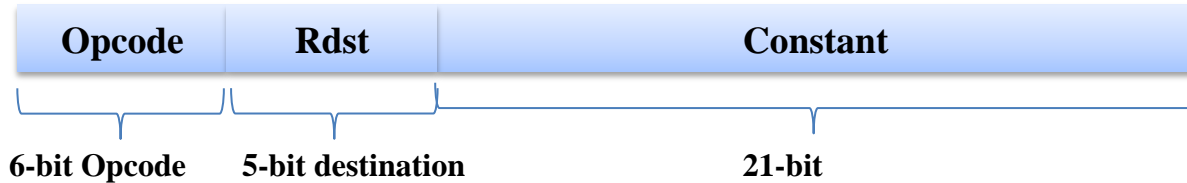


# Assignment 1: Implementation of a MIPS like processor (10Marks)

Design and implement (in Verilog) datapath and control unit for a single cycle MIPS like processor (including instruction memory) which has two classes of instructions. The two classes of instructions along with the example usage and instruction decoding to be used are as below

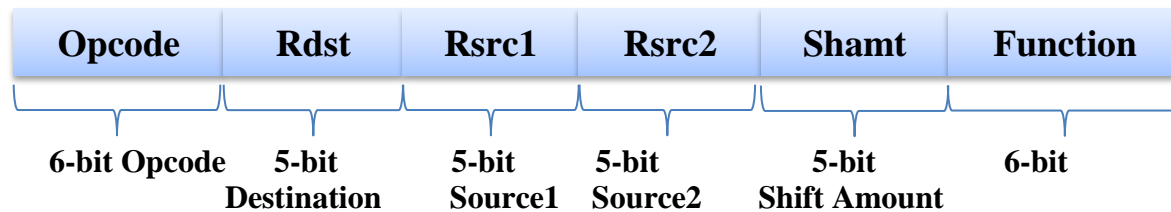
## 1. Immediate Type

Example: `li r1, constant` → loads immediate signed value specified in the instruction to the register R1



## 2. Register Type (R-type)

Example: `add r1, r2, r3` → adds the contents of registers r2 and r3. The result of addition is written in to the register r1



Assume there are 32 32-bit general purpose registers indicated by r0, r1, r2...r31 and corresponding register numbers (00000), (00001).....(11111).

Assume the Opcode for Immediate type and R-type instructions as below

Instruction Class	Opcode
Immediate type	111111
Register Type	000000

Additionally R-type instructions have multiple variations defined by their function codes. The R-type instructions should include **add**, **sub**, **AND**, **OR**, **srl** (Shift right logical), **sll** (shift left logical) .The different R-type instructions that the processor should support are tabulated below.

R-type Instruction	Example usage	Opcode	Rdst	Rsrc1	Rsrc2	shamt	Function
<b>add</b>	<code>add r0, r1, r2</code>	000000	00000	00001	00010	00000	100000
<b>sub</b>	<code>sub r4, r5, r6</code>	000000	00100	00101	00110	00000	100010
<b>AND</b>	<code>and r8, r9, r10</code>	000000	01000	01001	01010	00000	100100
<b>OR</b>	<code>and r9, r8, r10</code>	000000	01001	01000	01010	00000	100101
<b>sll</b>	<code>sll r11, r6, 6</code>	000000	01011	00110	00000*	00110	000000
<b>srl</b>	<code>srl r13, r9, 10</code>	000000	01101	01001	00000*	01010	000010

\*Second source is not used for shift operations

The processor module should have only two inputs CLK and Reset. When Reset is activated the Processor starts executing instructions from 0<sup>th</sup> location of instruction memory.

