Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών
Σχεδίαση Ψηφιακών Συστημάτων (ICE – 4005) Project_MIPS _2020
Γκούσαρης Κωνσταντίνος – 711171073 – cs171073@uniwa.gr

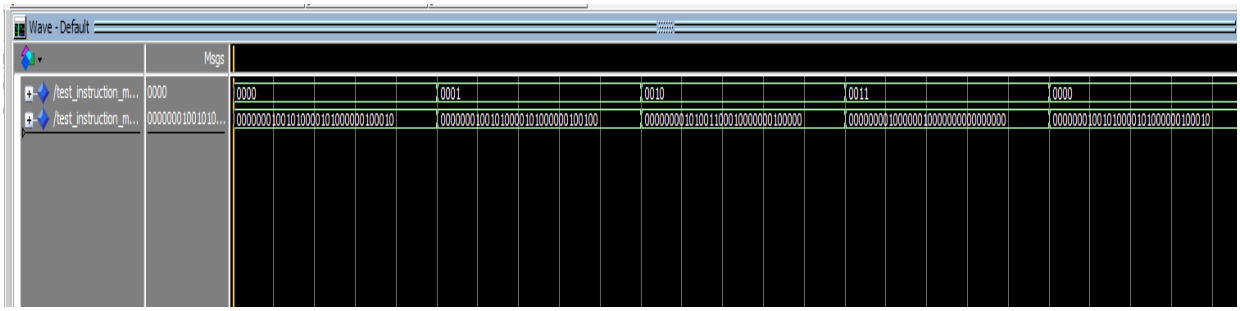
4: Instruction and Data Memory

4.1_InstructionMemory.vhd

```
--MIPS Part_4
--MemoryInstruction
--27/05/2020, Konstantinos Gkousaris, 711171073, UniWA

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY instrMemory IS PORT (
            Addr : IN STD_LOGIC_VECTOR(31 downto 0);
            C : out STD_LOGIC_VECTOR(31 downto 0));
END instrMemory;

ARCHITECTURE arch1 OF instrMemory IS

TYPE rom16x32 IS ARRAY (0 TO 15) OF STD_LOGIC_VECTOR(31 downto 0);

        --give default
      SIGNAL instrmem : rom16x32 := (
            "11111111111111111111111111111111",
            "00000000000000000000000000000000",
            "11111111111111111111111111111111",
            "00000000000000000000000000000000",
            "11111111111111111111111111111111",
            "00000000101001100010000000100000",
            "11111111111111111111111111111111",
            "11111111111111111111111111111111",
            "11111111111111111111111111111111",
            "00000000101001100010000000100000",
            "11111111111111111111111111111111",
            "11111111111111111111111111111111",
            "11111111111111111111111111111111",
            "11111111111111111111111111111111",
            "11111111111111111111111111111111",
            "11111111111111111111111111111111");

BEGIN
      C <= instrmem(to_integer(unsigned(Addr)));

END arch1;
```

4.1 InstructionMemory_testbench.vhd

```
--testbench code
--run for 4 instructions and jumps to address 0

LIBRARY ieee;
USE ieee.std_logic_1164.all;

USE ieee.numeric_std.ALL;

ENTITY test_instruction_memory IS
END test_instruction_memory;


ARCHITECTURE behabioral OF test_instruction_memory IS


COMPONENT instrMemory PORT (
            Addr : IN STD_LOGIC_VECTOR(3 downto 0);
            C : OUT STD_LOGIC_VECTOR(31 downto 0));
END COMPONENT;

      SIGNAL  Addr1 : STD_LOGIC_VECTOR(3 DOWNTO 0);
      SIGNAL  C1 : STD_LOGIC_VECTOR(31 DOWNTO 0);

BEGIN
      INSTRUCTIONMEMORY1 : instrMemory PORT MAP (
            Addr => Addr1,
            C => C1);

      read_instruction : PROCESS
      BEGIN
            FOR I IN 0 TO 3 LOOP
                  Addr1 <= std_logic_vector(to_unsigned(I, 4));
            WAIT FOR 50 ns;
            END LOOP;
      END PROCESS;
END;
```



