

ECE 424
DUE: Wednesday, Nov 18, 12:30 pm

Design Project #3: Single-Cycle Datapath Design (100 points)

COVER SHEET

Honor Code: _____ I have neither given or received, nor have I tolerated others' _____
_____ use of unauthorized aid. _____

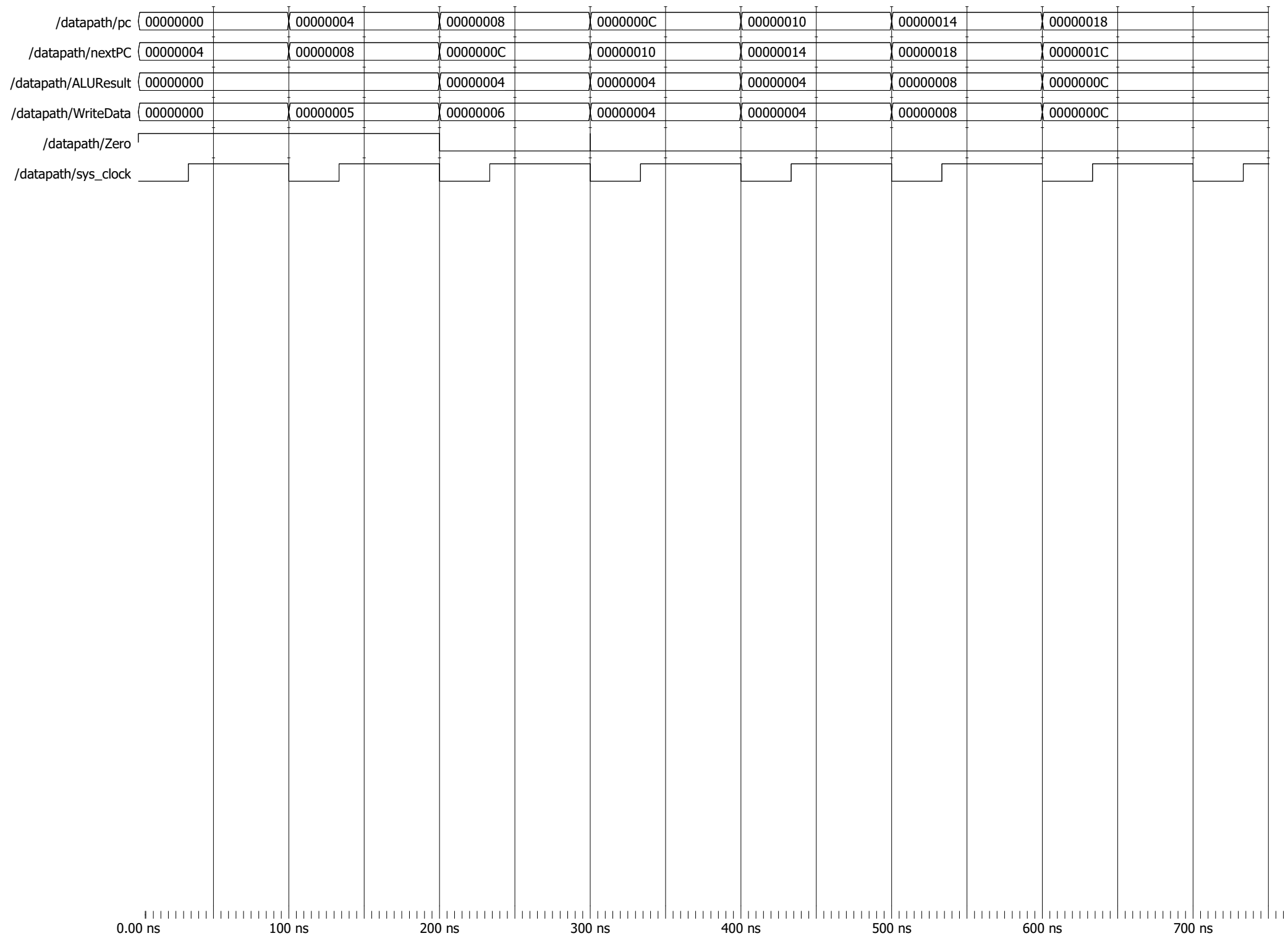
Name: _____ Joe Leveille _____ Signature: _____ *Joseph Leveille* _____

