

Gajendra Processor

Lab Report

submitted by

Aasritha Yadav C (CS22B064), K. A. Vignesh (CS22B076)

Under the supervision

of

Dr.Ayon Chakraborty



Computer Science and Engineering

Indian Institute of Technology Madras

Jul - Nov 2023

Contents

1	General CPU Register	5
1.1	Diagram	5
1.2	I/O Control	5
1.3	Working	5
2	Program Counter	6
2.1	Diagram	6
2.2	Working	6
2.3	PC_INC	6
2.4	PC_OUT	6
2.5	PC_LOAD	6
3	MEMORY	7
3.1	Diagram	7
3.2	MEM_OUT	7
4	MEMORY ADDRESS REGISTER	8
4.1	Diagram	8
4.2	MAR_IN	8
5	INSTRUCTION REGISTER	9
5.1	Diagram	9
5.2	INS_REG_IN	9
5.3	INS_REG_OUT	9
5.4	CTRL	9
6	INSTRUCTION DECODER	10
6.1	Diagram	10
6.2	INS_DEC_OUT	10
6.3	ALU_OUT	10
7	Arithmetic and Logic Unit(ALU)	11
7.1	ADD	11
7.2	SUB	11
8	Status Register	12
8.1	Diagram	12
8.2	I/O	12
8.3	Uses	12
9	Control Processor	13
9.1	Diagram	13
9.2	I/O	13
9.3	Working	13
9.4	Ring Counter	13
9.5	EEPROM Programming	13

10	Micro instructions and Controller Logic Design	14
10.1	NOP :	14
10.2	LDA :	14
10.3	ADD :	14
10.4	SUB :	15
10.5	LDI :	15
10.6	JMP:	15
10.7	JNZ :	15
11	Instruction Set	16
11.1	NOP :	16
11.2	LDA :	17
11.3	ADD :	17
11.4	SUB :	18
11.5	LDI :	18
11.6	JMP:	19
12	Sample Tasks	19
12.1	For Addition	19
12.2	For Subtraction	19
12.3	For both addition and subtraction	20

List of Figures

1	8-bit Register	5
2	Program Counter	6
3	Memory	7
4	Memory Address Register	8
5	Instruction Register	9
6	Instruction Decoder	10
7	Gajendra ALU	11
8	Status Register	12
9	Control Processor	13
10	Micro Instruction for NOP	14
11	Micro instruction for LDA	14
12	Micro instruction for ADD	14
13	Micro instruction for SUB	15
14	Micro instruction for LDI	15
15	Micro instruction for JMP	15
16	Micro instruction for JNZ	16
17	Instruction Set for NOP	16
18	Instruction Set for LDA	17
19	Instruction Set for ADD	17
20	Instruction Set for SUB	18
21	Instruction Set for LDI	18
22	Instruction Set for JMP	19
23	Rom data for addition of numbers 3,4	19
24	Rom data for subtraction of 2 from 5	19
25	Rom data for 5+3-2	20

