Σχεδίαση Ψηφιακών Συστημάτων

ΕΞΑΜΗΝΙΑΙΑ ΑΣΚΗΣΗ ΘΕΩΡΙΑΣ

ΜΕΡΟΣ 06

Τμήμα Μηχανικών Πληροφορικής

& Υπολογιστών

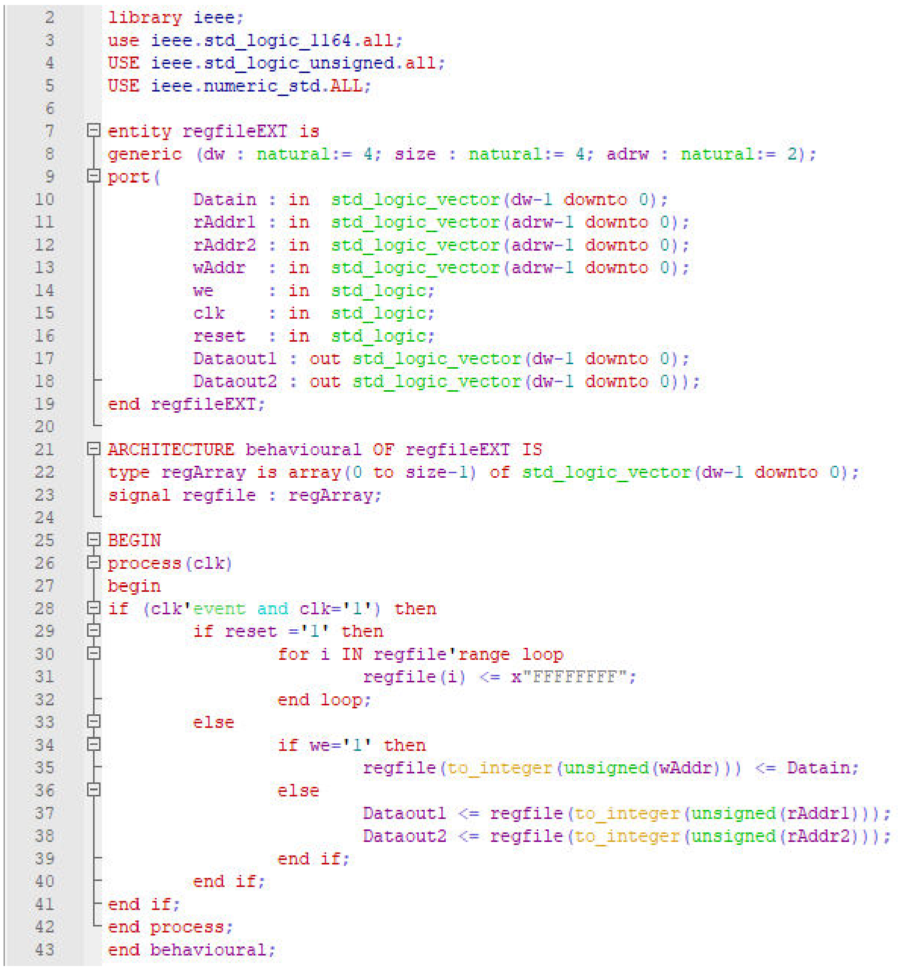
(Καθηγητής: Ιωάννης Βογιατζής)

