İstanbul Technical University

CEN 242E – Logical Circuits Lab

LAB 8: CPU Design

About LAB:

CPU Design

Stages and scores of LAB :

AvionCPU Design

1. Definition:

Within the scope of this project, the RTL design of a processor called Avion-CPU will be written in Verilog language and various code snippets written in machine language will be written on the designed processor. At the end of the project, it will be observed how the RAM, Control Unit and Registers in a simple processor can work together and execute code snippets in machine language.

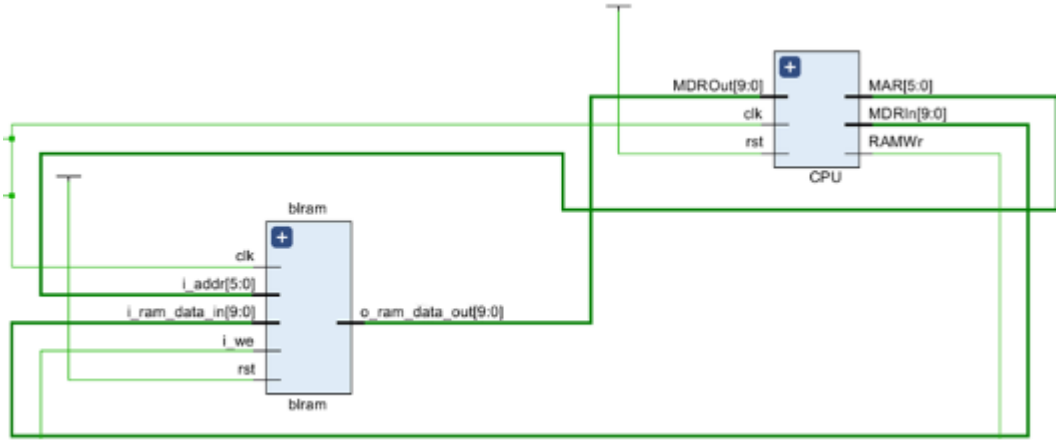
2. Design Requirements

Under this heading, the requirements of the AvionCPU to be designed are given.

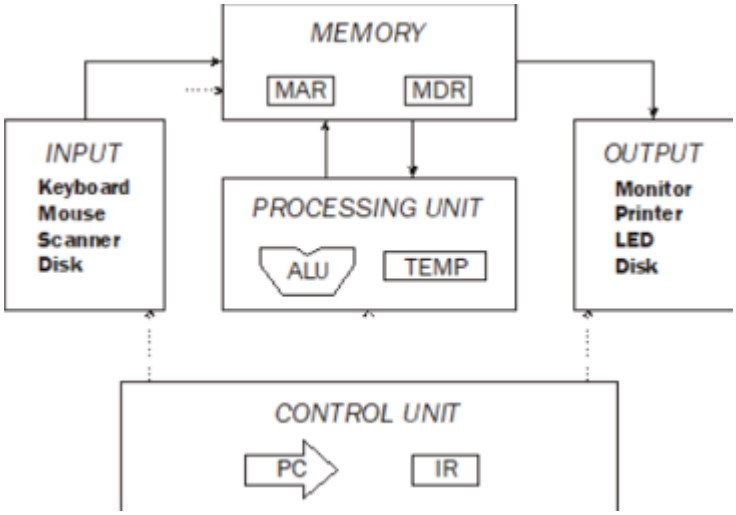
For the processor design, an initial design developed in Verilog language is given.

For the initial design, you can find it on course site.

Below figure shows the connections of the modules to each other in the testbench.



The desired AvionCPU RTL design is in Von Neumann architecture. Von Neumann Architecture is given in below figure.



This CPU has 4 main units.

- Registers
- Memory (RAM)
- Processing Unit (ALU)
- Control Unit

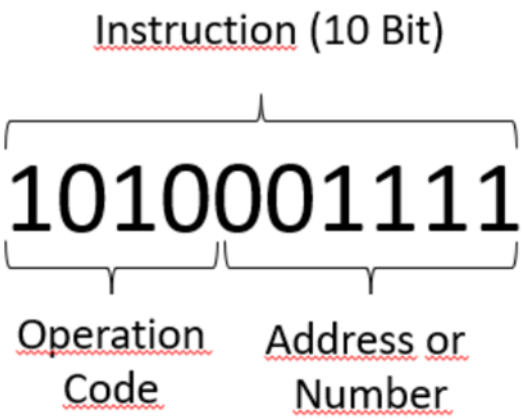
The Supported Instructions are given in following table

Table. AvionCPU ISA (Instruction Set Architecture)

