Processor Architecture Project Report

RISC-V ISA

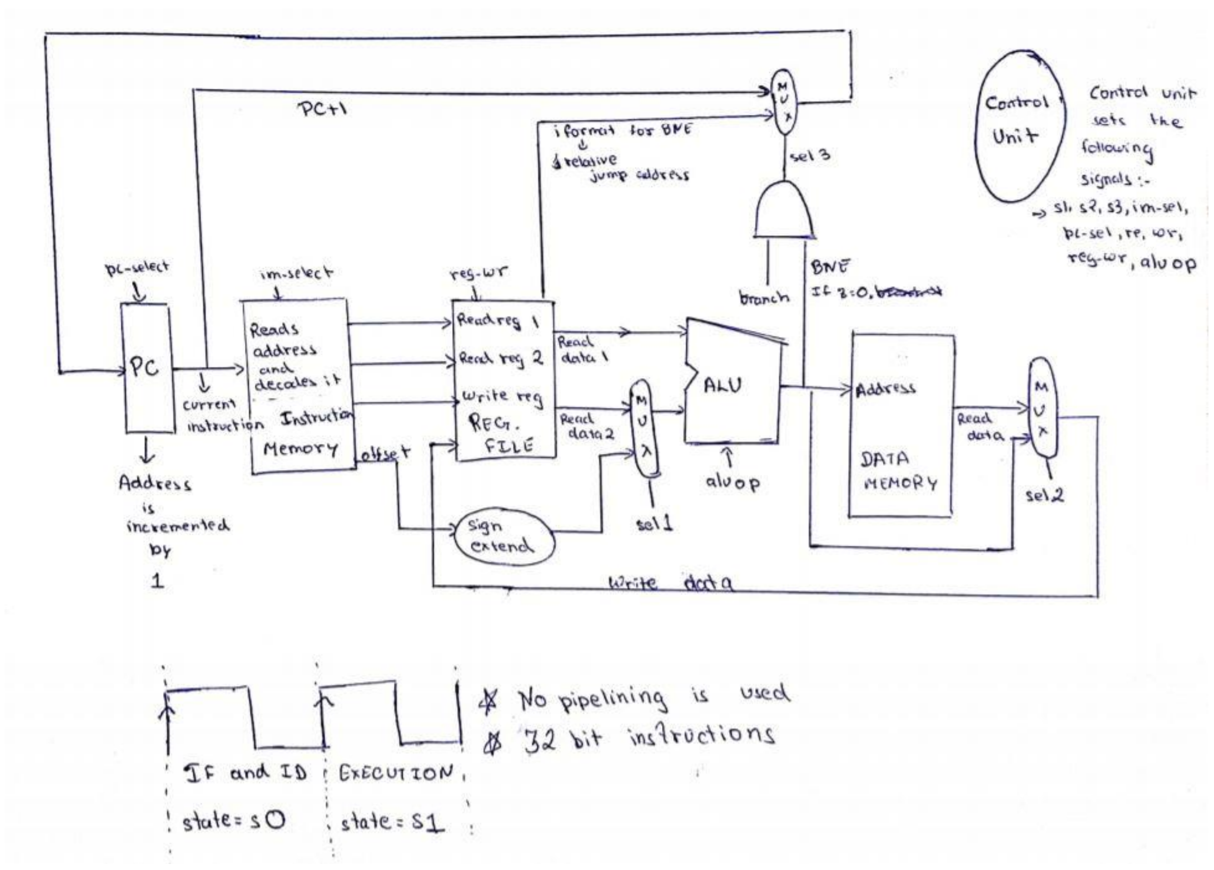
Utkarsh Pratap Singh (IMT2015522)
Alan Israel (IMT2015526)
Bharath N (IMT2015527)

Objective

To write a small assembly program to evaluate a factorial example. To design a single-cycle processor to accommodate RISC-V ISA and run this example.

Implementation

The RISC-V processor built using Verilog was inspired by the MIPS architecture. Harvard Architecture has been used as the instruction memory is separate from the data memory. It is a 32-bit instruction set processor with data word length as 16-bit. It is a non-pipelined processor which takes 2 cycles to complete the execution of a single instruction. In the first clock cycle it fetches the instruction from the instruction memory from the address put out by the program counter. The decoding of the instruction is also done in this phase as and when the instruction is dispatched from the instruction memory. There is one ALU unit which does the basic operations such as add, subtract and multiply. The basic architecture looks like this:



The modules comprising this architectural design in Verilog is as follows:

- **Risc module** – This is the top module which initiates the control and the data flow.
- **Control module** – This module sets the control signals that are needed to initiate the data flow in the processor. It basically has two state S0 and S1. The processor alternates between these states during Instruction Fetch/Instruction Decode stage and the Execution stage.
- **Datapath module** – This module determines the data flow across all the different units in the processor. The modules which are being used by this module are described below.
- **PC module** – The program counter module is used for incrementing the PC at every clock edge when selected.
- **Instruction_fetch module** – The module is used to put out the instruction from the instruction memory when an address from the PC is fed in. It also decodes the instruction.
- **Register module** – This module is used to issue the data in the registers and also to write back the result from either ALU or from the data memory to the register.
- **Sign module** – This module is used to expand a 12-bit address to a 16-bit address.
- **Latch module** – This module is used to hold the data for one clock cycle.
- **Mux module** – This module is a multiplexer module which selects between two inputs. There are three multiplexers implemented in the processor.
- **ALU module** – This module is the Arithmetic Logic Unit. This module performs the basic operations like add, subtract and multiply.
- **Datamemory module** – This module represents the data memory. It is used to read and write data from the data memory.

<u>**Observations and Results**</u>

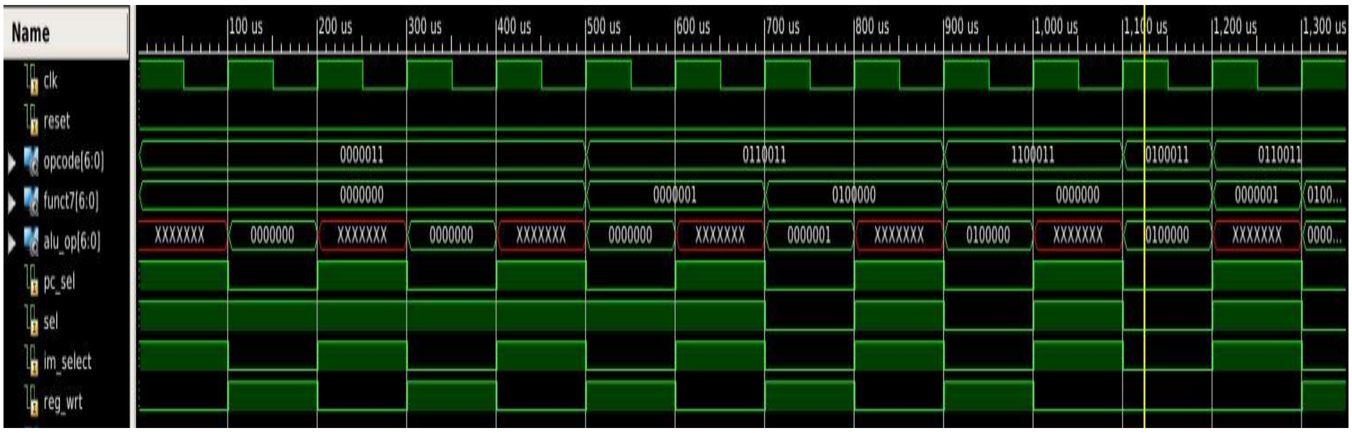
The assemble code for the factorial program:

```
LW R1,R0+data_loc(0)
LW R2,R0+data_loc(1)
LW R3,R0+data_loc(0)
MUL R1,R2,R1
SUB R2,R2,R3
BNE R2,R3,instruction_loc(3)
SW R1,R2+data_loc(2)
```

The above code translates to:

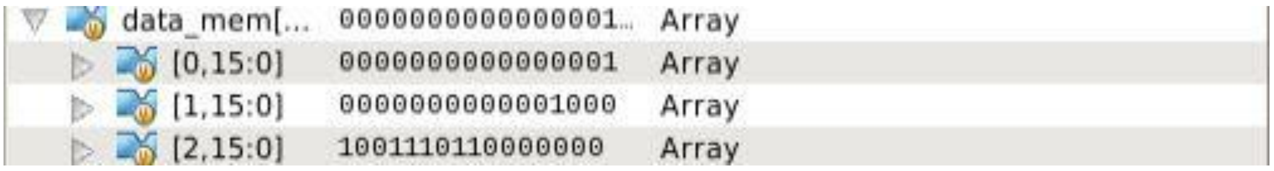
```
R1 = 1
R2 = n
R3 = 1
Loop:   R1 = R2*R1

        R2 = R2-R3

        R2 = R3 ? exit : Loop

Store R1 in R2+offset
```

The different signals produced during the running of the processor can be seen below:



The clock cycle time was given to be 100us. The pc_select can be seen on and off on alternate clock cycles. This represents the alternating between state S0 and S1 from which the IF/ID and then the EX/WB is done.

The result computed when the factorial of 8 was computed :



```
data_mem[...   0000000000000001...   Array
  [0,15:0]     0000000000000001      Array
  [1,15:0]     0000000000001000      Array
  [2,15:0]     1001110110000000      Array
```

Acknowledgement

We thank Prof.Nandita Rao ma’am for giving us the opportunity to work on this project.