INTRODUCTION

A basic computer can be understood as a hardware unit that is capable of simple arithmetic and logical calculation as per the instruction provided by the user. Calculator can be taken as example of basic computer.

The internal architecture of a basic computer system is defined by logic gates in the most fundamental level. Composed of those gates, registers, memory, various buses, etc. are used to build a computer.

The internal organization of a digital system is defined by the sequence of micro-operations that are performed on the data stored in memory and different registers. Instruction is sent to a computer by a set of bits which specify the micro-operation to be performed, the address of the operand and other fields. A defined list of instructions is already specified in the computer to execute basic arithmetic and logical operations.

In our mini project we have developed an basic computer of 256*12 bit. It states that our computer uses 12 bit as a instruction length and our computer is capable of loading 256 different instruction at same time.
The functionality, organization and implementation of our computer is included in this report.

DISCUSSION

COMPONENTS OF COMPUTER

Below are the list of the components used in our basic computer.

- RAM (Memory)
- Registers
    1. Instruction Register- 12bit
    2. Program Counter- 8bit
    3. Data Register-12bit
    4. Accumulator-12bit
    5. Address Register-8 bit
    6. Input Register-8 bit
    7. Output Register-8 bit
- Arithmetic and logical Unit (ALU)
- Control Unit
- Decoders
- Encoders
- Address Selector
- Instruction Selector
- Sequence Counter
- System Bus

Component Descriptions:

- **Memory (RAM):** Memory stores both the instruction and the data on which processing is to be performed. We have selected a memory of 256 words with a word size of 12-bitsfor our computer. This means that we require 8 bit address line to address all the words in the memory. Memory receives its address from the Address Register (AR), it

receives/send its data from the common bus system, read or write operations is distinguish from the control signal.

- **Registers:** There are 7 different types of Registers used in our system and all of them have similar configuration.

1. **Accumulator:** Accumulator is the main register of this computer. It is an 12-bit general purpose resister and stores the result of any Arithmetic or Logical operation. Input of the accumulator is linked with the output of the arithmetic and logic unit. The input of accumulator is not directly interface with the Common Bus System. Output of the AC goes to the common bus system as well as to ALU.

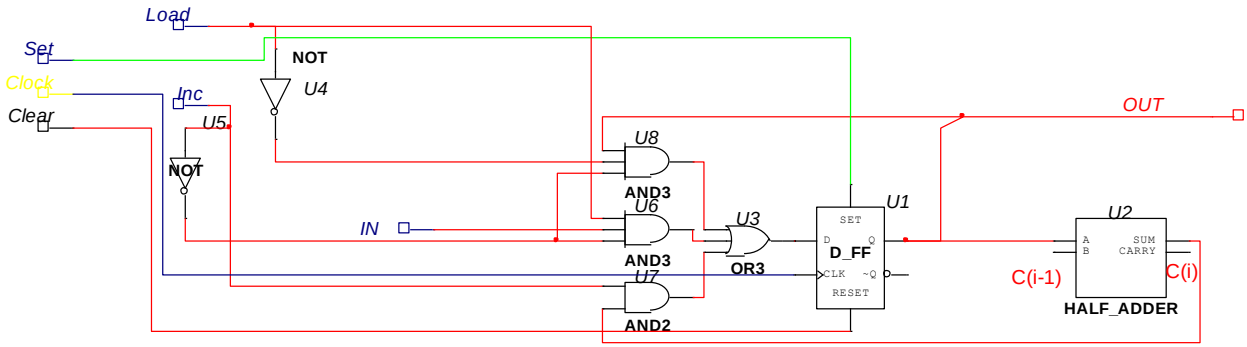


Fig: Single unit of Accumulator

2. **Instruction Register:** Instruction Register stores the 12-bit instruction read from the memory. At the start of each instruction cycle, the address contained in program counter is transferred to AR, and the content of the memory location pointer by AR is then transferred to the Instruction register through the common bus system. Once the instruction is loaded in the IR it is then decoded to obtain the decoded lines (D0-D7) which along with the timing signals are used to provide the necessary control signals for the computer to execute the stated operation.

