Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Project Report on

"SAAJA 16K*19 Computer"

[Code No: COMP 315]

(For partial fulfillment of COMP 315 III/I in Computer Engineering)

Submitted by :

Sushant Adhikari (05)

Ayush Aryal (07)

Aadarsha Dhakal (12)

Jamyang Gelek Gurung (16)

Ankush Niroula (58)

Submitted to :

Mr. Pankaj Raj Dawadi

Department of Computer Science and Engineering

Submission Date: Nov 4, 2022

Abstract

This paper includes the overview of the **SAAJA 16K*19 Computer**; the description of basic computer based on the knowledge of Basic computer of 4K*16 bits computer designed by Morris Mano. “SAAJA Computer” is a simple computer of size 16K*19 containing 14 bit address, 3 bit opcode and 2 bit addressing modes. It has 19-bit registers, 19-bit common bus system, 14-bit address lines, 7 Memory Reference Instructions, 12 Register Reference Instructions and 6 Input Output Instructions. The basic concept of architecture and organization of computer design, with all the necessary functional components to be working as a real computer, is covered in this paper which is simple to understand and comprehend the workings of computers and also assists in further understanding of more complex computer architecture.

KEYWORD: Basic computer, Memory, Register, Computer Architecture,

Table of Contents

Abstract **2**

Table of Contents **3**

List of Figures **5**

List of Tables **6**

Chapter 1 : Introduction **7**

Stored Program Concepts 7

Addressing Modes 8

Direct Addressing Mode 8

Indirect Addressing Mode 8

Immediate Addressing Mode 8

Assumptions 9

Chapter 2 : Design Considerations **10**

1. Computer Registers 10

2. Instruction format 11

I. Memory Reference Instruction: 11

II. Register Reference Instruction: 12

III. Input /Output Instruction: 13

3. Instruction cycle 14

Fetch 14

Decode 14

Execute 14

4. Interrupt cycle 15

5. Common Bus system 16

6. RTL and Control Signal 17

Fetch 17

Decode 17

Interrupt 17

Memory Reference Instructions 18

Immediate Memory Reference Instructions 18

Register Reference Instructions 19

Input/Output and Interrupt Instructions 19

Chapter 3 : Design **20**

Design of AR 20

Design of DR 20
Design of AC 21
Design of PC 22
Design of IR 24
Design of TR 24
Design of OUTR 24
Design of IEN 25
Design of E 26
Design of R 26
Design of S 27
Design of FGI 27
Design of FGO 28
Design of SC 29
Control of Common Bus 30
Design of Memory 33
Design of ALU 34
Control Unit 35
Common Bus System 36
Overall Architecture of SAAJA Basic Computer 37

Chapter 4 : Conclusion **38**

List of Figures

Figure 1.1: Stored Program Concept
Figure 2.1: FlowChart of Instruction Cycle
Figure 2.2: FlowChart of Interrupt Cycle
Figure 3.1: Design of AR
Figure 3.2: Design of DR
Figure 3.3: Design of AC
Figure 3.4: Design of PC
Figure 3.5: Design of IR
Figure 3.6: Design of TR
Figure 3.7: Design of OUTR
Figure 3.8: Design of IEN
Figure 3.9: Design of E
Figure 3.10: Design of R
Figure 3.11: Design of S
Figure 3.12: Design of FGI
Figure 3.13: Design of FGO
Figure 3.14: Design of SC
Figure 3.15: Design of Control of Common Bus
Figure 3.16: Design of Memory
Figure 3.17: Design of ALU
Figure 3.18: Design of Control Unit
Figure 3.19: Design of Common Bus System
Figure 3.20: Design of SAAJA Basic Computer

List of Tables

Table 2.1: Computer Register
Table 2.2: Memory Reference Instruction
Table 2.3: Register Reference Instruction
Table 2.4: Input/Output Instruction
Table 2.5: RTL of Memory Reference Instruction
Table 2.6: RTL of Immediate Memory Reference Instruction
Table 2.7: RTL of Register Reference Instruction
Table 2.8: RTL of Input/Output Instruction

Chapter 1 : Introduction

Computer organization refers to the operational unit and their interconnection that realize the architectural specification. Computer organization deals with how different part of a computer are organized and how various operations are performed between different part to do a specific task.The organization of the computer is defined by its internal registers ,timing and control structure ,and set of instructions that it uses.

Basic computers are simplifications of commercial computers which are simple enough to demonstrate the design process without too many complications and delves into the understanding of Computer Architecture. Basic computers enable us to show how operations can be specified with Register Transfer Language. Based on the knowledge of Basic Computer of 4K*16 bits designed by Morris Mano, we design similar computer architecture consisting of 16K*19 bits.

Stored Program Concepts

Our computer, like most modern computers, works on Von-Neumann architecture i.e. stored program concept - data and instructions are stored in the same memory. Before the introduction of this idea, instructions and data were considered two totally different entities and were thus stored separately. The term **Stored Program Control Concept** refers to the storage of instructions in computer memory to enable it to perform a variety of tasks in sequence or intermittently. The idea was introduced in the late 1040s by John von Neumann who proposed that a program be electronically stored in the binary-number format in a memory device so that instructions could be modified by the computer as determined by intermediate computational results.

Thus instructions like data can be read from the memory and written to the memory by the processor.The processor then addresses the memory, reads the corresponding instructions, executes them and according to the executed instruction, processes (reads and writes) data as well.

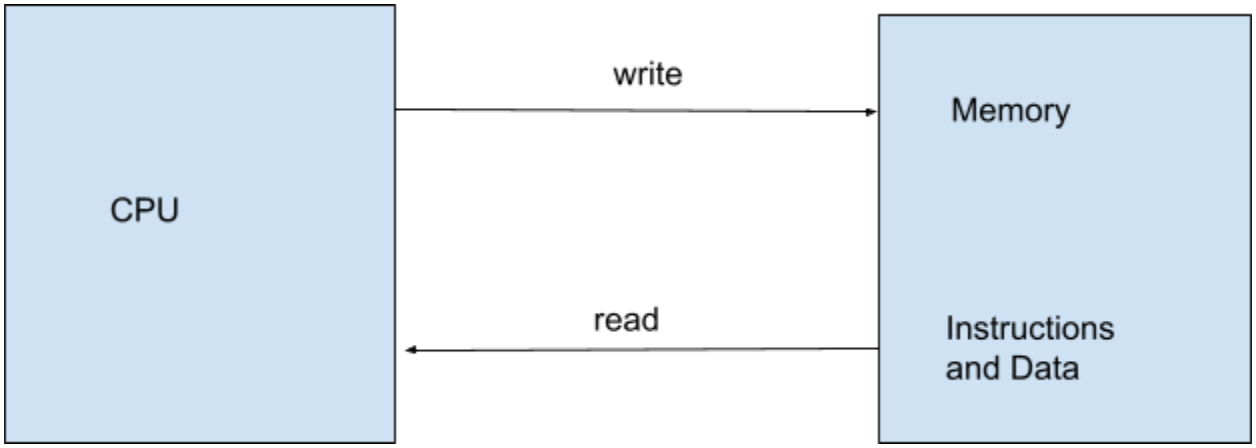


Figure 1.1 : Stored Program concept

Addressing Modes

It is sometimes convenient to use the address bits of an instruction code not as an address but as the actual operand. When the second part of an instruction code specifies an operand, the instruction is said to have an immediate operand. When the second part specifies the address of an operand, the instruction is said to have a direct address. This is in contrast to a third possibility called indirect address, where the bits in the second part of the instruction designate an address of a memory word in which the address of the operand is found. One bit of the instruction code can be used to distinguish between a direct and an indirect address. The computer will only have direct addressing.

Direct Addressing Mode

In this mode the address of data(operand) is specified in the instruction itself.That is, in this type of mode, the operand resides in memory and its address is given directly by the address field of the instruction. In this mode, the address field contains the Effective Address of operand.

Indirect Addressing Mode

In this mode, the address field of instruction gives the memory address where on, the operand is stored in memory.That is, in this mode, the address field of the instruction gives the address where the “Effective Address” is stored in memory.

Immediate Addressing Mode

In this mode , the operand is specified in the instruction itself. Instructions are longer but the operands are easily and immediately identified. In this mode the data is 8 bits or 16 bits long and data is the part of instruction.

Assumptions

Our computer (SAAJA) is designed to be 16K*19. Taking this into consideration, we have following properties :

- 16384 addressable memory with each of 19 bits or the RAM contains 2^{14} words.
- 19 bit Register
- 19 bit Common Bus System
- 14 bit address lines
- 7 Memory Reference Instructions
- 12 Register Reference Instructions
- 6 Input / Output Instructions

Registers : AC, DR, IR, AR, PC, TR, INR, OUTR
Flip Flops : I_0, I_1,R,IEN,FGI,E,S

Chapter 2 : Design Considerations

1. Computer Registers

SAAJA Computer consists of 8 registers. They are described as follows :

Register Symbol	Register Name	Number of bits	Function
AC	Accumulator	19	Stores the result of any operation after its completion.
DR	Data Register	19	Store the operand read from the memory that is to be processed
IR	Instruction Register	19	Stores the 19-bit instruction code read from the memory
AR	Address Register	14	Holds the address of the operand in the memory
PC	Program Counter	14	Stores address of the memory location where the next instruction is located.
TR	Temporary Register	19	Store temporary data during the working of the computer
INR	Input Register	8	Holds the input character feed from an input device
OUTR	Output Register	8	Holds the output character that needs to be displayed to the user.

Table 2.1 : Computer Register

2. Instruction format

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.

AM	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 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.

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 and 18 give the addressing mode. It is further specified that the instruction code is an MRI only if the op- code has the decimal value from 0 to 6 i.e. (000 to 110).

18	17	16	13	0
I1	I0	Op-code	Address	

Symbol	Hexadecimal Code			Description
	Direct	Indirect	Immediate	
LDA	08XXXX	10XXXX	18XXXX	Load memory word to AC
STA	09XXXX	11XXXX	19XXXX	Store content of AC in memory
ADD	0AXXXX	12XXXX	1AXXXX	Add memory word to AC

AND	0BXXXX	13XXXX	1BXXXX	Ands memory word to AC
BUN	0CXXXX	14XXXX	1CXXXX	Branch Unconditionally
BSA	0DXXXX	15XXXX	1DXXXX	Branch and Save Return Address
NAND	0EXXXX	16XXXX	1EXXXX	Nand memory word to AC

Table 2.2 : Memory Reference Instruction

II. Register Reference Instruction:

The instruction code is an RRI if the bits 14 to 16 have the decimal equivalent of the value 7 (i.e. 111) and the value of bit 17 and 18 is set to 0. In this case the bits 12 and 13 are neglected and the remaining bits 0 to 11 give the register operation code.