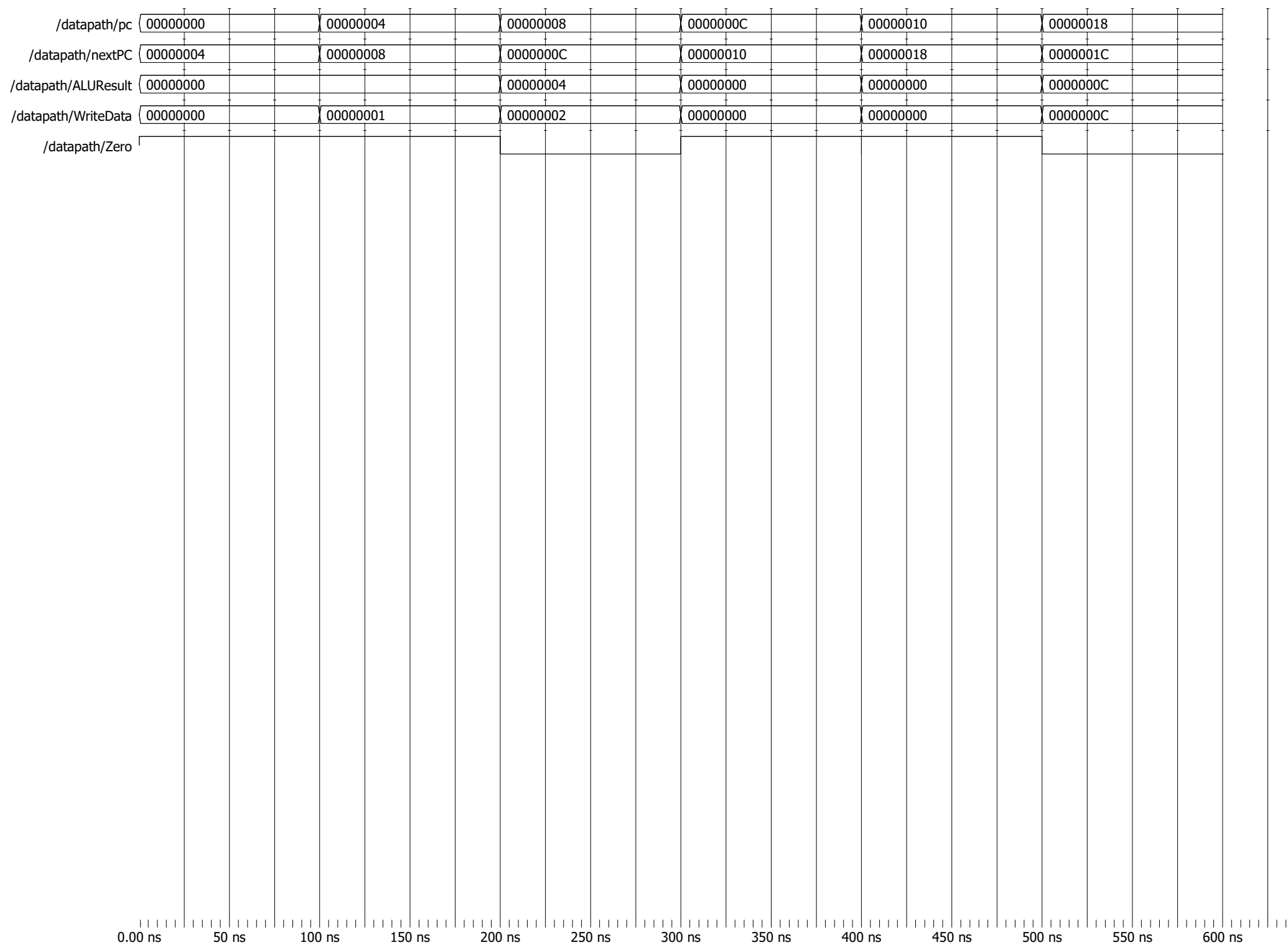
Honor Code: _____ I have neither given or received, nor have I tolerated _____
_____ others' use of unauthorized aid. _____