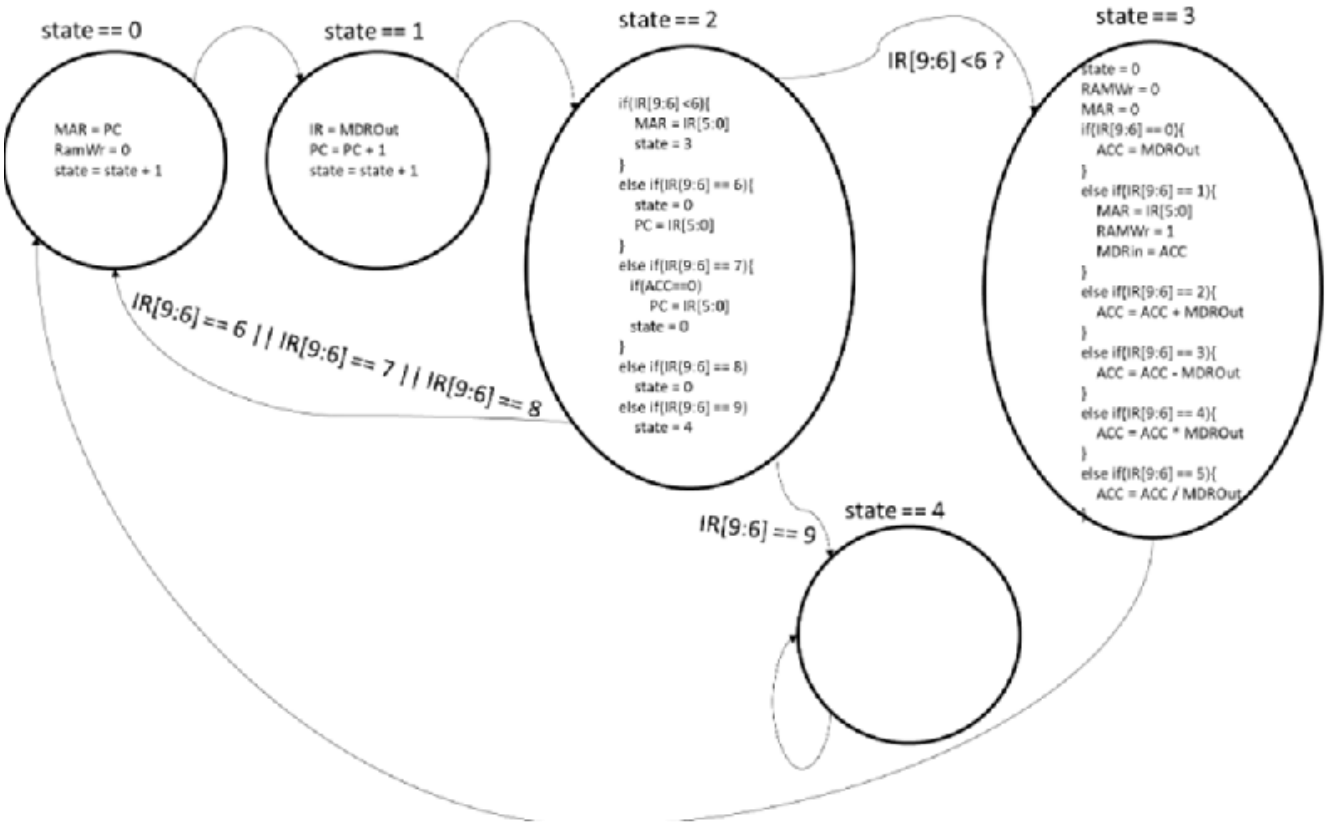
Instruction	Purpose	**Operation Code**
LOD ADDR	Load, It takes the value from the given address in memory and copies value to the ACC register. ACC = *(ADDR)	0000
STO ADDR	Store, It takes the value in ACC and writes it to the address given in memory. *(ADDR) = ACC	0001
ADD ADDR	It takes the value at the given address in memory, sums it with ACC, and overwrites to ACC. ACC = ACC +*(ADDR)	0010
SUB ADDR	It takes the value at the given address in	0011

	memory, subtracts it with ACC and overwrites to ACC. ACC = ACC - *(ADDR)	
MUL ADDR	It takes the value at the given address in memory, multiplies it with ACC and overwrites ACC. ACC = ACC * (*(ADDR))	0100
JMP NUM	PC will be given number	0110
JMZ NUM	ACC'ın value is 0, then given number will be assigned to PC otherwise PC will be only PC + 1.	0111
NOP	No Operation	1000
HLT	Halts(Stops) the execution of CPU	1001

The processor supports 9 instructions.



The state diagram of the AvionCPU is given in below figure. It shows the tasks that the processor must do step by step.



Registers

In the initial design, all registers that the AvionCPU will need are defined. There are 4 hiders in the design.

