Design of a 16-bit Computer

Project Report

Submitted to:

Pankaj R. Dawadi

Department of Computer Science and Engineering (DoCSE)

Dhulikhel, Kavre

Submitted by:
Anurag Adhikari(02)
Suyog Adhikari (03)
Sara Bade (06)
Bodhita Tuladhar(40)
Elina Baral(47)

Date: 2017/02/14

Contents

1. INTRODUCTION

A 2K X 16 computer has a very simple architecture compared to today's complex computers. But understanding the way it functions helps understand many complex concepts which are the pillars to the development of the modern computers. This designed computer has a very small instruction set. It supports basic arithmetic and logic operations such as addition, subtraction, and, or etc. Accumulator is used to store the output of any ALU operation. Data register is used to store data from memory before performing any operations on them. Address register and stack pointer are both used to specify memory address with the help of common address bus. Data is transferred between registers using common address bus. Program counter stores the next instruction to be executed and the computer interfaces with the user using input and output registers. All the instructions are executed by using the signals generated by the control unit consisting of a sequence counter and a large collection of logic gates. Our computer has a 4 bit opcode, 11 bit address length and 1 bit Memory Reference type Flag (I).

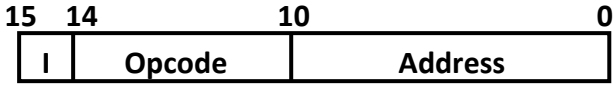
Following section describes the design process and the internal architecture of our computer as well as instruction set supported by this computer. We have also provided an explanation on how each instruction is executed.

Our computer consists of following hardware components:

1. A memory of 2K*16.
2. Eight registers: AR, PC, DR, AC, IR, OUTR, INPR and SC.
3. Five flip-flops: I, S, E, IEN, FGI and FGO.
4. Two decoders: a 4*16 operation decoder and a 3*8 timing decoder.
5. One encoder 8*3
6. A 16-bit common bus.
7. Control logic gates.
8. Adder, Subtractor and logic circuit connected to the input of AC.

2. Instruction Set

It is a group of bits that tells the computer to perform a specific operation.
As we are using 2K x 16 bit computer so the instruction set is divided into Address, Opcode and Addressing Mode.
$2.2^{10} = 2^{11}$ So, in 16 bits we need 11 bits for address, 4 bits for opcode and 1 bit for addressing



a. Memory Reference Instructions

Memory reference instructions load values into and store values from the general registers.
In our design, we have the following memory reference instructions:

Symbol	Opcode		Description
	I = 0 Direct	I = 1 Indirect	
LDA	00000X	10000X	Load accumulator with memory word
AND	00001X	10001X	AND the AC with memory word
ADD	00010X	10010X	ADD the memory word with AC and store in AC
BUN	00011X	10011X	Branch unconditionally
STA	00100X	10100X	Store value of AC to memory
XOR	00101X	10101X	XOR the AC with memory word
SUB	00110X	10110X	Subtract memory word with AC & store in AC

Table 1: Memory Reference Instructions
Here, 'X' is a 11 bit memory address.

b. Register Reference Instructions

Register reference instruction are the instruction which refers to registers instead of memory address or memory location for data.
In the Opcode part of the instruction they have the value 1111 and have value 0 in addressing mode.
In our design, we have the following register reference instructions:

Symbol	Instruction	Description
CLA	0 1111 10000000000	Clear AC
CLE	0 1111 01000000000	Clear E
CIR	0 1111 00100000000	Circulate Right AC and E
CIL	0 1111 00010000000	Circulate Left AC and E
INC	0 1111 00001000000	Increment AC
HLT	0 1111 00000100000	Halt computer

Table 2: Register Reference Instructions

c. Input Output Instructions

Input Output Instructions in a computer program causes transfer of data between peripheral devices and main memory, and enables the central processing unit to control the peripheral devices connected to it.
In the Opcode part of the instruction they have value 1111 and also have value 1 in the addressing mode.

Symbol	Instruction	Description
INP	1 1111 10000000000	Input character to AC
OUT	1 1111 01000000000	Output character from AC
SKI	1 1111 00100000000	Skip on input flag
SKO	1 1111 00010000000	Skip on output flag
ION	1 1111 00001000000	Interrupt on
IOF	1 1111 00000100000	Interrupt off

Table 3: Input Output Instructions

3. Common Bus System

The computer here designed has 8 registers, memory unit and control unit. Path must be provided for the communication between two registers in the computer. Use of wire from one register to other register may be excessive. So, the more efficient system is used for transferring the date between many registers called common bus system. Common bus system is the efficient scheme for transferring the information in system which has a many registers. If common bus is not use, then every register has to be connected to individual bus which may create a complex and impossible situation. So, it must require a scheme in which bus is connected to the memory as well as the registers.