1 General CPU Register

1.1 Diagram

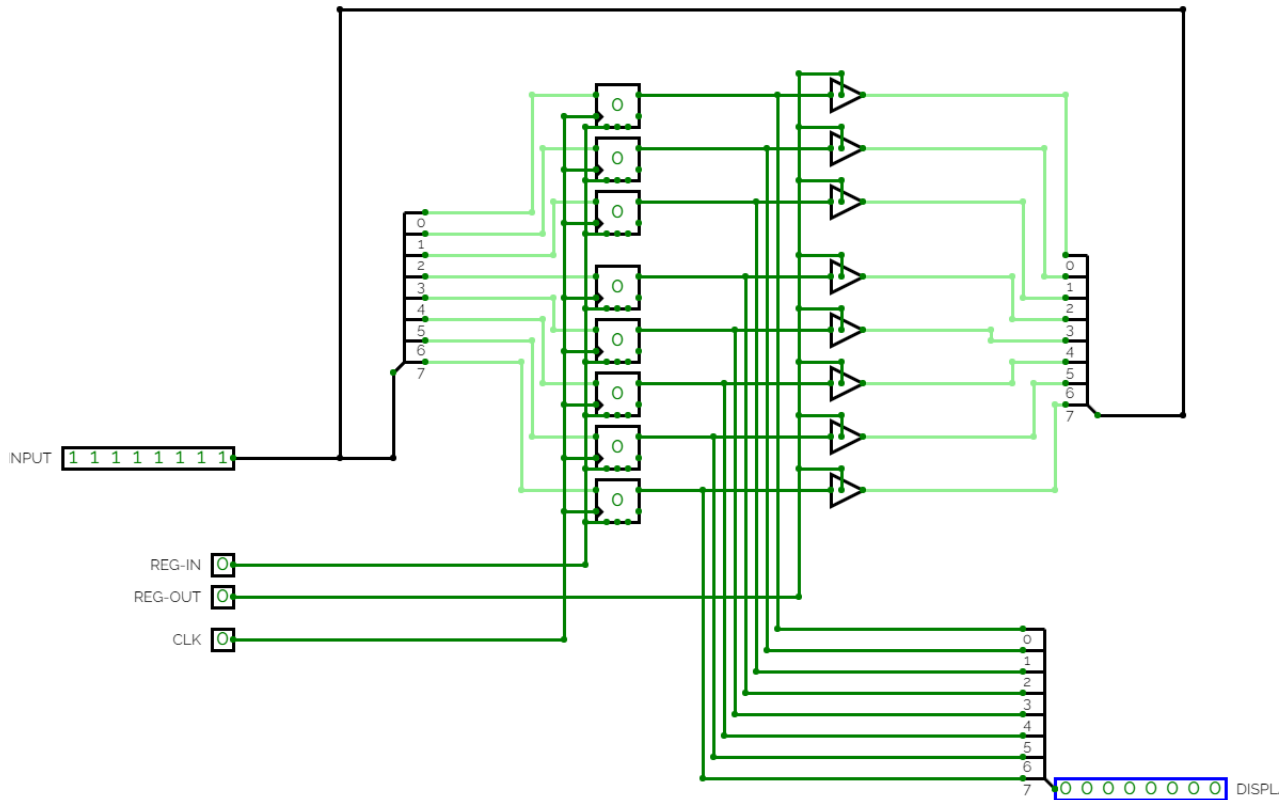


Figure 1: 8-bit Register

1.2 I/O Control

The register consists of a bidirectional common 'I/O' line which stores the 8 bit input from common bus. The 8 bit value can later be sent to output. The flow of data is controlled by the signals REG_in and REG_out.

1.3 Working

The register consists of D flip flops, responsible for the storage of memory. REG_act as the control signal for input, when it is 1 the input is read from the common bus while REG_out is the control signal for output in the same way.

3 MEMORY

3.1 Diagram

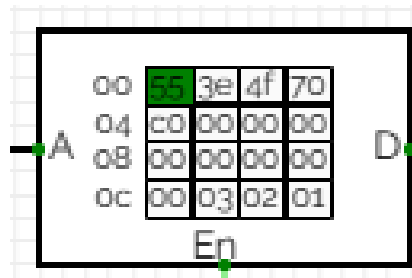


Figure 3: Memory

Memory is the ROM which contains both instruction and data required for a program. The following are the functionalities that are present in the Memory.

3.2 MEM_OUT

The ROM's output is controlled by a tri-state buffer. MEM_OUT acts as the enable line for this tri-state buffer. If MEM_OUT = 1 then the memory outputs a 8-bit output into the common bus.

4 MEMORY ADDRESS REGISTER

During a computer run , the 4-bit address in the Programming Counter is latched into the MAR .Then the MAR applies this 4-bit address to the Memory where a read operation is performed.

The following are the functionalities that are present in the MAR .