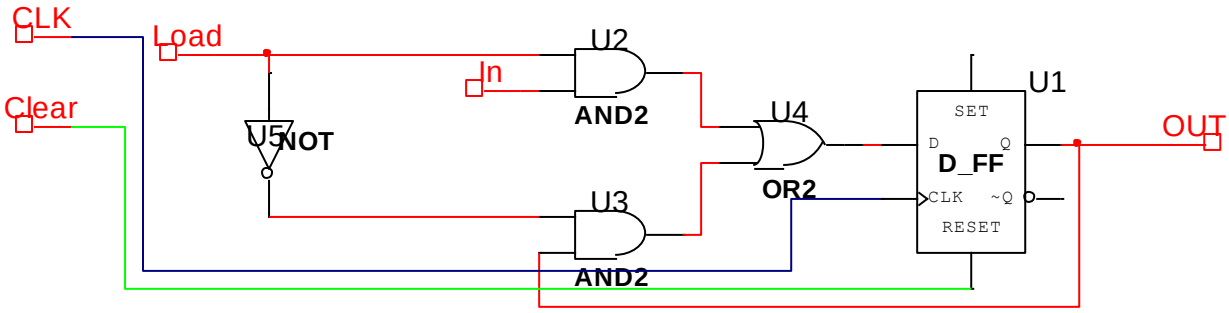


Fig: Single Unit of Instruction Register

3. **Data Register:** Data register holds the operand read from memory. It is 12-bit register that is used to store operand for any operations. Output of data register is interfaced with the ALU. Any valued that is to be added, anded etc. must at first be

transferred to the Data register. It is connected to common bus system for in or out flow of data.

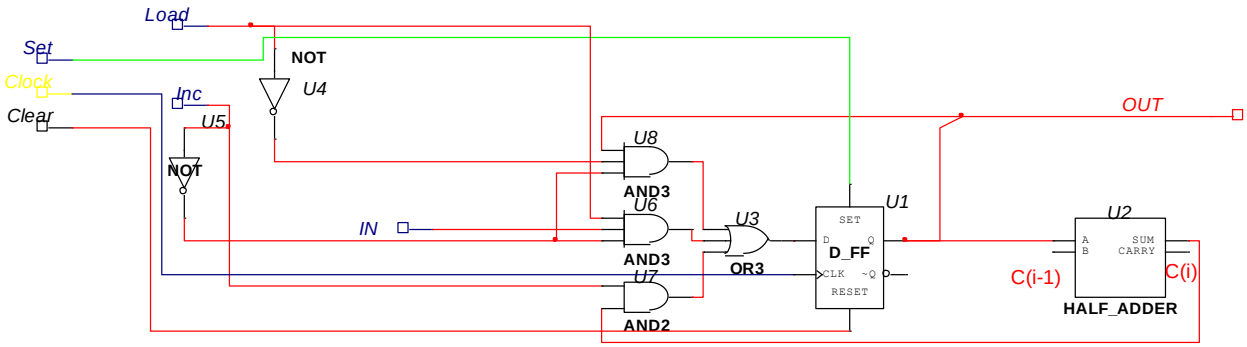


Fig: Single Unit of Data Register

4. **Address Register:** Address register is used to store the address of the memory for read from the memory and write into memory. It is 8-bit register that is used for storing the address of the instruction to fetch instruction from memory and to write the output data in certain address of memory it is connected to common bus system for in and out flow of bits. It is also directly interfaced with memory for pointing address.

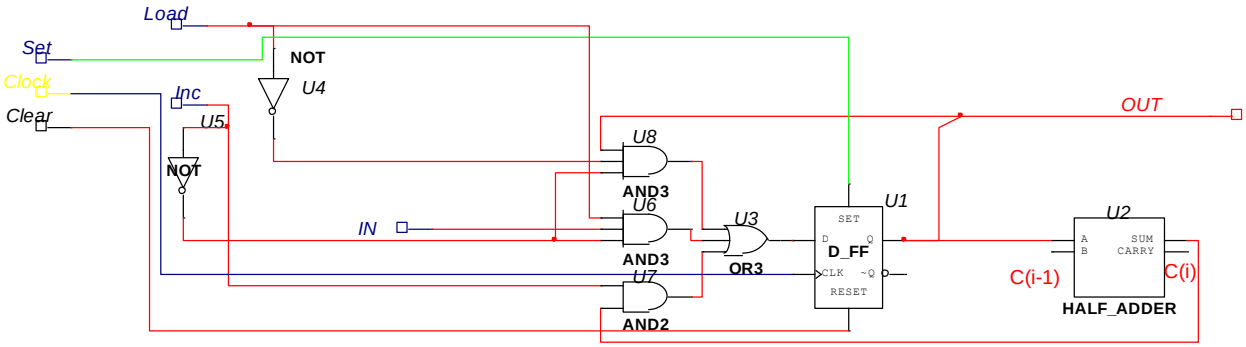


Fig: Single unit of Address Register

5. **Program Counter:** Program counter is a 8-bit register that contains the address of the memory location where the next instruction is located. At the beginning of each instruction cycle (fetch), the content of program counter is transferred to AR and then is incremented by one to point to next consecutive location It is interfaced with common bus system. .

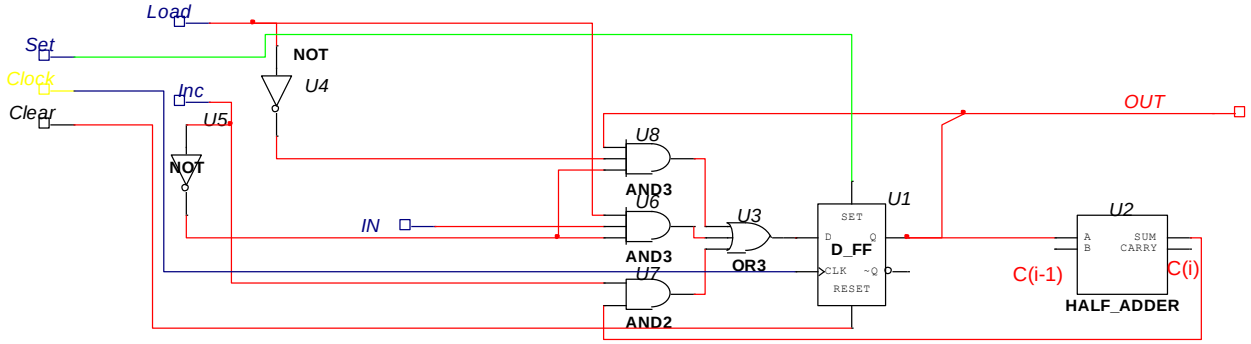


Fig: Single unit of Program Counter

6. **Input Register:** It is used to read input from user. It is 8-bit register as most data are stored in 8 bit system. It is interfaced with ALU for direct input.

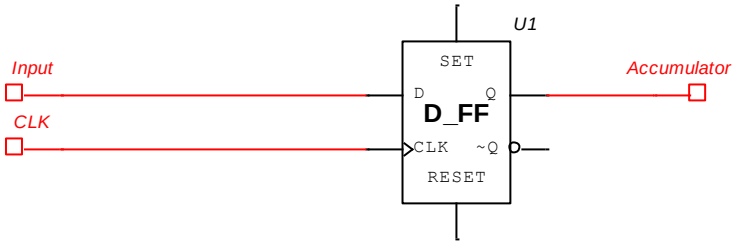


Fig: Single unit of Input Register

7. **Output Register:** It is 8 bit register that is used to display the desired output after the calculation. It is interfaced with system bus for data.

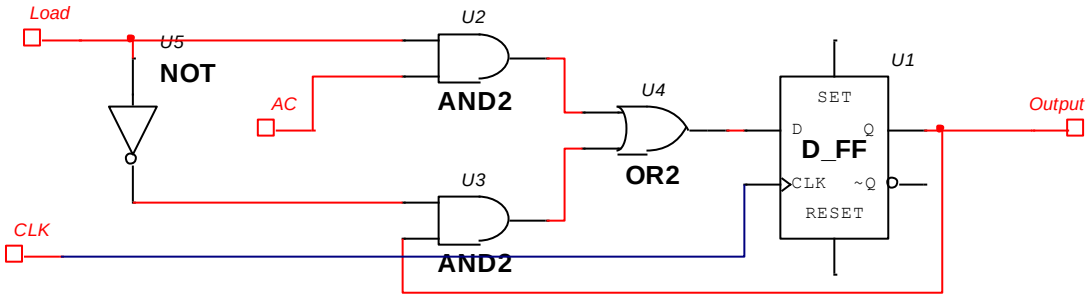


Fig: Single unit of Input Register

Arithmetic and Logical Unit (ALU): Arithmetic and logic unit in our computer design is of 12 bit and performs 7 main operations:

i. AND

ii. ADD

iii. Complement

iv. Load from Data Register

v. Read Input from Input Register

vi. Circular SHIFT LEFT

vii. Circular SHIFT RIGHT

Arithmetic and logic unit takes three inputs: accumulator (operand 1), data register (operand 2) and directly from the input register(INPR) for transfer operation to the AC. It performs Arithmetic and Logic operation and then sends its output directly to AC. It also gives one extra bit as output for carry from calculation.
Below is the unit state of ALU:

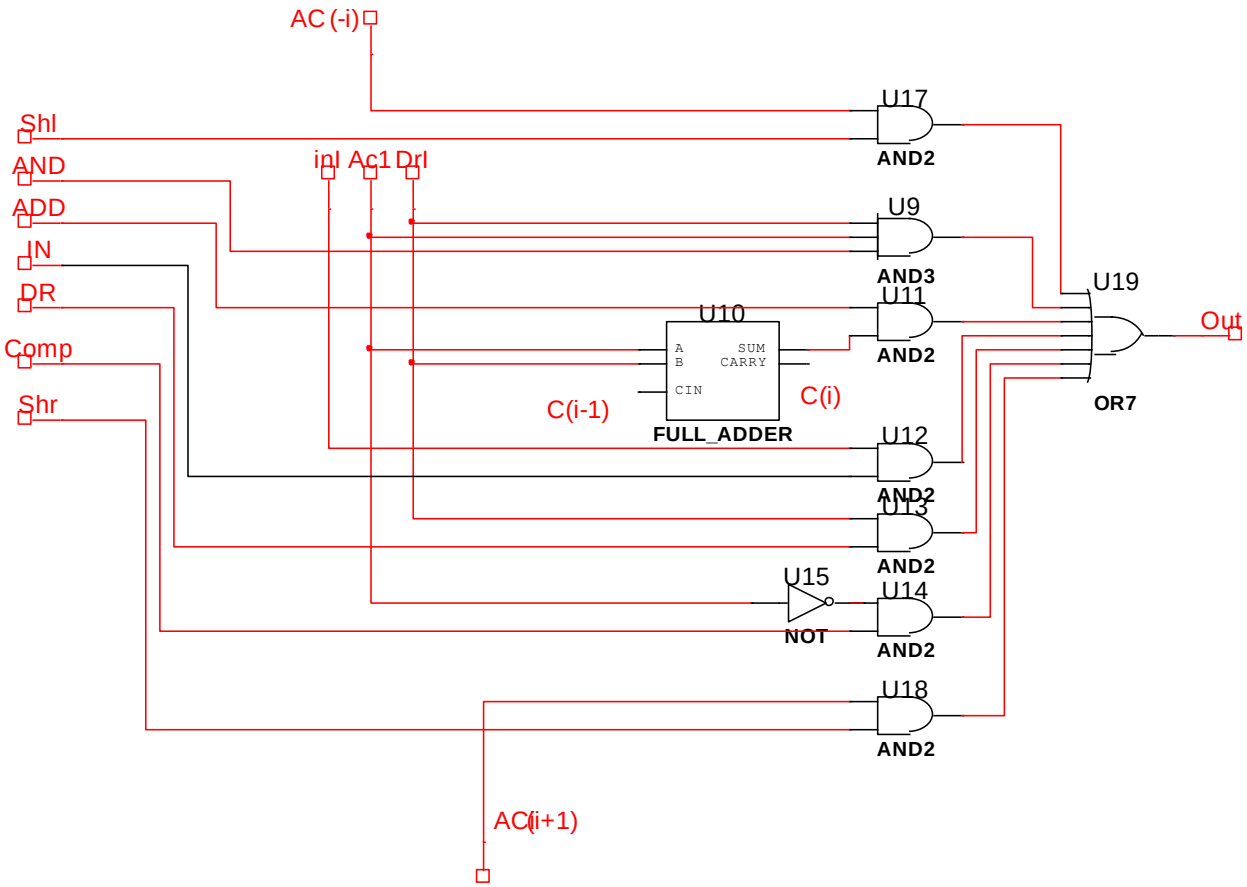


FIG: Unit State of ALU

- **Control Unit:** It is the unit to control the flow of the micro-operation for performing the requested instruction. Brief discussion is done below. Control unit (CU) generates the necessary control and timing signals to carry out the sequences of micro-operations in order to execute the given instruction.

- **Decoders**: it is used to decode the opcode into (D0-D7) outputs. As we have opcode of3 bit length the decoder of 3*8 is used it is also used to decode time from sequence counter into (T0-T7) individual time unit.

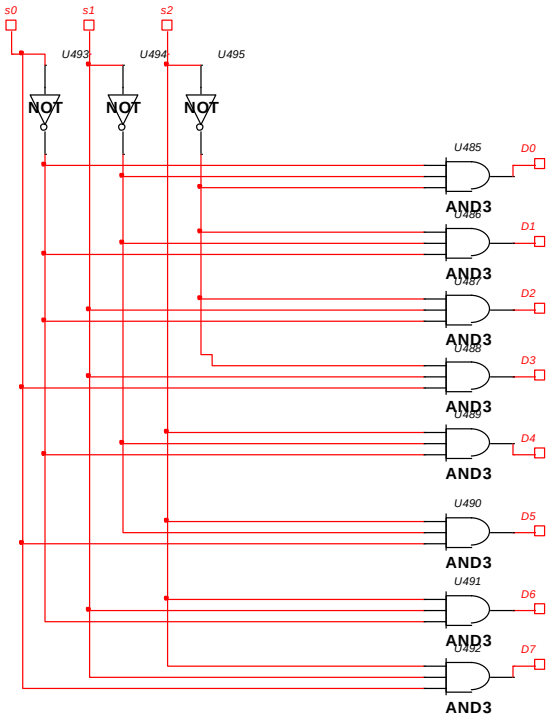


Fig: Decoder

- **Encoders:** It is used to encode the selection line from control unit to the system bus for selection of data in common bus system.

- **Address Selector:** It is the unit used for the selection of address of memory either to enter manually or from the address register as per the program flow.

- **Instruction Selector:** It is the unit used for the selection of instruction to write into memory either to enter manually or from the system bus as per the program flow.