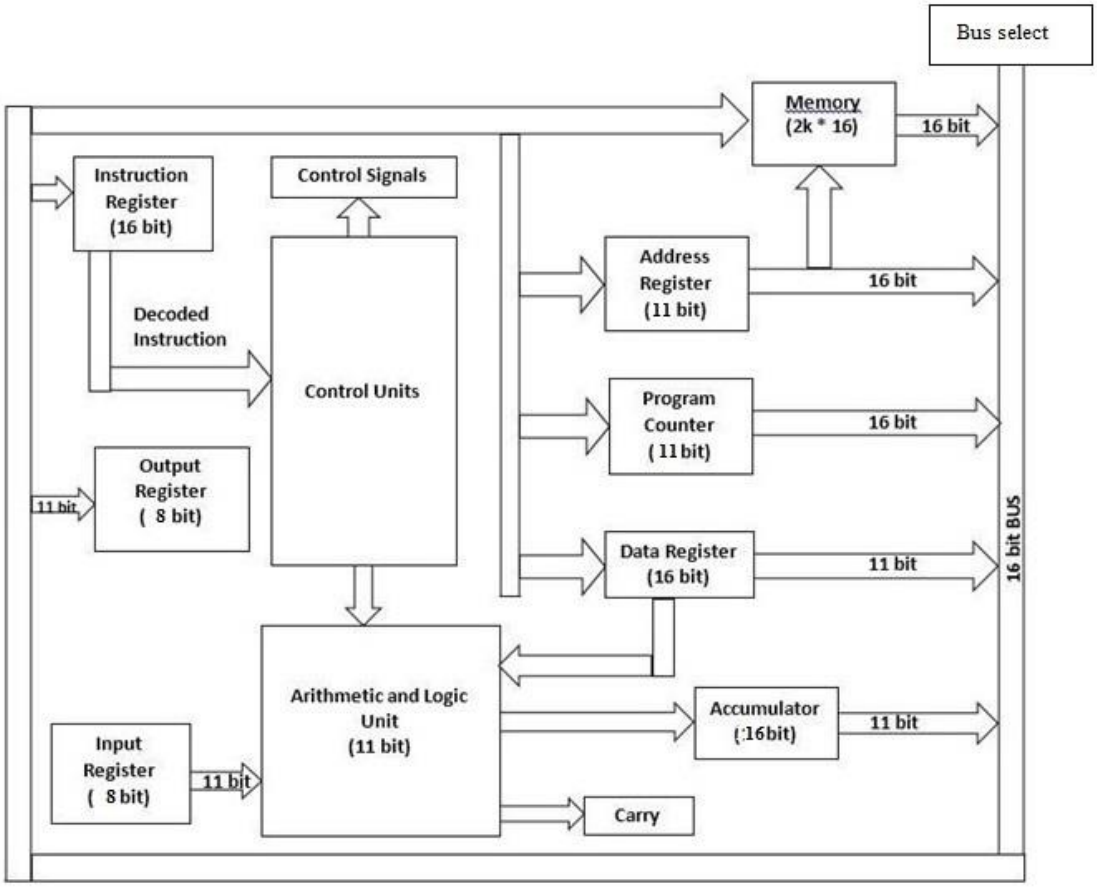


Figure 1 : Block Diagram

4. Components

The various components given below comes together to form a simple working computer system. The various components are selected by the Bus as per the output from the encoder which can be seen from the table below :

X1	X2	X3	X4	X5	X6	X7	S2	S1	S0	Selected Registers
0	0	0	0	0	0	0	0	0	0	None
1	0	0	0	0	0	0	0	0	1	Address Register
0	1	0	0	0	0	0	0	1	0	Program Counter
0	0	1	0	0	0	0	0	1	1	Data Register
0	0	0	1	0	0	0	1	0	0	Accumulator
0	0	0	0	1	0	0	1	0	1	Instruction Register
0	0	0	0	0	1	0	1	1	0	Temporary Register
0	0	0	0	0	0	1	1	1	1	Memory