Name: _____ Jon Bayert _____ Signature: _____ *Jonathon Bayert* _____

A complete assignment will contain:

1. This cover sheet
2. A printout of your VHDL code including your main file, modified imem.vhd, modified alu-control.vhd, and modified dmem.vhd.
3. Waveforms for the two different cases, showing the five (and only the five) signals in hex format.





```

1  -- Joe Leveille & Jon Bayert
2  -- Final VHDL Assignment - ECE 424 - Computer Architecture
3  --
4
5  library ieee;
6  use ieee.std_logic_1164.all;
7  use ieee.std_logic_arith.all;
8  use ieee.std_logic_unsigned.all;
9  use work.mips_types.all;
10
11  entity datapath is
12      PORT ( pc : IN STD_LOGIC_VECTOR ( 31 downto 0 );
13            nextPC: OUT STD_LOGIC_VECTOR (31 downto 0) );
14
15  END datapath;
16
17  architecture dataBoi of datapath is
18      COMPONENT alu_control
19          port( INSTR: in std_logic_vector(5 DOWNT0 0);
20                ALUOP: in std_logic_vector(1 DOWNT0 0);
21                ALUCTRL : OUT std_logic_vector(3 DOWNT0 0));
22      END COMPONENT;
23
24      COMPONENT control
25          port( INSTR: in std_logic_vector(5 DOWNT0 0);
26                REGDST: OUT std_logic;
27                BRANCH: OUT std_logic;
28                MEMREAD: OUT std_logic;
29                MEMTOREG: OUT std_logic;
30                ALUOP: OUT std_logic_vector(1 DOWNT0 0);
31                MEMWRITE: OUT std_logic;
32                ALUSRC: OUT std_logic;
33                REGWRITE: OUT std_logic);
34      END COMPONENT;
35
36      COMPONENT dmemory
37          port(
38              DIN: in mips_data;
39              ADDR: in mips_address;
40              DOUT: out mips_data;
41              WE, RE: in std_logic;
42              CLK: in std_logic
43          );
44      END COMPONENT;
45
46      COMPONENT imemory
47          port(
48              ADDR: in mips_address;
49              DOUT: out mips_data
50          );
51      END COMPONENT;
52
53      COMPONENT alu32
54          PORT( A, B: in std_logic_vector (31 downto 0);
55                ALUOp: in std_logic_vector (3 downto 0);
56                RESULT: out std_logic_vector (31 downto 0);
57                Z, V, C: out std_logic );
58      END COMPONENT;
59
60      COMPONENT mux2a
61          port( IN0: in mips_reg_addr;
62                IN1: in mips_reg_addr;
63                SEL: in STD_LOGIC;
64                DOUT: out mips_reg_addr);
65      end component;
66
67      COMPONENT reg_file
68          port (a1 : in mips_reg_addr;
69                q1 : out mips_data;

