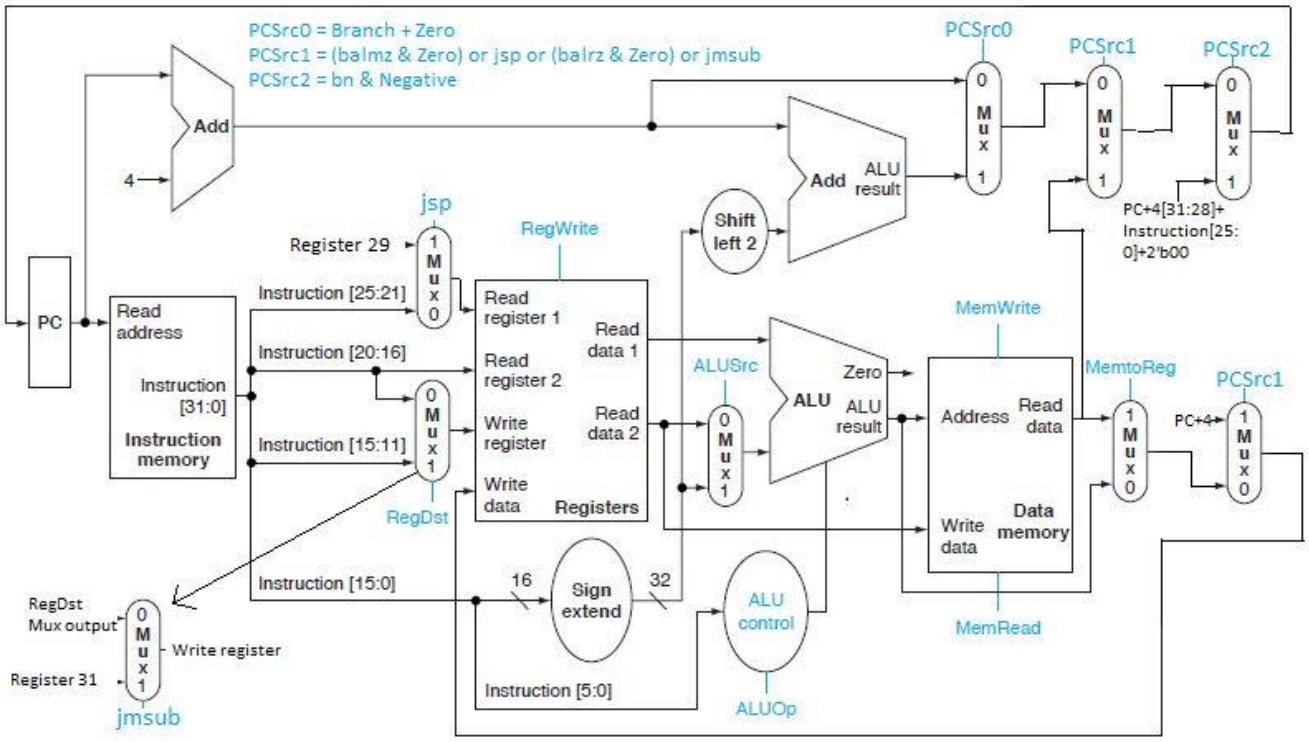
MARMARA UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

CSE3038 COMPUTER ORGANIZATION
PROJECT-2

Number	First Name	Last Name
150119566	Müslim	Yılmaz
150119037	Ömer	Kibar
150119036	Serkan	Korkut
150119804	Berkay	Ağar

GENERAL VIEW OF MODIFIED DATAPATH

Instructions: balmz, balrz, jmsub, bn, sllv, jsp



- **BALMZ**

INSTR	Type	Code	Meaning
balmz	I-type	opcode=23	if Status [Z] = 1, branches to address found in memory, link address is stored in $rt

- Verilog code:

Control.v: We set the control signals for balmz.

```
assign rformat=~|in;
assign lw=in[5]& (~in[4])&(~in[3])&(~in[2])&in[1]&in[0];
assign sw=in[5]& (~in[4])&in[3]&(~in[2])&in[1]&in[0];
assign beq=~in[5]& (~in[4])&(~in[3])&in[2]&(~in[1])&(~in[0]);
assign balmz=~in[5]& in[4]&(~in[3])&in[2]&in[1]&in[0];          //balmz => 23=010111
assign jsp=~in[5]& in[4]&(~in[3])&(~in[2])&in[1]&(~in[0]);             //jsp => 18=010010
assign bn=~in[5]& in[4]&in[3]&(~in[2])&(~in[1])&in[0];          //bn => 25=011001
assign regdest=rformat;
assign alusrc=lw|sw|balmz|jsp;
assign memtoreg=lw;
assign regwrite=rformat|lw|balmz;
assign memread=lw|balmz|jsp;
assign memwrite=sw;
assign branch=beq;
assign aluop1=rformat;
assign aluop2=beq;
```