These;

• state: The state machine keeps information about the state it is in.

• PC: Information on which address in RAM the command is running is kept.

• IR: The currently running instruction itself is kept.

• ACC: Temporary storage area.

No new registers will be added. All necessary registers are included in the design.

AvionCPU will be implemented with the state machine method. In other words, this processor will have a design that works in 5 different states, depending on the value of the register named state

All other registers would operate based on the state register. In other words, the progress of the system depends on the status signal.

The memory signals connected to the input and output ports in the design are given below.

• MAR (6 Bit): It is a register called Memory Address Register. This register is connected to the address input of RAM. Since RAM has 2^6 locations, MAR is 6 bits. The register is in the inside of RAM.

• MDRIn (10 Bit): Memory Data Register In is the register used when data is written to RAM. Since one location of RAM is 10 bits, the register is 10 bits. The register is in the inside of RAM.

• RAMWr (1 Bit): It is activated when data will be written to RAM. If it is not 1, no data is written to RAM. The register is in the inside of RAM.

• MDROut (10 Bit): Memory Data Register is the register used when data is read from RAM. Since one location of RAM is 10 bits, the register is 10 bits. The register is in the inside of RAM.

Memory (RAM, Random Access Memory)

There is a Block RAM mechanism where the AvionCPU reads the commands and writes back the calculated values. The memory instantiated by the test code is located in the bram module. There are 4 registers, clock and reset signals connected to RAM. The functions of the registers connected to RAM are explained in the registers section.

Processing Unit (ALU, Arithmetic Logic Unit)

This is the section where arithmetic operations are performed. There are 3 arithmetic operations in AvionCPU. These perform operations such as addition, subtraction and multiplication according to the incoming operation code and write them to the ACC register.

Control Unit

Registers are responsible for transferring data between the Arithmetic Processing Unit and RAM. Manages intra-processor data flow. In the initial verilog design given for the processor design, the Processing and Control Unit is missing. These missing places are in states 2 and 3 in the state machine.

Example Software

In the AvionCPU 10-bit instruction, the first 4 bits [9:6] represent the operation code and the last 6 bits [5:0] represent the address.

Whether the processor is working correctly or not will be tested with the following code snippets.

Test Example 1

For AvionCPU, develop an application that saves the sum of the two numbers at addresses 50 and 51 in memory to address 52.

0: 0000_110010 // LOD 50, (ACC = *50), Hex = 32

1: 0010_110011 // ADD 51, ACC = ACC + (*51), Hex = B3

2: 0001_110100 // STO 52, (*52) = ACC, Hex = 74

3: 1001_000000 // Halt, Hex = 240

50: 0000000101 // Hex = 5

51: 0000001010 // Hex = A

Test Example 2

For AvionCPU, develop an application that saves the multiplication of the two numbers at addresses 50 and 51 in memory to address 52.

0: 0000_110010 // LOD 50, (ACC = *50), Hex = 32

1: 0100_110011 // ADD 51, ACC = ACC * (*51), Hex = 133

2: 0001_110100 // STO 52, (*52) = ACC, Hex = 74

3: 1001_000000 // Halt, Hex = 240

50: 0000000101 // Hex = 5

51: 0000001010 // Hex = A

Test Example 3

For AvionCPU, develop an application that saves the multiplication of two numbers at addresses 50 and 51 in memory to address 52. However, do not use the multiplication operation. For multiplication, add the number in address 50 times the number in address 51 and write it to address 52.

0: 0000_110011 // LOD 51, ACC = *51, Hex = 33

1: 0011_110001 // SUB 49, ACC = ACC - *49, Hex = F1

2: 0111_001010 // JMZ 10, If the loop is finished, it will exit the loop (ACC-49 == 0), 10. Row, Hex = 1CA

3: 0000_110000 // LOD 48, load temp value, initially 0, Hex = 30

4: 0010_110010 // ADD 50, add the second number above ACC, Hex = B2

5: 0001_110000 // STO 48, Assign value of ACC to temp, Hex = 70

6: 0000_110001 // LOD 49, ACC = i, Hex = 31

7: 0010_101110 // ADD 46, ACC = i + 1, Hex = AE

8: 0001_110001 // STO 49, i = i + 1, Hex = 71

9: 0110_000000 // JMP 0, return to the beginning of the loop line 0, Hex = 180

10: 0000_110000 // LOD 48, ACC = temp, Hex = 30

11: 0001_110100 // STO 52, *52 = ACC, Hex = 74

10: 1001_000000// HLT, halting, Hex = 240

46: 1 // 1 sayısı

48: 0 // Hex = 0, temp

49: 0 // Hex = 0, i index

50: 0000000101 // Hex = 5

51: 0000001010 // Hex = A