EE 443

Computer Engineering Analysis and Design Laboratory

*MIPS Part 3: Designing a 16-bit RISC style microprocessor*

Lab *6*: Instruction and Data memory; basic components and datapath for I- and J- format instructions

*Due: Apr 4, 2023*

**Prepared by:**

*Lisa Jacklin*

Instructor/TA comments and grading

**1.** **Objective and Background**

The purpose of this lab is to continue to gain a better working understanding of how a sixteen bit risc style microprocessor operates. In this labs case, through the coding of datapaths that can accept instruction and jump format.

**2. Equipment**

* *Altera Quartus II*

**3. Procedure**

To begin the lab, a new project is created making sure that the correct chip is connected so we can later upload these programs to the experimenter boards and check that all operations are working as expected. Next, several different verilog components are coded and tested separately before setting up the final complex memory component.

The very first component that was created was *SHLONE.vhd* which was to shift the entire sixteen bit number to the left by one. This was done by concatenating the values fifteen down to one with zero, which would be the new low value to the shifted value. The next component created was *ADD16.vhd,* which used the four bit adder created in one of the last labs, to add together two sixteen bit values. With the creation of both the *SHLONE.vhd* and *ADD16.vhd,* the setup for the branch instructions is complete and the next set of components can be created.

The next set of components are created to work with instruction fetch and PC incrementation. The first of these components was *INCTWO.vhd* which added two to a sixteen bit value.This was done using the ADD16 created prior to easily add two to the original sixteen bit value. Note that since our ALU takes sixteen bit values rather than thirtytwo, an incrementation of two is used for the PC counter rather than an increment of four. *DCD5x32.vhd* was also created to take in the five bits of address, and turn them into 32 bits of data that can be stored in instruction memory which will be programmed next. For this to be implemented, several DCD3x8 decoders were used from a previous lab with port map to create the correct 32 output.

The last component that was set up to work with instruction fetch was *INS\_MEM.vhd*. As the name implies, the instruction memory will take the five bits from the address that displays what operations are being requested, then after going through the thirtytwo bit decoder, points to the value of the word that contains the operation. Note that within the instruction memory component, the first eight bits are held in one word, while the last eight bits are held in the last word, separating them into the most significant and least significant bytes. Once this component was created, chip resources, timing delays and waveforms were used to test and check that the component was working as expected.

With the instruction memory components created, the memory access components and related are created. The first of these components being *SGNEXT6x16.vhd,* which extends a six bit number to a sixteen bit number. In a similar fashion to how the SHLONE component was created, the six bit value was concatenated by the following ten zeros to extend the value. The next component was the *DCD4x16.vhd,* using the combination of two 3x8 decoders.