Processor.v: We created 2 new mux to write data and change PC.

```
//AND gate
assign pcsrc0=branch && statusZNV[2];
assign pcsrc1=(balmz && statusZNV[2]) || jsp || (balrz && statusZNV[2]) || jmsub;
assign pcsrc2=bn && statusZNV[1];

//mux with (Balmz&ALUZero and jsp) control
mult2_to_1_32 mult5(out5, out4,dpack,pcsrc1);

//mux with Balmz, Balrz, Jmsub control
mult2_to_1_32 mult3_1(out3_1, out3,adderlout,pcsrc1);
```

- Example run:

add $2, $0, $0 = 00001020

balmz $2, imm4($1) = 5c220004

```
 0PC 00000004  SUM 00000000   INST 00001020   REGISTER 00000008 000000ff
20PC 00000004  SUM 00000000   INST 00001020   REGISTER 00000008 00000000
40PC 00000008  SUM 0000000c   INST 5c220004   REGISTER 00000008 00000000
60PC 00000008  SUM 0000000c   INST 5c220004   REGISTER 00000008 0000000c
80PC 00000004  SUM 00000000   INST 00001020   REGISTER 00000008 0000000c

Data Memory[12]= 00
Data Memory[13]= 00
Data Memory[14]= 00
Data Memory[15]= 04
```

- **BALRZ**

INSTR	Type	Code	Meaning
balrz	R-type	funct=22	if Status [Z] = 1, branches to address found in register, $rs link address is stored in $rd (which defaults to 31)

- Verilog code:

Alucont.v: We set the alu control signals for balrz.

```
if (f5&~(f4|f3|f2|f1|f0))gout=3'b010;   //function code=100000,ALU control=010
if (f5&~(f4)&f3&~(f2)&f1&~(f0))gout=3'b111;                    //function code
if (f5&~(f4)&~(f3)&~(f2)&f1&~(f0))gout=3'b110;          //function code=100010,
if (f5&~(f4)&~(f3)&f2&~(f1)&f0)gout=3'b001;             //function code=100101,
if (f5&~(f4)&f2&~(f0))gout=3'b000;               //function code=10x1x0,ALU cont
if (~(f5)&~(f4)&~(f3)&f2&~(f1)&~(f0))gout=3'b011;       //function code=000100,
if (~(f5)&f4&~(f3)&f2&f1&~(f0))
begin
        gout=3'b100;    //function code=010110,ALU control=100 (balrz)
        balrz = 1'b1;
end
if (f5&~(f4)&~(f3)&~(f2)&f1&f0)
begin
        gout=3'b110;    //function code=100011,ALU control=110 (sub)(jmsub)
        jmsub = 1'b1;   //bozuk
end
```

Processor.v: We used the muxes we created for balmz.

```
//AND gate
assign pcsrc0=branch && statusZNV[2];
assign pcsrc1=(balmz && statusZNV[2]) || jsp || (balrz && statusZNV[2]) || jmsub;
assign pcsrc2=bn && statusZNV[1];

//mux with (Balmz&ALUZero and jsp and balrz and jmsub) control
mult2_to_1_32 mult5(out5, out4,dpack,pcsrc1);

//mux with Balmz, Balrz, Jmsub control
mult2_to_1_32 mult3_1(out3_1, out3,adderlout,pcsrc1);
```

- Example run:

add $2, $0, $0 = 00001020

balrz $4, $1 = 00800816

```
 0PC 00000004  SUM 00000000   INST 00001020   REGISTER 00000008 000000ff 000000ff 0000000c
20PC 00000004  SUM 00000000   INST 00001020   REGISTER 00000008 00000000 000000ff 0000000c
40PC 00000008  SUM 0000000c   INST 00800816   REGISTER 00000008 00000000 000000ff 0000000c
60PC 00000008  SUM 0000000c   INST 00800816   REGISTER 0000000c 00000000 000000ff 0000000c
80PC 00000004  SUM 00000000   INST 00001020   REGISTER 0000000c 00000000 000000ff 0000000c

Data Memory[12]= 00
Data Memory[13]= 00
Data Memory[14]= 00
Data Memory[15]= 04
```

- **JMSUB**

INSTR	Type	Code	Meaning
jmsub	R-type	funct=35 (instead of 34)	Jumps to address found in memory [$rs-rt], link address is stored in $31

- Verilog code:

Alucont.v: We set the alu control signals for jmsub.

```
begin
        gout=3'b100;    //function code=010110,ALU control=100 (balrz)
        balrz = 1'b1;
end
if (f5&~(f4)&~(f3)&~(f2)&f1&f0)
begin
        gout=3'b110;    //function code=100011,ALU control=110 (sub)(jmsub)
        jmsub = 1'b1;   //bozuk
end
```

Processor.v: We created a new mux to be able to select register 31 and used to mux and for PC and write data we used the mux we created in balmz

```
//AND gate
assign pcsrc0=branch && statusZNV[2];
assign pcsrc1=(balmz && statusZNV[2]) || jsp || (balrz && statusZNV[2]) || jmsub;
assign pcsrc2=bn && statusZNV[1];

//mux with (Balmz&ALUZero and jsp and balrz and jmsub) control
mult2_to_1_32 mult5(out5, out4,dpack,pcsrc1);

//mux with Balmz, Balrz, Jmsub control
mult2_to_1_32 mult3_1(out3_1, out3,adderlout,pcsrc1);

//mux with jmsub control
mult2_to_1_5  mult1_1(out1_1, out1,5'b11111,jmsub);
```

- Example run:

Jmsub $2, $3

```
 0PC 00000004  SUM 00000000   INST 00632022   REGISTER 00000008 000000ff 000000ff xxxxxxxx    Data Memory[0]= 00
40PC 00000008  SUM 00000000   INST 00430023   REGISTER 00000008 000000ff 000000ff xxxxxxxx    Data Memory[1]= 00
60PC 00000008  SUM 00000000   INST 00430023   REGISTER 00000008 000000ff 000000ff 0000000c    Data Memory[2]= 00
80PC 00000001  SUM 00000000   INST 63202200   REGISTER 00000008 000000ff 000000ff 0000000c    Data Memory[3]= 01
```

- **BN**

INSTR	Type	Code	Meaning
bn	J-type	opcode=25	if Status [N] = 1, branches to pseudo-direct address (formed as j does)

- Verilog code:

Control.v: We set the control signals for bn.

```
assign rformat=~|in;
assign lw=in[5]& (~in[4])&(~in[3])&(~in[2])&in[1]&in[0];
assign sw=in[5]& (~in[4])&in[3]&(~in[2])&in[1]&in[0];
assign beq=~in[5]& (~in[4])&(~in[3])&in[2]&(~in[1])&(~in[0]);
assign balmz=~in[5]& in[4]&(~in[3])&in[2]&in[1]&in[0];          //balmz => 23=010111
assign jsp=~in[5]& in[4]&(~in[3])&(~in[2])&in[1]&(~in[0]);              //jsp => 18=010010
assign bn=~in[5]& in[4]&in[3]&(~in[2])&(~in[1])&in[0];          //bn => 25=011001
assign regdest=rformat;
assign alusrc=lw|sw|balmz|jsp;
assign memtoreg=lw;
assign regwrite=rformat|lw|balmz;
assign memread=lw|balmz|jsp;
assign memwrite=sw;
assign branch=beq;
assign aluop1=rformat;
assign aluop2=beq;
```

Processor.v: We created a new mux for pseudo-direct address

```
//AND gate
assign pcsrc0=branch && statusZNV[2];
assign pcsrc1=(balmz && statusZNV[2]) || jsp || (balrz && statusZNV[2]) || jmsub;
assign pcsrc2=bn && statusZNV[1];


//mux with bn control
assign   target = {adderlout[31:28], instruc[25:0], {2'b00}};
mult2_to_1_32 mult6(out6, out5,target,pcsrc2);
```

- Example run:

sub $3, $1, $2 = 00221822

bn imm(1) = 64000001

```
  0PC 00000004  SUM ffffff09   INST 00221822   REGISTER 00000008 000000ff 000000ff
 20PC 00000004  SUM ffffff09   INST 00221822   REGISTER 00000008 000000ff ffffff09
 40PC 00000008  SUM 00000000   INST 64000001   REGISTER 00000008 000000ff ffffff09
 80PC 00000004  SUM ffffff09   INST 00221822   REGISTER 00000008 000000ff ffffff09
```

- **SLLV**

INSTR	Type	Code	Meaning
sllv	R-type	func=4	shift register $rt to left by the value in register $rs, and store the result in register $rd.

