KATHMANDU UNIVERSITY

PROJECT REPORT

ON

Design of a Basic Computer

Submitted to

Mr. Pankaj Dawadi

Department of Computer Science and Engineering (DoCSE)

Eliza Giri (21044)

Sandip Katuwal (21049)

Anav Katwal (21050)

Bishal Khadka (21051)

20 July, 2017

Table of Contents

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 set of registers, control structures and set of instructions that is uses. This organization uses sequence of microoperations that it performs on the data stored in its registers in order to carry out the whole instruction. This done 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.

Although there are quite more functions and operations that a commercial computer can do now a days, the computer designed in this project is limited with just a few of them. These operations are the basic operations used in computers and only from the knowledge gained with the architecture of such primitive operations, the modern computer could have been built.

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

- The RAM (memory) contains 2^{14} words.
- Each word contains 18 bits.
- Seven memory reference operations can be performed by the computer.
- Seven computer registers for storing and processing data and addresses.

2. 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.

Op-code	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 play 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.

i. Memory Reference Instruction (MRI)
ii. Register Reference Instruction (RRI)
iii. Input / Output Instruction (IOI)

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

I. 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 14	13 0
I	Op-code	Address

II. 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 the bits 12 and are neglected and the remaining bits 0 to 11 give the register operation code.

17	16 14	13 0
1	0 0 0	Register operation

III. 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 the bits 12 and 13 are neglected and the remaining bits 0 to 11 give the input / output operation code.

17	16 14	13 0
0	0 0 0	I/O operation

The following table gives the description of the instruction for the designed computer.

Memory Reference Instruction

Symbol	I=0	I=1	Description
LDA	1xxxx	9xxxx	Load the memory word to the accumulator
STA	2xxxx	Axxxx	Store the accumulator content to memory
ADD	3xxxx	Bxxxx	Add memory word to the accumulator
AND	4xxxx	Cxxxx	AND memory word to the accumulator
BUN	5xxxx	Dxxxx	Branch unconditionally
BSA	6xxxx	Exxxx	Branch and save return address
NAND	7xxxx	Fxxxx	NAND memory word to the accumulator

Note: x denotes the hexadecimal equivalent of the address.

Register Reference Instruction

Symbol	Instruction code	Description
CLA	8800	Clear AC
CMA	8400	Complement AC
INC	8200	Increment AC
CIR	8100	Circulate right AC and E
CIL	8080	Circulate left AC and E
SPA	8040	Skip next instruction if AC is positive
SNA	8020	Skip next instruction if AC is negative
SZA	8010	Skip next instruction if AC is zero
CLE	8008	Clear E
CME	8004	Complement E
SZE	8002	Skip next instruction if E is positive
HLT	8001	Halt computer

Note: The instruction code denotes the hexadecimal code of the RRI neglecting the bits 12 and 13 of the instruction register.

Input / Output Instruction

Symbol	Instruction code	Description
INP	0800	Input character to AC
OUT	0400	Output character from AC
SKI	0200	Skip on input flag
SKO	0100	Skip on output flag
ION	0080	Interrupt on
IOF	0040	Interrupt off

Note: The instruction code denotes the hexadecimal code of the IOI neglecting the bits 12 and 13 of the instruction register.

3. Computer Resistors

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

- **Accumulator (AC)**: Accumulator is the main register of this computer. It is of 18-bit and stores the result of any operation after its completion. Input of the accumulator comes from the output of the arithmetic and logic out. Output of the AC goes to the common bus system as well as to 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 data register is interfaced with the ALU. Any valued 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 provided 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 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 of 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.

4. 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.

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

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

II. **Decode:** During the decode cycle, the bit 17 if the IR is transferred to a flip flop I. The bits 15 to 17 are decoded and the bits 0 to 13 are transferred to the AR. The RTL for the microoperation of given below.
$R'T_2$: I←IR(17), D0-D7←Decode IR(14-16), AR←IR(0-13)
$R'T_3I$: AR←M[AR]

III. **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 latter in the report.

5. 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 to programmer to with options 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.

A flow chart of the instruction cycle including the interrupt cycle is given later in the report.

6. **Common Bus System:**

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

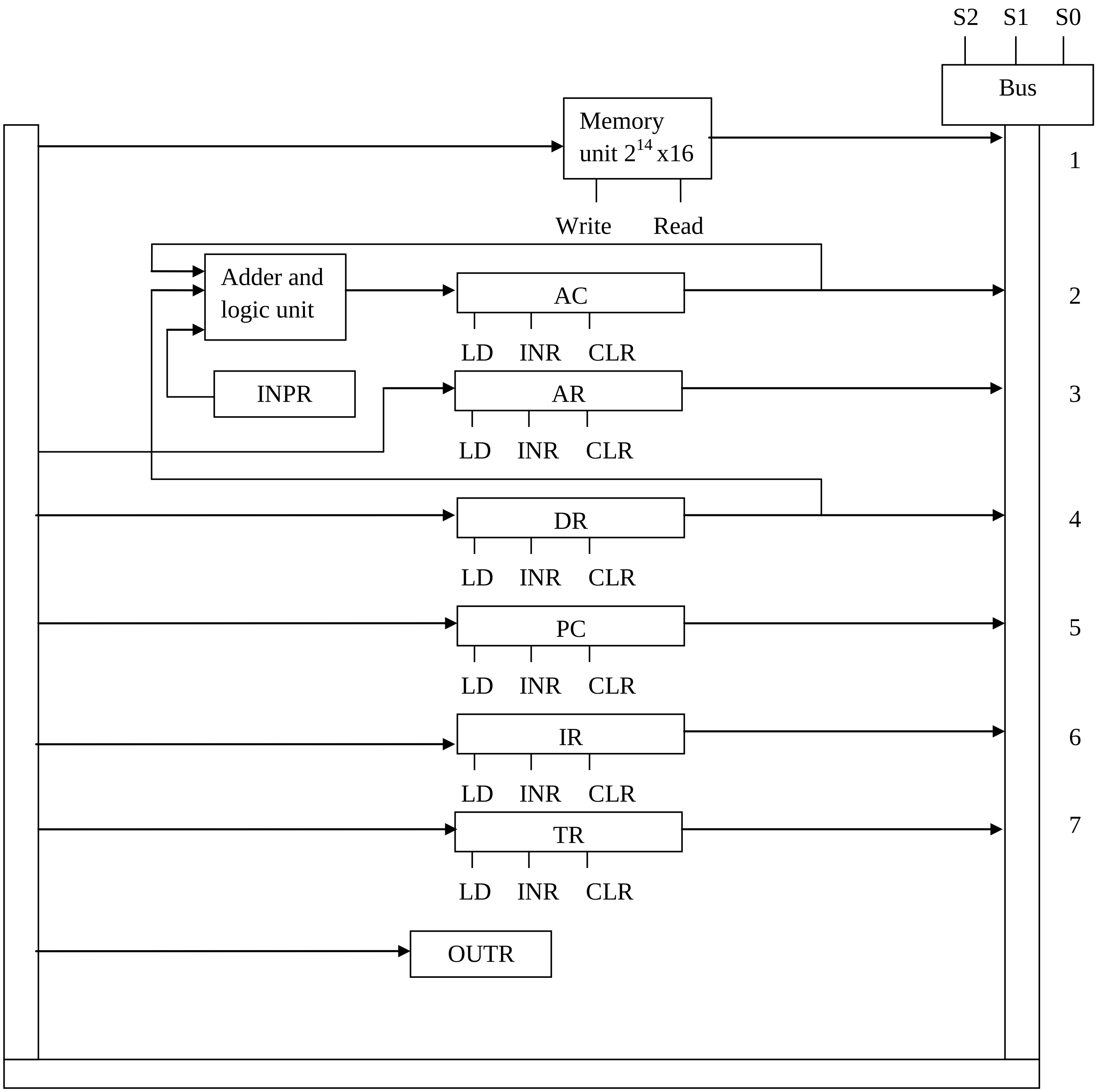

Figure 1: Block diagram of the Common Bus System

7. RTL and Control Signals:

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

Fetch:

$R'T_0$: AR←PC

$R'T_1$: IR←M[AR], PC←PC+1

Decode:

$R'T_2$: I←IR(17), D0-D7←Decode IR(14-16), AR←IR(0-13)

$D_0'T_3I$: AR←M[AR]

Interrupt:

$T_0'T_1'T_2'$(IEN)(FGO+FGI): R←1

RT_0: AR←0, TR←PC

RT_1:M[AR]←TR, PC←0

RT_2:PC←PC+1, IEN←0, R←0, SC←0

Memory Reference Instruction:

LDA: D_1T_4: DR←M[AR]

D_1T_5: AC←DR, SC←0

STA: D_2T_4: M[AR]←AC, SC←0

ADD: D_3T_4: DR←M[AR]

D_3T_5: AC←AC+DR, E←C_{out}, SC←0

AND: D_4T_4: DR←M[AR]

D_4T_5: AC←AC^DR, SC←0

BUN: D_5T_4: PC←AR, SC←0

BSA: D_6T_4: M[AR]←PC, AR←AR+1

D_6T_5: PC←AR, SC←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←0

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

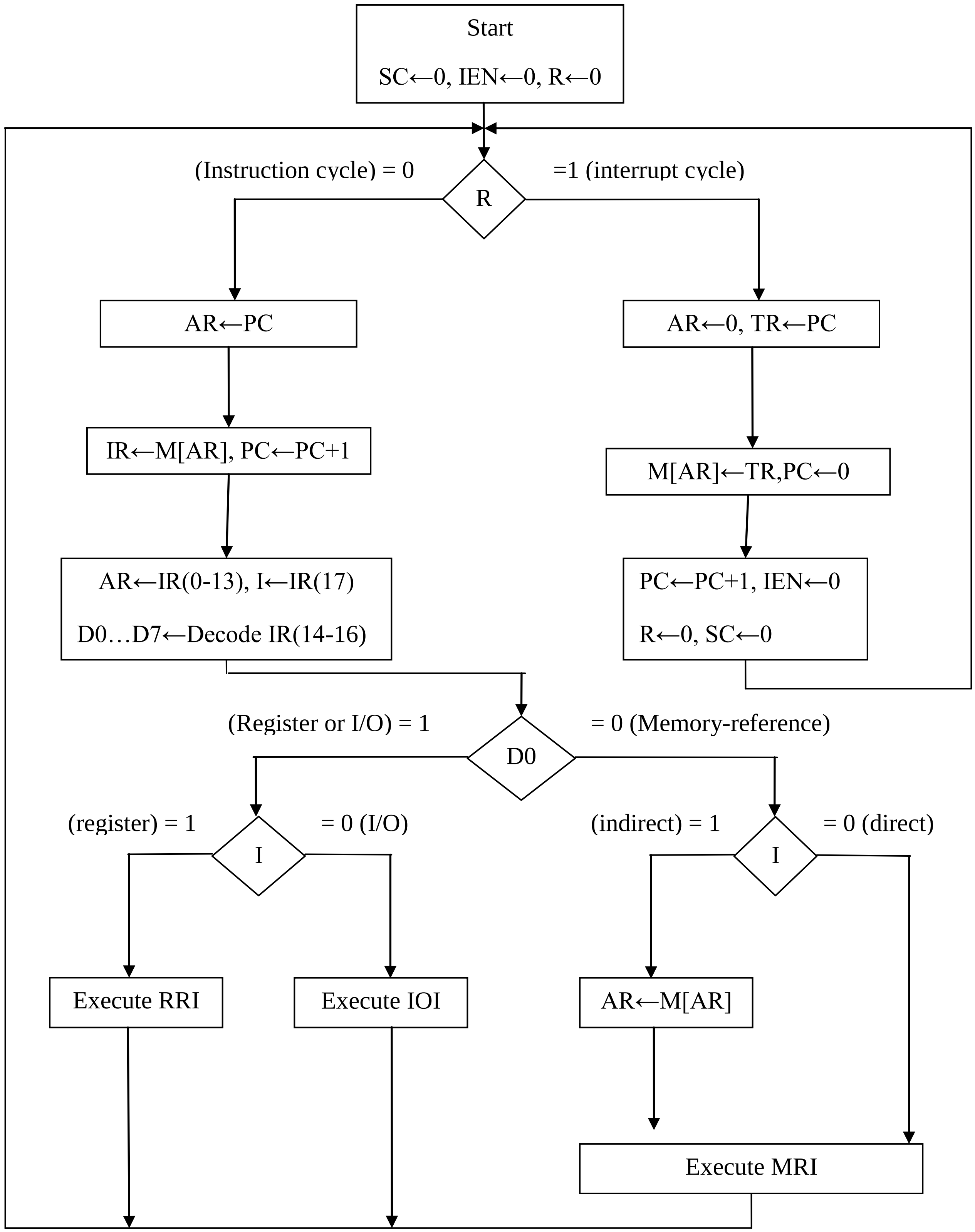

Figure 2: Program Flow Chart

8. **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 follow figure shows the block diagram of the control unit along with its inputs from the Instruction Register and the sequence counter.