```

```

70         a2 : in mips_reg_addr;
71         q2 : out mips_data;
72         a3 : in mips_reg_addr;
73         d3 : in mips_data;
74         write_en : in std_logic;
75         clk : in std_logic);
76 end COMPONENT;
77
78 COMPONENT signextend
79     port( INDATA: in STD_LOGIC_VECTOR(15 DOWNTO 0);
80           OUTDATA: out mips_data);
81 end COMPONENT;
82
83 COMPONENT mux2
84     port( IN0: in mips_data;
85           IN1: in mips_data;
86           SEL: in std_logic;
87           DOUT: out mips_data);
88 end COMPONENT;
89
90 COMPONENT shiftleft
91     port( INDATA: in mips_data;
92           OUTDATA: out mips_data);
93 end COMPONENT;
94
95 SIGNAL sys_clock : std_logic := '0';
96 CONSTANT Tcycle : time := 100 ns;
97
98 SIGNAL inst, ALUResult, outMuxIn1, branchALUin1, wDataMuxin1 : std_logic_vector( 31
99     downto 0);
100 SIGNAL defNextPC, WriteData, aluIn0, aluIn1, mux2in0, mux2in1 : std_logic_vector(
101     31 downto 0);
102 SIGNAL REGDST, BRANCH, MEMREAD, MEMTOREG, MEMWRITE, ALUSRC, REGWRITE, Zero,
103     outMuxSel : std_logic;
104 SIGNAL ALUOpDatapath : std_logic_vector(1 downto 0);
105 SIGNAL wRegIn : std_logic_vector(4 downto 0);
106 SIGNAL mainALUctl : std_logic_vector(3 downto 0);
107
108 BEGIN
109
110     -- create clock process
111     clk_gen: process
112     begin
113         sys_clock <= '1' after Tcycle/3, '0' after Tcycle;
114         wait until sys_clock = '0';
115     end process clk_gen;
116
117     Imem: imemory PORT MAP(
118         ADDR => pc, DOUT => inst
119     );
120
121     PCplus4: alu32 PORT MAP(
122         A => pc, B=> X"00000004", ALUOp => "0010", RESULT=>defNextPC, Z => oPeN, V=>
123         OPeN, C=>oPeN
124     );
125
126     CTL: control PORT MAP(
127         INSTR => inst(31 downto 26), REGDST => RegDst, BRANCH => Branch,
128         MEMREAD => MEMREAD, MEMTOREG => MEMTOREG, MEMWRITE => MEMWRITE, ALUSRC => ALUSRC,
129         REGWRITE => REGWRITE, ALUOP => ALUOpDatapath
130     );
131
132     writeRegSel: mux2a PORT MAP(
133         IN0 => inst(20 downto 16), IN1 => inst(15 downto 11), SEL => REGDST, DOUT =>
134         wRegIn
135     );
136
137     reg: reg_file PORT MAP(
138         al=> inst(25 downto 21), a2 => inst(20 downto 16), a3 => wRegIn,

```

```

134         q1 => aluIn0, q2 => mux2in0, d3=> WriteData, write_en=> REGWRITE, CLK =>
        sys_clock
135     );
136
137     sign: signextend PORT MAP(
138         INDATA => inst(15 downto 0), OUTDATA => mux2in1
139     );
140
141     muxTheSecond: mux2 PORT MAP(
142         IN0=>mux2in0, IN1=> mux2in1, SEL=>ALUSRC, DOUT => aluIn1
143     );
144
145     ALUctl: alu_control PORT MAP(
146         INSTR=> inst(5 downto 0), ALUOp => ALUOpDatapath, ALUctrl => mainALUctl
147     );
148
149     mainALU: alu32 PORT MAP(
150         a=>aluIn0, b=>aluIn1, ALUOp=> mainALUctl, RESULT => ALUResult, Z=>Zero,
        V=>open, C=>oPEN
151     );
152
153     shift: shiftleft PORT MAP(
154         INDATA=>mux2in1, OUTDATA=>branchALUin1
155     );
156
157     branchALU: alu32 PORT MAP(
158         a=>defNextPC, b=>branchALUin1, ALUOP=>"0010", RESULT=>outMuxIn1, Z=>open,
        C=>opEn, V=>OpEN
159     );
160
161     outMuxSel <= branch and Zero;
162
163     outMux: mux2 PORT MAP(
164         IN0=>defNextPC, IN1=>outMuxIn1, SEL=> outMuxSel, DOUT => nextPC
165     );
166
167     dmem: dmemory PORT MAP(
168         ADDR => ALUResult, DIN => mux2in0, DOUT => wDataMuxin1, CLK => sys_clock, WE =>
        memWrite, RE =>memRead
169     );
170
171     wDataMux: mux2 PORT MAP(
172         IN0 => ALUResult, IN1=>wDataMuxin1, sel=>memToReg, DOUT=>WriteData
173     );
174
175 end dataBoi;
176

