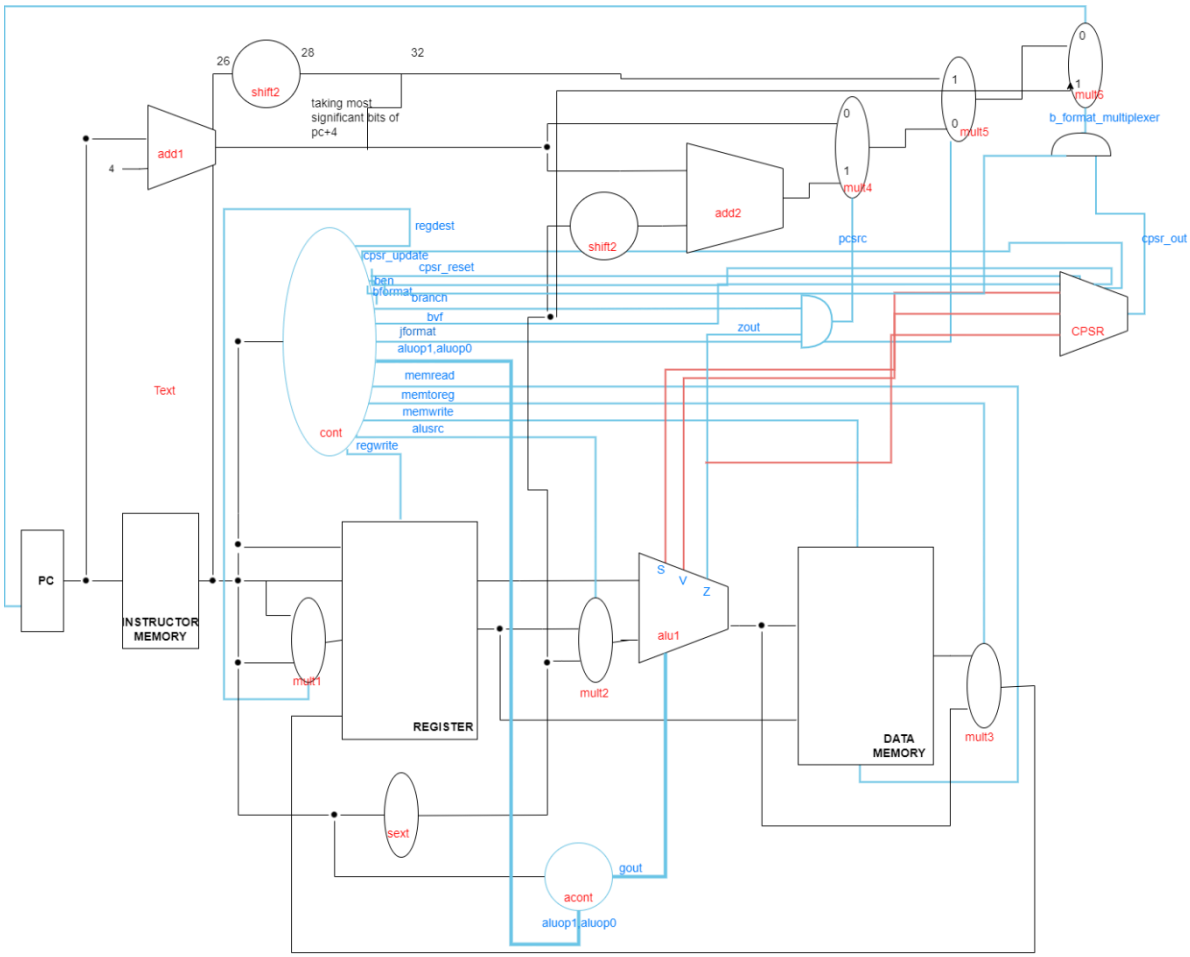
280201056 Ceng311 HW3 Report Gencay Turgut

Here is the cpu that I designed:



I added some components related to the cpu that I've designed.

J-FORMAT

Added these in processor.v:

```
wire cpsr_out,pcsrc,regdest,alusrc,memtoreg,regwrite,memread,ben,bvf,sout,vout,zout,cpsr_reset,memwrite,branch,aluop1,aluop0,cpsr_update,
b_format_multiplexer,j_format,b_format;
```

```
//shift jump adress to the left 2 times
assign jumpaddress = inst25_0[25:0];//26
assign jumpaddress2 = {jumpaddress}<<2;//28
//add pc+4 to jump adress
assign jumpaddress3= {adder1out[31:28],jumpaddress2[27:0]};
```

```
mult2_to_1_32 mult5(out5, out4,jumpadress3,j_format);//jump
```