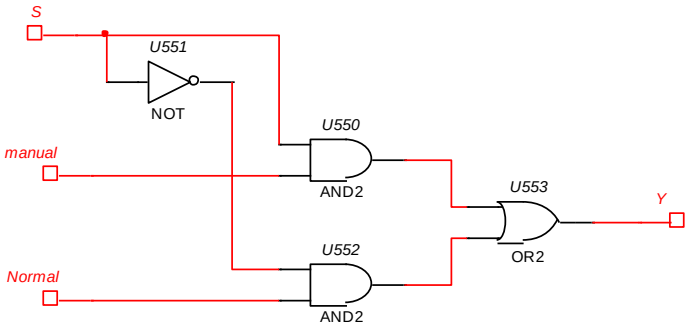


Fig: Single unit of address and instruction selector

- **Sequence Counter:** It is the unit that develops the time cycle as per the clock transition. We have used normal Synchronous up counter which counts from 0 to 7.

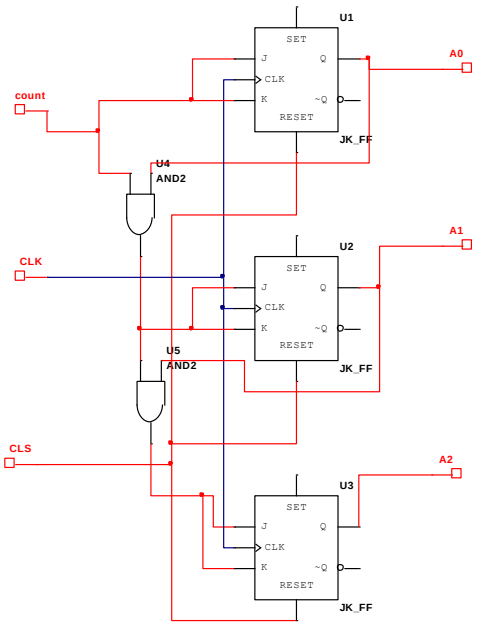


Figure: Sequence Counter

- **Common System Bus:** Common System Bus is the unit that performs the task pf flowing the data bat from one register to next or from memory to register and vice versa as per the control statement.

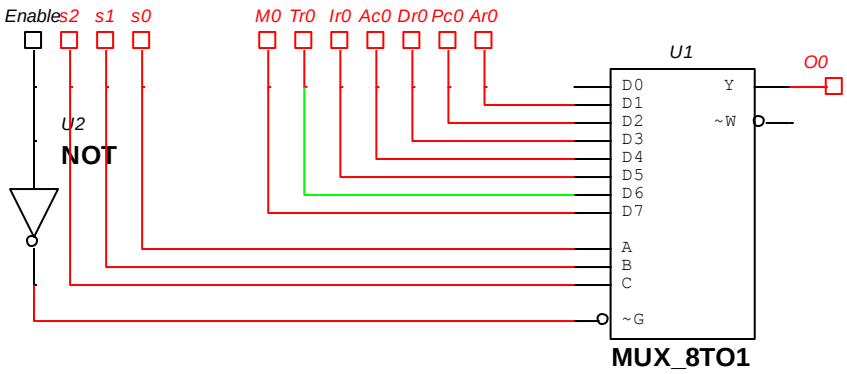


Fig: Single unit state of common Bus system.

*since the Mux need 0 as enable so not gate is used.

(*Block diagram of the computer with circuit is attached with report)

INSTRUCTION FORMAT

As we have 12 bit computer the instruction has 12 bit length and the 12 bit length is divided into two parts address part which is 8 bit and opcode part which is 4 bit long.

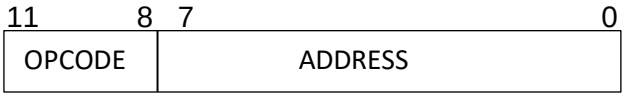


Fig: Instruction Format

Opcode: The upper 4 bits are defined as the opcode for the instruction. The opcode are the combination of 4 bits that defines the micro-operation to be performed during instruction execution. It differs on the type of instruction and they are

1. **Memory Reference Instruction (MRI):**
   In memory reference instruction most significant-bits read from the memory during the fetch cycle is used to specify the direct or indirect addressing mode and the consecutive 3 bit is used to specify memory operation, and next consecutive 8 bits read from the memory contains the memory address of the operand. The opcode of the MRI varies from 000 to 110. Instruction code format for the MRI is.