18	17	16	13	0
I1	I0	Op-code	Register options	

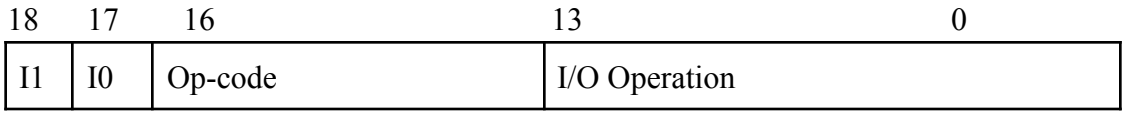
symbol	Hexadecimal code	Description
HLT	07X800	Halt Computer
CLA	07X400	Clear Accumulator
CMA	07X200	Complement A
INC	07X100	Increment AC
SPA	07X080	Skip next instruction if AC is $+ ve$
SNA	07X040	Skip next instruction if AC is $- ve$
SZA	07X020	Skip next instruction if AC is 0
SZE	07X010	Skip next instruction if E is 0
CLE	07X008	Clear E F/F
CME	07X004	Complement E F/F

CIR	07X002	Circular Shift Right
CIL	07X001	Circular Shift Left

Table 2.3 : Register Reference Instruction

III. Input /Output Instruction:

The instruction code is an IOI only if the bits 14 to 16 have the decimal equivalent of the value 7 (i.e. 111) and the value of bits 17 and 18 is set to 1. In this case the bits 12 and 13 are neglected and the remaining bits 0 to 11 give the input / output operation code.



symbol	Instruction code	Description
INP	1FX800	Input Character to AC
OUT	1FX400	Output Character from AC
SKI	1FX200	Skip on Input Flag
SKO	1FX100	Skip on Output Flag
ION	1FX080	Interrupt On
IOF	1FX040	Interrupt Off

Table 2.4 : Input / Output Instruction

3. Instruction cycle

Fetch

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

$R'.T_0: AR \leftarrow PC$

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

Decode

During this cycle, the bit 18 of the IR is transferred to a flip flop I_1 and the bit 17 of the IR is transferred to a flipflop I_2 . The bits 14 to 16 are decoded and the bits 0 to 13 are transferred to the AR. The RTL for the microoperation are given below:

$R'.T_2: I_1 \leftarrow IR(18), I_0 \leftarrow IR(17),\ D_0 \ldots D_7 \leftarrow IR(16 - 14),\ AR \leftarrow IR(13 - 0)$

Execute

This cycle differs for the different types of instructions. In this cycle the instructions are executed accordingly . The complete set of RTL and the control functions for different types of instructions will be listed later in the report.

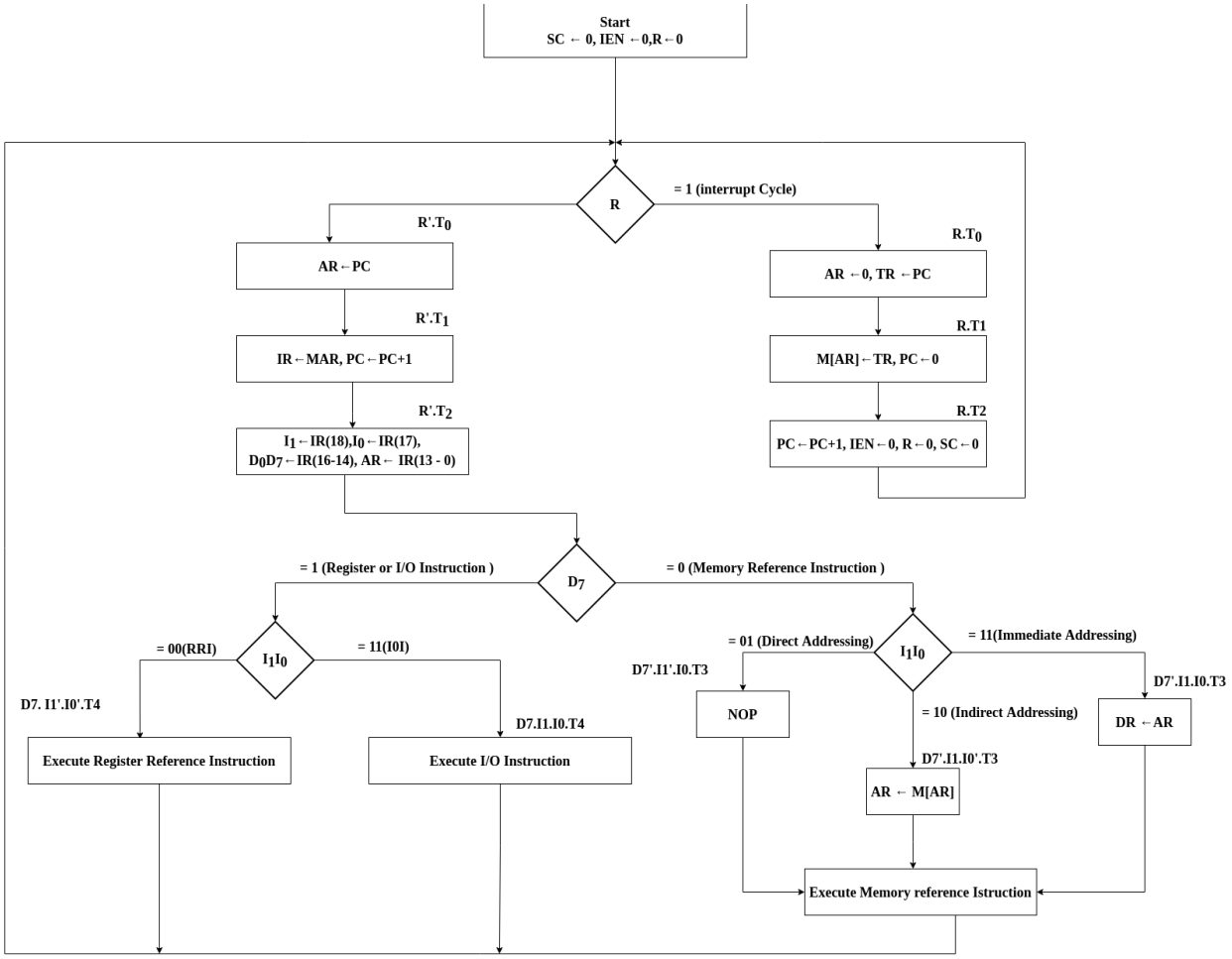


Figure 2.1 : Flow chart of Instruction Cycle or Program Flow

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 programs 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.

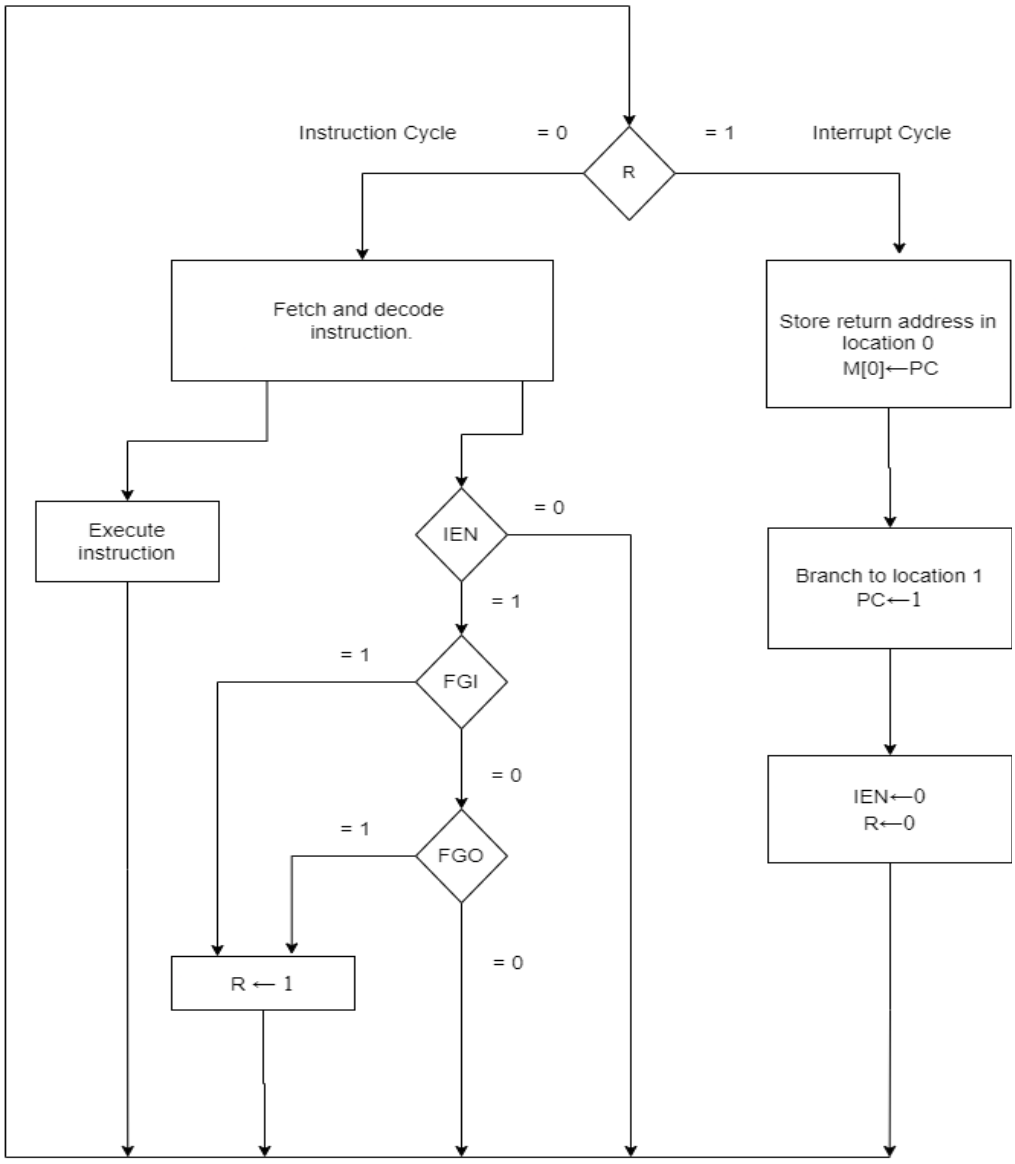


Figure 2.2 : Flow chart of interrupt Cycle

5. Common Bus system

In the common bus system diagram there are 7 components (six registers and a single memory) connected with a common bus. So, to represent these seven components of a common bus system in binary, we require three select lines S0, S1, and S2. Either one of the registers will have its load signal activated, or the memory will have its read signal activated.

- Six registers and a memory are connected to a bus.

- The input register INPR and OUTR has 8 bits each.
- The seven registers, memory, INPR, and OUTR are driven by a single phase clock pulse.
- The particular register whose load (LD) input is enabled receives the data from the bus during the next clock pulse.
- Five registers have three control inputs: load(LD), Increment(INR), and clear(CLR). Two registers IR and OUTR have only LD inputs.
- The result is transferred to AC and end carry is transferred to flip-flop E.

6. RTL and Control Signal

Fetch

$R'.T_0: AR \leftarrow PC$

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

Decode

$R'.T_2: I_1 \leftarrow IR(18), I_0 \leftarrow IR(17),\ D_0 \ldots D_7 \leftarrow IR(16 - 14),\ AR \leftarrow IR(13 - 0)$

Addressing Modes

$D_7'.I_1'.I_0.T_3: NOP$ (For Direct AM)

$D_7'.I_1.I_0'.T_3: AR \leftarrow M[AR]$ (For Indirect AM)

$D_7'.I_1.I_0.T_3: DR \leftarrow AR$ (For Immediate AM)

Interrupt

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

$R.T_0: TR \leftarrow PC$

$R.T_1: M[AR] \leftarrow TR,\ PC \leftarrow 0$

$R.T_2: PC \leftarrow PC + 1,\ IEN \leftarrow 0,\ R \leftarrow 0,\ SC \leftarrow 0$

Memory Reference Instructions

LDA	$D_0.(I_0 \oplus I_1).T_4: DR \leftarrow M[AR]$ $D_0.T_5: AC \leftarrow DR,\ SC \leftarrow 0$
STA	$D_1.T_4: M[AR] \leftarrow AC,\ SC \leftarrow 0$
ADD	$D_2.(I_0 \oplus I_1).T_4: DR \leftarrow M[AR]$ $D_2.T_5: AC \leftarrow AC + DR,\ SC \leftarrow 0, E \leftarrow C_{out}$
AND	$D_3.(I_0 \oplus I_1).T_4: DR \leftarrow M[AR]$ $D_3.T_5: AC \leftarrow AC \wedge DR,\ SC \leftarrow 0$
BUN	$D_4.T_4: PC \leftarrow AR,\ SC \leftarrow 0$
BSA	$D_5.T_4: M[AR] \leftarrow PC,\ AR \leftarrow AR + 1$ $D_5.T_5: PC \leftarrow AR,\ SC \leftarrow 0$
NAND	$D_6.(I_0 \oplus I_1).T_4: DR \leftarrow M[AR]$ $D_6.T_5: AC \leftarrow (AC \wedge DR)',\ SC \leftarrow 0$

Table 2.5 : RTL of Memory Reference Instruction

Immediate Memory Reference Instructions

(The data is immediately transferred from AR to DR , so the step of transfer is not necessary which is why the t4 cycle is NOP)

LDA	$D_0.(I_0 \oplus I_1)'.T_4: NOP$ $D_0.T_5: AC \leftarrow DR,\ SC \leftarrow 0$
ADD	$D_2.(I_0 \oplus I_1)'.T_4: NOP$ $D_2.T_5: AC \leftarrow AC + DR,\ SC \leftarrow 0, E \leftarrow C_{out}$
AND	$D_3.(I_0 \oplus I_1)'.T_4: NOP$ $D_3.T_5: AC \leftarrow AC \wedge DR,\ SC \leftarrow 0$
NAND	$D_6.(I_0 \oplus I_1)'.T_4: NOP$ $D_6T_5: AC \leftarrow (AC \wedge DR)',\ SC \leftarrow 0$

Table 2.6 : RTL of Immediate Memory Reference Instruction

Register Reference Instructions

$D_7.\ I_1'.I_0'.T_4$ is denoted by r

	$r: SC \leftarrow 0$
HLT	$r.B_{11}: S \leftarrow 0$
CLA	$r.B_{10}: AC \leftarrow 0$
CMA	$r.B_9: AC \leftarrow AC'$
INC	$r.B_8: AC \leftarrow AC + 1$
SPA	$r.B_7: If\ AC(18) = 0\ then\ PC \leftarrow PC + 1$
SNA	$r.B_6: If\ AC(18) = 1\ then\ PC \leftarrow PC + 1$
SZA	$r.B_5: If(AC = 0)\ then\ PC \leftarrow PC + 1$
SZE	$r.B_4: If(E = 0)\ then\ PC \leftarrow PC + 1$
CLE	$r.B_3: E \leftarrow 0$
CME	$r.B_2: E \leftarrow E'$
CIR	$r.B_1: AC(18) \leftarrow E, E \leftarrow AC(0), AC \leftarrow shr(AC)$
CIL	$r.B_0:, AC(0) \leftarrow E, E \leftarrow AC(18),\ AC \leftarrow shl(AC)$

Table 2.7 : RTL of Register Reference Instruction

Input/Output and Interrupt Instructions

$D_7.I_1.I_0.T_4$ is denoted by p

	$p: SC \leftarrow 0$
INP	$p.B_{11}: AC(0-7) \leftarrow INPR,\ FGI \leftarrow 0$
OUT	$p.B_{10}: OUTR \leftarrow AC(0-7),\ FGO \leftarrow 0$
SKI	$p.B_9: If\ (FGI = 1)\ then\ PC \leftarrow PC + 1$
SKO	$p.B_8: If\ (FGO = 1)\ then\ PC \leftarrow PC + 1$
ION	$p.B_7: IEN \leftarrow 1$
IOF	$p.B_6: IEN \leftarrow 0$

Table 2.8 : RTL of Input/ Output Instruction

Chapter 3 : Design

Design of AR

LD : (AR <-)

$R'.T_0: AR \leftarrow PC$

$R'.T_2: I_1 \leftarrow IR(18), I_0 \leftarrow IR(17),\ D_0 \ldots D_7 \leftarrow IR(16-14),\ AR \leftarrow IR(13-0)$

$D_7'.I_1.I_0'.T_3: AR \leftarrow M[AR]$ (For Indirect AM)

CLR : $R.T_0: AR \leftarrow 0,\ TR \leftarrow PC$

INR : $D_5.T_4: M[AR] \leftarrow PC,\ AR \leftarrow AR + 1$

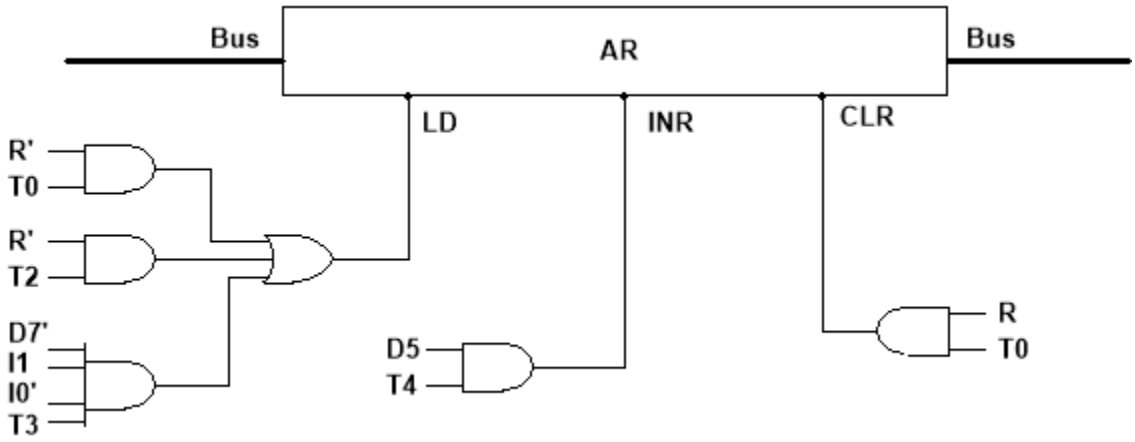


Figure 3.1 : Design of AR

Design of DR

LD:

$D_7'.I_1.I_0.T_3:\ DR \leftarrow AR$ (For Immediate AM)

$D_0.(I_0 \oplus I_1).T_4: DR \leftarrow M[AR]$

$D_2.(I_0 \oplus I_1).T_4: DR \leftarrow M[AR]$

$D_3.(I_0 \oplus I_1).T_4: DR \leftarrow M[AR]$

$D_6.(I_0 \oplus I_1).T_4: DR \leftarrow M[AR]$

CLR:

INR:

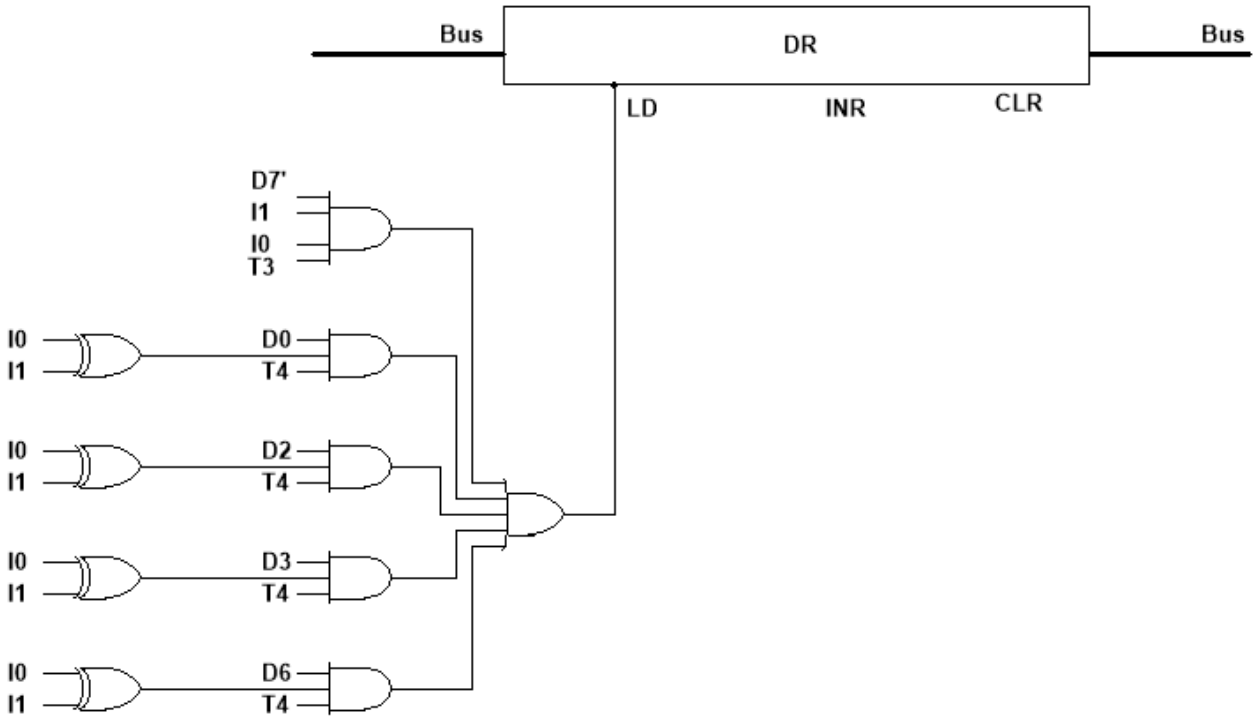


Figure 3.2 : Design of DR

Design of AC

LD :