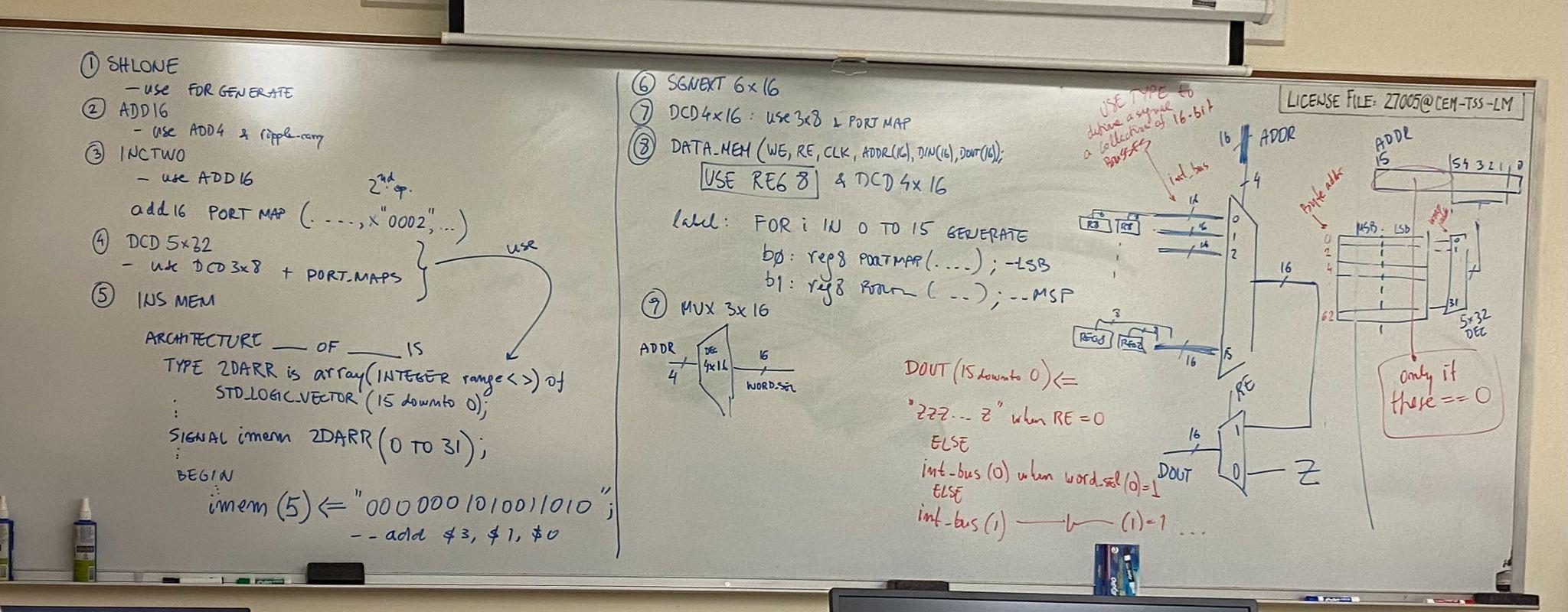
The final component for memory access was the *DAT\_MEM.vhd*. This component took two sixteen bit inputs, three single bit inputs for write, read, and clock, and a single sixteen bit output. Using REG8 from a previous lab as well as the DCD4x16 component created previously, the data memory component was created using port maps. Just as the instruction memory was tested, the data memory component was tested for resources, timing delay and waveforms for correct operations.

The very last component that was created in this lab was *MUX3x16.vhd,* which was created for jump instructions which will be implemented more in the next laboratory.

**6. Conclusion**

This lab helped me understand how each component of hardware works together to create a small section of an arithmetic logic unit within a central processing unit. When considering how each component we created operates, the diagram displayed in *Attachment 1,* helps me wrap my head around how different address labels can be decoded into a memory label, and from there, can allow the correct output to be displayed if there is any output to be had.

**7. Attachments**

**

*Attachment 1: Board notes from in Lab including diagrams and order of the project items being created.*

| *Instruction Binary*  *ADD R1, R2 0001000000100010*  *SUB R1, R2 0001000100100010*  *AND R1, R2 0001001000100010*  *OR R1, R2 0001001100100010*  *MOV R1, R2 0001010000100010*  *JMP 010000 0010000001000000*  *JZ 010000 0010010001000000*  *JNZ 010000 0010100001000000*  *JGZ 010000 0010110001000000*  *JLZ 010000 0011000001000000*  *LD R1, 0100 0100000100000100*  *ST R1, 0100 0100010100000100*  *CALL 010000 0100100001000000*  *RET 0100110000000000*  *HALT 0101000000000000* |
| --- |

*Attachment 2: A table of all the binary code (machine language) that was created for this lab and will be used in the next set of labs.*

| *--Lisa Jacklin*  *--EE 443 LAb 6*  *--INCTWO.vhd*  *-----------------------------------------------*  *LIBRARY IEEE;*  *USE IEEE.STD\_LOGIC\_1164.ALL;*  *ENTITY INCTWO IS*  *PORT (INCIN :IN STD\_LOGIC\_VECTOR(15 DOWNTO 0);*  *INCOUT : OUT STD\_LOGIC\_VECTOR (15 DOWNTO 0));*  *END INCTWO;*  *ARCHITECTURE INCTWO\_BEHAVIOR OF INCTWO IS*  *COMPONENT ADD16 IS --IS THIS ADDITION TRULY NEEDED WHEN THE ADDER CAN ONLY OUTPUT 4 BITS?*  *PORT( INA, INB :IN STD\_LOGIC\_VECTOR(15 DOWNTO 0);*  *CAIN :IN STD\_LOGIC;*  *ADDOUT :OUT STD\_LOGIC\_VECTOR(15 DOWNTO 0);*  *CAOUT :OUT STD\_LOGIC);*  *END COMPONENT;*    *SIGNAL CAO,CIN :STD\_LOGIC;*    *BEGIN*    *A1: ADD16 PORT MAP (INCIN(15 DOWNTO 0), X"0002", CIN,INCOUT(15 DOWNTO 0), CAO);*      *END INCTWO\_BEHAVIOR;* |
| --- |