4.1 Diagram

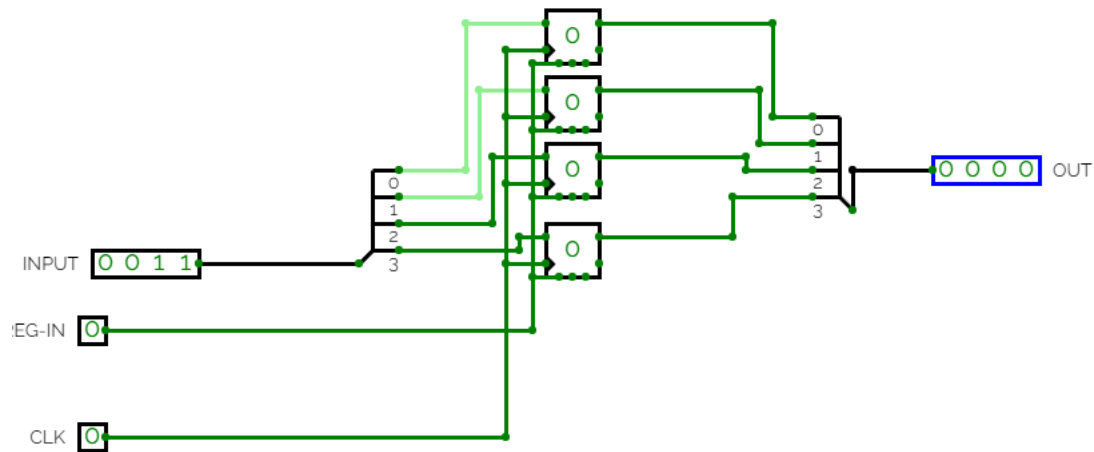


Figure 4: Memory Address Register

4.2 MAR_IN

If $MAR_IN = 1$, then the MAR takes the 4-bit address (Least Significant in common bus) from the Program Counter through common bus.

5 INSTRUCTION REGISTER

5.1 Diagram

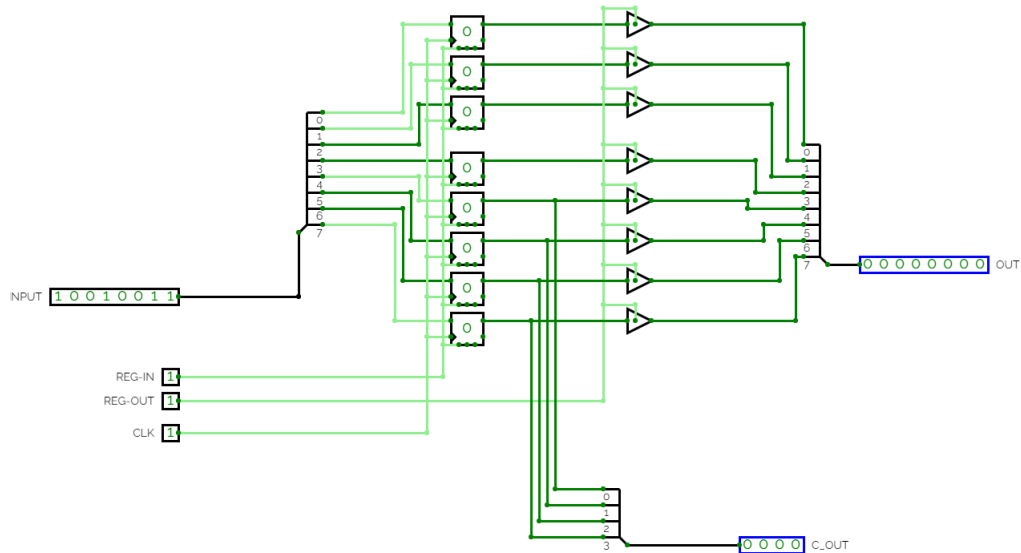


Figure 5: Instruction Register

To fetch an instruction from the memory the computer does a memory read operation . This places the contents of the addressed memory location on the common bus. The instruction register loads this content into it on the next positive clock edge .

