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

List of Figures

1 General CPU Register

1.1 Diagram

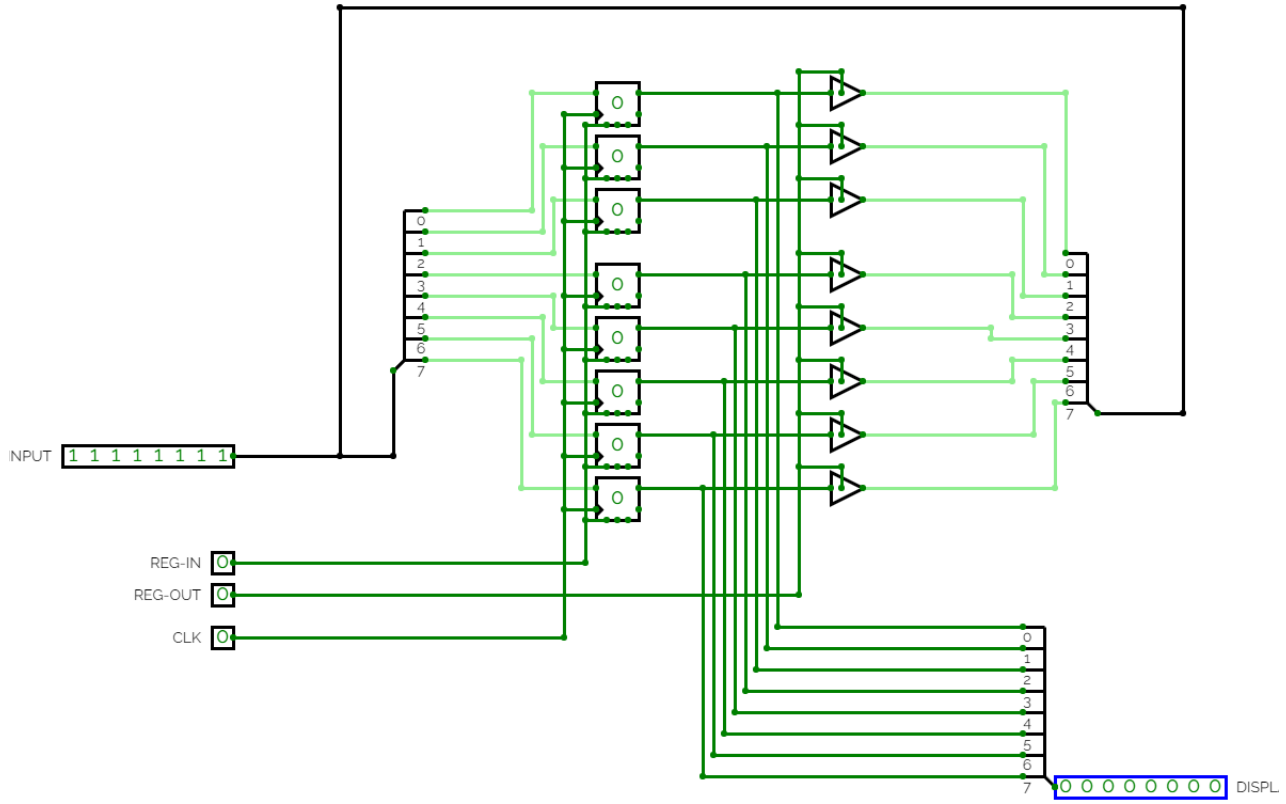

Figure 1: 8-bit Register

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_acts 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.