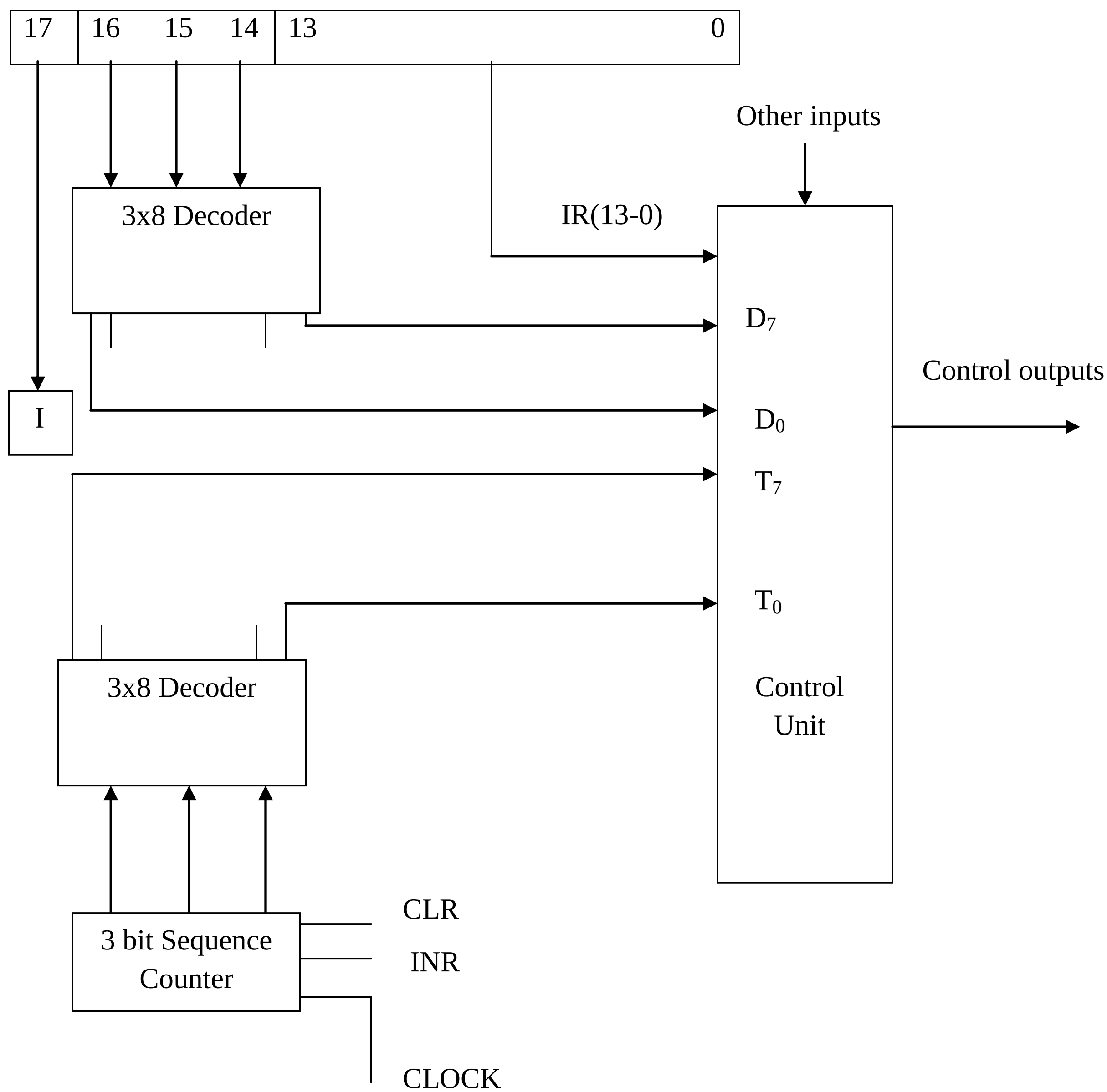

Figure 3: Block Diagram of the Control Unit

9. 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)=$R'T_0+R'T_2+D_0'T_3I$

CLR(AR)=RT_0

INR(AR)=D_6T_4

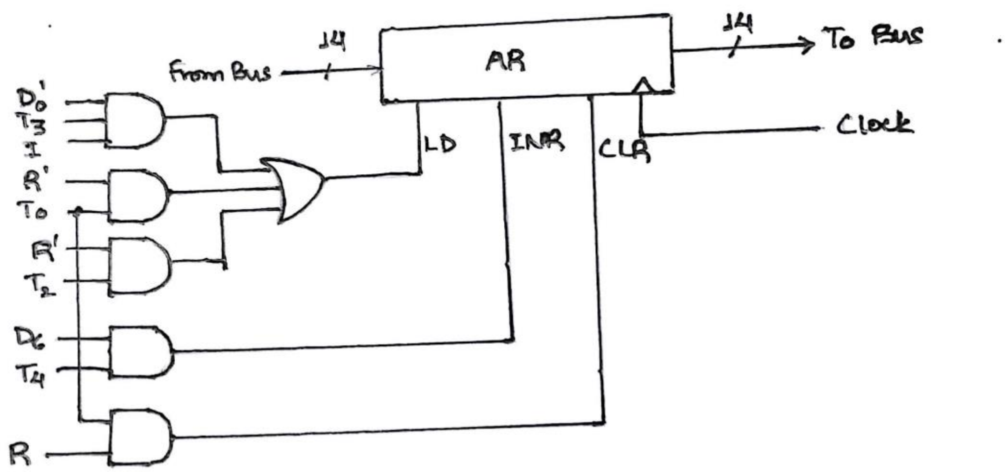

Figure 4: Control Circuit for AR

Memory:

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

Write= $RT_1+ D_2T_4+ D_6T_4$

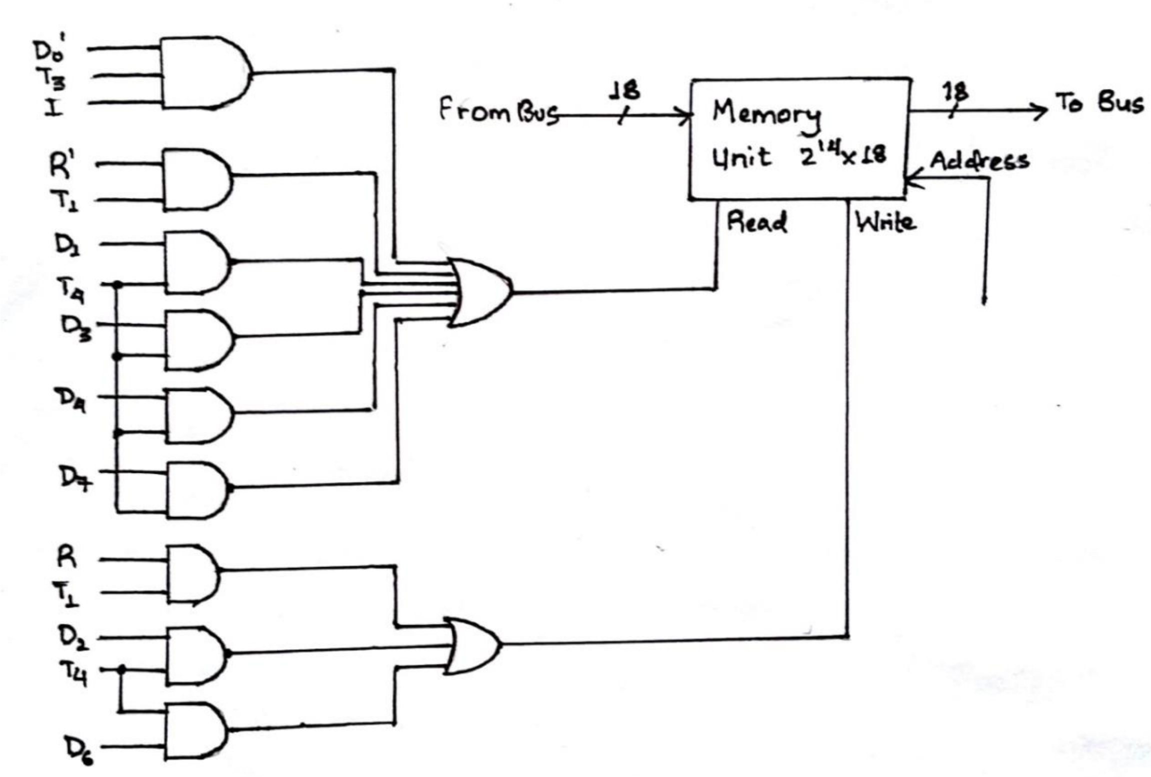

Figure 5: Control Circuit for Memory

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}

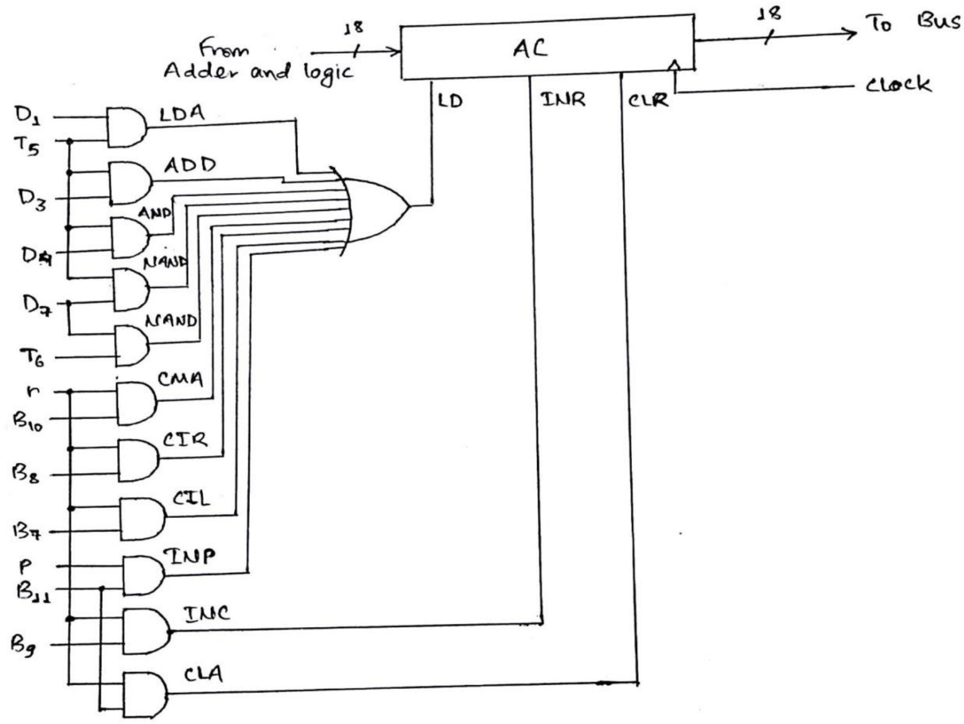

Figure 6: Control Circuit for AC

DR:

LD= D_1T_4+ D_3T_4+ D_4T_4+D_7T_4

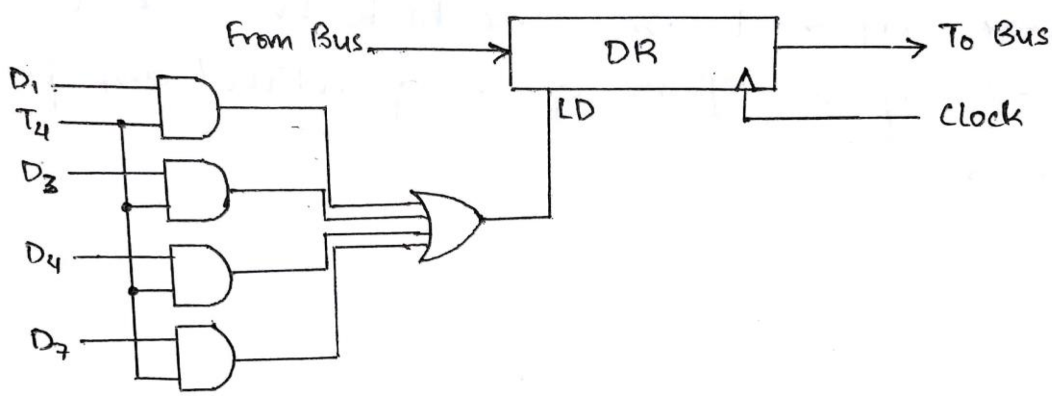

Figure 7: Control Circuit for DR

IR:

LD= $R'T_1$

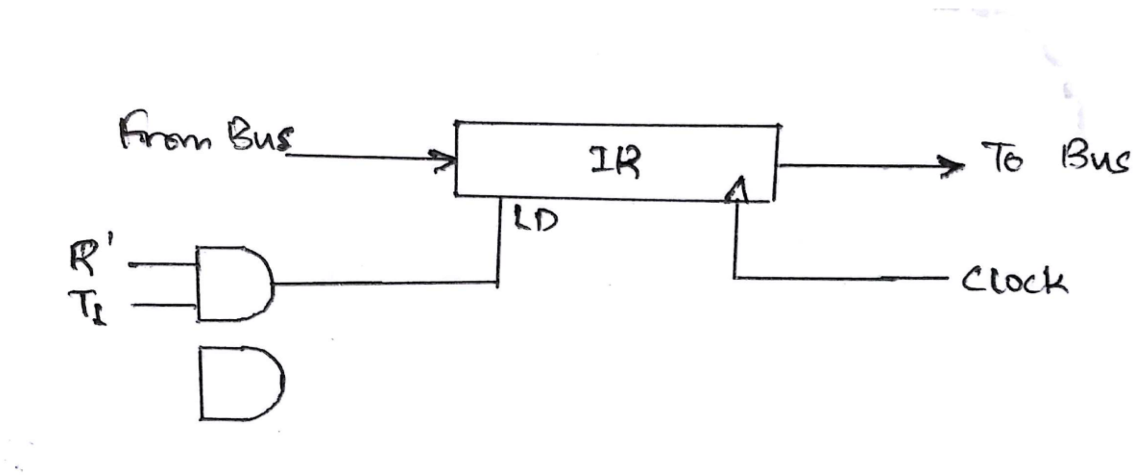

Figure 8: Control Circuit for IR

TR:

LD=RT_0

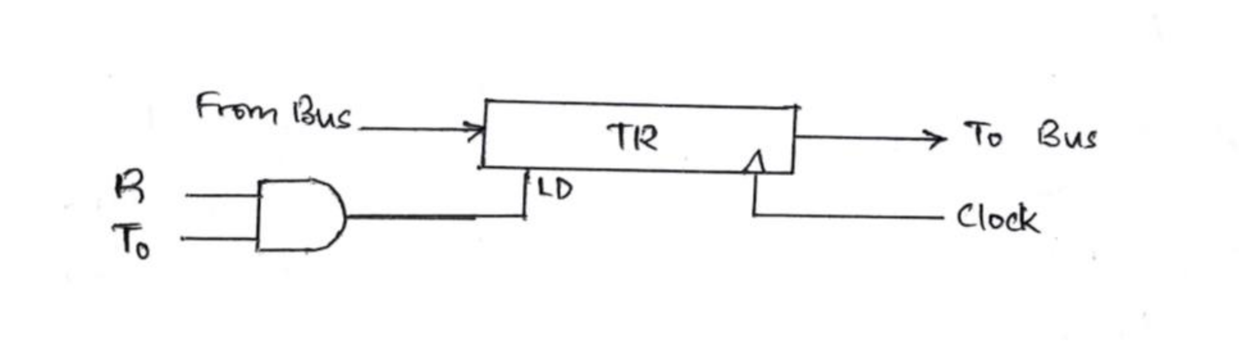

Figure 9: Control Circuit for TR

PC:

LD= D_5T_4+ D_6T_5

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

CLR= RT_1

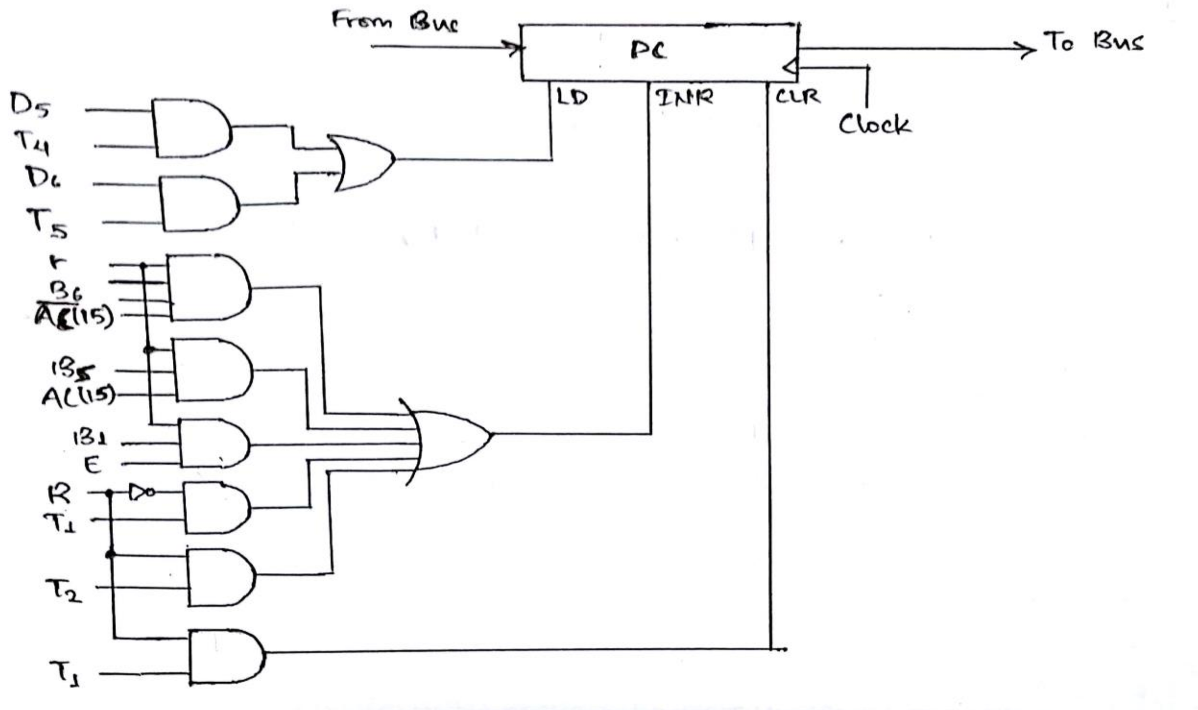

Figure 10: Control Circuit for PC

OUTR:

LD= pB_{10}

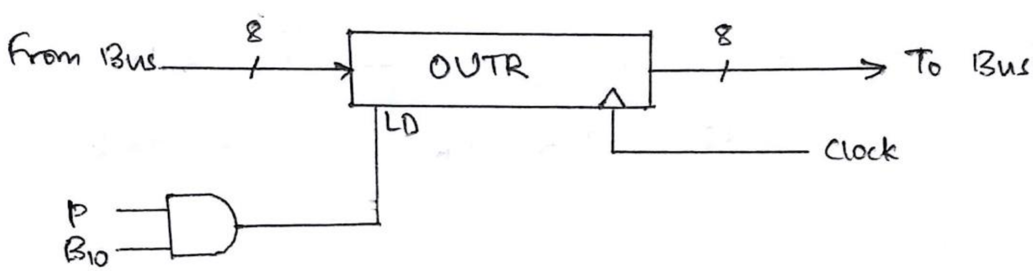

Figure 11: Control Circuit for OUTR

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’