The contents of the instruction register are split into two nibbles of 4-bits each (least 4 significant bits , most 4-significant bits) , The most 4-significant bits are sent into **Controller** and it will also send the 8-bit content into the **Instruction decoder** .

The following are the functionalities that are present in the Instruction register .

5.2 INS_REG_IN

If $INS_REG_IN = 1$, then the INS register takes the 8-bit content as input from the common bus .

5.3 INS_REG_OUT

If $INS_REG_OUT = 1$, then INS register loads its 8-bit content into Instruction decoder and thus into common bus.

5.4 CTRL

The Most Significant 4-bits(4-7) load in to the **Controller** and **INS Decoder**.

6 INSTRUCTION DECODER

6.1 Diagram

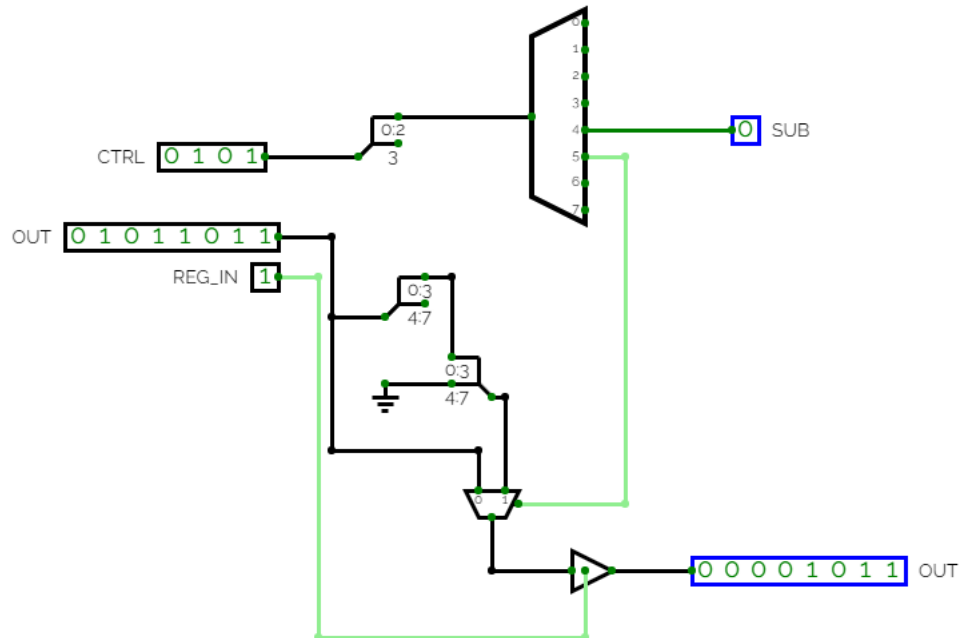


Figure 6: Instruction Decoder

It helps the ALU by giving the instruction whether to add or subtract , by using the most significant 4-bits (i.e CTRL) . It also helps us with the instruction **LDI** by making the 4 - most significant bits into 0000 , so that the remaining 4-bits gets stored in our **accumulator (i.e register A)**.

The following are the functionalities that are present in the Instruction Decoder .

6.2 INS _DEC _OUT

If $INS_DEC_OUT = 1$, then it takes the 8-bit content from OUT of INS register.

6.3 ALU_OUT

The INS_DEC generates a 1-bit input for the **ALU**.

It generates the following outputs for the corresponding instructions .

$$0 \implies \textit{Addition}$$

$$1 \implies \textit{Subtraction}$$

7 Arithmetic and Logic Unit(ALU)

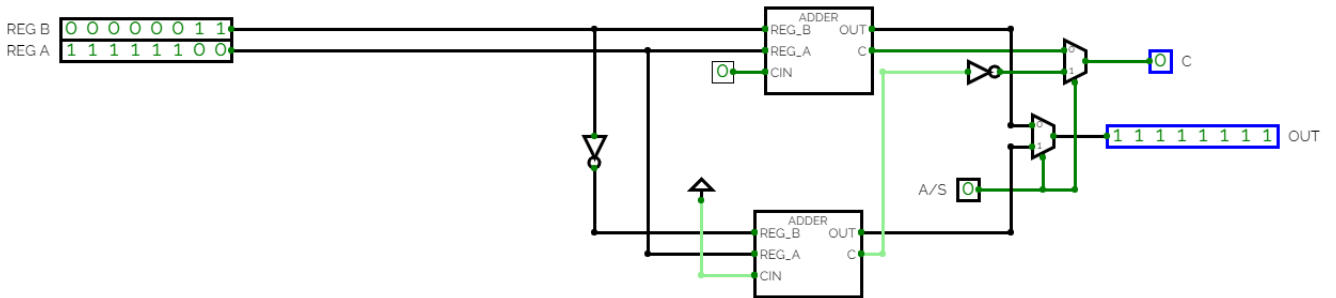


Figure 7: Gajendra ALU

The ALU performs the arithmetic operations, executed by the computer. Supported operations are Addition(ADD) and Subtraction(SUB). The final 8-bit data is then loaded into Accumulator or REG _A .

7.1 ADD

The ADD operation is used to add two 8-bit numbers using a 8-bit full adder which adds without carry .

7.2 SUB

The SUB operation is used to subtract two 8-bit numbers using a 8-bit full adder with initial carry in as 1. This one is supplied by INS_DEC.

8 Status Register

8.1 Diagram

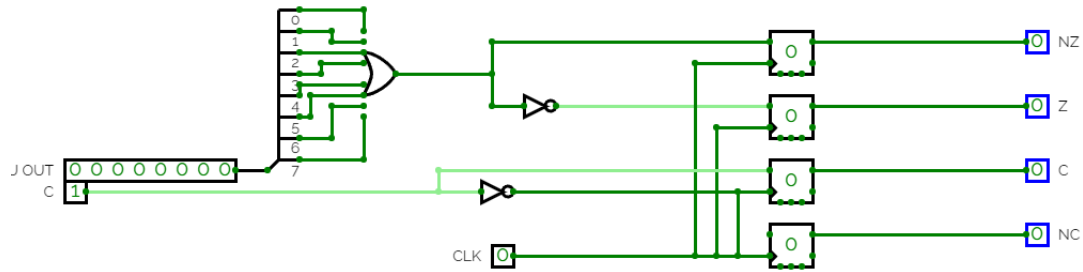


Figure 8: Status Register

8.2 I/O

It takes 2 inputs ALU _OUT and C (carry out from ALU) and gives four outputs namely: **C** (1 if carry = 1), **NC**(**negation of C**) , **Z**(1 if(**ALU _OUT** = 1) and **NZ** (**negation of Z**). Tise NZ line is used as input to the controller to aid in the **JNZ** function.

8.3 Uses

It is used to implement JUMP and JUMP if not zero conditions.

9 Control Processor

9.1 Diagram

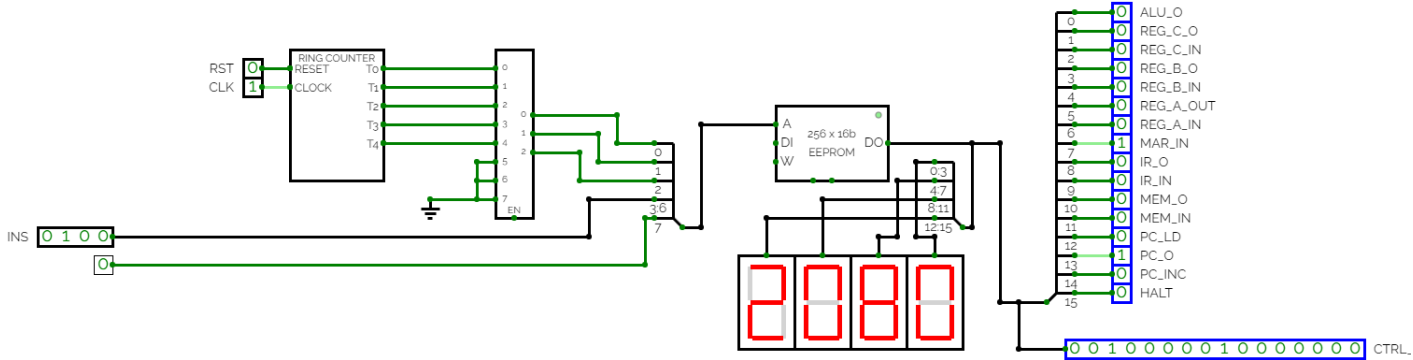


Figure 9: Control Processor

9.2 I/O

The control processor takes 4 bit instruction from INS_REG .We use a ring counter of length 5 indicating the 5 T-states. The EEPROM in the control processor takes 8-bit address is split into 3 parts. The first bit zero and next 4 bits are the instruction and the last 3 bits are the T-state that is to be implemented. The EEPROM gives 16-bit output which is the control word which is also the output of the Control Processor.

9.3 Working

The control word consists of all inputs of the program which are: REG_in, REG_out, PC_load, PC_out, PC_inc, flag, MEM_in, MEM_out, MAR_in, INS_out, INS_out and ALU_out. It controls all the other components of the program and is hence the most important component of the program .

9.4 Ring Counter

Every instruction has micro instructions called 'T states' that are needed to be implemented in a particular sequence one after another. This is done using a ring counter. We use a ring counter of length 5 bits in the program which governs the micro instructions.

9.5 EEPROM Programming

Each memory location in EEPROM contains 8 bits necessary to execute its micro instructions. The each memory location contains the control word of that particular micro instruction.

10 Micro instructions and Controller Logic Design

10.1 NOP :

NOP :	1<<PC_OUT 1<<MAR_IN	T0
	1<<PC_INC 1<<MEM_OUT 1<<IR_IN	T1
	0	T2
	0	T3
	0	T4

Figure 10: Micro Instruction for NOP

10.2 LDA :

LDA :	1<<PC_OUT 1<<MAR_IN	T0
	1<<PC_INC 1<<MEM_OUT 1<<IR_IN	T1
	1<<IR_OUT 1<<MAR_IN	T2
	1<<MEM_OUT 1<<REGA_IN	T3
	0	T4

Figure 11: Micro instruction for LDA

10.3 ADD :

STA :	1<<PC_OUT 1<<MAR_IN	T0
	1<<PC_INC 1<<MEM_OUT 1<<IR_IN	T1
	1<<MEM_IN 1<<REGA_OUT	T2
	0	T3
	0	T4

Figure 12: Micro instruction for ADD

10.4 SUB :

SUB	:1<<PC_OUT 1<<MAR_IN	T0
	1<<PC_INC 1<<MEM_OUT 1<<IR_IN	T1
	1<<IR_OUT 1<<MAR_IN	T2
	1<<MEM_IN 1<<REGA_OUT	T3
	0	T4

Figure 13: Micro instruction for SUB

10.5 LDI :

LDI	:1<<PC_OUT 1<<MAR_IN	T0
	1<<PC_INC 1<<MEM_OUT 1<<IR_IN	T1
	1<<IR_OUT 1<<REGA_IN	T2
	0	T3
	0	T4

Figure 14: Micro instruction for LDI

10.6 JMP:

JMP	:1<<PC_OUT 1<<MAR_IN	T0
	1<<PC_INC 1<<MEM_OUT 1<<IR_IN	T1
	1<<IR_OUT 1<<PC_LOAD	T2
	0	T3
	0	T4

Figure 15: Micro instruction for JMP

10.7 JNZ :

JNZ	:1<<PC_OUT 1<<MAR_IN	T0
	1<<PC_INC 1<<MEM_OUT 1<<IR_IN	T1
	1<<IR_OUT 1<<PC_LOAD	T2
	0	T3
	0	T4

Figure 16: Micro instruction for JNZ

11 Instruction Set

11.1 NOP :

Description

Loads the Instruction Register and increments the Program counter. No other operation is carried out.