$D_0.T_5: AC \leftarrow DR,\ SC \leftarrow 0$

$D_2.T_5: AC \leftarrow AC\ +\ DR,\ SC \leftarrow 0, E \leftarrow C_{out}$

$D_3.T_5: AC \leftarrow AC \wedge DR,\ SC \leftarrow 0$

$D_6.T_5: AC \leftarrow (AC \wedge DR)',\ SC \leftarrow 0$

$r.B_9: AC \leftarrow AC'$

$r.B_1: AC(18) \leftarrow E, E \leftarrow AC(0), AC \leftarrow shr(AC)$

$r.B_0:, AC(0) \leftarrow E, E \leftarrow AC(18),\ AC \leftarrow shl(AC)$

$p.B_{11}: AC(0\ -\ 7) \leftarrow INPR,\ FGI \leftarrow 0$

CLR:

$r.B_{10}: AC \leftarrow 0$

INR :

$r.B_8: AC \leftarrow AC\ +\ 1$

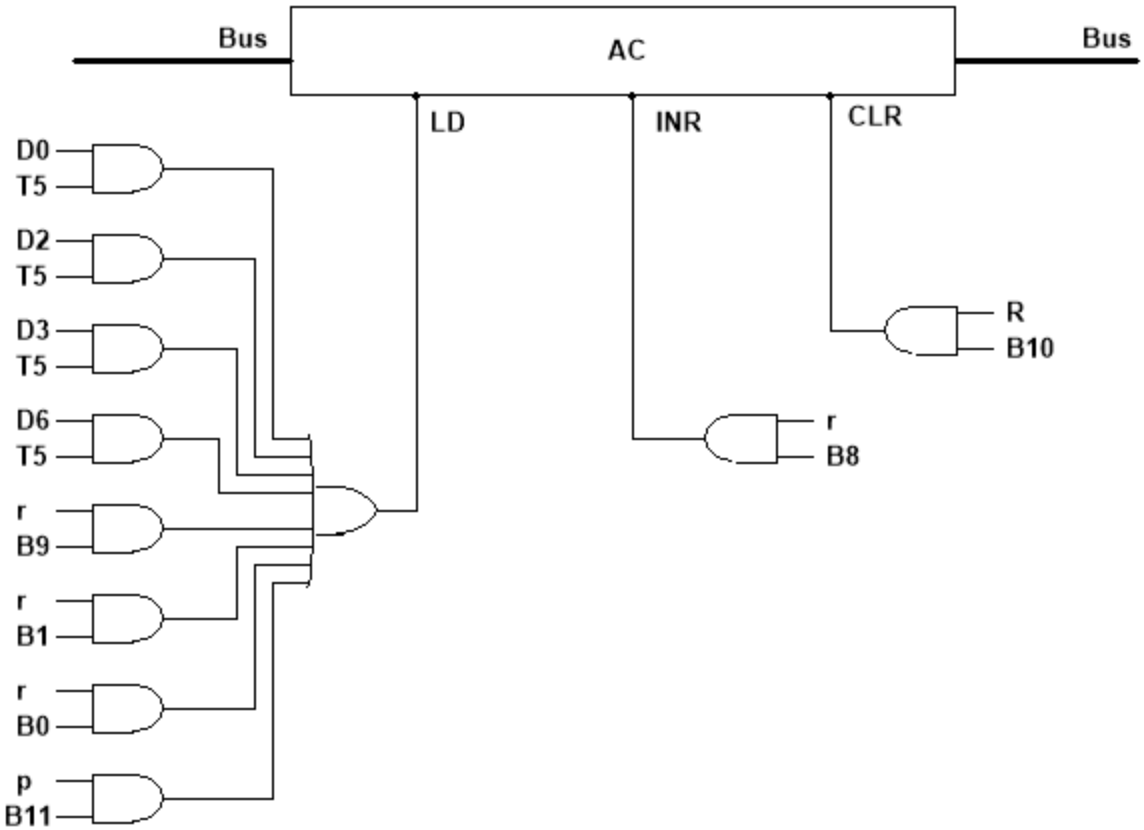


Figure 3.3 : Design of AC

Design of PC

LD:

$D_4.T_4: PC \leftarrow AR,\ SC \leftarrow 0$

$D_5.T_5: PC \leftarrow AR,\ SC \leftarrow 0$

CLR:

$R.T_1: M[AR] \leftarrow TR,\ PC \leftarrow 0$

INR:

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

$R.T_2: PC \leftarrow PC + 1,\ IEN \leftarrow 0,\ R \leftarrow 0,\ SC \leftarrow 0$

$r.B_7: If AC(18) = 0\ then\ PC \leftarrow PC + 1$

$r.B_6: If AC(18) = 1\ then\ PC \leftarrow PC + 1$

$r.B_5: If(AC = 0)\ then\ PC \leftarrow PC + 1$

$r.B_4: If(E = 0)\ then\ PC \leftarrow PC + 1$

$p.B_9: If\ (FGI = 1)\ then\ PC \leftarrow PC + 1$

$p.B_8: If\ (FGO = 1)\ then\ PC \leftarrow PC + 1$

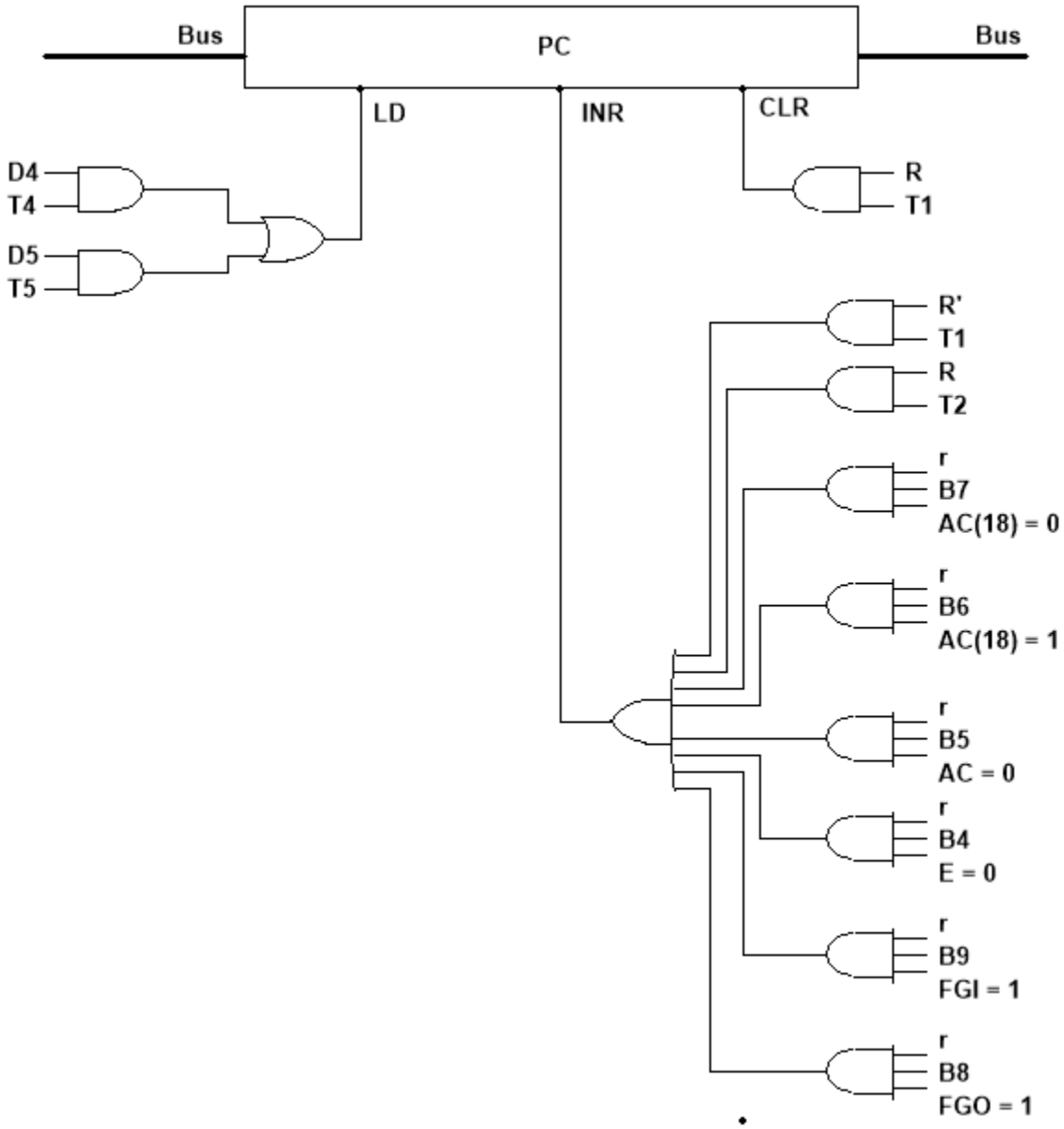


Figure 3.4 : Design of PC

Design of IR

LD:

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

CLR:

INR:

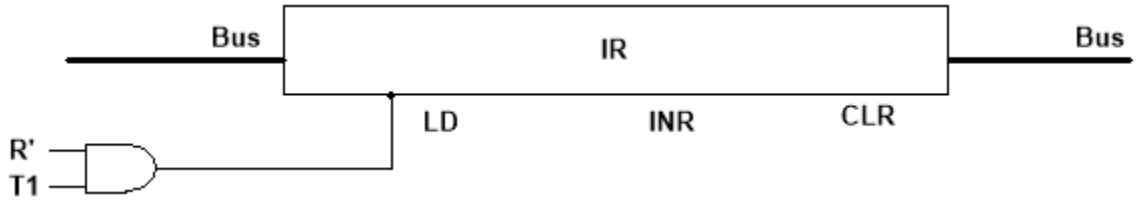


Figure 3.5 : Design of IR

Design of TR

LD :

$R.T_0: AR \leftarrow 0,\ TR \leftarrow PC$

INR :

CLR :

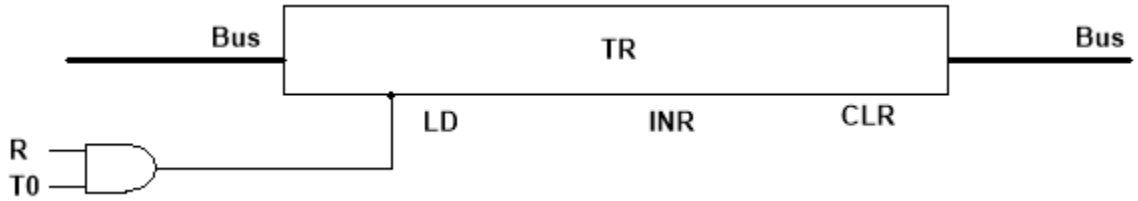


Figure 3.6 : Design of TR

Design of OUTR

LD :

$D_7.I_1.I_0.T_4$ is denoted by p

$p.B_{10}: OUTR \leftarrow AC(0 - 7),\ FGO \leftarrow 0$

INR :