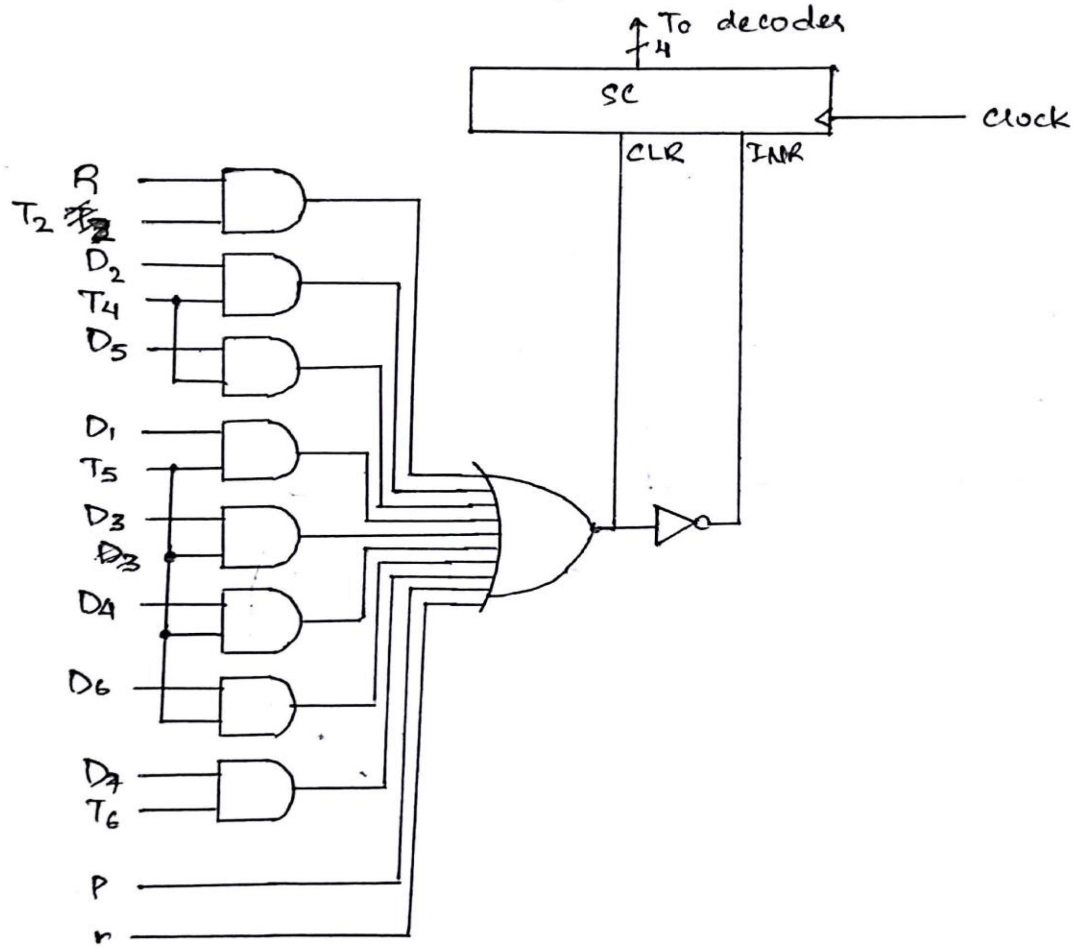

Figure 12: Control Circuit for SC

IEN:

RESET: RT_2+ pB_6

SET: pB_7

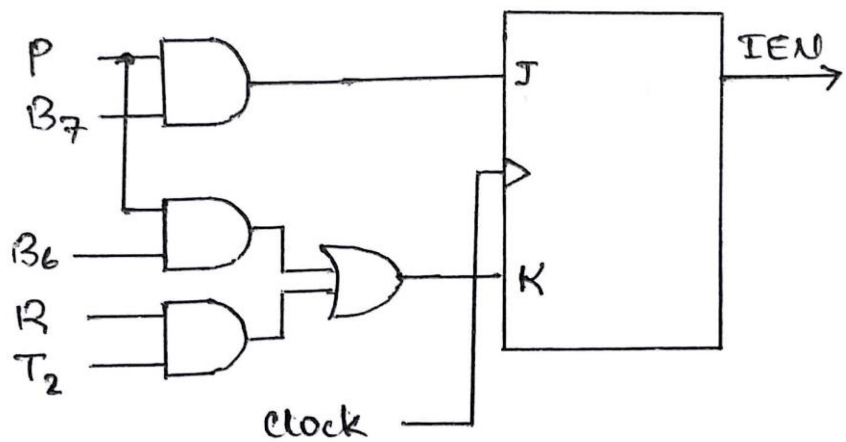

Figure 13: Control Circuit for IEN

R:

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

RESET= RT_2

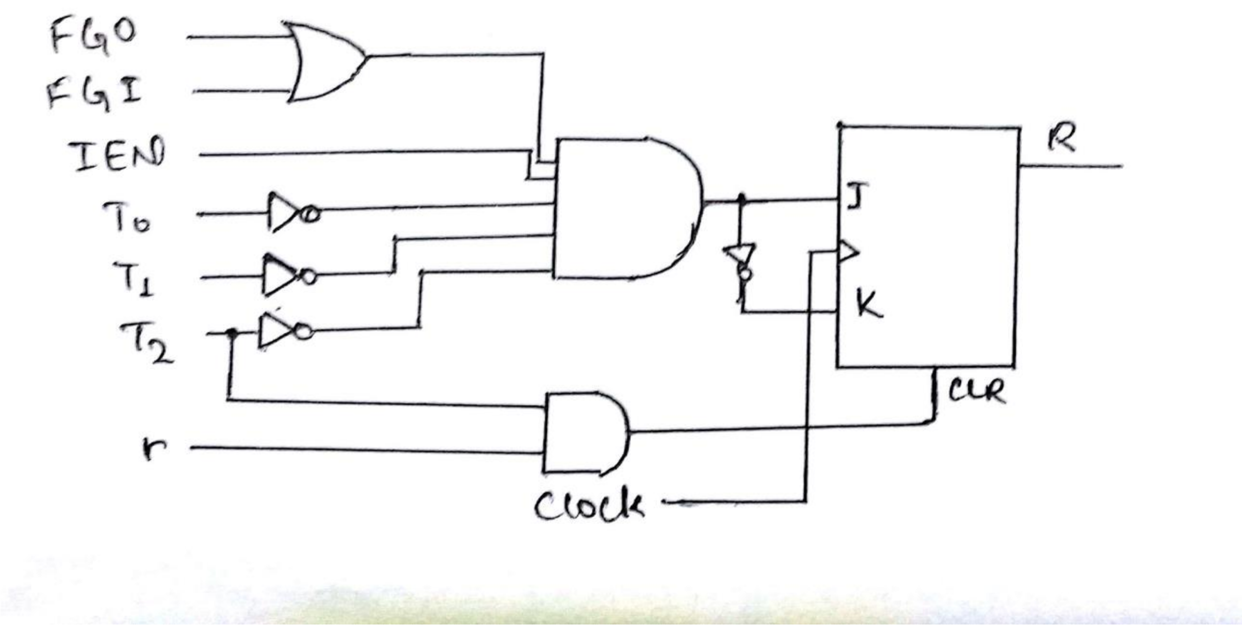

Figure 14: Control Circuit for R

E:

LD= D_3T_5+ rB_2+ rB_8+ rB_7

CLR= rB_3

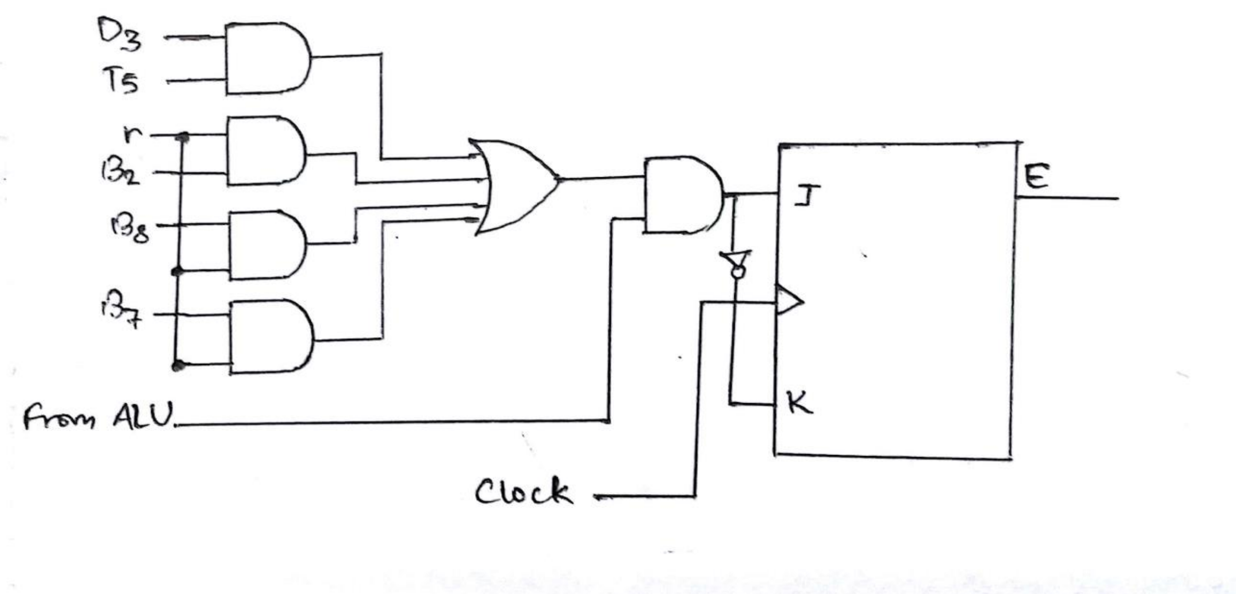

Figure 15: Control Circuit for E

Decoder Design for the Operation Code (3x8):

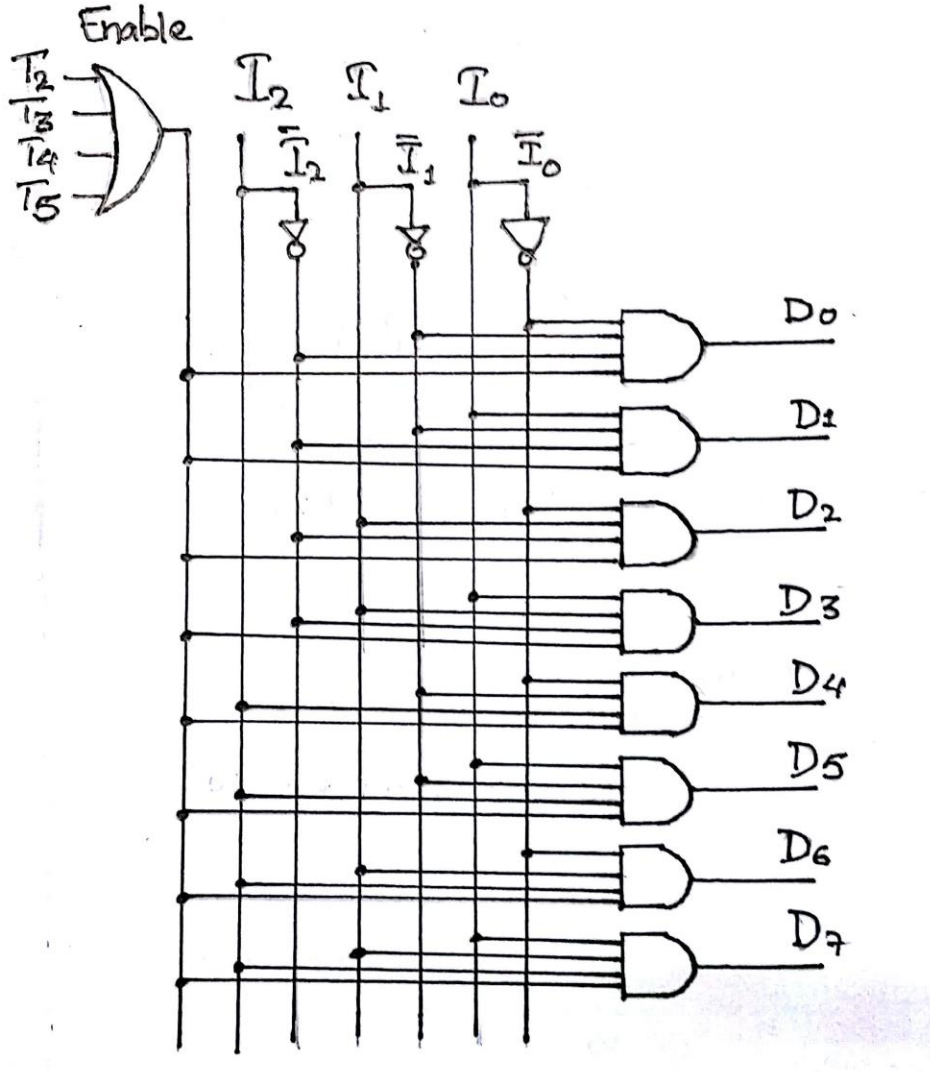

Figure 16: Decoder Circuit

10. 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 selected for bus**
X_1	X_2	X_3	X_4	X_5	X_6	X_7	S_2	S_1	S_0	
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.