$X1 = AR = D_3.T_4$
$X2 = PC = R'.T_0 + R.T_0$
$X3 = DR = D_0.T_5$
$X4 = AC = D_4.T_4 + p.B_{10}$
$X5 = IR = R'.T_2$
$X6 = TR = R.T_1$
$X7 = Memory = D_0.T_4 + D_1.T_4 + D_2.T_4 + D_5.T_4 + D_6.T_4 + R'.T_1$

a. Instruction Register

Size: 16 bits

Instruction Register stores the 16-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 (CBS). Once the instruction is loaded in the IR it is then decoded to obtain the decoded lines (D0-D15) and B0-B10 which along with the timing signals is used to provide the necessary control signals for the computer to execute the stated operation. The composition of the component of the Instruction Register is given as :
$Load = R.T_0$

b. Program Counter

Size: 11 bits

Program counter 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.
In case of subroutines, the content of program counter (PC) is saved in the memory and is loaded with the location of memory containing the subroutine procedure. Finally at

the end of subroutine call PC is loaded with the previously saved address and the execution cycle continues.
The composition of the components of the program counter is given as :
Load = $D_3.T_4$
Increment = $R'.T_1 + R.T_2$
Clear = $R.T_1$

c. Data Register

Size: 16 bits

Data register is used to store the operand read from the memory that is to be processed. Output of data register is an input to the ALU.
The composition of the components of the Data Register is given as :
Load = $D_0.T_4 + D_1.T_4 + D_2.T_4 + D_6.T_4$

d. Temporary Register

Temporary Registers are used to store the temporary value during the execution. Its composition is given by :
Load = $R'.T_1$

e. Accumulator

Size: 16 bits

Accumulator is the main register for most of the operations. It stores the result of any operation after its completion. Input of the accumulator comes from the output of the ALU. The input of accumulator is not directly interfaced with the CBS. Output of the AC goes to the common bus system as well as to ALU as the first operand, second being the Data register.
The composition of the components of the program counter is given as :
Load = $D_0.T_5 + D_1.T_5 + D_2.T_5 + D_5.T_5 + D_6.T_5 + p.B_{11} + r.B_3 + r.B_2$
Increment = $r.B_5$
Clear = $r.B_1$

f. Input Output Register

Size: 8 bits

There are two register each of 8 bit for both input and output. Input register (INPR) is connected to the ALU and the content of the input register is transferred to the ALU on selection of the transfer operation.
Output register (OUTR) is used by the computer to provide output to the external world. It receives its output from the common bus system which usually comes from the accumulator. This output may be then connected to an external display.

g. Control Unit

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.

h. Memory Unit

*Size: 2K*16*

Memory stores both the instruction and the data on which processing is to be performed.
We have selected a memory of 2048 words with a word size of 16-bits for our computer. This means that we require 11 bit address line to address all the word of 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.
The composition for the components of the memory unit is :
Read = $D_0.T_4 + D_1.T_4 + D_2.T_4 + D_5.T_4 + D_5.T_5 + D_6.T_4 + D_6.T_5 + R'.T_1 + R'.T_2$
Write = $D_4.T_4 + R.T_1$

i. Address Register

Memory Address Register (MAR) is the CPU register that either stores the memory address from which data will be fetched from the CPU or the address to which data will be sent and stored. Its component composition is given as :
Load = $R'.T_0 + R'.T_2 + D_7'.I.T_3$
Clear = $R.T_0$

j. Sequence Counter

It is the part of the computer that determines whether the given instruction has completed its execution or not. The completion of the instruction is denoted by SC ←0. Its component composition is given as :
Clear = $p + r + R.T_2 + D_0.T_5 + D_1.T_5 + D_2.T_5 + D_3.T_4 + D_4.T_4 + D_5.T_5 + D_6.T_5$

k. Arithmetic and Logic Unit (ALU)

An arithmetic logic unit (ALU) is a digital circuit used to perform arithmetic and logic operations. It represents the fundamental building block of the central processing unit (CPU) of a computer. The ALU uses operands and code that tells it which operations to perform for input data. After the information has been processed by the ALU, it is sent to the computer's memory. Arithmetic and logic unit in our computer performs 9 main operations:

1) AND with DR
2) ADD with DR
3) SUB with DR
4) XOR with DR
5) LOAD FROM INPUT
6) LOAD FROM DR
7) COMPLEMENT
8) LOGICAL SHIFT RIGHT
9) LOGICAL SHIFT LEFT