CLR :

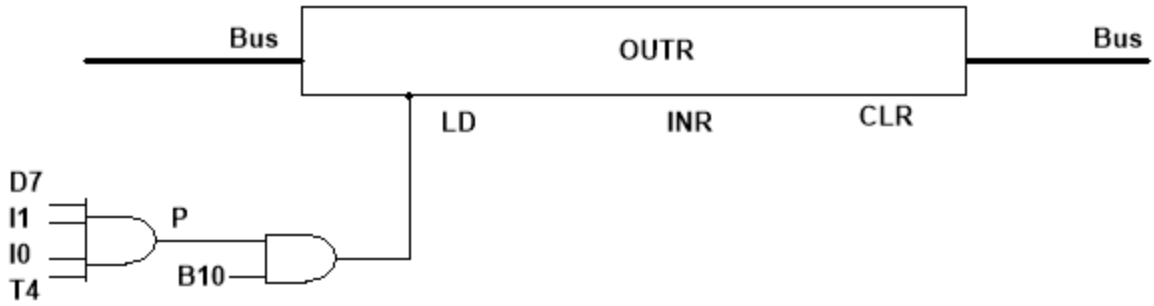


Figure 3.7 : Design of OUTR

Design of IEN

Set:

$p.B_7: IEN \leftarrow 1$

Reset:

$R.T_2: PC \leftarrow PC + 1,\ IEN \leftarrow 0,\ R \leftarrow 0,\ SC \leftarrow 0$

$p.B_6: IEN \leftarrow 0$

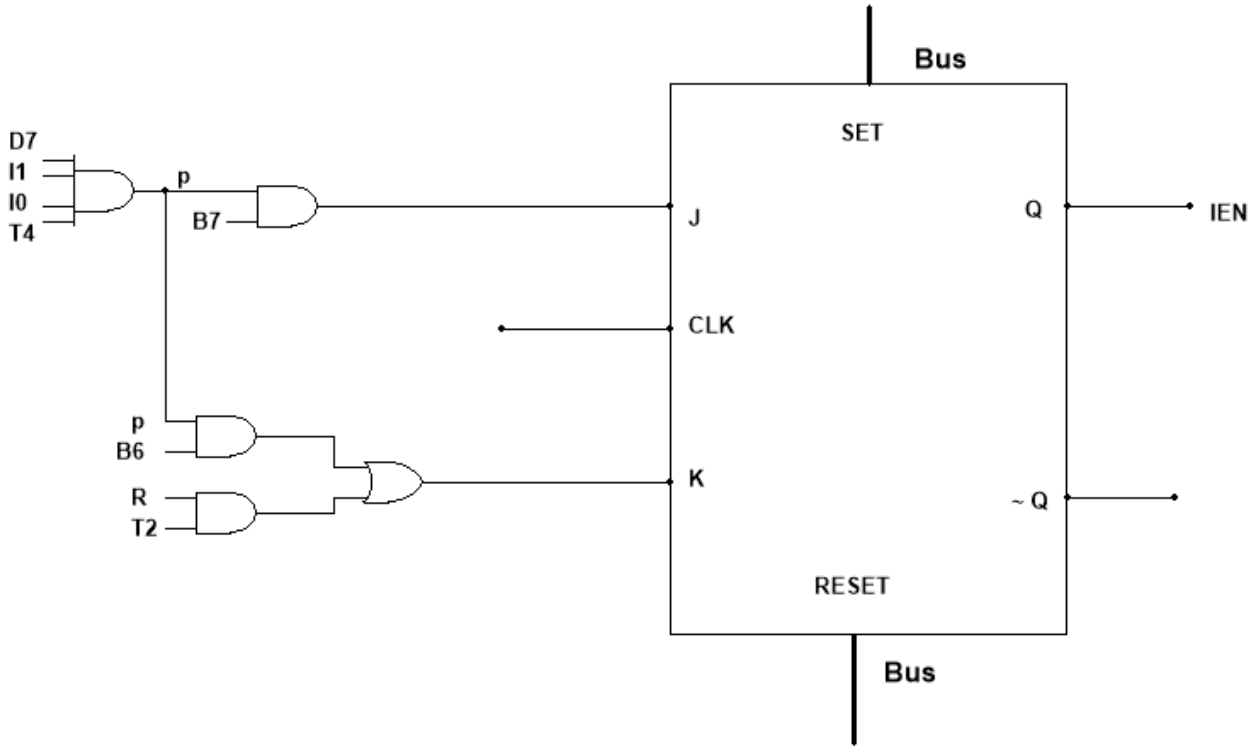


Figure 3.8 : Design of IEN

Design of E

LD :

$D_2.T_5: AC \leftarrow AC + DR,\ SC \leftarrow 0, E \leftarrow C_{out}$

$r.B_2: E \leftarrow E'$

$r.B_1: AC(18) \leftarrow E, E \leftarrow AC(0), AC \leftarrow shr(AC)$

$r.B_0:, AC(0) \leftarrow E, E \leftarrow AC(18),\ AC \leftarrow shl(AC)$

RESET :

$r.B_3: E \leftarrow 0$

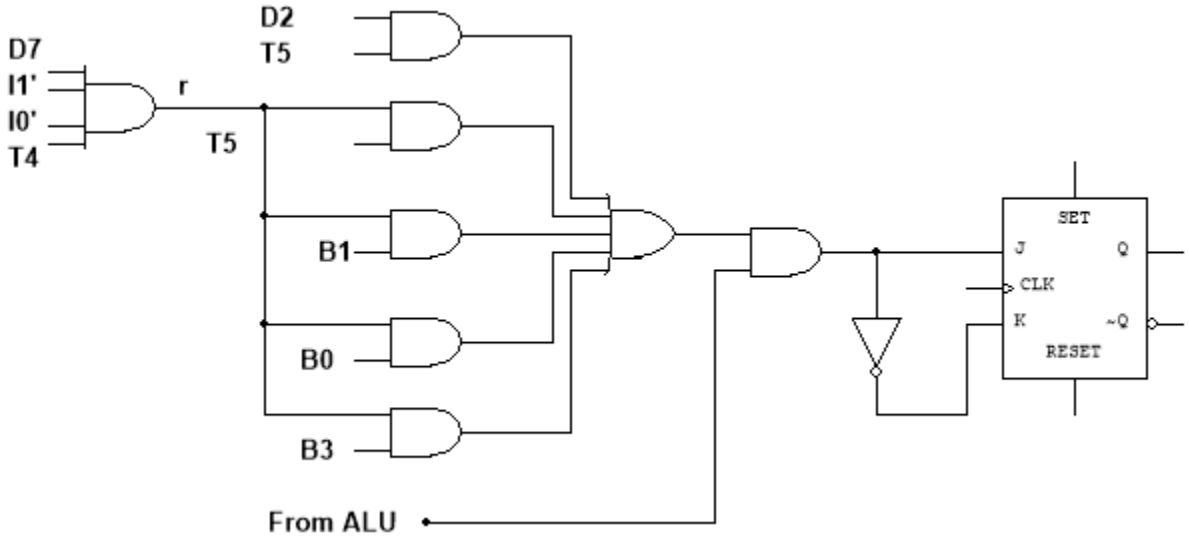


Figure 3.9 : Design of E

Design of R

SET :

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

RESET:

$R.T_2: PC \leftarrow PC + 1,\ IEN \leftarrow 0,\ R \leftarrow 0,\ SC \leftarrow 0$

INR :

CLR :

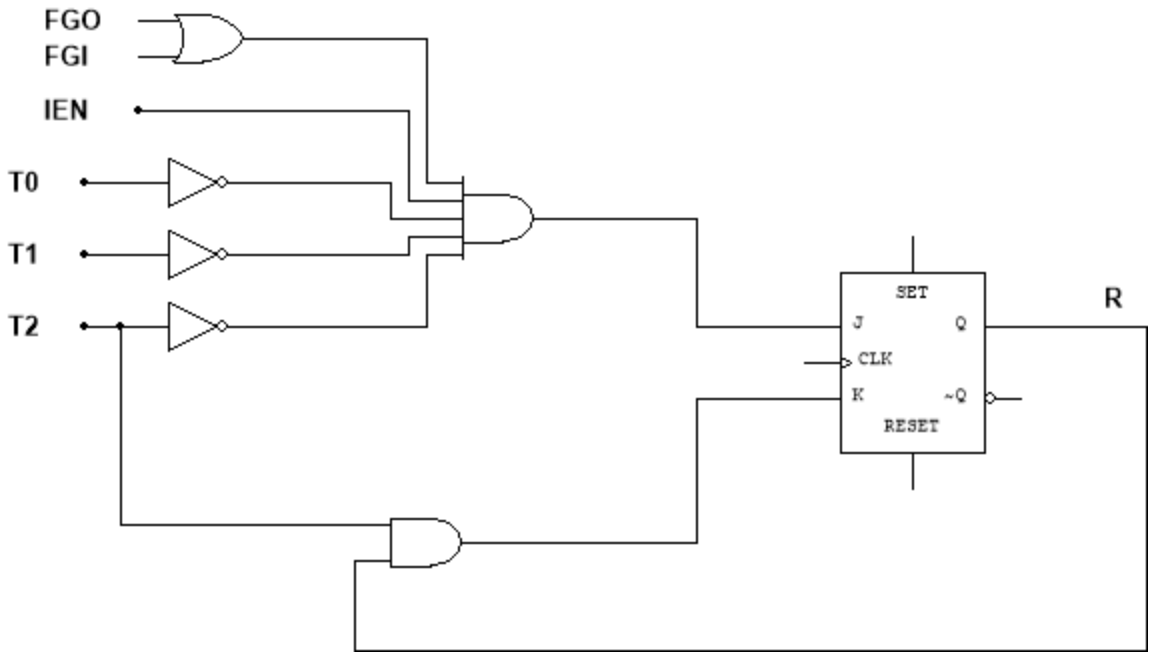


Figure 3.10 : Design of R

Design of S

LD :

INR :

CLR :

$r.B_{11}: S \leftarrow 0$

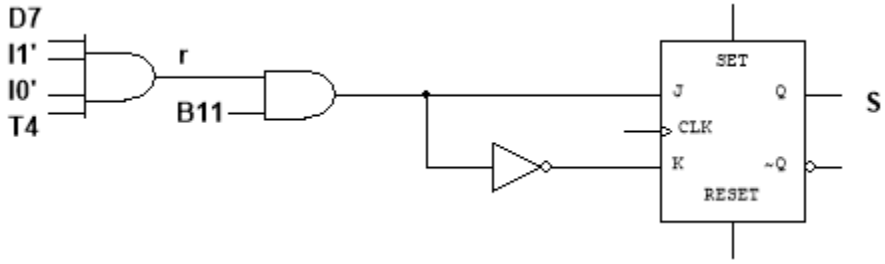


Figure 3.11 : Design of S

Design of FGI

LD :

INR :
CLR :

$p.B_{11}: AC(0 - 7) \leftarrow INPR,\ FGI \leftarrow 0$

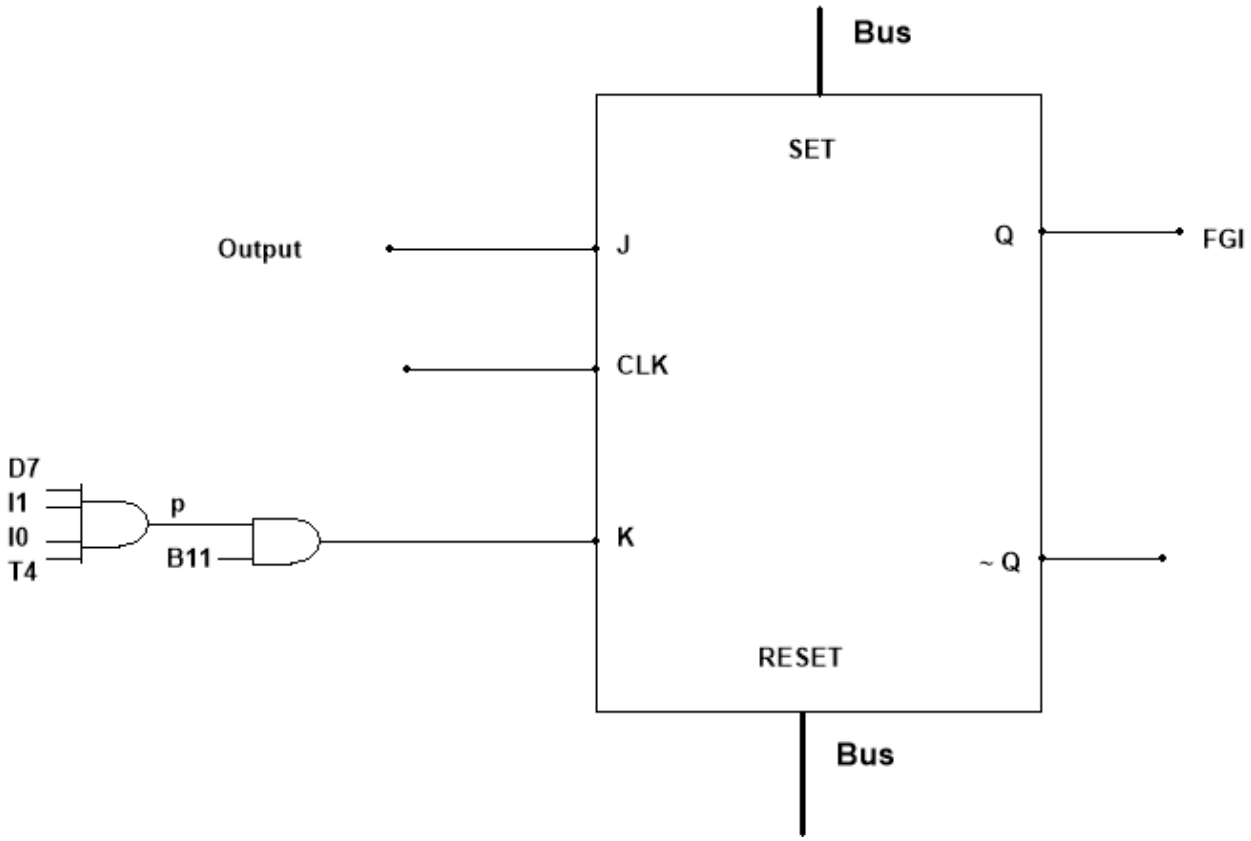


Figure 3.12 : Design of FGI

Design of FGO

LD :

INR :
CLR :

$p.B_{10}: OUTR \leftarrow AC(0 - 7),\ FGO \leftarrow 0$

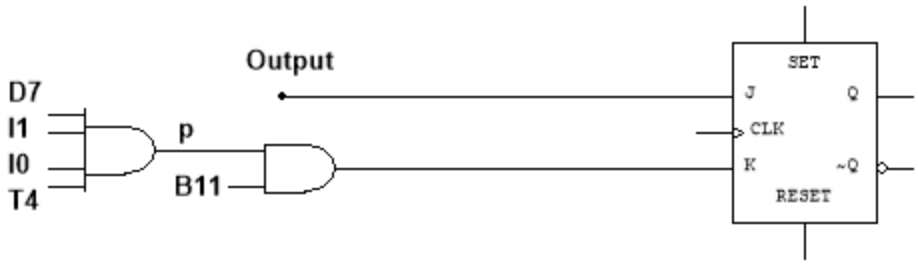


Figure 3.13 : Design of FGO

Design of SC

LD :

INR :

CLR :

$R.T_2: PC \leftarrow PC + 1,\ IEN \leftarrow 0,\ R \leftarrow 0,\ SC \leftarrow 0$

$D_0.T_5: AC \leftarrow DR,\ SC \leftarrow 0$

$D_1.T_4: M[AR] \leftarrow AC,\ SC \leftarrow 0$

$D_2.T_5: AC \leftarrow AC + DR,\ SC \leftarrow 0, E \leftarrow C_{out}$

$D_3.T_5: AC \leftarrow AC \wedge DR,\ SC \leftarrow 0$

$D_4.T_4: PC \leftarrow AR,\ SC \leftarrow 0$

$D_5.T_5: PC \leftarrow AR,\ SC \leftarrow 0$

$D_6.T_5: AC \leftarrow (AC \wedge DR)',\ SC \leftarrow 0$

$r: SC \leftarrow 0$

$p: SC \leftarrow 0$

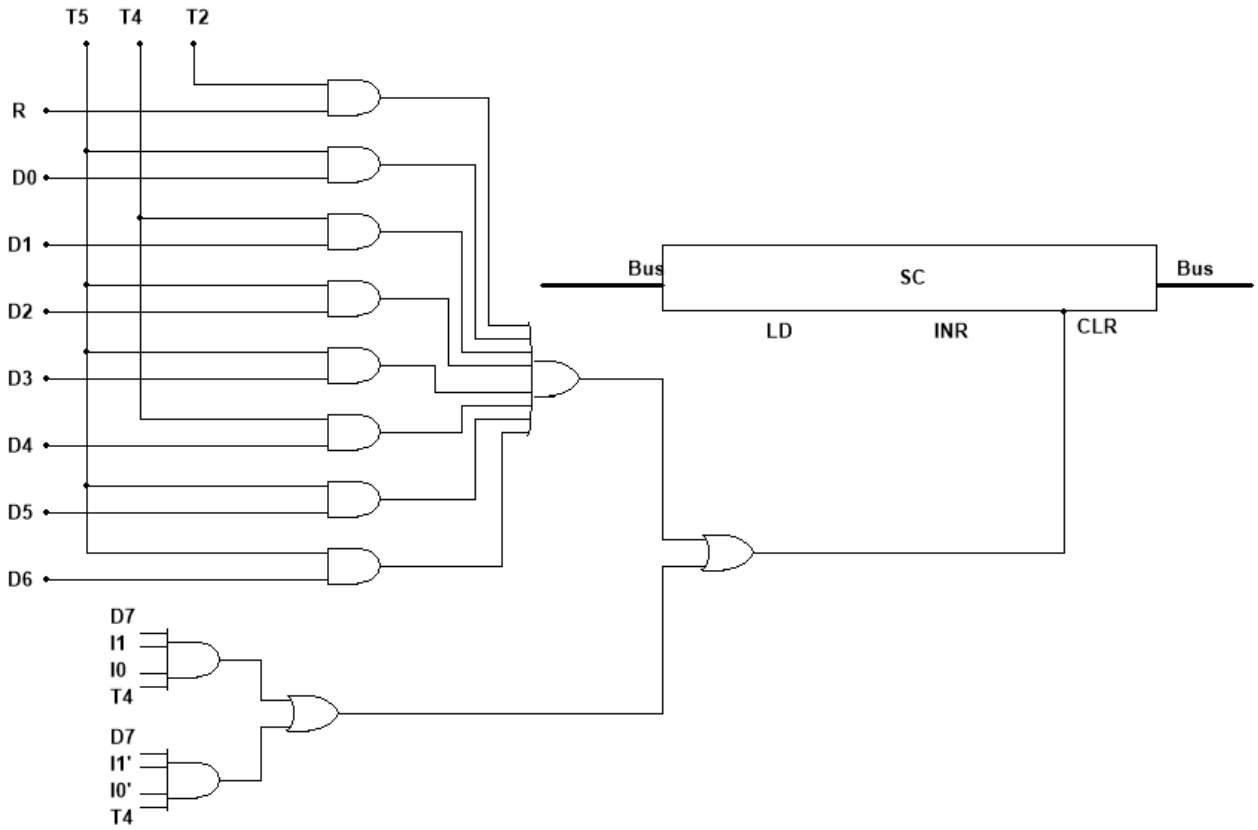


Figure 3.14 : Design of SC

Control of Common Bus

Inputs								Outputs			Register selection
X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	S_2	S_1	S_0	
1	0	0	0	0	0	0	0	0	0	0	None
0	1	0	0	0	0	0	0	0	0	1	PC
0	0	1	0	0	0	0	0	0	1	0	AC
0	0	0	1	0	0	0	0	0	1	1	AR
0	0	0	0	1	0	0	0	1	0	0	DR
0	0	0	0	0	1	0	0	1	0	1	Memory
0	0	0	0	0	0	1	0	1	1	0	IR

0	0	0	0	0	0	0	1	1	1	1	TR

Table 3.1 : Control of common Bus

Memory

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

$D_7'.I_1.I_0'.T_3: AR \leftarrow M[AR]$ (For Indirect AM)

$D_0.(I_0 \oplus I_1).T_4: DR \leftarrow M[AR]$

$D_2.(I_0 \oplus I_1).T_4: DR \leftarrow M[AR]$

$D_3.(I_0 \oplus I_1).T_4: DR \leftarrow M[AR]$

$D_6.(I_0 \oplus I_1).T_4: DR \leftarrow M[AR]$

AC

$D_1.T_4: M[AR] \leftarrow AC,\ SC \leftarrow 0$

$p.B_{10}: OUTR \leftarrow AC(0 - 7),\ FGO \leftarrow 0$

DR

$D_0.T_5: AC \leftarrow DR,\ SC \leftarrow 0$

IR

$R'.T_2: I_1 \leftarrow IR(18), I_0 \leftarrow IR(17),\ D_0 \ldots D_7 \leftarrow IR(16 - 14),\ AR \leftarrow IR(13 - 0)$

AR

$D_7'.I_1.I_0.T_3: DR \leftarrow AR$ (For Immediate AM)