*Attachment 3:INCTWO.vhd*

| *--Lisa Jacklin*  *-- EE 443 Lab 6*  *--DCD5x32.vhd*  *----------------------------------------------*  *LIBRARY IEEE;*  *USE IEEE.STD\_LOGIC\_1164.ALL;*  *ENTITY DCD5x32 IS*  *PORT (DCDIN :IN STD\_LOGIC\_VECTOR(4 DOWNTO 0);*  *DCDOUT :OUT STD\_LOGIC\_VECTOR(31 DOWNTO 0));*  *END DCD5X32;*  *ARCHITECTURE DCD5X32\_BEHAVIOR OF DCD5X32 IS*  *COMPONENT DCD3x8 IS*  *PORT (S :IN STD\_LOGIC\_VECTOR(2 DOWNTO 0);*  *Q :OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0));*  *END COMPONENT;*  *--TO CONNECT THE VALUES FOR THE MINI DECODERS*  *--NOTE THAT THE VECTOR IS LARGE BECAUSE THE OUTPUT IS 5 LARGER THAN THE INPUT*  *SIGNAL INTERN : STD\_LOGIC\_VECTOR(23 DOWNTO 0);*    *BEGIN*  *--USING SEVERAL PORT MAPS, EACH CHUNK OF THE LARGE ONE WILL BE SEPARATED INTO MANAGEABLE PIECES*    *D1: DCD3X8 PORT MAP (DCDIN (2 DOWNTO 0), INTERN(7 DOWNTO 0));*  *D2: DCD3X8 PORT MAP (DCDIN (4 DOWNTO 2), INTERN(15 DOWNTO 8));*  *D3: DCD3X8 PORT MAP (DCDIN (3 DOWNTO 1), INTERN(24 DOWNTO 16));*    *--NOW, TO SETUP THE ACTUAL OUTPUT FACTORS TO BE OUTPUTS FOR THE 5X32 DECODER*  *DCDOUT (31 DOWNTO 24) <= INTERN(24 DOWNTO 16);*  *DCDOUT (23 DOWNTO 16) <= INTERN(15 DOWNTO 8);*  *DCDOUT (15 DOWNTO 8) <= INTERN(7 DOWNTO 0);*  *DCDOUT (7 DOWNTO 0) <= "00000001" WHEN DCDIN = "10000" ELSE "00000000";*    *END DCD5X32\_BEHAVIOR;* |
| --- |

*Attachment 4:DCD5x32.vhd*

| *--Lisa Jacklin*  *-- EE 443 LAb 6*  *-- 3/28/2023*  *--INS\_MEM.vhd*  *---------------------------------------------------------*  *LIBRARY IEEE;*  *USE IEEE.STD\_LOGIC\_1164.ALL;*  *ENTITY INS\_MEM IS*  *PORT(*  *ADDR : IN STD\_LOGIC\_VECTOR(15 DOWNTO 0);*  *DOUT : OUT STD\_LOGIC\_VECTOR(15 DOWNTO 0)*  *);*  *END INS\_MEM;*  *ARCHITECTURE INS\_MEM\_BEHAVIOR OF INS\_MEM IS*  *TYPE TWODARR IS ARRAY(INTEGER RANGE <>) OF STD\_LOGIC\_VECTOR(15 DOWNTO 0);*    *COMPONENT DCD5x32 IS*  *PORT(*  *DCDIN : IN STD\_LOGIC\_VECTOR(4 DOWNTO 0);*  *DCDOUT : OUT STD\_LOGIC\_VECTOR(31 DOWNTO 0)*  *);*  *END COMPONENT;*    *COMPONENT DCD3x8 IS*  *PORT(*  *S : IN STD\_LOGIC\_VECTOR(2 DOWNTO 0);*  *Q : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0)*  *);*  *END COMPONENT;*    *SIGNAL IMEM : TWODARR(0 TO 31);*  *SIGNAL RE : STD\_LOGIC;*    *BEGIN*  *PROCESS(ADDR)*  *BEGIN*  *IF ADDR(15 DOWNTO 6) = "00000000" THEN*  *RE <= '1';*  *ELSE*  *RE <= '0';*  *END IF;*  *END PROCESS;*    *D1: DCD5X32 PORT MAP(ADDR(15 DOWNTO 1), IMEM(31 DOWNTO 0));*    *PROCESS(RE, IMEM)*  *BEGIN*  *IF RE = '0' THEN*  *DOUT <= (OTHERS => 'Z');*  *END IF;*    *IF RE = '1' THEN*  *IMEM(0) <= "0000000000000000";*  *IMEM(1) <= "0001000000000000";*  *IMEM(2) <= "0010000000000000";*  *IMEM(3) <= "0011000000000000";*  *IMEM(4) <= "0100000000000000";*  *IMEM(5) <= "0101000000000000";*  *IMEM(6) <= "0110000000000000";*  *IMEM(7) <= "0111000000000000";*  *IMEM(8) <= "1000000000000000";*  *IMEM(9) <= "1001000000000000";*  *IMEM(10) <= "1010000000000000";*  *IMEM(11) <= "1011000000000000";*  *IMEM(12) <= "1100000000000000";*  *IMEM(13) <= "1101000000000000";*  *IMEM(14) <= "1110000000000000";*  *IMEM(15) <= "1111000000000000";*  *IMEM(16) <= "0000000000000000";*  *IMEM(17) <= "0000000000000000";*  *IMEM(18) <= "0000000000000000";*  *IMEM(19) <= "0000000000000000";*  *END IF;*    *DOUT <= IMEM(CONV\_INTEGER(ADDR(5 DOWNTO 0)));*    *END PROCESS;* |
| --- |

*Attachment 5:INS\_MEM.vhd*

| *--Lisa Jacklin*  *-- EE 443 Lab 6 continued...*  *-- 3/28/2023*  *--SGNEXT 6x16.vhd*  *---------------------------------------------------*  *library ieee;*  *use ieee.std\_logic\_1164.all;*  *entity SGNEXT6x16 IS*  *PORT (SIGNIN :IN STD\_LOGIC\_VECTOR ( 6 DOWNTO 0);*  *SIGNOUT:OUT STD\_LOGIC\_VECTOR (15 DOWNTO 0));*    *END SGNEXT6x16;*    *ARCHITECTURE SGN\_BEHAVIOR OF SGNEXT6x16 IS*  *BEGIN*  *--CONCATINATE THE ADDON OF FORWARDING ZEROS TO THE ORIGINAL 6 INPUT VALS*  *SIGNOUT <= "000000000" & SIGNIN;*      *END SGN\_BEHAVIOR;* |
| --- |

*Attachment 6:SGNEXT6x16.vhd*

| *--Lisa Jacklin*  *--EE 443 Lab 6*  *--DCD4x16.vhd*  *------------------------------------------*  *LIBRARY IEEE;*  *USE IEEE.STD\_LOGIC\_1164.ALL;*  *ENTITY DCD4X16 IS*  *PORT( IN4 :IN STD\_LOGIC\_VECTOR(3 DOWNTO 0);*  *CLK :IN STD\_LOGIC;*  *OUT16 :OUT STD\_LOGIC\_VECTOR (15 DOWNTO 0));*    *END DCD4X16;*  *ARCHITECTURE DCD\_BEHAVIOR OF DCD4X16 IS*  *---------COMPONENTS--------------*  *COMPONENT DCD3x8 IS*  *PORT (S :IN STD\_LOGIC\_VECTOR(2 DOWNTO 0);*  *Q :OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0));*  *END COMPONENT;*  *SIGNAL INTERMED1, INTERMED2 : STD\_LOGIC\_VECTOR(7 DOWNTO 0);*  *BEGIN*  *--GOING TO NEED PORTMAPS TO PULL FOUR TO A SIXTEEN CONVERSION*  *D1: DCD3X8 PORT MAP (IN4 (2 DOWNTO 0), INTERMED1);*    *D2:DCD3X8 PORT MAP (IN4(3)& IN4(1 DOWNTO 0), INTERMED2);*    *--CONCATINATING THE TWO 3X8 RESULTS*  *OUT16 <= INTERMED1 & INTERMED2;*    *END DCD\_BEHAVIOR;* |
| --- |

*Attachment 7: DCD4x16.vhd*

| *--Lisa Jacklin*  *--EE 443 LAb 6*  *-- DATA\_MEM.vhd*  *----------------------------------------*  *LIBRARY IEEE;*  *USE IEEE.STD\_LOGIC\_1164.ALL;*  *ENTITY DATA\_MEM IS*  *PORT( ADDR, DIN :IN STD\_LOGIC\_VECTOR (15 DOWNTO 0);*  *WE, RE, CLK :IN STD\_LOGIC;*  *DOUT :OUT STD\_LOGIC\_VECTOR (15 DOWNTO 0));*  *END DATA\_MEM;*  *ARCHITECTURE DATA\_BEHAVIOR OF DATA\_MEM IS*  *------COMPONENTS-----------------------*  *COMPONENT DCD3x8 IS*  *PORT (S :IN STD\_LOGIC\_VECTOR(2 DOWNTO 0);*  *Q :OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0));*  *END COMPONENT;*  *COMPONENT DCD4X16 IS*  *PORT( IN4 :IN STD\_LOGIC\_VECTOR(3 DOWNTO 0);*  *CLK :IN STD\_LOGIC;*  *OUT16 :OUT STD\_LOGIC\_VECTOR (15 DOWNTO 0));*  *END COMPONENT;*    *COMPONENT REG8 IS*  *PORT (D :IN STD\_LOGIC\_VECTOR(7 DOWNTO 0);*  *WE, CLK :IN STD\_LOGIC; --NOTE THAT EN HERE IS THE WE (WRITE ENABLED)*  *Q :OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0 ));*  *END COMPONENT;*  *---------------------------------------*  *SIGNAL REGOUT1, REGOUT2 :STD\_LOGIC\_VECTOR(7 DOWNTO 0);*  *SIGNAL ADDR4: STD\_LOGIC\_VECTOR(3 DOWNTO 0);*  *SIGNAL INT\_BUS: STD\_LOGIC\_VECTOR(15 DOWNTO 0);*  *SIGNAL WORD\_SEL: STD\_LOGIC\_VECTOR(0 DOWNTO 0);*    *BEGIN*  *--SETTING UP THE TWO REGISTERS THAT ARE BEING PULLED FROM*  *INPUTS: for i in 0 to 15 GENERATE*  *b0: reg8 port map(DIN(7 DOWNTO 0), WE, CLK, REGOUT1 ); --LSB*  *b1: reg8 port map(DIN(15 downto 8), WE, CLK, REGOUT2); --MSB*    *END GENERATE INPUTS;*    *--HANDLING THE ADDRESS NOW, 16 TO 4*  *ADDR4 <= ADDR(3 DOWNTO 0);*    *--NOW, TO WORK ON THE MULTIPLEXER to choose the correct output*  *WITH ADDR4 SELECT*  *INT\_BUS <= REGOUT1 WHEN "0000",*  *REGOUT2 WHEN OTHERS;*    *--using the output from the mux,*  *WORD\_SEL(0) <= ADDR(4);*    *DOUT (15 DOWNTO 0) <= "ZZZZZZZZZZZZZZZZ" WHEN RE = '0'*  *ELSE INT\_BUS WHEN WORD\_SEL(0) = '1' ELSE (OTHERS => 'Z');*    *END DATA\_BEHAVIOR;* |
| --- |