Μάρκος Παντελιδάκης - 18390228 - ice18390228@uniwa.gr

MIPS.vhd

6.0

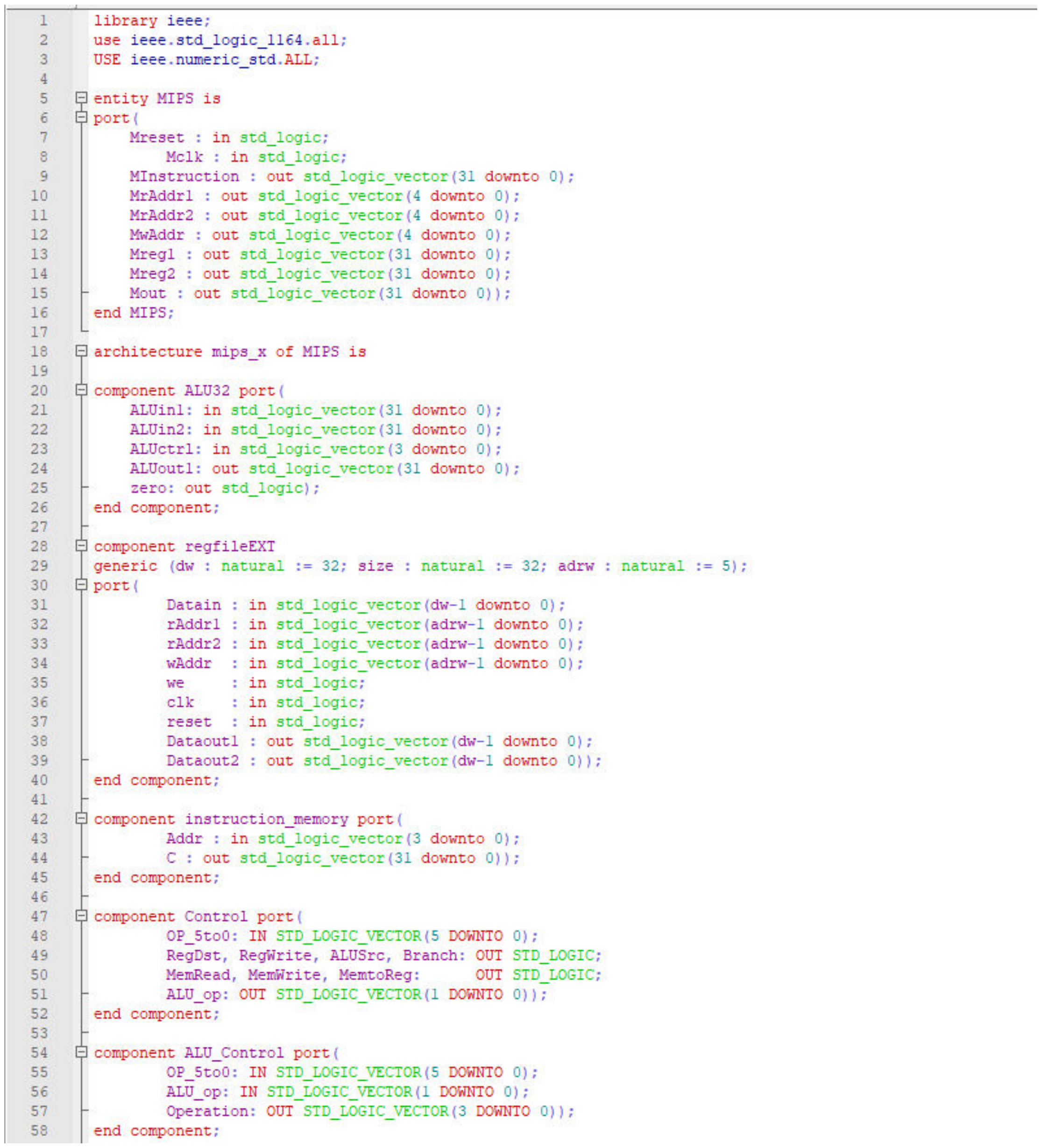
Τροποποιούμε το register file, ώστε όταν το reset είναι ενεργοποιημένο, όλοι οι καταχωρητές να αρχικοποιούνται στο 0xFFFFFFFF.



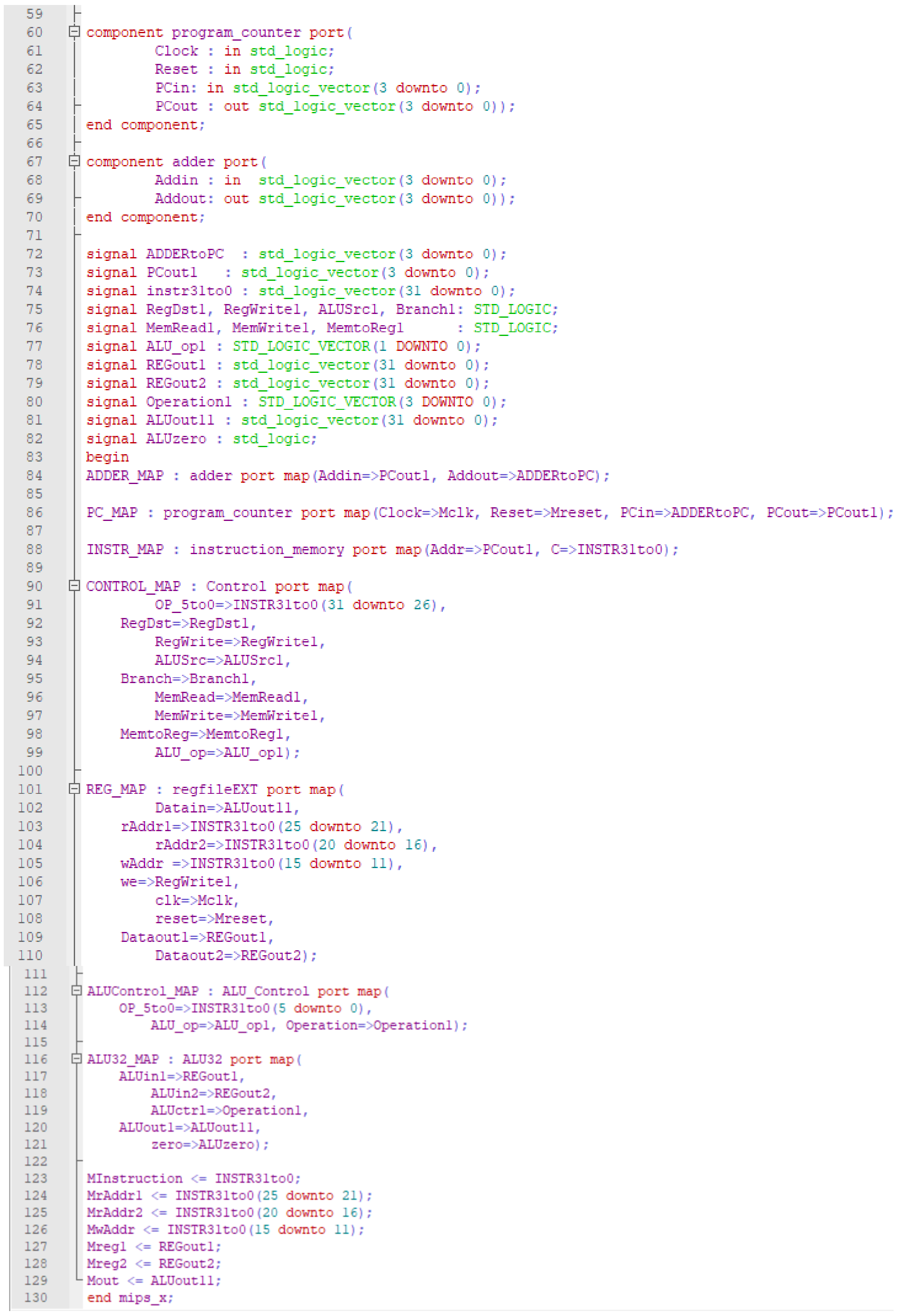
```vhdl
library ieee;
use ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

entity regfileEXT is
generic (dw : natural:= 4; size : natural:= 4; adrw : natural:= 2);
port(
        Datain : in  std_logic_vector(dw-1 downto 0);
        rAddr1 : in  std_logic_vector(adrw-1 downto 0);
        rAddr2 : in  std_logic_vector(adrw-1 downto 0);
        wAddr  : in  std_logic_vector(adrw-1 downto 0);
        we     : in  std_logic;
        clk    : in  std_logic;
        reset  : in  std_logic;
        Dataout1 : out std_logic_vector(dw-1 downto 0);
        Dataout2 : out std_logic_vector(dw-1 downto 0));
end regfileEXT;

ARCHITECTURE behavioural OF regfileEXT IS
type regArray is array(0 to size-1) of std_logic_vector(dw-1 downto 0);
signal regfile : regArray;

BEGIN
process(clk)
begin
if (clk'event and clk='1') then
        if reset ='1' then
                for i IN regfile'range loop
                        regfile(i) <= x"FFFFFFFF";
                end loop;
        else
                if we='1' then
                        regfile(to_integer(unsigned(wAddr))) <= Datain;
                else
                        Dataout1 <= regfile(to_integer(unsigned(rAddr1)));
                        Dataout2 <= regfile(to_integer(unsigned(rAddr2)));
                end if;
        end if;
end if;
end process;
end behavioural;
```