Operation

(i) None

Syntax:

Operands:

Program Counter

(i) NOP X

None

$PC \leftarrow PC + 1$

8-bit op-code:

0000	XXXX
------	------

Figure 17: Instruction Set for NOP

11.2 LDA :

Description

Loads Register A with the value of the common bus (this value is obtained from the memory).

Operation

(i) $R_A \leftarrow (X)$

Syntax: Operands: Program Counter

(i) LDA X $0 \leq X \leq 15$ $PC \leftarrow PC + 1$

8-bit op-code:

0001	0xX(XXXX)
------	-----------

Figure 18: Instruction Set for LDA

11.3 ADD :

Description

Add the value at the given address to accumulator.

Operation

(i) $R_B \leftarrow (X)$

(ii) $R_A \leftarrow R_A + R_B$

Syntax: Operands: Program Counter

(i) ADD X $0 \leq X \leq 15$ $PC \leftarrow PC + 1$

8-bit op-code:

0011	0xX(XXXX)
------	-----------

Figure 19: Instruction Set for ADD

11.4 SUB :

Description

Subtract the value at the given address from the accumulator.

Operation

- (i) $R_B \leftarrow (X)$
- (ii) $R_A \leftarrow R_A - R_B$

Syntax: Operands: Program Counter

- (i) ADD X $0 \leq X \leq 15$ $PC \leftarrow PC + 1$

8-bit op-code:

0100	0xX(xxxx)
------	-----------

Figure 20: Instruction Set for SUB

11.5 LDI :

Description

Loads the value given along with the instruction to the Register A.

Operation

- (i) $R_A \leftarrow X$

Syntax: Operands: Program Counter

- (i) LDI X $0 \leq X \leq 15$ $PC \leftarrow PC + 1$

8-bit op-code:

0101	0xX(xxxx)
------	-----------

Figure 21: Instruction Set for LDI

11.6 JMP:

Description

Jumps to the instruction of the program which is present in the given address.

Operation

(i) $PC \leftarrow (X)$

Syntax:

Operands:

Program Counter

(i) $JMP X$

$0 \leq X \leq 15$

$PC \leftarrow X$

8-bit op-code:

0110	0xX(XXXX)
------	-----------

Figure 22: Instruction Set for JMP

12 Sample Tasks

12.1 For Addition

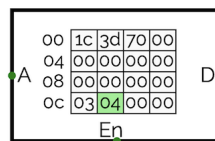


Figure 23: Rom data for addition of numbers 3,4

It first loads the data which is stored in the address 0x0c (i.e 3) and then as the program counter is increased it will take the instruction addition stored in the address 0x01 and load the number stored in the memory address 0x0d(i.e 4), then after it takes the instruction out and gives the output 7 at the display of the register C.

12.2 For Subtraction

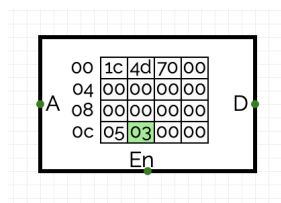


Figure 24: Rom data for subtraction of 2 from 5

It first loads the data which is stored in the address 0x0c (i.e 5) and then as the program counter is increased it will take the instruction subtraction stored in the address 0x01 and load

the number stored in the memory address 0x0d(i.e 3), then after it takes the instruction out and gives the output 2 at the display of the register C.

12.3 For both addition and subtraction

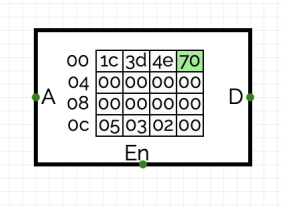


Figure 25: Rom data for 5+3-2