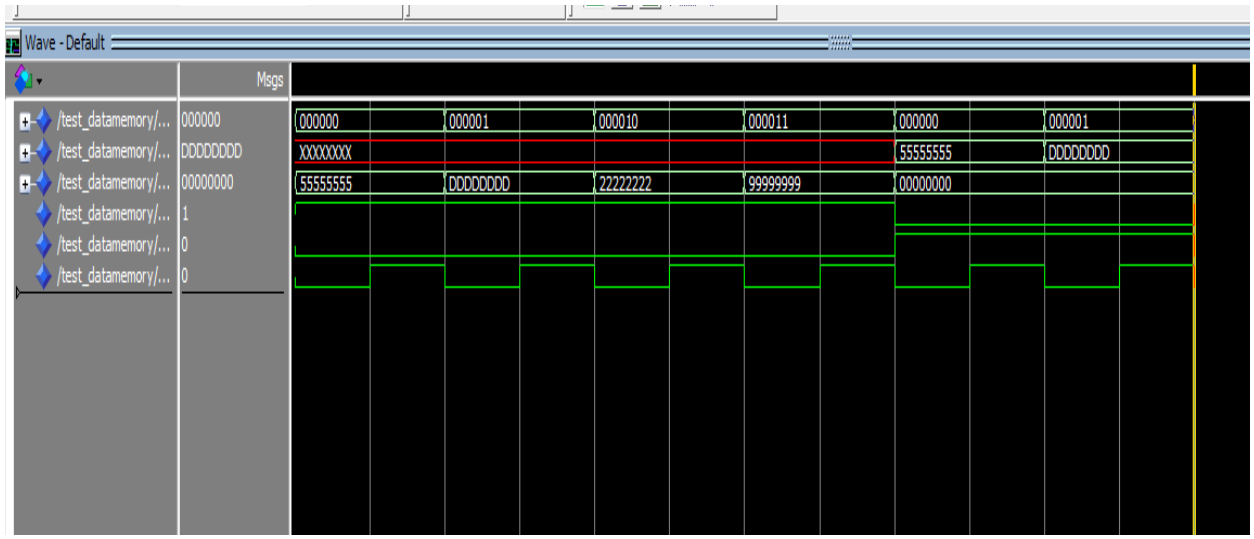
4.2 DataMemory.vhd

```
--MIPS Part_4
--DATA MEMORY
--18/05/2020, Konstantinos Gkousaris, 711171073, UniWA

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;
ENTITY dataMemory IS PORT (
      Addr : IN STD_LOGIC_VECTOR(5 DOWNTO 0);
      writeD : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
      we : IN STD_LOGIC;
      re : IN STD_LOGIC;
      readD : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
      clk : IN STD_LOGIC);
END dataMemory;

--add a clock input, because the process works with clock
--i dont know if i use it in MIPS implentation yet
--or change the component, but for the separate trial
--needs clock.

ARCHITECTURE behavioral OF datamemory IS

TYPE memArray IS ARRAY(0 TO 63) OF STD_LOGIC_VECTOR(31 downto 0);
SIGNAL memfile : memArray;
      BEGIN
      PROCESS(clk)
      BEGIN
            IF (clk'event and clk='0') THEN
                  IF we='1' THEN
                        memfile(to_integer(unsigned(Addr))) <= writeD;
                  END IF;
            END IF;
            IF re='1' THEN
                  readD <= memfile(to_integer(unsigned(Addr)));
            END IF;
      END PROCESS;
END behavioral;
```

4.2 DataMemory_testbench.vhd

```
--testbench code
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY test_datamemory IS
END test_datamemory;

ARCHITECTURE behavioral OF test_datamemory IS

COMPONENT dataMemory IS PORT (
      Addr : IN STD_LOGIC_VECTOR(5 DOWNTO 0);
      writeD : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
      we : IN STD_LOGIC;
      re : IN STD_LOGIC;
      readD : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
      clk : IN STD_LOGIC);
END COMPONENT;

      SIGNAL AddrS : STD_LOGIC_VECTOR(5 DOWNTO 0);
      SIGNAL readD_S, writeD_S : STD_LOGIC_VECTOR(31 DOWNTO 0);
      SIGNAL we_S, re_S, clkS : STD_LOGIC;

      constant clk_period : time := 100 ns;


BEGIN

      TESTDATAMEMORY1 : dataMemory PORT MAP (AddrS, writeD_S, we_S, re_S,
readD_S, clkS);

      --process for clock
      clk_process :PROCESS
      BEGIN
            clkS    <= '0'; wait for clk_period/2;
            clkS    <= '1'; wait for clk_period/2;
      END PROCESS;

      simulation_Address: PROCESS
      BEGIN
            AddrS   <= "000000"; wait for 100 ns;
            AddrS   <= "000001"; wait for 100 ns;
            AddrS   <= "000010"; wait for 100 ns;
            AddrS   <= "000011"; wait for 100 ns;
            AddrS   <= "000000"; wait for 100 ns;
            AddrS   <= "000001"; wait for 100 ns;

      END PROCESS;

      simulation_write: PROCESS
      BEGIN
            we_S  <= '1'; wait for 100 ns;
            we_S  <= '1'; wait for 100 ns;
            we_S  <= '1'; wait for 100 ns;
            we_S  <= '1'; wait for 100 ns;
            we_S  <= '0'; wait for 100 ns;
            we_S  <= '0'; wait for 100 ns;
      END PROCESS;
```

```
	simulation_read: PROCESS
	BEGIN
		re_S  <= '0'; wait for 100 ns;
		re_S  <= '0'; wait for 100 ns;
		re_S  <= '0'; wait for 100 ns;
		re_S  <= '0'; wait for 100 ns;
		re_S  <= '1'; wait for 100 ns;
		re_S  <= '1'; wait for 100 ns;
	END PROCESS;

	simulation_data: PROCESS
	BEGIN
		writeD_S <= "01010101010101010101010101010101"; wait for 100 ns;
		writeD_S <= "11011101110111011101110111011101"; wait for 100 ns;
		writeD_S <= "00100010001000100010001000100010"; wait for 100 ns;
		writeD_S <= "10011001100110011001100110011001"; wait for 100 ns;
		writeD_S <= "00000000000000000000000000000000"; wait;
	END PROCESS;

END;
```



***Πίνακας Λειτουργίας**

Address	Operation	Περιεχόμενα RAM
000000	write	01010101010101010101010101010101
000001	write	11011101110111011101110111011101
000010	write	00100010001000100010001000100010
000011	write	10011001100110011001100110011001
000000	read	**01010101010101010101010101010101**
000000	read	**11011101110111011101110111011101**