Εικ6.0: Τροποποιημένο Register file.

6.1

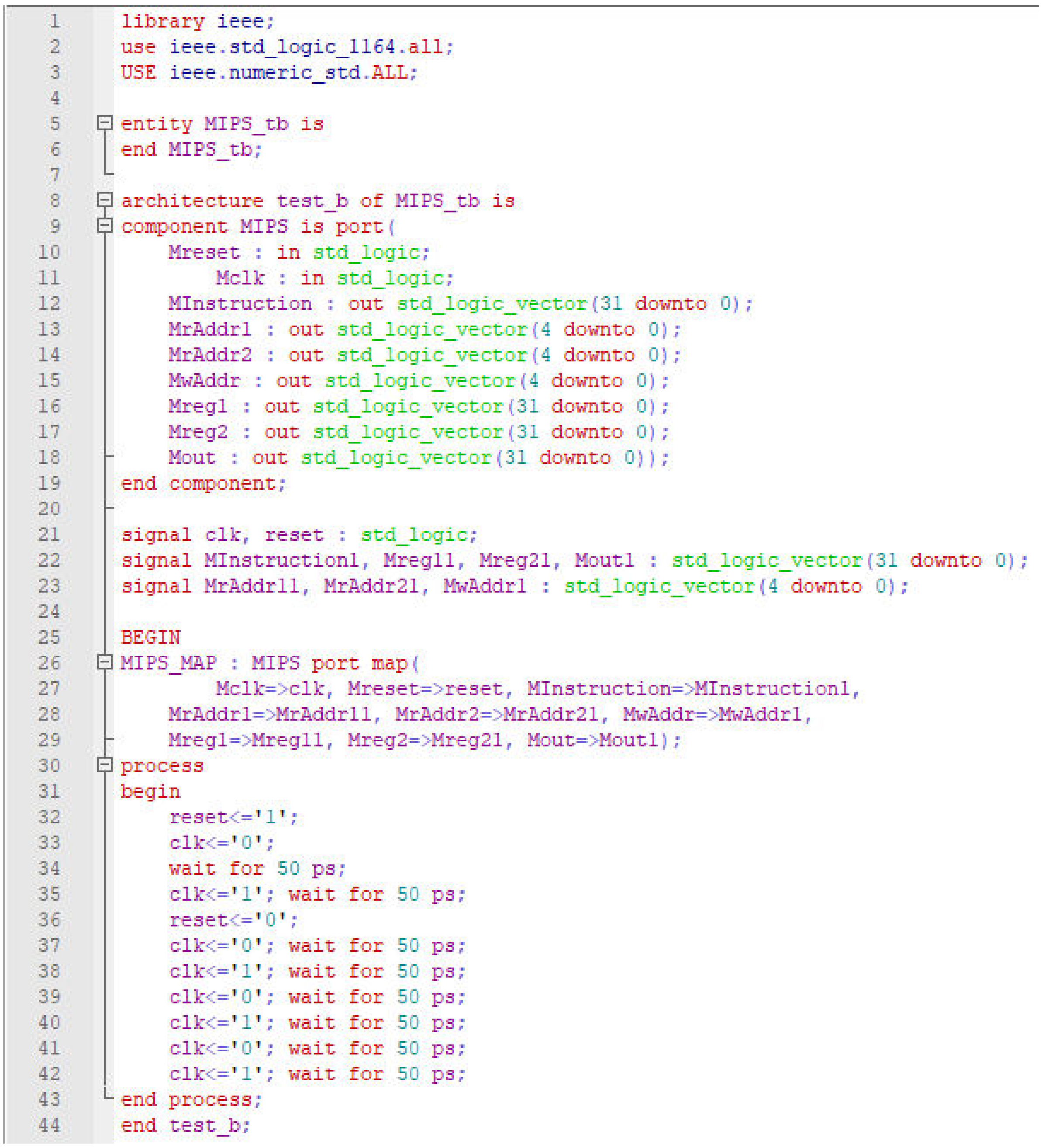


```vhdl
library ieee;
use ieee.std_logic_1164.all;
USE ieee.numeric_std.ALL;

entity MIPS is
port(
    Mreset : in std_logic;
        Mclk : in std_logic;
    MInstruction : out std_logic_vector(31 downto 0);
    MrAddr1 : out std_logic_vector(4 downto 0);
    MrAddr2 : out std_logic_vector(4 downto 0);
    MwAddr : out std_logic_vector(4 downto 0);
    Mreg1 : out std_logic_vector(31 downto 0);
    Mreg2 : out std_logic_vector(31 downto 0);
    Mout : out std_logic_vector(31 downto 0));
end MIPS;

architecture mips_x of MIPS is

component ALU32 port(
    ALUin1: in std_logic_vector(31 downto 0);
    ALUin2: in std_logic_vector(31 downto 0);
    ALUctrl: in std_logic_vector(3 downto 0);
    ALUout1: out std_logic_vector(31 downto 0);
    zero: out std_logic);
end component;

component regfileEXT
generic (dw : natural := 32; size : natural := 32; adrw : natural := 5);
port(
        Datain : in std_logic_vector(dw-1 downto 0);
        rAddr1 : in std_logic_vector(adrw-1 downto 0);
        rAddr2 : in std_logic_vector(adrw-1 downto 0);
        wAddr  : in std_logic_vector(adrw-1 downto 0);
        we     : in std_logic;
        clk    : in std_logic;
        reset  : in std_logic;
        Dataout1 : out std_logic_vector(dw-1 downto 0);
        Dataout2 : out std_logic_vector(dw-1 downto 0));
end component;

component instruction_memory port(
        Addr : in std_logic_vector(3 downto 0);
        C : out std_logic_vector(31 downto 0));
end component;

component Control port(
        OP_5to0: IN STD_LOGIC_VECTOR(5 DOWNTO 0);
        RegDst, RegWrite, ALUSrc, Branch: OUT STD_LOGIC;
        MemRead, MemWrite, MemtoReg:      OUT STD_LOGIC;
        ALU_op: OUT STD_LOGIC_VECTOR(1 DOWNTO 0));
end component;

component ALU_Control port(
        OP_5to0: IN STD_LOGIC_VECTOR(5 DOWNTO 0);
        ALU_op: IN STD_LOGIC_VECTOR(1 DOWNTO 0);
        Operation: OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
end component;
```



```vhdl

component program_counter port(
        Clock : in std_logic;