Added these in control.v:

```
module control(in, regdest, alusrc, memtoreg, regwrite, memread, memwrite,j_format, branch, aluop1, aluop2, cpsr_reset,
cpsr_update, b_format,ben,bvf);
input [5:0] in;
output regdest, alusrc, memtoreg, regwrite, memread, memwrite,j_format, branch, aluop1, aluop2,b_format,cpsr_update, ben,
bvf,cpsr_reset;

wire b_format,r_format,lw,sw,beq,itype,addi;
```

```
assign j_format = (~in[5])& (~in[4])&(~in[3])&(~in[2])&in[1]&(~in[0]);
```

NOR

Added this in alucont.v:

```
if (f2 & f1 & f0 &~(f3))
    gout = 3'b100;
```

Added this in alu32.v:

```
3'b100: alu_out = ~(a | b);//bitwise NOR
```

ADDI

Added these in control.v:

```
assign addi = (~in[5])& (~in[4])&(in[3])&(~in[2])&(~in[1])&(~in[0]);
```

```
assign itype= addi|lw|sw;
```

```
assign alusrc = lw|sw|addi;
```

```
assign regwrite = rformat|lw|addi;
```

CPSR

Added these in processor.v:

created cpsr_out, ben, bvf, sout, vout, cpsr_rest, cpsr_update, cpsr_read, b_format_multiplexer, b_format wires additionally.

Here, ben and bvf stands for reading. I check them in control and sending a signal to cpsr unit. If there is a stack overflow, bvf gives output 1. If the value stored in cprs is negative or zero, ben gives output 1.

```
wire cpsr_out,pcsrc,regdest,alusrc,memtoreg,regwrite,memread,ben,bvf,sout,vout,zout,cpsr_reset,memwrite,branch,aluop1,aluop0,cpsr_update,
b_format_multiplexer,j_format,b_format;
```

```
mult2_to_1_32 mult6(out6, out5,extad,b_format_multiplexer);//btype
```

```
control cont(instruc[31:26],regdest,alusrc,memtoreg,regwrite,memread,memwrite,j_format,branch,
aluop1,aluop0,cpsr_reset,cpsr_update,b_format,ben,bvf);

cpsr cpsr_(ben,bvf,sout,vout, zout , cpsr_out, cpsr_reset, cpsr_update);
```

```
assign b_format_multiplexer = b_format && cpsr_out;
```

Added these in control.v:

```
module control(in, regdest, alusrc, memtoreg, regwrite, memread, memwrite,j_format, branch, aluop1, aluop2, cpsr_reset, cpsr_update, b_format,ben,bvf);
input [5:0] in;
output regdest, alusrc, memtoreg, regwrite, memread, memwrite,j_format, branch, aluop1, aluop2,b_format,cpsr_update, ben, bvf,cpsr_reset;

wire b_format,r_format,lw,sw,beq,j_format,itype,addi;// ,addi
```

```
//assign bvf to 0x5
assign bvf = (~in[5])& (~in[4])&(~in[3])&in[2]&(~in[1])&in[0];
//assign ben to 0x6
assign ben = (~in[5])& (~in[4])&(~in[3])&in[2]&in[1]&(~in[0]);
```

```
assign cpsr_update = r_format|itype;
assign cpsr_reset = b_format | branch | (~cpsr_update) | j_format;
```

Added these in alu32.v:

```
module alu32(alu_out, a, b, zout, vout, sout, alu_control);
output reg [31:0] alu_out;
input [31:0] a,b;
input [2:0] alu_control;

reg [31:0] less;
output zout, vout, sout;
reg zout, vout, sout;
```

```
zout=~(|alu_out);
sout = alu_out[31];
begin
    if (alu_control==3'b010)
    begin
        if (a[31]==b[31])
            if (a[31]!=alu_out[31])
                vout=1;
    end
    if (alu_control==3'b110)
    begin
        if (a[31]!=b[31])
            if (a[31]!=alu_out[31])
                vout=1;
    end
end

end
endmodule
```

Created a unit called “cpsr.v”:

```
module cpsr(ben,bvf,sout,vout, zout , out, reset, update);
input ben, bvf, sout, vout, zout,reset,update;
reg [2:0] flags;
output reg out;
reg clk;

always@(posedge clk)
begin
if (reset)
//sout = 0 , vout = 0 , zout = 0
flags = 3'b000;
else if (update)
flags = {sout, vout, zout};
end

always @(ben|bvf)
begin
if (ben&&flags[0])
out = flags[0];
else if (bvf&&flags[1])
out = flags[1];
else if (ben&&flags[2])
out = flags[2];
else
out = 0;
end
```

```
initial
begin
clk=0;
end

always #5 clk = ~clk;

endmodule
```

Here are my test results: