

**lab report 7**

Andrew Nady

900184042

**Codes**

* All code is in a separate folder
* Risc-v module:

|  |
| --- |
| **`timescale** **1**ns / **1**ps  **module** Full\_dataPath( **input** clk,**input** fpga, **input** rst ,**input** [**1**:**0**] LedSel ,  **input** [**3**:**0**] ssdSel , **output** **reg** [**7**:**0**] leds, **output** [**6**:**0**]SSDout, **output**[**3**:**0**] anodes);  **wire** [**31**:**0**] adder1,adder2,PCmux,PCout;  **wire** cout1\_ripple1,cout1\_ripple2;  **wire** [**31**:**0**]ints\_out;  **wire** [**31**:**0**]readdata1, readdata2 ;  **wire** [**31**:**0**] gen\_out; //should be reg || wire  **wire** branch;  **wire** memread;  **wire** memtoreg;  **wire** [**1**:**0**]aluop;  **wire** memwrite;  **wire** alusrc;  **wire** regwrite;  **wire** [**3**:**0**]aluS;  **wire** [**31**:**0**]outmux\_input\_alu;  **wire** [**31**:**0**]ALU\_result;  **wire** zero;  **wire** [**31**:**0**] dataMem\_out;  **wire** [**31**:**0**] writingData;  **wire** [**31**:**0**] shiftout;  **reg** [**12**:**0**]num;  // wires of piplened regs  **wire** [**31**:**0**] **IF\_ID\_PC**, IF\_ID\_Inst; // input to REG1\_ID  //output of ID\_EX  **wire** memtoreg\_ID\_out;  **wire** regwrite\_ID\_out;  **wire** memread\_ID\_out;  **wire** memwrite\_ID\_out;  **wire** branch\_ID\_out;  **wire** [**1**:**0**]aluop\_ID\_out;  **wire** alusrc\_ID\_out;  **wire** [**31**:**0**]IF\_ID\_PC\_ID\_out;  **wire** [**31**:**0**]readdata1\_ID\_out;  **wire** [**31**:**0**]readdata2\_ID\_out;  **wire** [**31**:**0**]gen\_out\_ID\_out;  **wire** [**3**:**0**]ints\_out\_ID\_out\_up;  **wire** [**4**:**0**]ints\_out\_ID\_out\_down;  //out of EX/MEM  **wire** memtoreg\_EX\_out;  **wire** regwrite\_EX\_out;  **wire** memread\_EX\_out;  **wire** memwrite\_EX\_out;  **wire** branch\_EX\_out;  **wire** [**31**:**0**]adder\_branch\_Ex\_out;  **wire** zero\_EX\_out;  **wire** [**31**:**0**]ALU\_result\_EX\_out;  **wire** [**31**:**0**]readdata2\_EX\_out;  **wire** [**4**:**0**]ints\_out\_EX\_out\_down;  //out of MEM\_WB  **wire** memtoreg\_MEM\_out;  **wire** regwrite\_MEM\_out;  **wire** [**31**:**0**] dataMem\_out\_MEM\_out;  **wire** [**31**:**0**]ALU\_result\_MEM\_out;  **wire** [**4**:**0**]ints\_out\_MEM\_out\_down;  //instantiate 32bitreg PCin PCout  //module OneReg(input clk,input S,input [31:0] in,input rst, output [31:0] Q );  OneReg pc( clk,**1**, PCmux, rst, PCout );  ripple\_carry ripplecar0(PCout,  **32'd4**,  **0**,  adder1,  cout1\_ripple1  );  //adder branch (DONE)  ripple\_carry ripplecar1(IF\_ID\_PC\_ID\_out,  shiftout,  **0**,  adder2,  cout1\_ripple2  );  //mux  ThirtytwoMUX mux3(adder1,adder2,branch\_EX\_out & zero\_EX\_out ,PCmux);  //register (PCmux, PCin) //reload the PC  //instmem  InstMem instmem(PCout[**7**:**2**],ints\_out);  //---------------------------------------If/ID----------------------------------  sixtyFour\_reg #(**64**) **IF\_ID** (clk, **1'b1**,{PCout,ints\_out},rst,{**IF\_ID\_PC**,IF\_ID\_Inst});  //control unit (DONE)  Control\_unit controlunit(IF\_ID\_Inst[**6**:**2**], branch,  memread, memtoreg,  aluop, memwrite, alusrc, regwrite );  //alu\_control (DONE)  ALU\_Control aluControl(aluop\_ID\_out, ints\_out\_ID\_out\_up[**2**:**0**],ints\_out\_ID\_out\_up[**3**] ,aluS);  //registerfile DONE  RegisterFile regfile( clk, rst,  IF\_ID\_Inst[**19**:**15**], IF\_ID\_Inst[**24**:**20**], ints\_out\_MEM\_out\_down,  writingData,  regwrite\_MEM\_out,  readdata1, readdata2 );  //immGenerator (DONE)  immediate\_generator immgen( gen\_out, IF\_ID\_Inst );  //------------------------------- ID\_EX--------------------------------  sixtyFour\_reg #(**145**) **ID\_EX** (clk, **1'b1**,  {memtoreg,regwrite,memread,memwrite,  branch,aluop,alusrc,**IF\_ID\_PC**,readdata1, readdata2  ,gen\_out,{IF\_ID\_Inst[**30**],IF\_ID\_Inst[**14**:**12**]},IF\_ID\_Inst[**11**:**7**] },rst,    {memtoreg\_ID\_out,regwrite\_ID\_out, memread\_ID\_out, memwrite\_ID\_out, branch\_ID\_out,aluop\_ID\_out  ,alusrc\_ID\_out,IF\_ID\_PC\_ID\_out, readdata1\_ID\_out,readdata2\_ID\_out,  gen\_out\_ID\_out  ,ints\_out\_ID\_out\_up, ints\_out\_ID\_out\_down});  //shifting left (done)  shiftLeft sh(gen\_out\_ID\_out ,shiftout);  //mux (DONE)  ThirtytwoMUX mux (readdata2\_ID\_out,gen\_out\_ID\_out,alusrc\_ID\_out ,outmux\_input\_alu);  //ALU (Done)  **ALU** alu(  readdata1\_ID\_out, outmux\_input\_alu,  aluS,  ALU\_result, zero );    //----------------------EX/MEM----------------------------------------------------  sixtyFour\_reg #(**107**) **EX\_MEM** (clk, **1'b1**,  {memtoreg\_ID\_out,regwrite\_ID\_out, memread\_ID\_out, memwrite\_ID\_out, branch\_ID\_out,  adder2,zero,ALU\_result,readdata2\_ID\_out,ints\_out\_ID\_out\_down },rst,    { memtoreg\_EX\_out,  regwrite\_EX\_out,  memread\_EX\_out,  memwrite\_EX\_out,  branch\_EX\_out,  adder\_branch\_Ex\_out,  zero\_EX\_out,  ALU\_result\_EX\_out,  readdata2\_EX\_out,  ints\_out\_EX\_out\_down});    //dataMem  DataMem datamem( clk, memread\_EX\_out, memwrite\_EX\_out,  ALU\_result\_EX\_out[**7**:**2**] , readdata2\_EX\_out, dataMem\_out);  //----------------------MEM/WB----------------------------------------------------  sixtyFour\_reg #(**71**) **MEM\_WB** (clk, **1'b1**,  {regwrite\_EX\_out,memtoreg\_EX\_out,  dataMem\_out,ALU\_result\_EX\_out,ints\_out\_EX\_out\_down},rst,    {    regwrite\_MEM\_out,  memtoreg\_MEM\_out,  dataMem\_out\_MEM\_out,ALU\_result\_MEM\_out,  ints\_out\_MEM\_out\_down  });        //mux after the datamem  ThirtytwoMUX mux\_writing (ALU\_result\_MEM\_out,dataMem\_out\_MEM\_out,memtoreg\_MEM\_out ,writingData);  //switch or if on input ledsel to generate outputs leds  **always** @(\*)  **begin**  **case**(LedSel)  **2'b00** :  **begin**  leds[**0**]= regwrite;  leds[**1**]= alusrc;  leds[**3**:**2**]= aluop;  leds[**4**]= memread;  leds[**5**]= memwrite;  leds[**6**]= memtoreg;  leds[**7**]= branch;  **end**  **2'b01**:  **begin**  leds[**3**:**0**]=aluS;  leds [**4**]=zero;  leds[**5**]=branch&zero;  leds[**7**:**6**] =**0**;    **end**  **2'b11**:  leds[**7**:**0**]= {**1'b0**,ints\_out[**6**:**0**]}; // monotring the opcode  ////switch or if on input ssdsel to generate ssdnumber  **default** : leds[**7**:**0**]= **0**;  **endcase**  **end**  **always**@(\*) **begin**  **case** (ssdSel)  **4'b0000**:num = PCout;  **4'b0001**: num =PCout+**4**;  **4'b0010**: num = adder2;  **4'b0011**: num = PCmux;  **4'b0100**: num = readdata1;  **4'b0101**: num= readdata2;  **4'b0110**: num= writingData;  **4'b0111**: num= gen\_out;  **4'b1000**: num= shiftout;  **4'b1001**:num=outmux\_input\_alu;  **4'b1010**: num= ALU\_result;  **4'b1011**:num=dataMem\_out;  **default** : num = **0**;  **endcase**  **end**  //instantiate 7 seg  seven ssd ( fpga, num,anodes,SSDout);  **endmodule** |

**Simulation screenshots**

**Here is the data out from memory**



**Here is the writing data**



**Here is the writing data and the reg write signal**



**Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Pc Output** | **Pc input  (branch adder or PC+4)** | **Writing data (Reg file)** | **Writing data (data mem)** |
| **lw x1, 0(x0)** | **4** | **8** | **17** | **17** |
| **lw x2, 4(x0)** | **20** | **24** | **9** | **9** |
| **lw x3, 8(x0)** | **36** | **40** | **25** | **25** |
| **or x4, x1, x2** | **52** | **56** | **25** | **0** |
| **beq x4, x3, 16** | **68** | **16\*2+68=100** | **0** | **0** |
| **add x3, x1, x2** | **84** | **20** |  |  |
| **add x5, x3, x2** | **100** | **104** | **25+9=34** | **0** |
| **sw x5, 12(x0)** | **116** | **120** | **0** | **0** |
| **lw x6, 12(x0)** | **132** | **136** | **34** | **34** |
| **and x7, x6, x1** | **150** | **154** | **34&17=0** | **0** |
| **sub x8, x1, x2** | **164** | **170** | **17-9=8** | **0** |
| **add x0, x1, x2** | **180** | **184** | **0** | **0** |
| **add x9, x0, x1** | **196** | **200** | **17** | **0** |

**As x4, x3 are equal so the PC will jump**

**minimize the number of NOP**

* **as know that and, add, … needs 2 bubbles (2 Nop) -> data hazard**
* **load word need 3 bubbles (3 NOP) ->data hazard**
* **beq need 3 bubbles (3 nop)-> control hazard**

**add x0, x0, x0**

**lw x1, 0(x0)**

**lw x2, 4(x0)**

**lw x3, 8(x0)**

NOP ->for x1(data hazard)

NOP -> for x2(data hazard)

**or x4, x1, x2**

NOP -> (data hazard)

NOP -> (data hazard)

**beq x4, x3, 16**

NOP -> (control hazard)

NOP -> (control hazard)

NOP -> (control hazard)

**add x3, x1, x2**

NOP -> (data hazard)

NOP -> (data hazard)

**add x5, x3, x2**

NOP -> (data hazard)

NOP -> (data hazard)

**sw x5, 12(x0)**

**lw x6, 12(x0)**

NOP -> (data hazard)

NOP -> (data hazard)

NOP -> (data hazard)

**and x7, x6, x1**

**sub x8, x1, x2**

**add x0, x1, x2**

**add x9, x0, x1**

**add x0, x0, x0**

**lw x1, 0(x0)**

**lw x2, 4(x0)**

**lw x3, 8(x0)**

NOP -> (data hazard)

NOP -> (data hazard)

**or x4, x1, x2**

NOP -> (data hazard)

NOP -> (data hazard)

**beq x4, x3, 16**

NOP -> (control hazard)

NOP -> (control hazard)

NOP -> (control hazard)

**add x3, x1, x2**

NOP -> (data hazard)

NOP -> (data hazard)

**add x5, x3, x2**

NOP -> (data hazard)

NOP -> (data hazard)

**sw x5, 12(x0)**

**lw x6, 12(x0)**

**sub x8, x1, x2**

**add x0, x1, x2**

**add x9, x0, x1**

**and x7, x6, x1**

Modifying the program

**My program**

lw t4, 0(zero)

lw t5, 4(zero)

lw t6, 8(zero)

NOP

NOP

NOP

loop:

beq t4,t6,exit

NOP

NOP

NOP

add t4,t4,t5 1

beq zero,zero,loop

NOP

NOP

NOP

exit:

sw t4, 12(zero)

Modifying the program

lw t4, 0(zero)

lw t5, 4(zero)

lw t6, 8(zero)

loop:

beq t4,t6,exit

add t4,t4,t5

beq zero,zero,loop

exit:

sw t4, 12(zero)