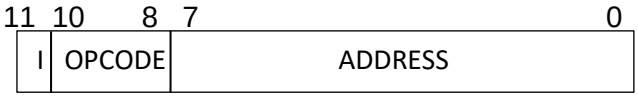


Fig: Instruction Format RRI

2. **Register Reference Instruction (RRI):**
   In RRI the opcode of the instruction including I bit is always fixed. It is always 0111. The opcode 111 defines is same for RRI and Input/output instruction and it is distinguished by the 12th bit i.e. I bit where 0 specifies RRI and 1 specifies I/O instruction. The instruction format for MRI is:

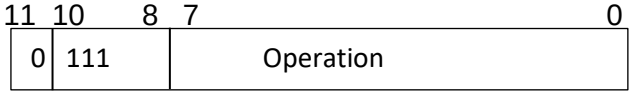


Fig: Instruction Format for RRI

3. **Input / Output Instruction (I/O Operation) :** Input/output instructions are required in order to establish the communication between the computer and the outside world. There are only 2 input output operations one for getting the input from the outside world and another for sending output to the LED display. Instruction Format for Input/output instruction code is given below:

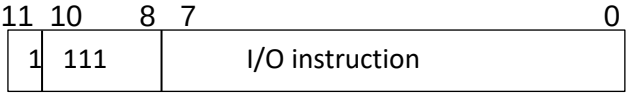


Fig: Instruction Format Input Output

INSTRUCTION CYCLE and CONTROL SIGNALS

Fetch and Decode Cycle

Fetch Cycle:

During the fetch cycle the instruction is read from the memory and loaded in the instruction register. For this, the address of the next location containing the instruction in the memory is contained in program counter. This address is transferred to Address Register during the first T_0 timing signal. On the next timing signal the content of the memory pointed by the address register is copied to the instruction register in the meantime, Program counter as well as address register is incremented by one. Following RTL denotes the fetch cycle:

T_0: **AR ← PC**

T_1: **IR ← M[AR], PC ← PC + 1, AR ← AR + 1**

Decode Cycle

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

T_2: **D_0 … D_7 ← Decode [IR(12-14)], I ← IR(15), AR ← M[AR]**

Indirect

After decoding, if I is found to be 1, then the content of AR becomes effect address which stores the true address of operand. So, effective address is transferred to AR.

T_3: **AR← M[AR**

Table: Micro-operation for fetch and decode cycle

Cycle	Control Signal	RTL	Description
Fetch	T_0	**AR ← PC**	Load AR with the address of new instruction to be read from the memory
	T_1	**IR ← M[AR], PC ← PC + 1**	Copy the instruction from the memory to the instruction register and increase program counter.
Decode	T_2	**D_0…D_7←Decode[IR(1214)], I ← IR(15), AR ← M[AR]**	Decode the Instruction
Indirect	T_3	**AR← M[AR]**	Indirect addressing

Execution Cycle

Memory Reference Instruction (MRI)

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

   D_0T_4: DR ← M[AR]

   D_0T_5 AC ← AC^DR,SC ← 0

2. **ADD:** During the first cycle the operand specified by the address in copied to the DR. On the next cycle the content of DR is added to the current content of the AC and the result is stored back to AC. If any carry is generated than the carry flag is set. Operation of adding two data is performed inside the ALU using cascaded full-adders. RTL for this instruction is:

   D_1T_4: DR ← M[AR]

   D_1T_5: AC ← AC+DR, E ← C_{out} , SC ← 0

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

   D_2T_4: DR ← M [AR]

   D_2T_5: AC ← DR, SC ← 0

4. **STA**: This instruction stores the content of AC into the memory word specified by the effective address. Since the output of AC is applied to the bus and the data input of memory is connected to the bus, we can execute this instruction with one micro operation:

   D_3T_4: M[AR] ← AC, SC ← 0

5. **BUN:** This instruction causes the computer to branch unconditionally to the specified address and then continue the program execution from that location. It takes only one clock cycle to execute this instruction. Executing this instruction copies the content of the address register to program counter and then resets sequence counter.

   D6T5: PC ← AR, SC ←0

Table: Micro-operation for Execution of Memory Reference Instruction

MRI	Control Signal	RTL	Description
AND	D_0T_4	DR ← M[AR]	AND the memory word with AC and store the result in AC
	D_0T_5	AC ← AC^DR,SC ← 0	
ADD	D_1T_4	DR ← M[AR]	Add the memory word with AC and store the result in AC
	D_1T_5	AC ← AC+DR, E ← C_{out} , SC ← 0	
LDA	D_2T_4	DR ← M[AR]	Load the AC with memory location pointed by AR
	D_2T_5	AC ← DR, SC ← 0	
STA	$D3T_4$	M[AR] ← AC, SC ← 0	Store AC to given memory location
BUN	$D4T_4$	PC← AR, SC ← 0	Branch Unconditionally

Register Reference Instruction (RRI): r= D7I' T_3

RRI are recognized by control when D7 is high and I is 0. This instruction used bit 0 to 7 for specifying the instruction. These bits are available in IR(0-7). The logic for RRI is denoted as $D_7I'T_3 = r$ as the RRI gets executed in T3 cycle.

