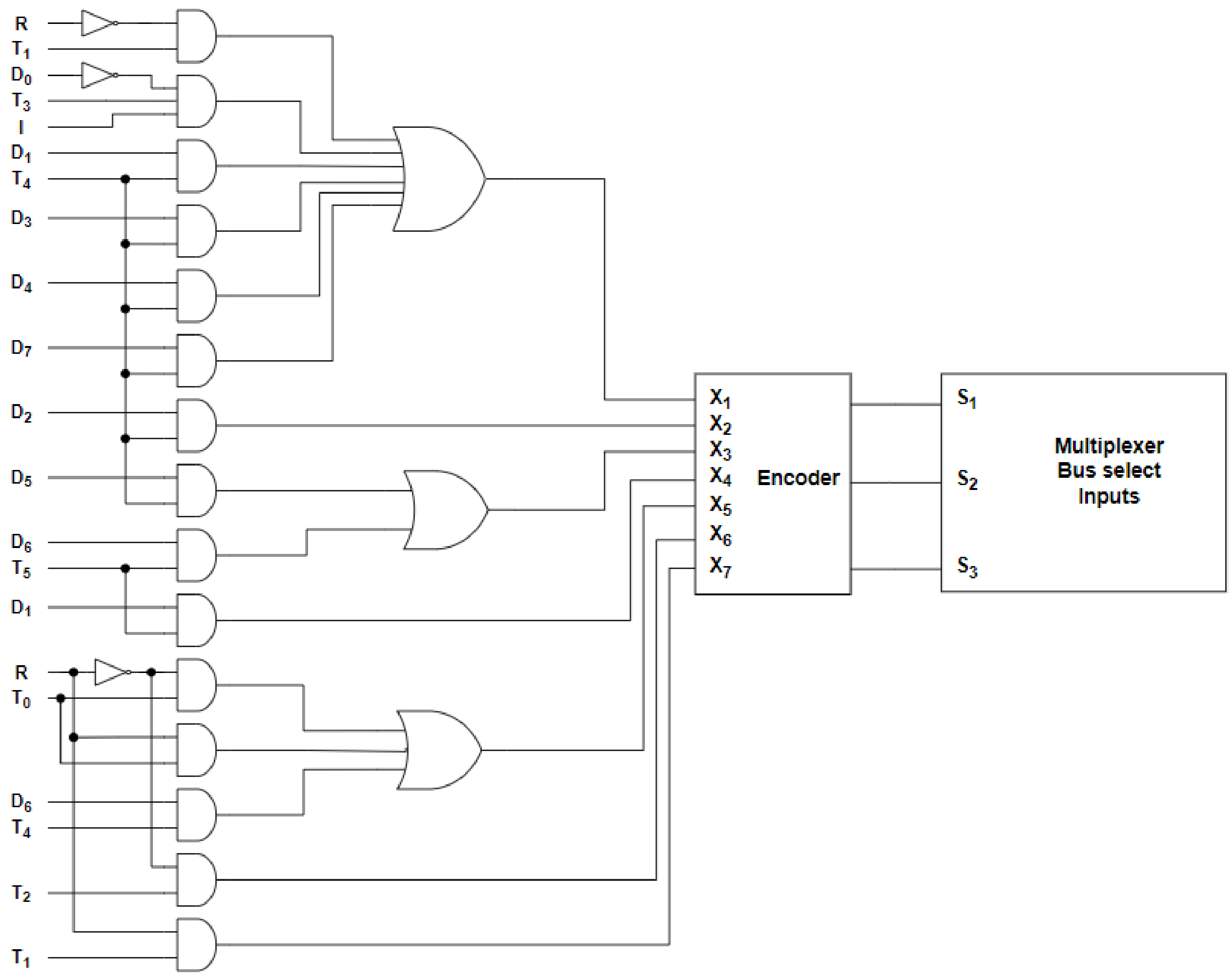

Figure 18: Control Circuit for CBS

3.4 Arithmetic and Logic Unit

As the name suggests, this unit performs all the arithmetic and logical calculations in the computer. Its output is directly connected to the input of the accumulator and therefore is of 18 bits in the computer design. Its inputs are connected to the outputs of the Data Register, Accumulator and the Input Register. The solution obtained from the calculation in the ALU is sent to the Accumulator, so it is from here that the output goes to the bus to its destination (i.e, Memory or the OUTR). A detailed design of a bit of the ALU is given below.

