REPORT ON PROCESSOR IMPLEMENTATION:

Ammannaidu Gedela (EE13M072)
Laxmeesha S (EE13M074)

Below are the specifications of the microprocessor designed :

Address Bus - 8bits
Data Bus - 8 bits
Number of registers - 8 (each of width 8)

ALU :

- No status flags implemented
- 3 bit bus for function selection
- Two 8 bit inputs (in1 and in2)
- One 8bit output
- ALU sel (mux) to select between register content and immediate content.

This is implemented using case statement. Supports the following operations:
Addition , Subtraction, And , OR.
File name : alu.v

REGISTER FILE:

- Eight 8 bit registers r0 to r7
- Register r0 intialised to 0
- Two 3 bit bus signals sr1 and sr2 to read out register contents for in1 and in2
- Two eight bit outputs r1 and r2
- One 8 bit bus rin and one 3 bit address bus srin to write data into register file
- Clk signal. Always at clock read r1 and r2 and then write rin.

Implemented using registers in verilog.

File name: reg_file.v

INSTRUCTION MEMORY:

- 8 bit address bus.
- 16 bit data bus.

This 16 bit data (instruction) is split in the below manner :

Type	15-12	11-9	8-6	5-3	2-0
R (register)	Opcode	Rd	Ra	Rb	Func
I (immediate)	Opcode	Imm[5:3]	Ra	Rb	Imm[2:0]
J (jump)	Opcode	- don't care -		Addr (use 6:0 inst of 5:0)	

If Jump = 0 , then PC (Program counter is incremented), else the PC points to new jump address.

The instruction memory can be instantiated in Xilinx as a ROM and a pre-programmed coe file (which has machine equivalent of the assembly) is loaded. Here for simplicity, the instruction memory is implemented using a register IMEM and the contents are loaded into this using “readmemb” in verilog.

File name: inst_memory.v

DATA MEMORY:

- 8 bit address bus
- 8 bit dout bus and 8 bit din bus
- Write-enable and clk

The RAM is instantiated from the block memory core in Xilinx for Data memory.

CONTROL UNIT:

This is implemented using state machines (case statement). The instruction from IMEM is parsed into the corresponding fields i.e. opcode, sr1, sr2, func, imm and jump address.

Following are the states:

Instruction	Opcode/Func
Add rd, ra, rb	0000 / 000
Sub rd, ra, rb	0000 / 010
And rd, ra, rb	0000 / 100
Or rd, ra, rb	0000 / 101
Addi rd,ra, imm	0100
Lw rd, imm(ra)	1011
Sw rd, imm(ra)	1111
Beq rd, ra, imm	1000
J addr	0010

Along with this 2 more states are added :

Instruction	Opcode
End	1001
Read from DMEM	1101

The Program counter is modified accordingly inside the control unit , based on the opcode (if jump or branch instruction) and other previous conditions. The stepping of PC is stopped when “end” is received in the code.

Testing the processor:

To test the processor functionality a multiplication code is executed. The objective of code is to multiply contents in the register r1 and r2 and store result in r3. To multiply contents of register r5 and r6 and store the result in r7. Then copy the contents of r3 and r7 into the data memory i.e. the RAM (address 0 and 1 respectively).

Below is the assembly code for the same:

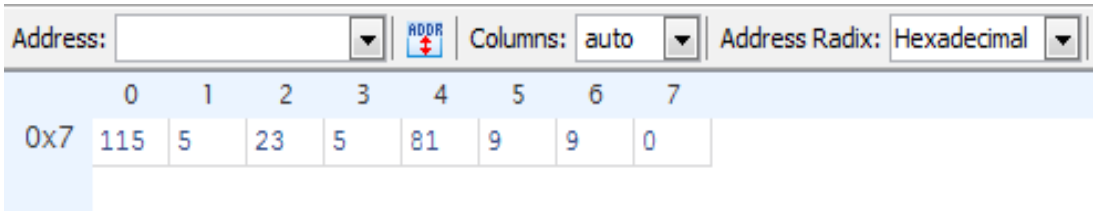
```
0 :    AND r3,r3,r0              // clearing r3 register
1 :   AND r4,r4,r0               // clearing r4 register
```

```
2 :   BEQ r2,r4,#6              // Branch to address 6
3 :   ADD r3,r3,r1
4 :   ADDi r4,r4,#1
5 :   J    #2                   //Unconditional jump to address 2
6 :   AND r7,r7,r0              //clear r7
7 :   AND r4,r4,r0              //clear r4
8 :   BEQ r6,r4,#12
9 :   ADD r7,r7,r5              //result in r7
10:   ADDi r4,r4,#1
11:   J    #8
12:   END
```

The corresponding machine code is as below :

```
0  : 0000011011000100
1  : 0000100100000100
2  : 1000000010100010
3  : 0000011011001000
4  : 0100000100100001
5  : 0010000000000001
6  : 0000111111000100
7  : 0000100100000100
8  : 1000000110100101
9  : 0000111111101000
10: 0100000100100001
11: 0010000000000100
12: 1001000000000000
13: 0000000000000000
```

The module is simulated with ISIM and the outputs are found to be proper.



R1 -> 9 R5->23

R2 ->9 R6->5

R3 ->81 R7->115

TIMING REPORT:

Minimum period: 6.985ns (Maximum Frequency: 143.164MHz)

Minimum input arrival time before clock: 0.336ns

Maximum output required time after clock: 3.259ns

Maximum combinational path delay: No path found

Then VIO and ILA are instantiated to view these registers in chip scope.

The corresponding verilog files and the list files for this project are attached.