1. **CLA:** This instruction clears the content of the accumulator. Clear signal of the AC is activated to clear its content. RTL for the instruction is:

   rB_7: AC ← 0

2. **CLE:** This instruction clears the content of the carry bit. RTL for the instruction is:

   rB_6: E ← 0

3. **CMA:** This instruction complements the content of Accumulator. The content of the AC is passed though the ALU where the complement operation is selected, and finally the complemented result is stored back in the ALU.

   rB5:AC← $\overline{AC}$

4. **CME:** This instruction complements the content of Carry bit. The content of the E is passed though the ALU where the complement operation is selected, and finally the complemented result is stored back in the ALU.

   rB4:E←$\overline{E}$

5. **CIR:** This operation shifts each bits AC to right by one position. The value at Least Significant Bit is set to E where as the Most Significant Bit(MSB) i.e. AC(11) is set with the value in E. Shifting the bits to right is equivalent to dividing the number by 2.

   rB3: AC←cir AC

6. **CIL:** This operation shifts each bits AC to left by one position. The Least Significant Bit is set with the value in E where as the Most Significant Bit AC(11) is set to E. Shifting the bits to left is equivalent to multiplying the number by 2.

   rB2: AC←cil AC

7. **HLT:** Stop the execution of the program by resetting the start-stop flip-flop.

   rB_0: S ← 0

Table: Table: Micro-operation for Execution of Register Reference Instruction

RRI	Control Signal	RTL	Description
-	**r**	**SC←0**	Clears Sequence Counter
CLA	**rB_7**	**AC←0**	Clears the accumulator
CLE	**rB_6**	**E←0**	Clears the carry bit
CMA	**rB_5**	**AC←$\overline{AC}$**	Complement the accumulator
CME	**rB_4**	**E←$\overline{E}$**	Complements the carry bit
CIR	**rB_3**	**AC←cirAC**	Circular Right
CIL	**rB_2**	**AC←cilAC**	Circular Left
INC	**rB_1**	**AC←AC+1**	Increment the Accumulator
HLT	**rB_0**	**S←0**	Halt the Computer

Input/output Instruction (I/O Operation): p= D7IT 3

Input/output Instruction are recognized by control when D7 is high and I is 1. This instruction used bit 2 last significant bits for specifying the instruction. These bits are available in IR(0-7). The logic for RRI is denoted as D_7IT_3 = r as the RRI gets executed in T3 cycle.

1. **INP:** This is an input instruction. It causes the computer to accept a 8-bit data from the input register and clears the input flag. Clearing the input flag FGI indicates that the data has been accepted so that inputting device can provide with next 8-bits of data.

   pB7: AC(0-7) ← INPR, FGI ← 0

2. **OUT:** This is an output instruction. It causes the computer to send 8-bit of data stored in the AC to be transferred to the output register and clear the output flag. Clearing the output flag indicated the external reading device that the content is ready in the output register to be copied.

   pB6: OUTR ← AC(0-7), FGO ← 0

Table: Table: Micro-operation for Execution of Input/output instruction

I/O	Control Signal	RTL	Description
-	**p**	**SC←0**	Clears Sequence Counter
INP	**pB_7**	**AC(0-4)←INPR, FGI←0**	Input data to AC
OUT	**pB_6**	**OUTR←AC(0-7), FGO←0**	Output a data from AC

Control for Register and Memory:

The registers and memory is connected to same common bus system so the control logic for load and read data from register is required for required flow of data. Following are logics for control of register and memory.

Table: Control logic of the registers and Memory.

S.N	Registers	Load logic	Read Logic	Increment Logic	Clear Logic
1	**Address**	**T0+T1+D'IT3**	**D4T4**	**-**	**-**
2	**Program Counter**	**D4T4**	**T0**	**T1**	**-**
3	**Data**	**D0T4+D1T4+D2T4**	**D2T5**	**-**	**-**
4	**Accumulator**	**D0T5+D1T5+D2T5+rB5+ rb3+rB2+pB7**	**D3T4+pB6**	**rB1**	**rB7**
5	**Instruction**	**T1**	**T2**	**-**	**-**
6	**Output**	**pB6**	**-**	**-**	**-**
7	**Memory**	**D3T4**	**T1+r+D0T4+D1T4+D2T4**		

Some other control Logics:

Some other logic required for the processing of instruction for the computers are