**As part of the assignment the following files should be submitted in zipped folder.**

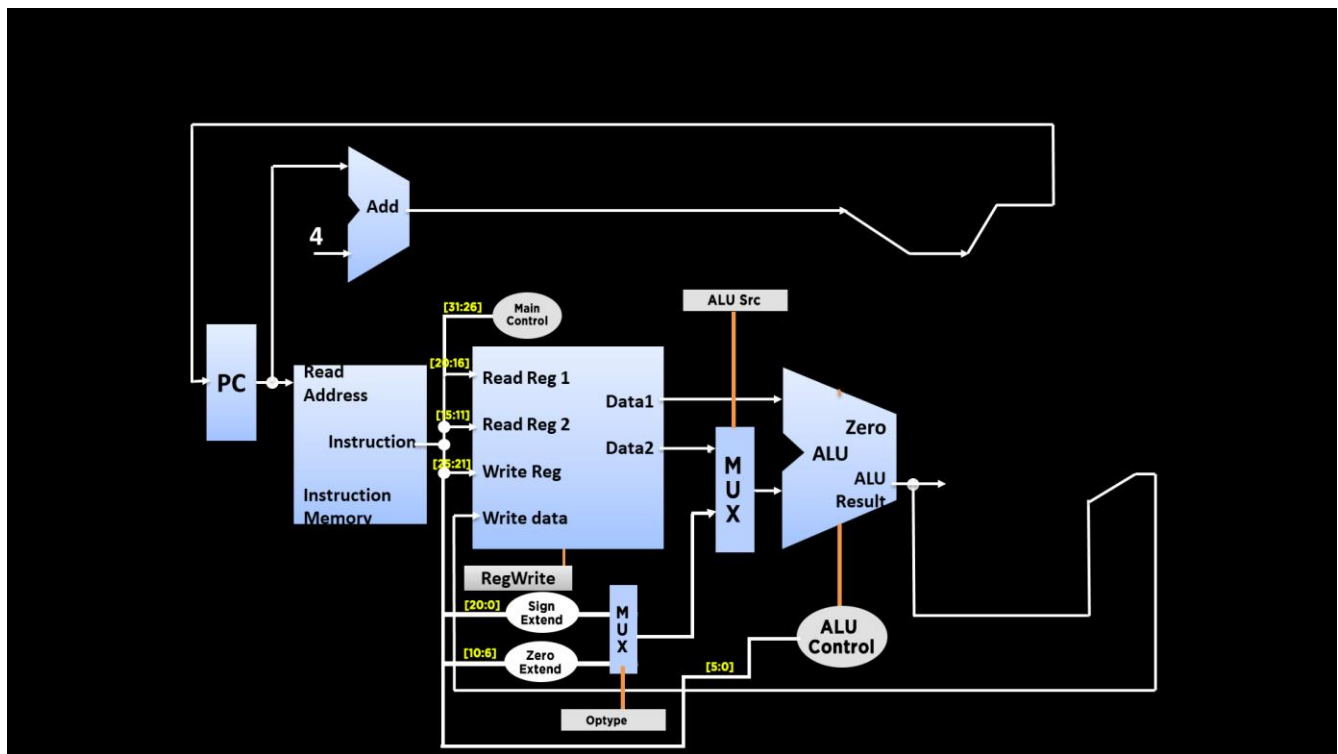
1. PDF version of this Document with all the Questions below answered with file name as **IDNO\_NAME.pdf**.
2. Design Verilog Files for all the Sub-modules (including control unit).
3. Design Verilog file for the main processor.

**The name of the zipped folder should be in the format IDNO\_NAME.zip**

**The due date for submission is 3-April-2019, 5:00 PM.**

**Q6.1. Draw the block level design of the processor (datapath + control unit) for above specifications. (you can modify the design given in the class ppts and copy the image of final design here)**

Answer:

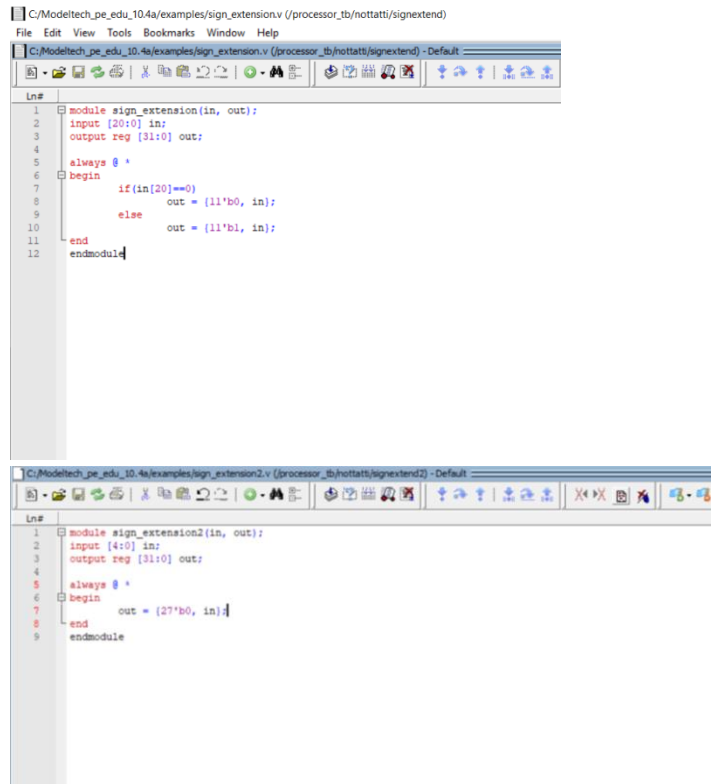


**Q6.2. List the different blocks that will be required for implementation of datapath of the above processor.**

Answer: Instruction Memory, Instruction fetch, instruction decode, ALU, 2:1 Mux, sign extend, zero extend, processor, control unit.