```

```

1  --
2  -- Copyright Jay Brockman
3  --
4  -- MIPS Processor Developement
5  -- Eric W. Johnson
6  -- Valparaiso University
7  --
8  --
9  -- Updated Jeffrey Will
10
11 library ieee;
12 use ieee.std_logic_1164.all;
13 use ieee.std_logic_arith.all;
14 use ieee.std_logic_unsigned.all;
15 use work.mips_types.all;
16
17
18 -- memory Entity Description
19 entity imemory is
20     port(
21         ADDR: in mips_address;
22         DOUT: out mips_data
23     );
24 end imemory;
25
26 -- memory Architecture Description
27 architecture rtl of imemory is
28     subtype ramword is bit_vector(31 downto 0);
29     type rammemory is array (0 to 1024) of ramword;
30
31     --*****
32     -- Students: You will have to hand-assemble the instructions
33     -- Given in the assignment, and fill in the values
34     -- below. (It is like you are "flashing" the instruction
35     -- memory with your assembly code
36     --*****
37
38     signal ram : rammemory := (
39         x"00000820", -- 00: add $1,$0,$0
40         x"8C220000", -- 01: lw $2,0($1)
41         x"8C230004", -- 02:
42         x"00432024", -- 03:
43         x"10800001", -- 04:
44         x"AC240008", -- 05:
45         x"AC24000C", -- 06:
46         others => x"00000000");
47 begin
48
49     read_Process: process(ram, ADDR)
50         variable raddr1 : integer range 0 to 1024;
51         variable tempdata : ramword;
52     begin
53         -- convert address to integer
54         raddr1 := conv_Integer(ADDR);
55         raddr1 := raddr1/4;
56         tempdata := (ram(raddr1));
57         DOUT <= to_stdlogicvector(tempdata);
58     end process read_Process;
59
60 end rtl;
61

```

```

1  --
2  -- MIPS Processor Developement
3  -- Eric W. Johnson
4  -- Valparaiso University
5  --
6  --
7  -- Modified by Jeffrey D. Will
8
9
10 library ieee;
11 use ieee.std_logic_1164.all;
12 use ieee.std_logic_arith.all;
13 use ieee.std_logic_unsigned.all;
14 use work.mips_types.all;
15
16 entity control is
17     port( INSTR: in std_logic_vector(5 DOWNTO 0);
18           REGDST: OUT std_logic;
19           BRANCH: OUT std_logic;
20           MEMREAD: OUT std_logic;
21           MEMTOREG: OUT std_logic;
22           ALUOP: OUT std_logic_vector(1 DOWNTO 0);
23           MEMWRITE: OUT std_logic;
24           ALUSRC: OUT std_logic;
25           REGWRITE: OUT std_logic);
26 end control;
27
28 architecture behavioral of control is
29 begin
30     PROCESS( INSTR )
31         variable temp : std_logic_vector ( 7 downto 0 );
32         variable instr_int : integer range 0 to 64;
33     BEGIN
34         temp := "00000000";
35         temp := temp + instr;
36         instr_int := conv_Integer(temp);
37         case instr_int is
38
39             -- *****
40             -- Students: YOU MUST FILL IN THESE CONTROL LINES
41             -- FOR EACH INDIVIDUAL INSTRUCTION... they are
42             -- set to all zeroes below, but the zeroes are
43             -- just placeholders...
44             -- *****
45         when 0 => REGDST <= '1';
46                     BRANCH <= '0';
47                     MEMREAD <= '0';
48                     MEMTOREG <= '0';
49                     ALUOP <= "10";           -- ALUOP ORDER OF BITS????
50                     MEMWRITE <= '0';
51                     ALUSRC <= '0';
52                     REGWRITE <= '1';
53         when 35 => REGDST <= '0';
54                     BRANCH <= '0';
55                     MEMREAD <= '1';
56                     MEMTOREG <= '1';
57                     ALUOP <= "00";
58                     MEMWRITE <= '0';
59                     ALUSRC <= '1';
60                     REGWRITE <= '1';
61         when 43 => REGDST <= '0';
62                     BRANCH <= '0';
63                     MEMREAD <= '0';
64                     MEMTOREG <= '0';
65                     ALUOP <= "00";
66                     MEMWRITE <= '1';
67                     ALUSRC <= '1';
68                     REGWRITE <= '0';
69         when 4 => REGDST <= '0';