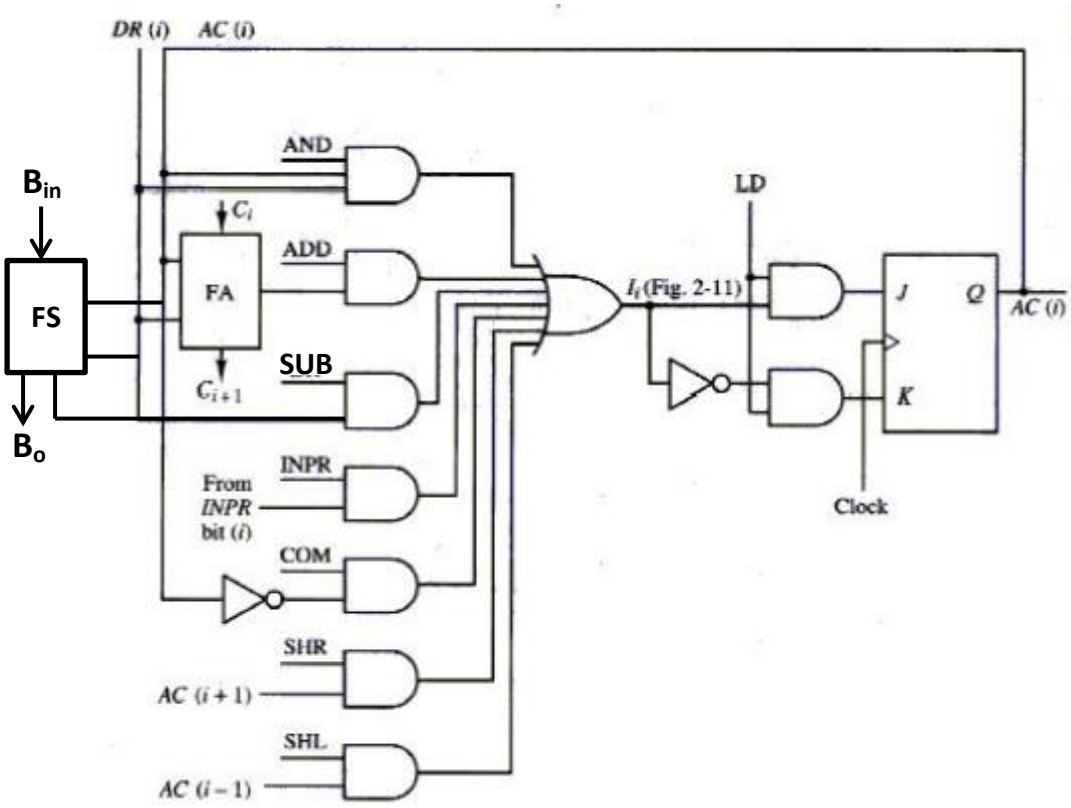


Figure 2 : ALU Block Diagram

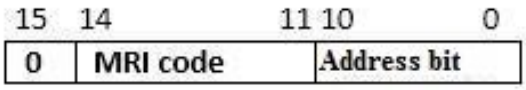
5. Opcodes

An opcode is the first byte of an instruction in machine language which tells the hardware what operation needs to be performed with this instruction. Every processor/controller has its own set of opcodes defined in its architecture. An opcode is followed by data like address, values etc. Opcodes for different instruction are designed based on the upper 4 bit stored in the instruction register. Following are the types of opcodes used:

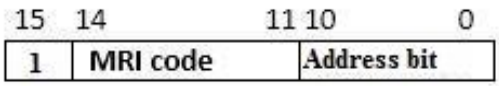
a. Memory Reference Instruction (MRI)

In memory reference instruction 5-bits read from the memory during the fetch cycle is used to specify the memory operation, and next consecutive 11 bits read from the memory contains the memory address of the operand.

Instruction code format for the MRI is:



Direct addressing

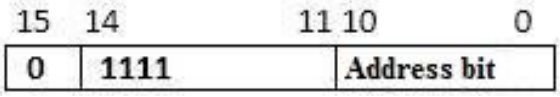


Indirect Addressing

b. Register Reference Instruction (RRI)

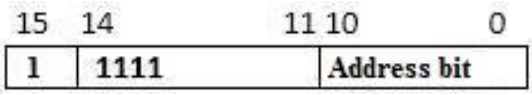
In RRI, the higher 5 bits of the instruction code is always 01111.
The lower 11 bit of the instruction code is then utilized to specify the specific register reference instruction. These lower eleven bits are then decoded to generate necessary control signals for executing the given instruction. Instruction format for RRI is given below:

,



c. Input Output Instruction (IOI)

In IOI, the higher 5 bits of the instruction code is always 11111.
The lower 11 bit of the instruction code is then utilized to specify the specific IOI. These lower eleven bits are then decoded to generate necessary control signals for executing the given instruction. Instruction format for IOI is given below:



6. Control Signals

Control signals for all the instructions are specified below:

a. Fetch and Decode Cycle

Cycle	Control Signal	RTL	Description
Fetch	T0	AR ←PC	Loads AR with the address of new instruction to be read from the memory
	T1	IR←M [AR] PC ←PC+1	Copy the instruction read from the memory to the program counter.
Decode	T2	D0…D15←Decode [IR (11-14)], I ←IR (15) AR ←IR(0-10)	Decode the instruction

Table 4: Fetch and Decode Cycle

b. Execution of Register-Reference instruction

D15I'T3 = r (Common to all register-reference instruction) IR(i) = Bi (i=0,1,2...,10) Specifies the particular RRI instruction

RRI	**Control Signal**	**RTL**	**Description**
	R	SC←0	Clear SC
CLA	rB_5	AC←0	Clear AC
CLE	rB_4	E←0	Clear E
CIR	rB_3	AC←shr AC, AC(15)←E, E←AC(0)	Circulate right
CIL	rB_2	AC←shl AC, AC(0)←E, E←AC(15)	Circulate left
INC	rB_1	AC←AC+1	Increment AC
HLT	rB_0	S ←0(S is a start-stop flip-flop)	Halt computer

Table 5: RRI Signals

c. Memory Reference Instruction (MRI)

MRI	Control Signal	RTL	Description
LDA	D_0T_4	DR ← M [AR]	Load the AC with memory location pointed by AR
	D_0T_5	AC ←DR, SC ←0	
AND	D_1T_4	DR ← M [AR]	AND the AC with DR
	D_1T_5	AC←AC ^ DR, SC ←0	
ADD	D_2T_4	DR ← M [AR]	Add the memory word with AC and store the result in AC
	D_2T_5	AC←AC+DR, E ←Cout , SC ←0	
BUN	D_3T_4	PC ← AR, SC ←0	Branch unconditionally to AR
STA	D_4T_4	M[AR] ← AC, SC ← 0	Store contents of AC in memory.
XOR	D_5T_4	DR←M[AR]	XOR the AC with DR
	D_5T_5	AC←AC $\oplus$DR, SC ←0	
SUB	D_6T_4	DR←M[AR]	Subtract the memory word with AC and store the result in AC
	D_6T_5	AC←AC+DR, SC ←0	

Table 6: MRI Signals

d. Input Output operations:

D15IT3 = p (Common to all IO instruction)

IR(i) = Bi (i=7,8,9,10) Specifies the particular RRI instruction

RRI	Control Signal	RTL	Description
	P	SC←0	Clear SC.
INP	pB11	AC ←INPR,FGI←0	Input a character from AC
OUT	pB10	OUTR←AC,FGI←0	Output a character from AC
SKI	pB9	If(FGI==1) then PC←PC+1	Skip on input flag
SKO	pB8	If(FGO==1) then PC←PC+1	Skip on output flag
ION	pB7	IEN ← 1	Sets interrupt flag
IOF	pB6	IEN ← 0	Resets interrupt flag

Table 7: RRI Signals

7. FlipFlops

FlipFlop	Description	Instruction
IEN	Start Stop Flip Flop	$pB_7 + pB_6 + RT_2$
I	Interrupt	$R'T_2$
R	Memory Reference type Flag	$T_0'.T_1'.D_2'.IEN.(FGI + FGO)$
E	Carry Flag	$r.B_3 + r.B_4 + r.B_2 + D_2.T_5$
FGI	Input Flag	$p.B_{11}$
FGO	Output Flag	$p.B_{10}$

Table 8: FlipFlops

8. Number of Components Used

S.N.	Component	Number
1	Memory(2K x 16)	1
2	Registers	8
3	FlipFlop	6
4	Full Adders	16
5	Full Subtractor	16
6	3 bit binary counter	1
7	Decoders(4 x 16) and (3 x 8)	2
8	Encoder (3*8)	1

9. Explanation of instruction

a. Fetch

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 T0 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.

When R →0, we use R' in the instruction
$R'T_0$: AR ← PC
$R'T_1$: IR ← M [AR], PC ← PC + 1

When Interrupt occurs R →1, so we use R in the instruction

RT_0 : AR ← 0, TR ← PC
RT_1 : M [AR] ← TR, PC ← 0

b. Decode

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. Following is the RTL for the decoding cycle:

When R →0
$R'.T_2$: $D0_2, \ldots, D7_7$ ← Decode IR(12-1_4), AR ← IR(0-11), I ← IR(15)

When R →1

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

c. Execution

I. Memory reference instructions

1. 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_0T_4: DR ←M [AR]
   D_0T_5: AC ←DR, SC ←0

2. 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_1T_4: DR←M [AR]
D_1T_5: AC←AC ^ DR, SC ←0

3. ADD

This instruction is used to add the data of the given memory address with the data loaded in the accumulator and the result thus obtained is stored in the accumulator. This instruction is executed in the following cycles

D_2T_4 : DR ← M [AR]
D_2T_5 : AC ← AC + DR, E ← Cout, SC ← 0

4. BUN

This instruction causes the computer to branch unconditionally to the given address and continue the execution of instruction for the location.
It takes only one step to execute this instruction. To execute this instruction content of the address is simply transferred to the program counter and the sequence counter is reset.

D_3T_4: PC ←AR, SC ←0

5. STA

This instruction stores the content of AC into the memory word specified by the effective address. The RTL for this instruction is:

D_4T_4: M[AR] ←AC, SC ←0

6. XOR

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 XORed with content of the accumulator and the result is stored in the accumulator. The XOR operation is performed inside the ALU. RTL for this instruction is:

D_5T_4: DR←M[AR]
D_5T_5: AC←ACDR, SC ←0

7. SUB

This instruction is used to subtract the data from the memory with the data stored in the accumulator and store the resulting data in the accumulator. Its instruction is :

D_6T_4: DR←M [AR]
D_6T_5: AC - DR, E ← Cout, SC ← 0

ii. Register reference instructions

$D_7I'T3 = r$ (Common to all register-reference instruction)

1. CLA

   This instruction clears the content of the accumulator.
   Clear signal of the AC is activated to clear its content.

   rB_5: AC←0

2. CLE

   This instruction clears the carry flag E.

   rB_4 : E←0

3. CIR

   This is the circular right shift instruction and causes the content of the accumulator to be circularly shifted right without loss of any bit. This operation is carried out in ALU.

   rB_3: AC←shr AC, AC(15)←E, E←AC(0)

4. CIL

   This is the circular left shift instruction and causes the content of the accumulator to be circularly shifted left without loss of any bit. This operation is carried out in ALU.

   rB_2: AC←shl AC, AC(0)←E, E←$AC(1_5)$

5. INC

   It increments the content of AC by one. This is carried out by applying the INR signal to the AC.

   rB_1: AC←AC+1

6. HLT

   Stop the execution of the program by resetting the start-stop flipflop.

   rB_0: S ←0

iii. Input/output operations

$D_7IT3 = p$

1. INP

   This is an input instruction. It causes the computer to accept a $_4$-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 4-bits of data.

   pB_{11}: AC ←INPR, FGI←0

2. OUT

   This is an output instruction. It causes the computer to send $_4$-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. Once the date is copied by the external device from the output register, the output flag is set allowing the computer to provided next $_4$-bits of data.

   pB_{10}: OUTR←AC,FGI←0

3. SKI

   Skip on input flag causes the computer to skip the next instruction if the input flag is set.

   pB_9: If(FGI==1) then PC←PC+1

4. SKO

   Skip on output flag causes the computer to skip the next instruction if the output flag is set.

   pB_8: If(FGO==1) then PC←PC+1

5. ION

   Sets the interrupt flip flop

   pB_7: IEN ← 1

6. IOF

   Resets the interrupt flip flop

   pB_6: IEN ← 0

10. CONCLUSION

The objective of designing this computer was not to build a computer that can match the modern computer in any aspect of computing. But the goal was to design a simple model to describe how computers function. The 2K X 16 computer is able to explain how microprocessors interact with input and output as well as the memory. It explains how an ALU works, what CU is and how registers work within the CPU. Hence, it fulfills it's purpose.

11. BIBLIOGRAPHY

1. Mano, M. Morris. Computer system architecture. 3rd ed. Englewood Cliffs, N.J.: Prentice Hall, 1993. Print.
2. Hennessy, John L., and David A. Patterson. Computer architecture: a quantitative approach. 5th ed. Waltham, MA: Morgan Kaufmann, 2012. Print.