2 Program Counter

2.1 Diagram

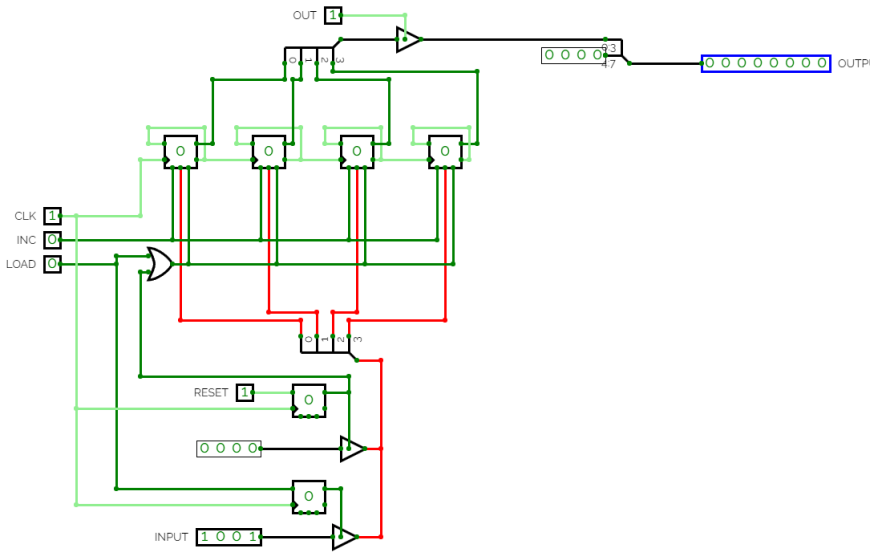

Figure 2: Program Counter

2.2 Working

Program counter is used to execute the program. It is a 4 bit data signal used to access the memory locations of the ROM. ROM consists of instructions and data of the program that has to be run . The counter starts the program at 0000 thereby accessing the instruction set in that location of the ROM via MAR. This is stored in Instruction Register which executes the program. The counter keeps incrementing by 1 and executes all the instructions in order. This cycle keep continuing until the program ends.

The following are the functionalities that are present in the Program Counter.

2.3 PC_INC

If PC _INC = 1 then the value of the address stored in the Program Counter is incremented by 1 in the next positive trigger of the clock.

2.4 PC_OUT

If PC_OUT = 1 then the Program Counter outputs the address stored in it into the **common bus** of the **Gajendra Processor** as Least Significant Bits (0-3).

2.5 PC_LOAD

If PC_LOAD = 1 then the Program Counter takes the Least Significant Bits (0-3) of the **common bus** into it through the **input** and updates the counter to that value (which is the next instruction for the ROM).