$D_4.T_4: PC \leftarrow AR,\ SC \leftarrow 0$

$D_5.T_5: PC \leftarrow AR,\ SC \leftarrow 0$

PC

$R'.T_0: AR \leftarrow PC$

$R.T_0: AR \leftarrow 0,\ TR \leftarrow PC$

$D_5.T_4: M[AR] \leftarrow PC,\ AR \leftarrow AR + 1$

TR

$R.T_1: M[AR] \leftarrow TR,\ PC \leftarrow 0$

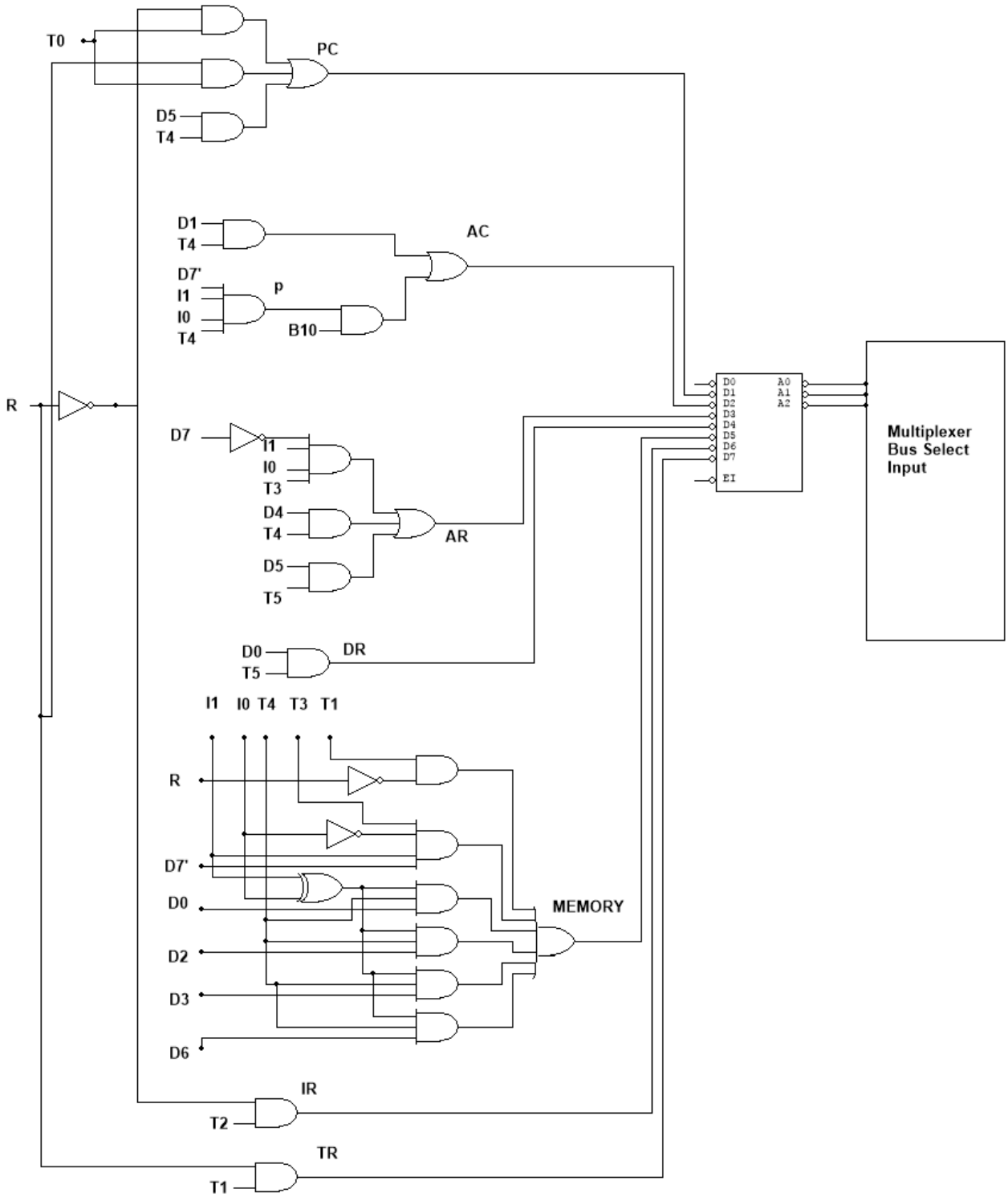


Figure 3.15 : Design of Control of Common Bus

Design of Memory

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

$D_7'.I_1.I_0'.T_3: AR \leftarrow M[AR]$ (For Indirect AM)

$D_0.(I_0 \oplus I_1).T_4: DR \leftarrow M[AR]$

$D_2.(I_0 \oplus I_1).T_4: DR \leftarrow M[AR]$

$D_3.(I_0 \oplus I_1).T_4: DR \leftarrow M[AR]$

$D_6.(I_0 \oplus I_1).T_4: DR \leftarrow M[AR]$

$R.T_1: M[AR] \leftarrow TR,\ PC \leftarrow 0$

$D_1.T_4: M[AR] \leftarrow AC,\ SC \leftarrow 0$

$D_5.T_4: M[AR] \leftarrow PC,\ AR \leftarrow AR + 1$

.