        Reset : in std_logic;
        PCin: in std_logic_vector(3 downto 0);
        PCout : out std_logic_vector(3 downto 0));
end component;

component adder port(
        Addin : in  std_logic_vector(3 downto 0);
        Addout: out std_logic_vector(3 downto 0));
end component;

signal ADDERtoPC  : std_logic_vector(3 downto 0);
signal PCout1   : std_logic_vector(3 downto 0);
signal instr31to0 : std_logic_vector(31 downto 0);
signal RegDst1, RegWrite1, ALUSrc1, Branch1: STD_LOGIC;
signal MemRead1, MemWrite1, MemtoReg1      : STD_LOGIC;
signal ALU_op1 : STD_LOGIC_VECTOR(1 DOWNTO 0);
signal REGout1 : std_logic_vector(31 downto 0);
signal REGout2 : std_logic_vector(31 downto 0);
signal Operation1 : STD_LOGIC_VECTOR(3 DOWNTO 0);
signal ALUout11 : std_logic_vector(31 downto 0);
signal ALUzero : std_logic;
begin
ADDER_MAP : adder port map(Addin=>PCout1, Addout=>ADDERtoPC);

PC_MAP : program_counter port map(Clock=>Mclk, Reset=>Mreset, PCin=>ADDERtoPC, PCout=>PCout1);

INSTR_MAP : instruction_memory port map(Addr=>PCout1, C=>INSTR31to0);

CONTROL_MAP : Control port map(
        OP_5to0=>INSTR31to0(31 downto 26),
    RegDst=>RegDst1,
        RegWrite=>RegWrite1,
        ALUSrc=>ALUSrc1,
    Branch=>Branch1,
        MemRead=>MemRead1,
        MemWrite=>MemWrite1,
    MemtoReg=>MemtoReg1,
        ALU_op=>ALU_op1);

REG_MAP : regfileEXT port map(
        Datain=>ALUout11,
    rAddr1=>INSTR31to0(25 downto 21),
        rAddr2=>INSTR31to0(20 downto 16),
    wAddr =>INSTR31to0(15 downto 11),
    we=>RegWrite1,
        clk=>Mclk,
        reset=>Mreset,
    Dataout1=>REGout1,
        Dataout2=>REGout2);

ALUControl_MAP : ALU_Control port map(
    OP_5to0=>INSTR31to0(5 downto 0),
        ALU_op=>ALU_op1, Operation=>Operation1);

ALU32_MAP : ALU32 port map(
    ALUin1=>REGout1,
        ALUin2=>REGout2,
        ALUctrl=>Operation1,
    ALUout1=>ALUout11,
        zero=>ALUzero);

MInstruction <= INSTR31to0;
MrAddr1 <= INSTR31to0(25 downto 21);
MrAddr2 <= INSTR31to0(20 downto 16);
MwAddr <= INSTR31to0(15 downto 11);
Mreg1 <= REGout1;
Mreg2 <= REGout2;
Mout <= ALUout11;
end mips_x;
```