- Verilog code:

Alucont.v:

```
begin
        if (f5&~(f4|f3|f2|f1|f0))gout=3'b010;   //function code=100000,ALU control=010 (add)
        if (f5&~(f4)&f3&~(f2)&f1&~(f0))gout=3'b111;                    //function code=101010,ALU control=111 (set on less than)
        if (f5&~(f4)&~(f3)&~(f2)&f1&~(f0))gout=3'b110;          //function code=100010,ALU control=110 (sub)
        if (f5&~(f4)&~(f3)&f2&~(f1)&f0)gout=3'b001;             //function code=100101,ALU control=001 (or)
        if (f5&~(f4)&f2&~(f0))gout=3'b000;              //function code=10x1x0,ALU control=000 (and)
        if (~(f5)&~(f4)&~(f3)&f2&~(f1)&~(f0))gout=3'b011;       //function code=000100,ALU control=011 (sllv)
        if (~(f5)&f4&~(f3)&f2&f1&~(f0))
        begin
                gout=3'b100;    //function code=010110,ALU control=100 (balrz)
                balrz = 1'b1;
        end
        if (f5&~(f4)&~(f3)&~(f2)&f1&f0)
        begin
                gout=3'b110;    //function code=100011,ALU control=110 (sub)(jmsub)
                jmsub = 1'b1;   //bozuk
        end
end
```

Alu32.v:

```
3'b000: sum=a & b;      //ALU control line=000, AND
3'b001: sum=a|b;                //ALU control line=001, OR
3'b011: sum=a<<b;               //ALU control line=011, SLLV
3'b100: sum=a;                  //ALU control line=100, balrz
default: sum=31'bx;
endcase
```

- Example run:

sllv $3, $2, $1 = 00411804

```
  0PC 00000004  SUM 0000ff00   INST 00411804   REGISTER 00000008 000000ff 000000ff
 20PC 00000004  SUM 0000ff00   INST 00411804   REGISTER 00000008 000000ff 0000ff00
 40PC 00000008  SUM 0000000c   INST 014b0023   REGISTER 00000008 000000ff 0000ff00
 80PC 00000004  SUM 0000ff00   INST 00411804   REGISTER 00000008 000000ff 0000ff00
```

- **JSP**

INSTR	Type	Code	Meaning
jsp	I-type	opcode=18	jumps to address found in memory where the memory address is written in register ($sp).

- Verilog code:

Control.v: We set the control signals for jsp.

```
assign rformat=~|in;
assign lw=in[5]& (~in[4])&(~in[3])&(~in[2])&in[1]&in[0];
assign sw=in[5]& (~in[4])&in[3]&(~in[2])&in[1]&in[0];
assign beq=~in[5]& (~in[4])&(~in[3])&in[2]&(~in[1])&(~in[0]);
assign balmz=~in[5]& in[4]&(~in[3])&in[2]&in[1]&in[0];          //balmz => 23=010111
assign jsp=~in[5]& in[4]&(~in[3])&(~in[2])&in[1]&(~in[0]);              //jsp => 18=010010
assign bn=~in[5]& in[4]&in[3]&(~in[2])&(~in[1])&in[0];          //bn => 25=011001
assign regdest=rformat;
assign alusrc=lw|sw|balmz|jsp;
assign memtoreg=lw;
assign regwrite=rformat|lw|balmz;
assign memread=lw|balmz|jsp;
assign memwrite=sw;
assign branch=beq;
assign aluop1=rformat;
assign aluop2=beq;
```

Processor.v: We created a new mux to be able to select register 29 and used to mux and for PC we used the mux we created in balmz

```
//AND gate
assign pcsrc0=branch && statusZNV[2];
assign pcsrc1=(balmz && statusZNV[2]) || jsp || (balrz && statusZNV[2]) || jmsub;
assign pcsrc2=bn && statusZNV[1];
```

```
// registers
mult2_to_1_32  multread1(dataa, registerfile[inst25_21],registerfile[29],jsp);
```

```
//mux with (Balmz&ALUZero and jsp) control
mult2_to_1_32 mult5(out5, out4,dpack,pcsrc1);
```

- Example run:

jsp = 48000000

```
  0PC 00000004  SUM 0000ff00   INST 00411804
 20PC 00000004  SUM 0000ff00   INST 00411804
 40PC 00000008  SUM 0000000c   INST 48000000
 80PC 00000004  SUM 0000ff00   INST 00411804

Register[28]= 00000000
Register[29]= 0000000c
Register[30]= 000000ff

Data Memory[12]= 00
Data Memory[13]= 00
Data Memory[14]= 00
Data Memory[15]= 04
```