3 MEMORY

3.1 Diagram

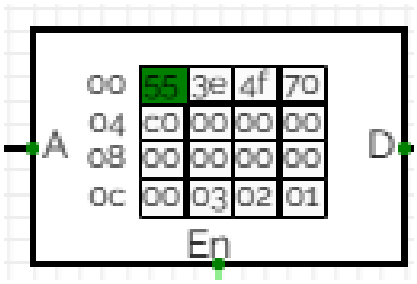

Figure 3: Memory

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

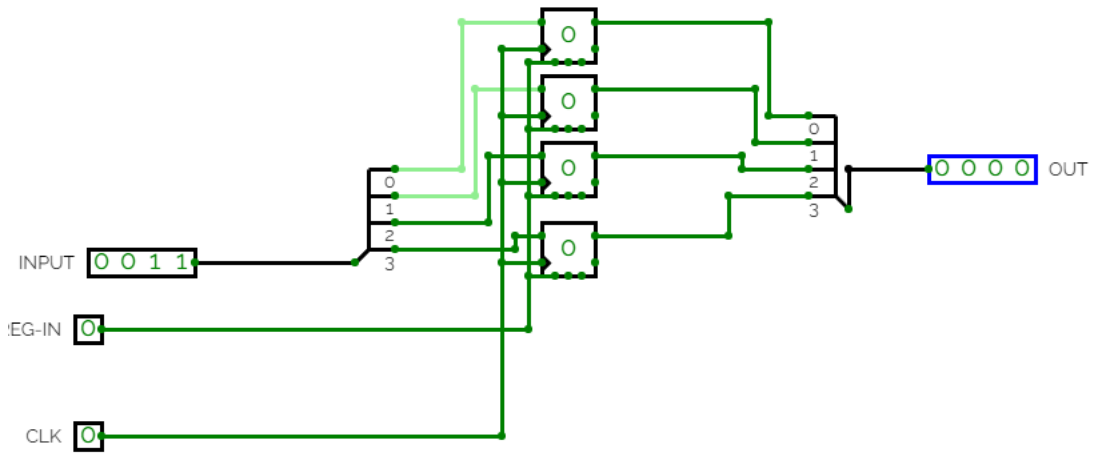

Figure 4: Memory Address Register

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

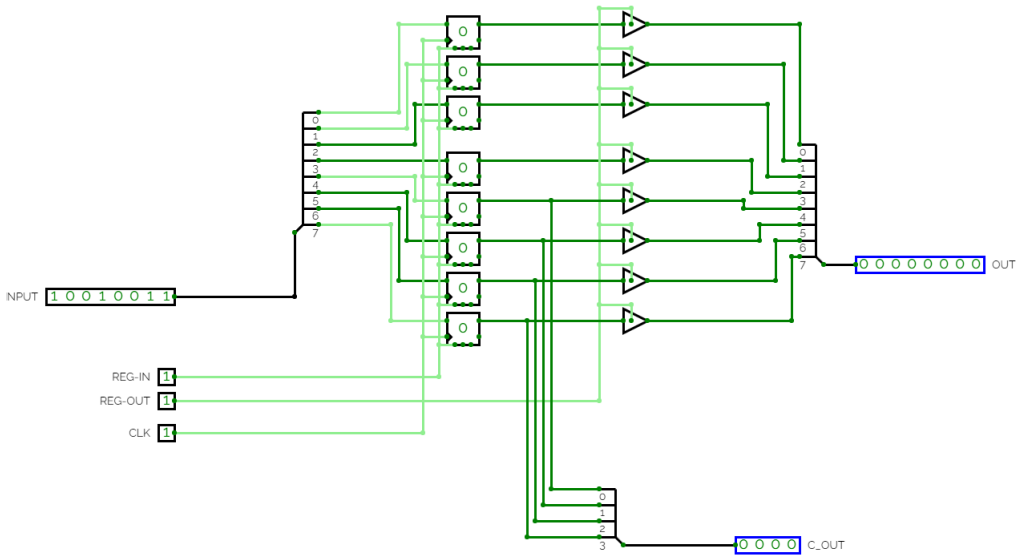

Figure 5: Instruction Register

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

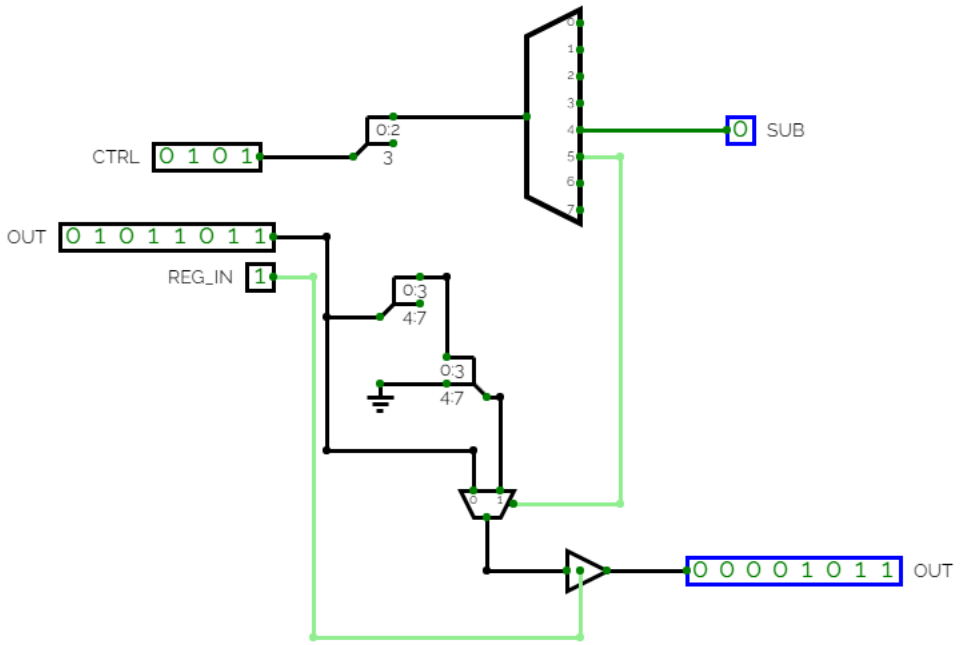

Figure 6: Instruction Decoder

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)

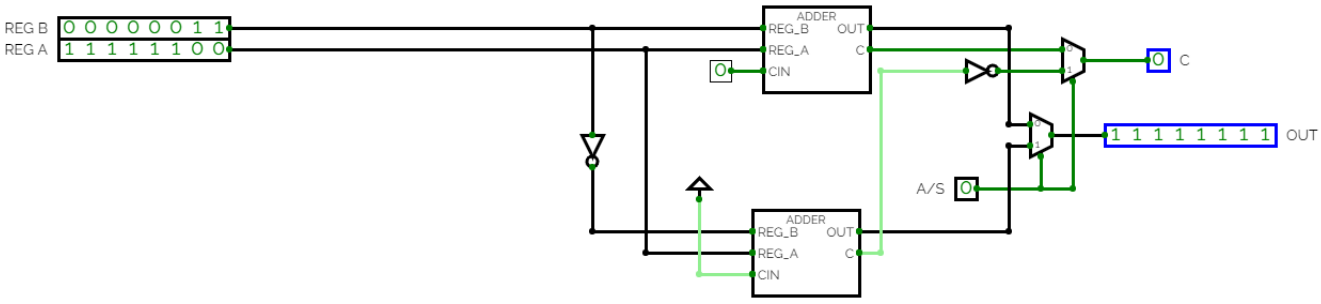

Figure 7: Gajendra ALU

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

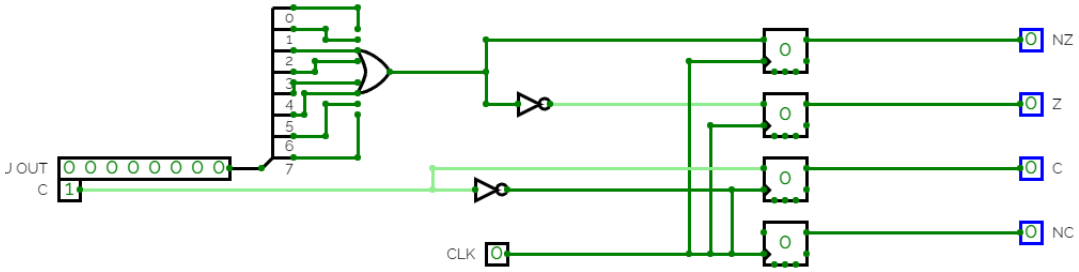

Figure 8: Status Register

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

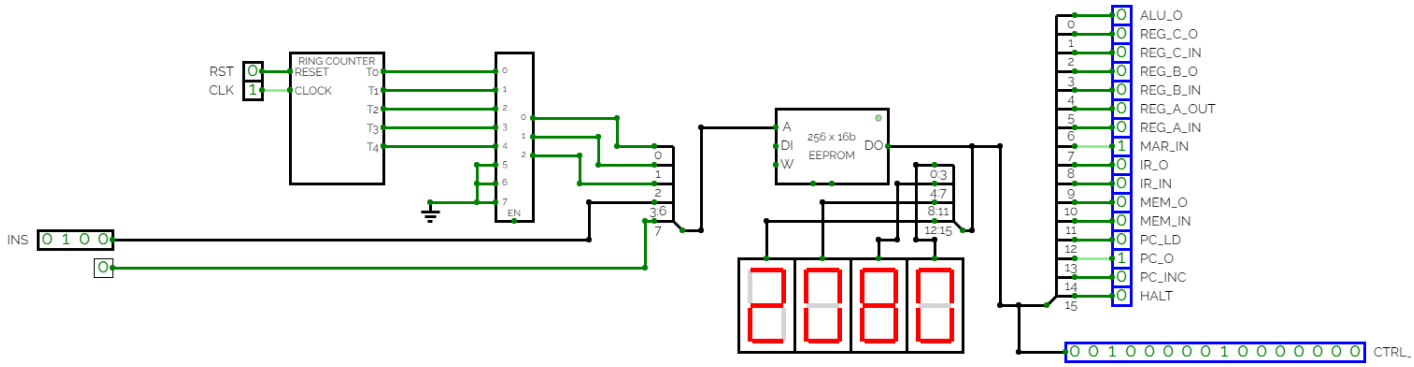

Figure 9: Control Processor

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 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 15: Micro instruction for JNZ