**Q6.3. Most of the datapath blocks that are listed above have already been implemented as part of previous labs. Implement the blocks which have not been implemented in the previous labs and copy the images of those Verilog codes here.**

Answer: sign extend



The first screenshot shows a Verilog module named `sign_extension` with the following code:

```

1 module sign_extension(in, out);
2   input [20:0] in;
3   output reg [31:0] out;
4
5   always @ *
6   begin
7     if(in[20]==0)
8       out = {11'b0, in};
9     else
10      out = {11'b1, in};
11   end
12 endmodule

```

The second screenshot shows a Verilog module named `sign_extension2` with the following code:

```

1 module sign_extension2(in, out);
2   input [4:0] in;
3   output reg [31:0] out;
4
5   always @ *
6   begin
7     out = {27'b0, in};
8   end
9 endmodule

```

**Q6.4. Assume Main control unit generates all the control signals. List different control signals that will be required for the above processor. Also specify the value of the control signals for different instructions.**

Answer:

Control Signal Name →	ALU Src	ALU Control	regwrite	optype	
li r1, 8	1	111	1	1	
add r0, r1, r2	0	000	1	0	
sub r4, r5, r6	0	001	1	0	
and r8, r9, r10	0	010	1	0	

or r9, r8, r10	0	011	1	0	
sll r11, r6, 6	1	100	1	0	
srl r13, r9, 10	1	101	1	0	

**Q6.5. Implement the main control unit and copy the image of Verilog code of Main control unit here.**

Answer:

```

C:/Modeltech_pe_edu_10.4a/examples/control_unit.v (/processor_tb/nottatti/ctrl)
File Edit View Tools Bookmarks Window Help
C:/Modeltech_pe_edu_10.4a/examples/control_unit.v (/processor_tb/nottatti/ctrl) - Default

Ln#
1 module control_unit(opcode, funct, alu_control, alu_src, optype);
2
3 input[5:0]opcode, funct;
4 output reg [2:0] alu_control;
5 output reg alu_src;
6 output reg optype;
7
8 always @ *
9 begin
10     if(opcode[5]==0)
11         optype = 0;
12     else
13         optype = 1;
14 end
15
16 always @ *
17 begin
18     if(opcode[5] == 0)
19     begin
20         if(funct[5]==1)
21             alu_src = 0;
22         end
23     else
24         alu_src = 1;
25 end
26
27 always @ *
28 begin
29     case ({opcode[5],funct})
30     7'b0100000 : alu_control = 3'b000;
31     7'b0100010 : alu_control = 3'b001;
32     7'b0100100 : alu_control = 3'b010;
33     7'b0100101 : alu_control = 3'b011;
34     7'b0000000 : alu_control = 3'b100;
35     7'b0000010 : alu_control = 3'b101;
36     default : alu_control = 3'b111;
37     endcase
38 end
39 endmodule

```

**Q6.6. Implement complete processor in Verilog (Instantiate all the datapath blocks and main control unit as modules). Copy the image of Verilog code of the processor here.**