*Attachment 8:DAT\_MEM.vhd*

| *--Lisa Jacklin*  *--EE 443 Lab 6*  *--SHLONE.vhd*  *----------------------------------------------*  *LIBRARY IEEE;*  *USE IEEE.STD\_LOGIC\_1164.ALL;*  *ENTITY SHLONE IS*  *PORT( VALU :IN STD\_LOGIC\_VECTOR(15 DOWNTO 0);*  *OUTP :OUT STD\_LOGIC\_VECTOR(15 DOWNTO 0));*    *END SHLONE;*  *ARCHITECTURE SHLONE\_BEHAVIOR OF SHLONE IS*  *BEGIN*  *--RECOMMENDED TO USE FOR GENERATE...HOW?*  *--FOR I IN 0 TO 15 GENERATE*    *OUTP (15 DOWNTO 0) <= VALU (14 DOWNTO 0) & '0';*  *END SHLONE\_BEHAVIOR;* |
| --- |

*Attachment 9: SHLONE.vhd*

| *--Lisa Jacklin*  *-- EE 443 Lab 6*  *--ADD16.vhd*  *---------------------------------------------*  *library ieee;*  *USE IEEE.STD\_LOGIC\_1164.ALL;*  *ENTITY ADD16 IS*  *PORT( INA, INB :IN STD\_LOGIC\_VECTOR(15 DOWNTO 0);*  *CAIN :IN STD\_LOGIC;*  *ADDOUT :OUT STD\_LOGIC\_VECTOR(15 DOWNTO 0);*  *CAOUT :OUT STD\_LOGIC);*  *END ADD16;*  *ARCHITECTURE ADD16\_BEHAVIOR OF ADD16 IS*  *--THE 16 BIT ADDER CONSISTS OF ADD4 PERFORMED IN RIPPLE CARRY*  *COMPONENT ADD4 IS*  *PORT ( X,Y :IN STD\_LOGIC\_VECTOR(3 DOWNTO 0);*  *CIN :IN STD\_LOGIC;*  *SUM :OUT STD\_LOGIC\_VECTOR(3 DOWNTO 0);*  *COUT :OUT STD\_LOGIC);*  *END COMPONENT;*    *--GOING TO ASSIGN SIGNALS HERE*  *SIGNAL COUTS,CO,SUM1,SUM2 : STD\_LOGIC\_VECTOR(3 DOWNTO 0);*    *BEGIN*  *--BEGINNING PORTMAPS WHICH TAKE THE FIRST 16 BITS*  *A1: ADD4 PORT MAP (INA(3 DOWNTO 0), INB (3 DOWNTO 0), CAIN, ADDOUT(3 DOWNTO 0), COUTS(0) );*  *A2: ADD4 PORT MAP (INA(7 DOWNTO 4), INB (7 DOWNTO 4), COUTS(0), ADDOUT(7 DOWNTO 4), COUTS(1) );*  *A3: ADD4 PORT MAP (INA (11 DOWNTO 8), INB (11 DOWNTO 8), COUTS(1), ADDOUT(11 DOWNTO 8), COUTS(2));*  *A4: ADD4 PORT MAP (INA (15 DOWNTO 12), INB (15 DOWNTO 12), COUTS(2), ADDOUT (15 DOWNTO 12), COUTS(3));*    *CAOUT <= COUTS(0) AND COUTS(1) AND COUTS(2) AND COUTS(3);*    *END ADD16\_BEHAVIOR;* |
| --- |

*Attachment 10: ADD16.vhd*

| *--Lisa Jacklin*  *--EE 443 Lab 6*  *--MUX3x16.vhd*  *-------------------------------*  *library IEEE;*  *use IEEE.STD\_LOGIC\_1164.all;*  *entity mux3x16 is*  *port (*  *A : in std\_logic\_vector(15 downto 0);*  *B : in std\_logic\_vector(15 downto 0);*  *C : in std\_logic\_vector(15 downto 0);*  *sel : in std\_logic\_vector(1 downto 0);*  *Y : out std\_logic\_vector(15 downto 0)*  *);*  *end entity mux3x16;*  *architecture behavior of mux3x16 is*  *begin*  *Y <= A when sel = "00" else*  *B when sel = "01" else*  *C;*  *end architecture behavior;* |
| --- |

*Attachment 11: MUX3x16.vhd*