Reset Sequence Counter: D0T5+D1T5+D2T5+D3T4+D4T4+r+p

Stop counter: rB0

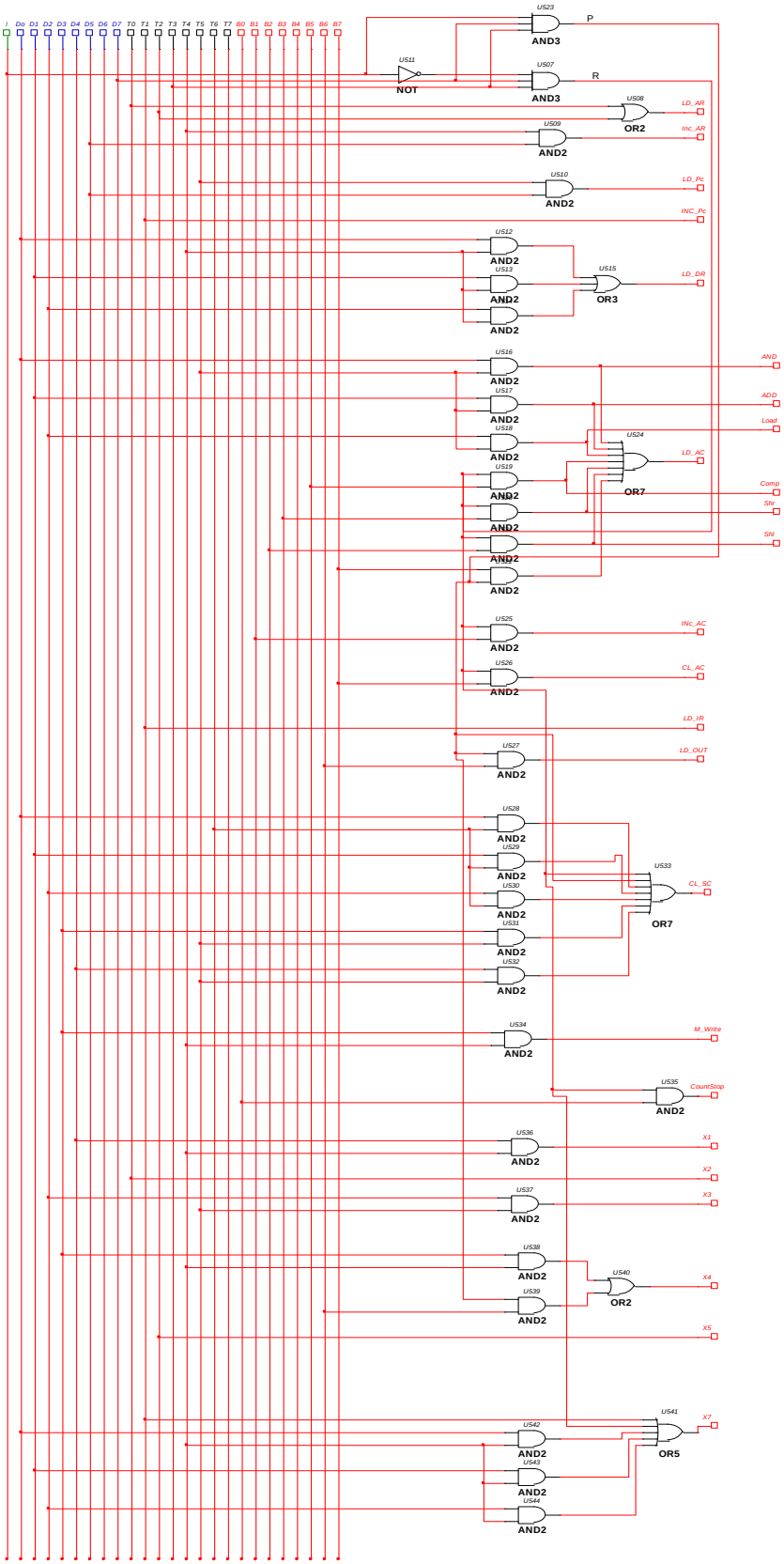
Control for Common Bus System:

The common bus system is controlled by 3bit of selection line in a multiplexer of 8*1 configurations. First of all the control signal for different register is given by control units and by encoding the control logic from control units it is passed into MUX to select the desired register's data to transmit in common bus system. The logics from control units are denoted by x1-x7 for different registers. Control logics are:

Table: Control for Common Bus System

S.N	Register	x	Selection line			Control Logic
			S2	S1	S0	
1	Address	X1	0	0	1	D4T4
2	Program Counter	X2	0	1	0	T0
3	Data	X3	0	1	1	D2T5
4	Accumulator	X4	1	0	0	D3T4+pB6
5	Instruction	X5	1	0	1	T2
6	Memory	X6	1	1	1	T1+r+D0T4+D1T4+D2T4

Diagram of Control Unit:



CONCLUSION

Designing a 12 bit CPU was very interesting. It helped to interconnect ideas learned during computer architecture course. Although it was somehow complex, the end result was worthfull. In overall, we got an idea how CPU works and how machine language is constructed. We also learned to use electrical simulation software during this design process. In this way the mini project was completed successfully.

BLIOGRAPHY

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

REFERENCE

1. http://digital.ni.com/public.nsf/allkb/25B2B005B379C3888625754C003DE11E [jan/30/2017]
2. http://www.ni.com/tutorial/11219/en/ [jan/20/2017]