```



```
70          BRANCH <= '1';
71          MEMREAD <= '0';
72          MEMTOREG <= '0';
73          ALUOP <= "01";
74          MEMWRITE <= '0';
75          ALUSRC <= '0';
76          REGWRITE <= '0';
77      when others => REGDST <= '0';
78          BRANCH <= '0';
79          MEMREAD <= '0';
80          MEMTOREG <= '0';
81          ALUOP <= "00";
82          MEMWRITE <= '0';
83          ALUSRC <= '0';
84          REGWRITE <= '0';
85
86      END CASE;
87
88      END PROCESS;
89
90  end behavioral;
91
```

```

1  --
2  -- MIPS Processor Developement
3  -- Eric W. Johnson
4  -- Valparaiso University
5  --
6  --
7  --
8  -- Modified by Jeffrey Will
9
10 library ieee;
11 use ieee.std_logic_1164.all;
12 use ieee.std_logic_arith.all;
13 use ieee.std_logic_unsigned.all;
14
15 entity alu_control is
16     port( INSTR: in std_logic_vector(5 DOWNTO 0);
17           ALUOP: in std_logic_vector(1 DOWNTO 0);
18           ALUCTRL : OUT std_logic_vector(3 DOWNTO 0));
19 end alu_control;
20
21 -----
22 --Students: it is your job to fill in
23 -- each of the "---" with the appropriate 1's and 0's
24 -----
25 architecture behavioral of alu_control is
26 begin
27     PROCESS(INSTR,ALUOP)
28     BEGIN
29         case ALUOP is
30             when "00" => ALUCTRL <= "0010";
31             when "01" => ALUCTRL <= "0110";
32             when "10" => case INSTR is
33                 when "100000" => ALUCTRL <= "0010";
34                 when "100010" => ALUCTRL <= "0110";
35                 when "100100" => ALUCTRL <= "0000";
36                 when "100101" => ALUCTRL <= "0001";
37                 when "101010" => ALUCTRL <= "0111";
38                 when others => ALUCTRL <= "0000";
39             end case;
40             when others => ALUCTRL <= "0000";
41
42         END CASE;
43
44     END PROCESS;
45
46 end behavioral;
47

```

```

1  --
2  -- MIPS Processor Developement
3  -- Eric W. Johnson
4  -- Valparaiso University
5  --
6  --
7
8  library ieee;
9  use ieee.std_logic_1164.all;
10 use ieee.std_logic_arith.all;
11 use ieee.std_logic_unsigned.all;
12 use work.mips_types.all;
13
14 --
15 -- Copyright Jay Brockman, Feb 1997
16 -- Updated Eric W. Johnson, Feb 1998
17 --
18 -- Modified June 2011 Jeffrey Will
19
20 -- memory Entity Description
21 entity dmemory is
22     port(
23         DIN: in mips_data;
24         ADDR: in mips_address;
25         DOUT: out mips_data;
26         WE, RE: in std_logic;
27         CLK: in std_logic
28     );
29 end dmemory;
30
31
32 -- memory Architecture Description
33 architecture rtl of dmemory is
34     subtype ramword is bit_vector(31 DOWNTO 0);
35     type rammemory is array (0 to 4096) of ramword;
36
37     -----
38     -- Students: This is where you modify the contents of
39     -- Data memory.
40     -----
41     signal ram : rammemory := (
42         x"00000001", -- 00 through 03
43         x"00000002", -- 04 through 07
44         x"00000003", --
45         x"00000004", --
46         x"00000000", --
47         others => x"00000000");
48 begin
49
50     read_Process: process(RE, ADDR)
51         variable raddr1 : integer range 0 to 4096;
52         variable tempdata : ramword;
53     begin
54         -- convert address to integer
55         IF ( RE = '1' ) THEN
56             raddr1 := conv_Integer(ADDR);
57             raddr1 := raddr1/4;
58             tempdata := (ram(raddr1));
59             DOUT <= to_stdlogicvector(tempdata);
60         END IF;
61     end process read_Process;
62
63     write_Process: process(WE, CLK)
64         variable waddr : integer range 0 to 4096;
65     begin
66         if ( WE = '1' AND CLK'EVENT AND CLK = '1' ) then
67             -- convert address to integer
68             waddr := conv_Integer(ADDR);
69             waddr := waddr/4;

```

```
70     ram(waddr) <= to_bitvector(DIN);  
71     end if;  
72 end process write_Process;  
73 end rtl;  
74
```