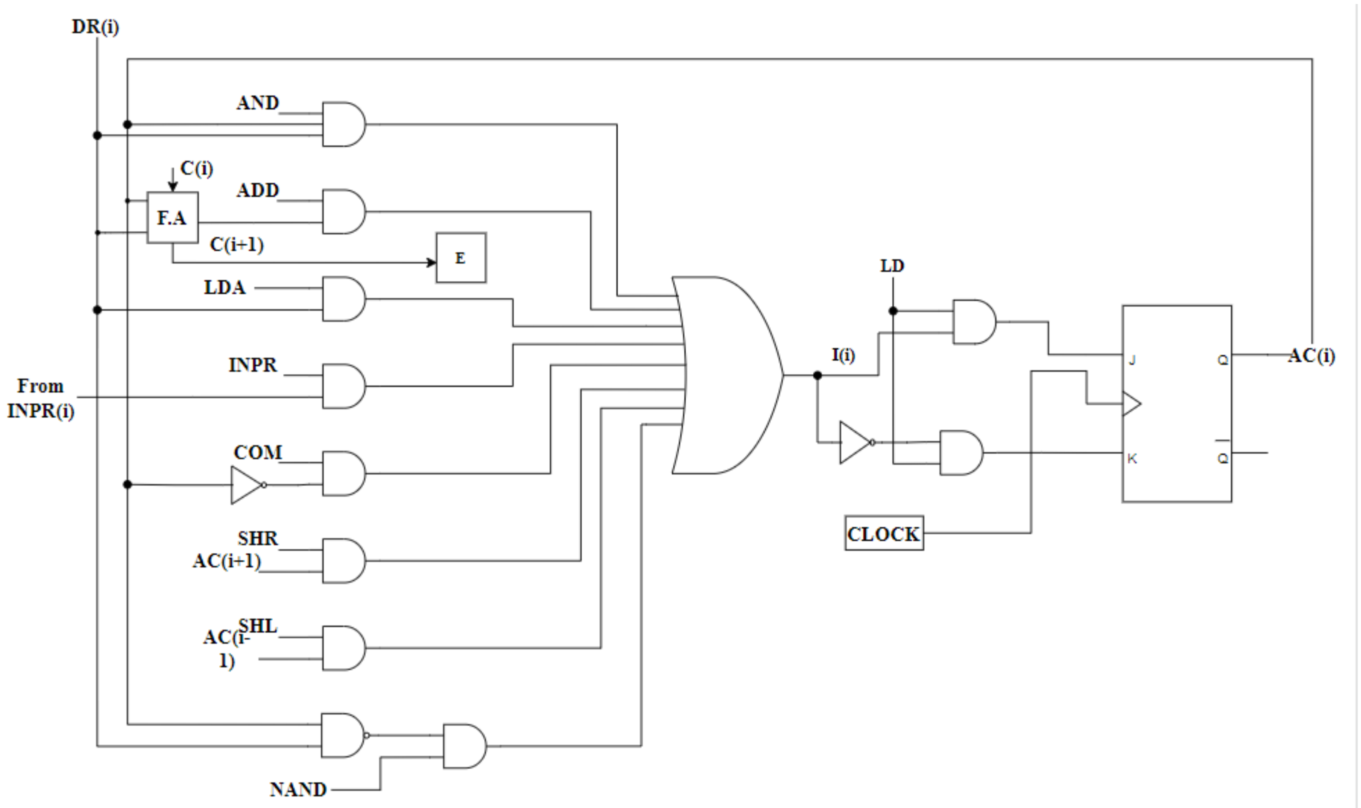

Figure 19: Circuit design for ALU

Chapter 4: Conclusion

By the above-mentioned architecture process a computer can be built. However, a commercial computer has much more memory capacity and operations that it can perform. Even in terms of hardware the computer has more functionality than the basic computer that is built in the project. But this basic computer acts as a prototype for the commercial computer and has a great role in helping to understand the architecture of a computer.