Answer:

```
C:/Modeltech_pe_edu_10.4a/examples/processor.v (/processor_tb/nottatti)
File Edit View Tools Bookmarks Window Help
C:/Modeltech_pe_edu_10.4a/examples/processor.v (/processor_tb/nottatti) - Default

Ln#
1 module processor(clk,reset);
2 input clk;
3 input reset;
4
5 wire [31:0] instr_code;
6 wire [4:0] id_read_reg_1, id_read_reg_2, id_write_reg;
7 wire [31:0] id_write_data;
8 wire [31:0] id_read_data_1, id_read_data_2, alu_result, signed_const, signed_const_li, signed_const_sh, alu_input_1, alu_input_2;
9 wire id_regwrite;
10 wire alu_src, optype;
11 wire [2:0] alu_control;
12
13 assign alu_input_1 = id_read_data_1;
14 assign id_read_reg_1 = instr_code[20:16];
15 assign id_read_reg_2 = instr_code[15:11];
16 assign id_write_data = alu_result;
17 assign id_write_reg = instr_code[25:21];
18 assign id_regwrite = 1'b1;
19
20 instr_fetch if1(clk,reset,instr_code);
21 //instr_mem insmem(pc, reset, instr_code);
22 id_instr_dec(id_read_reg_1, id_read_reg_2, id_write_reg, id_write_data, id_read_data_1, id_read_data_2, id_regwrite, clk);
23 alu_mod al(alu_input_1, alu_input_2, alu_control, alu_result);
24 sign_extension1 signextend1(instr_code[20:0], signed_const_li);
25 sign_extension2 signextend2(instr_code[10:6], signed_const_sh);
26 mux_2ip_3lb mu2(optype, signed_const_sh, signed_const_li, signed_const);
27 mux_2ip_3lb mu(alu_src, id_read_data_2, signed_const, alu_input_2);
28 control_unit ctrl(instr_code[31:26], instr_code[5:0], alu_control, alu_src, optype);
29
30
31 endmodule
32
33
```

**Q6.7. Test the processor design by initializing the instruction memory with a set of instructions (at least 5 instructions). List below the instructions you have used to initialize the instruction memory. Verify if the register file is changing according to the instructions. (Register file contains unknowns, you can initialize the register file or you can load values into the register file using li instruction specified earlier).**

Sequence of Instructions Implemented:

1. add r0, r1, r2 //r0 = 1+2 = 3
2. sub r4, r5, r6 //r4 = 5-6 = -1 = 32'hfffffff
3. add r3, r0, r4 //r3 = 3+ (-1) = 2
4. li r6, 25 //r6 = 25
5. sll r11, r6, 3 //r11 = 25 <<3 = 25 x 8 = 200
6. li r12, 16 //r12 = 16
7. srl r12, r12, 4 //r12 = 16 >>4 = 16/16 = 1

Register file is loaded with values.

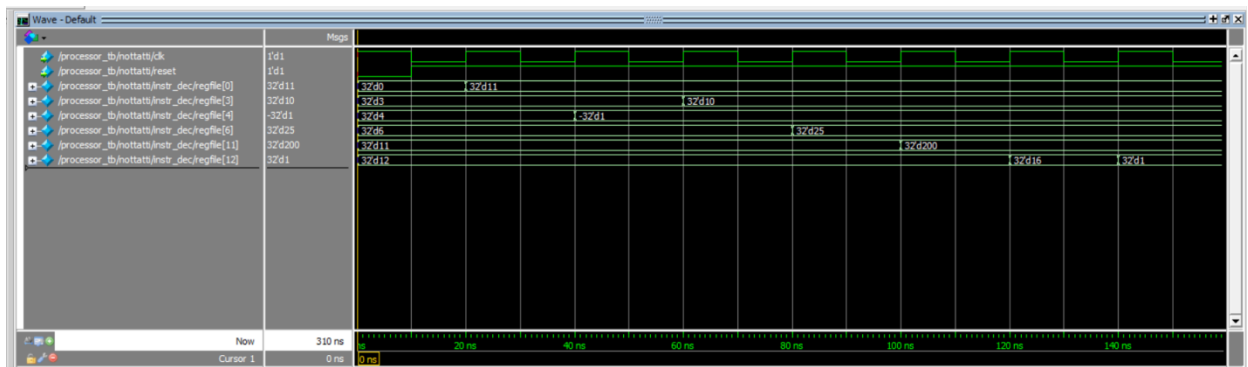
r0 contains 0

r1 contains 1

r2 contains 2 ...

**Q6.8. Verify if the register file is getting updated according to your sequence of instructions (mentioned earlier).**

Copy verified **Register file** waveform here (show only the Registers that get updated, CLK, and RESET):



### Unrelated Questions

What were the problems you faced during the implementation of the processor?

Answer: Implementation of the control unit and inputs to the ALU.

Did you implement the processor on your own? If you took help from someone whose help did you take? Which part of the design did you take help for?

Answer: On my own.

### Honor Code Declaration by student:

- My answers to the above questions are my own work.
- I have not shared the codes/answers written by me with any other students. (I might have helped clear doubts of other students).
- I have not copied other's code/answers to improve my results. (I might have got some doubts cleared from other students).

**Name:** Neil Thanawala  
**ID No.:** 2015A8PS0517G

**Date:** April 3, 2019