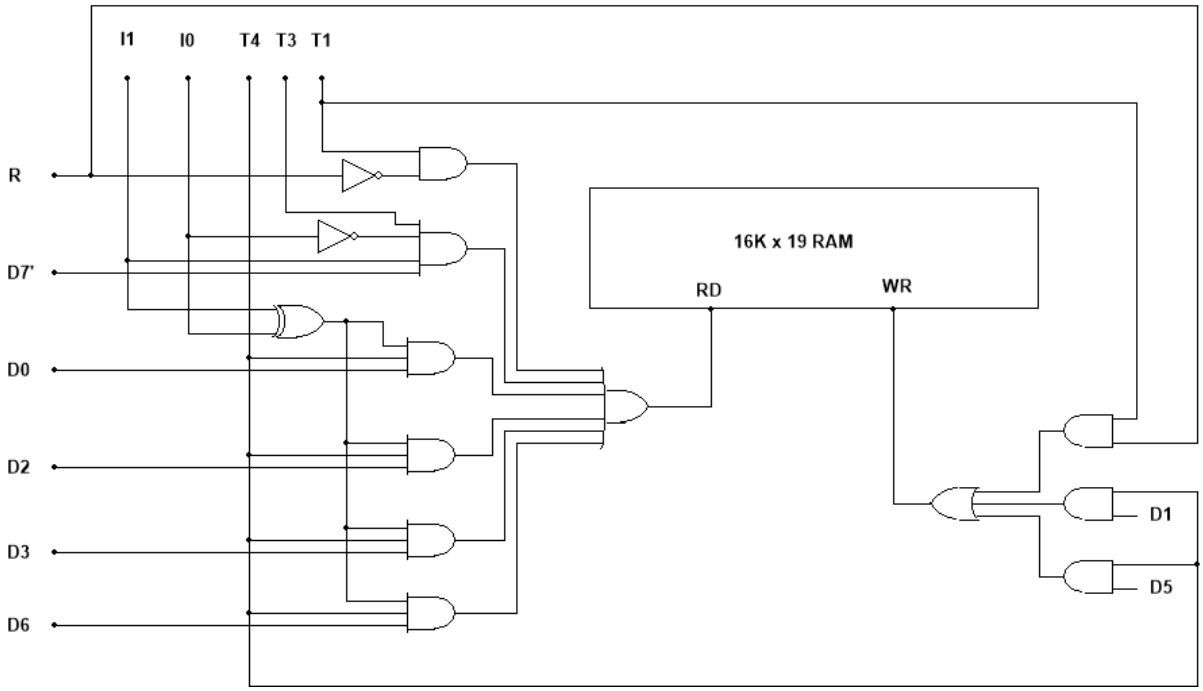


Figure 3.16 : Design of Memory

Design of ALU

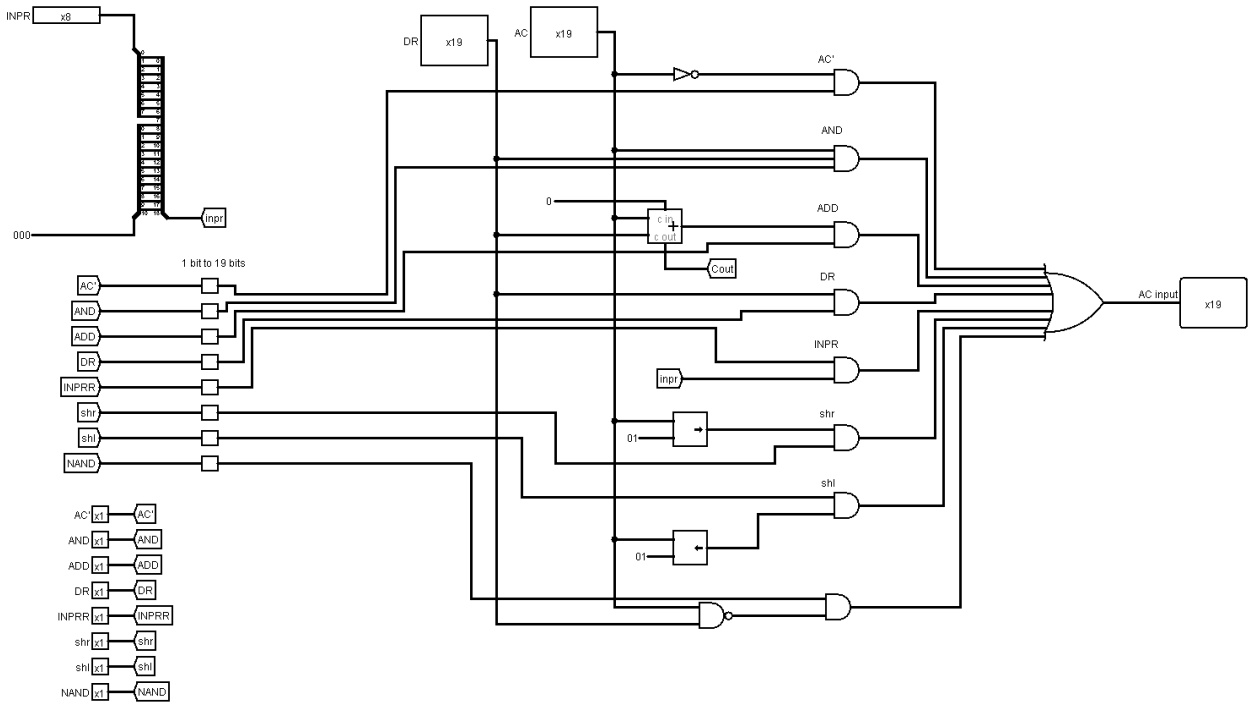


Figure 3.17 : Design of ALU

Control Unit

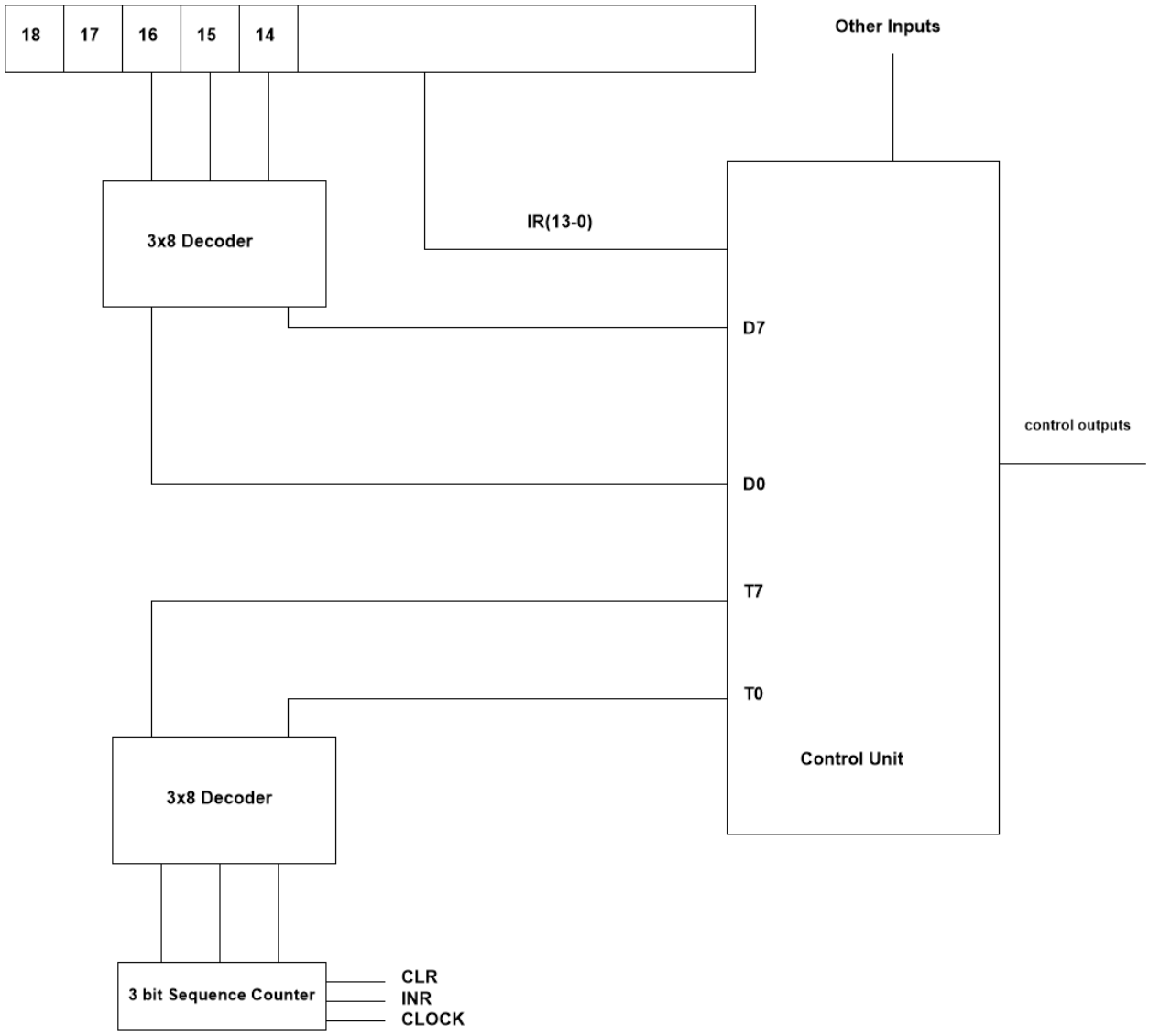


Figure 3.18 : Design of CU

Common Bus System

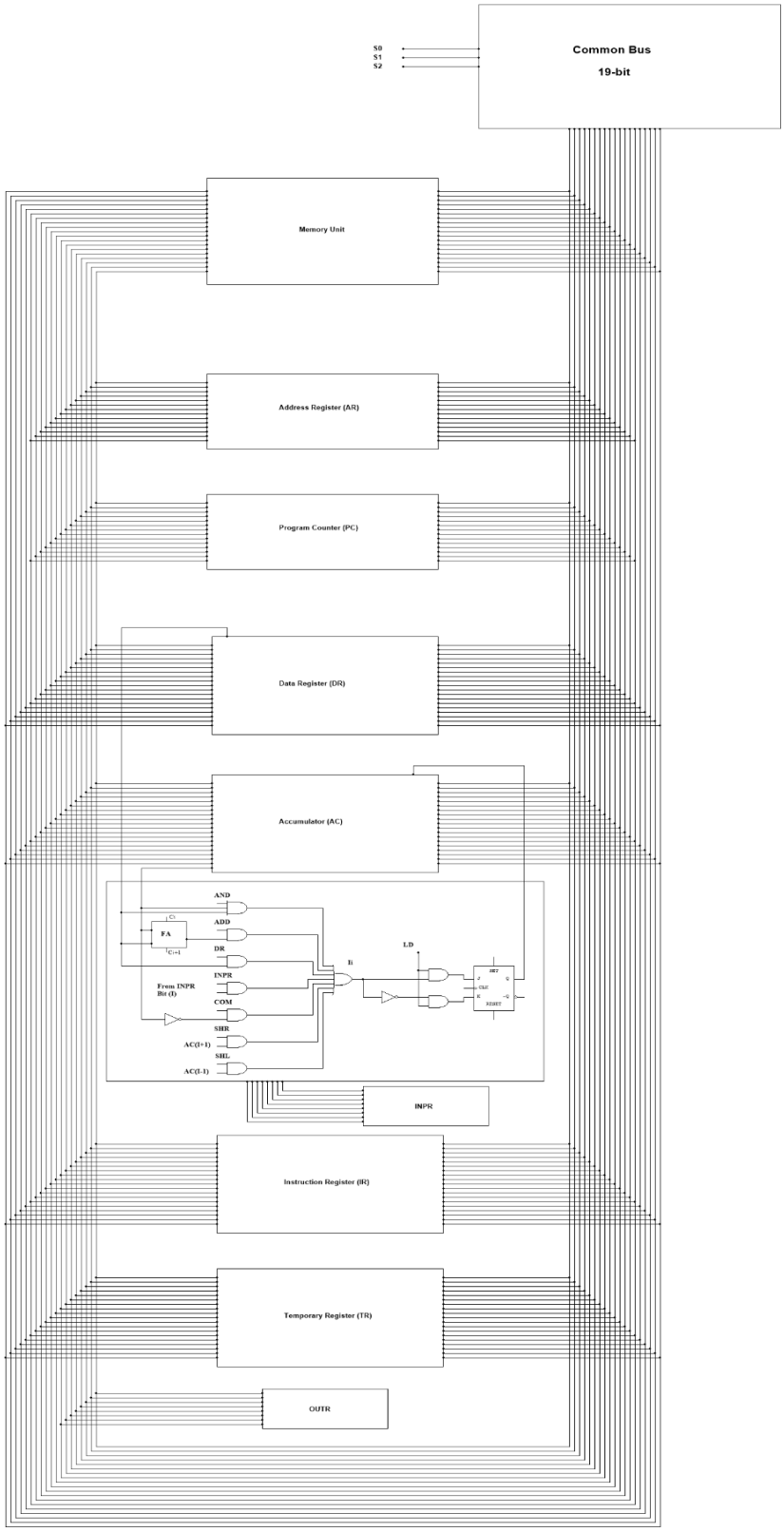


Figure 3.19 : Design of Common Bus System

Overall Architecture of SAAJA Basic Computer

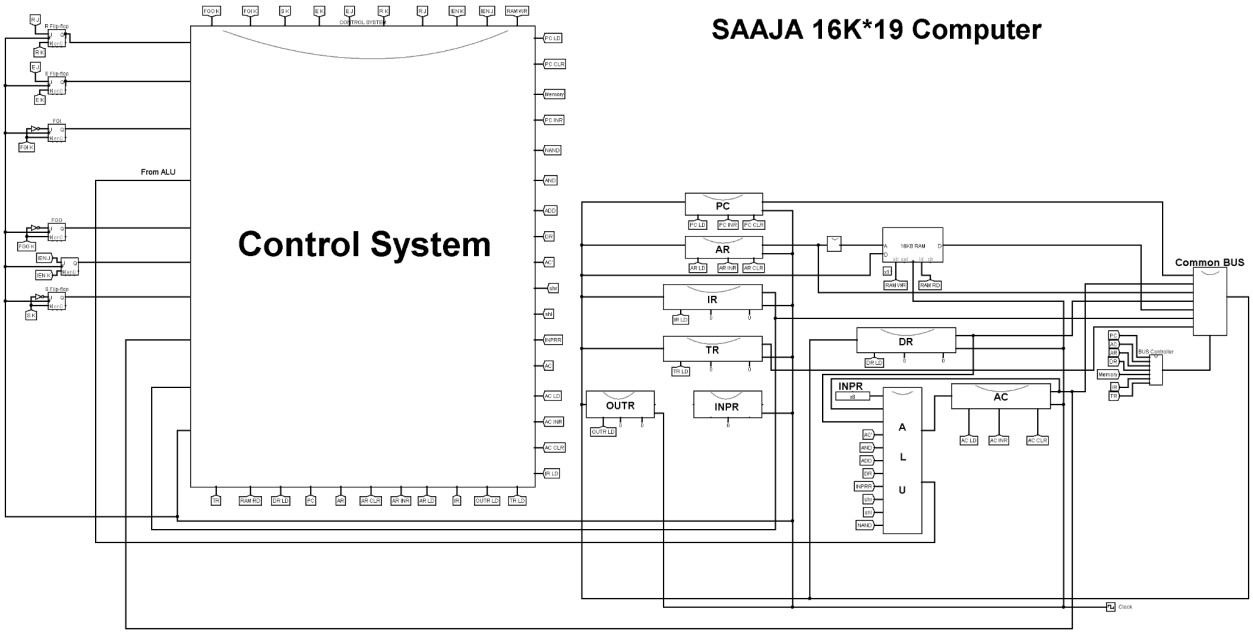


Figure 3.20 : Design of SAAJA Computer

Chapter 4 : Conclusion

This is a basic Computer which acts as a prototype for the commercial computer and help us understand the architecture of the computer. This basic computer help us understand the basic fetch-decode-execute cycle and how computers actually operate under the hood.The simple architecture was combined with an ISA and an assembly language, with emphasis given to the relationship between these two, allowing us to write programs for this simple computer. Bare essential components of the computer such as ALU and control unit was implemented and understood as a part of the design process for the computer. The design presented in this project contains a very simple architecture designed specially to illustrate the concepts without bogged down in too many technical details. It has 16K 19-bit words of main memory, uses 19-bit instructions and has seven registers. The simple computer design is able to accomplish instructions that are illustrated in basic instructions completeness. With this project, we are able to understand what makes a basic computer operate and allows us to make basic instructions run and operate.