$X1= R'T_1+ D_0'T_3I+ D_1T_4+ D_3T_4+ D_4T_4+ D_7T_4$

$X2= D_2T_4$

$X3= D_5T_4+ D_6T_5$

$X4= D_1T_5$

$X5= R'T_0+ RT_0+ D_6T_4$

$X6= R'T_2$

$X7=RT_1$

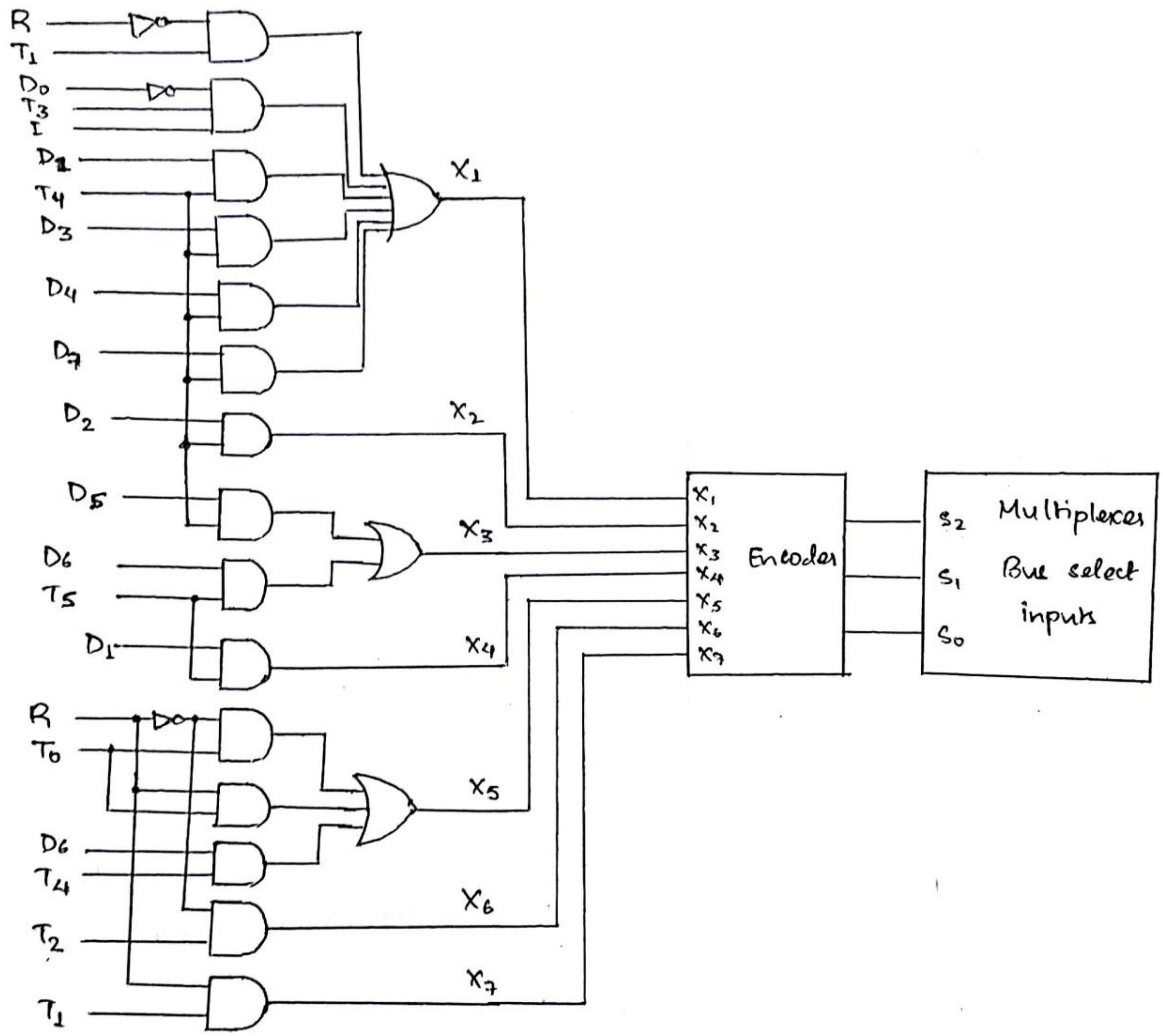

Figure 17: Control Circuit for CBS

11. 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 accumulator and therefore is of 18 bits in the computer designed. 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 detail design of a bit of the ALU is given below.

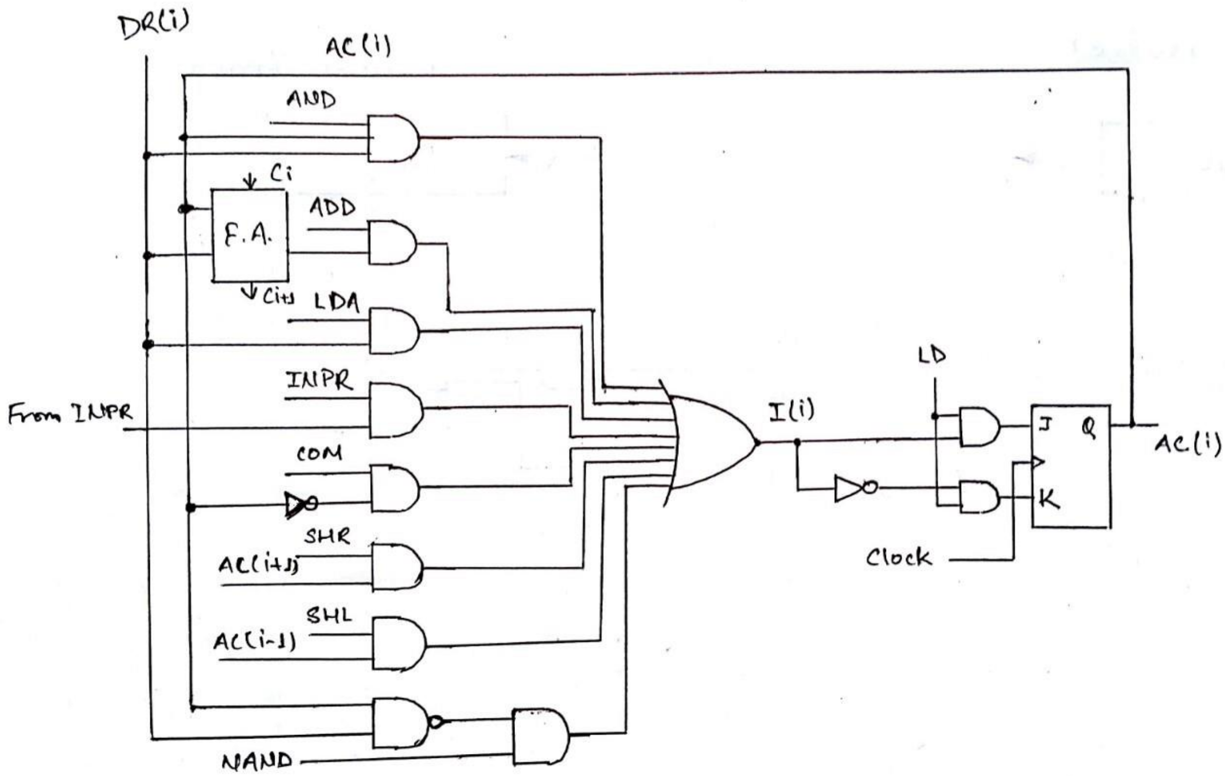

Figure 18: Circuit Design for ALU

12. 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.