Εικ6.1: VHDL κώδικας με MIPS entity.



```vhdl
library ieee;
use ieee.std_logic_1164.all;
USE ieee.numeric_std.ALL;

entity MIPS_tb is
end MIPS_tb;

architecture test_b of MIPS_tb is
component MIPS is port(
    Mreset : in std_logic;
        Mclk : in std_logic;
    MInstruction : out std_logic_vector(31 downto 0);
    MrAddr1 : out std_logic_vector(4 downto 0);
    MrAddr2 : out std_logic_vector(4 downto 0);
    MwAddr : out std_logic_vector(4 downto 0);
    Mreg1 : out std_logic_vector(31 downto 0);
    Mreg2 : out std_logic_vector(31 downto 0);
    Mout : out std_logic_vector(31 downto 0));
end component;

signal clk, reset : std_logic;
signal MInstruction1, Mreg11, Mreg21, Mout1 : std_logic_vector(31 downto 0);
signal MrAddr11, MrAddr21, MwAddr1 : std_logic_vector(4 downto 0);

BEGIN
MIPS_MAP : MIPS port map(
        Mclk=>clk, Mreset=>reset, MInstruction=>MInstruction1,
    MrAddr1=>MrAddr11, MrAddr2=>MrAddr21, MwAddr=>MwAddr1,
    Mreg1=>Mreg11, Mreg2=>Mreg21, Mout=>Mout1);
process
begin
    reset<='1';
    clk<='0';
    wait for 50 ps;
    clk<='1'; wait for 50 ps;
    reset<='0';
    clk<='0'; wait for 50 ps;
    clk<='1'; wait for 50 ps;
    clk<='0'; wait for 50 ps;
    clk<='1'; wait for 50 ps;
    clk<='0'; wait for 50 ps;
    clk<='1'; wait for 50 ps;
end process;
end test_b;
```

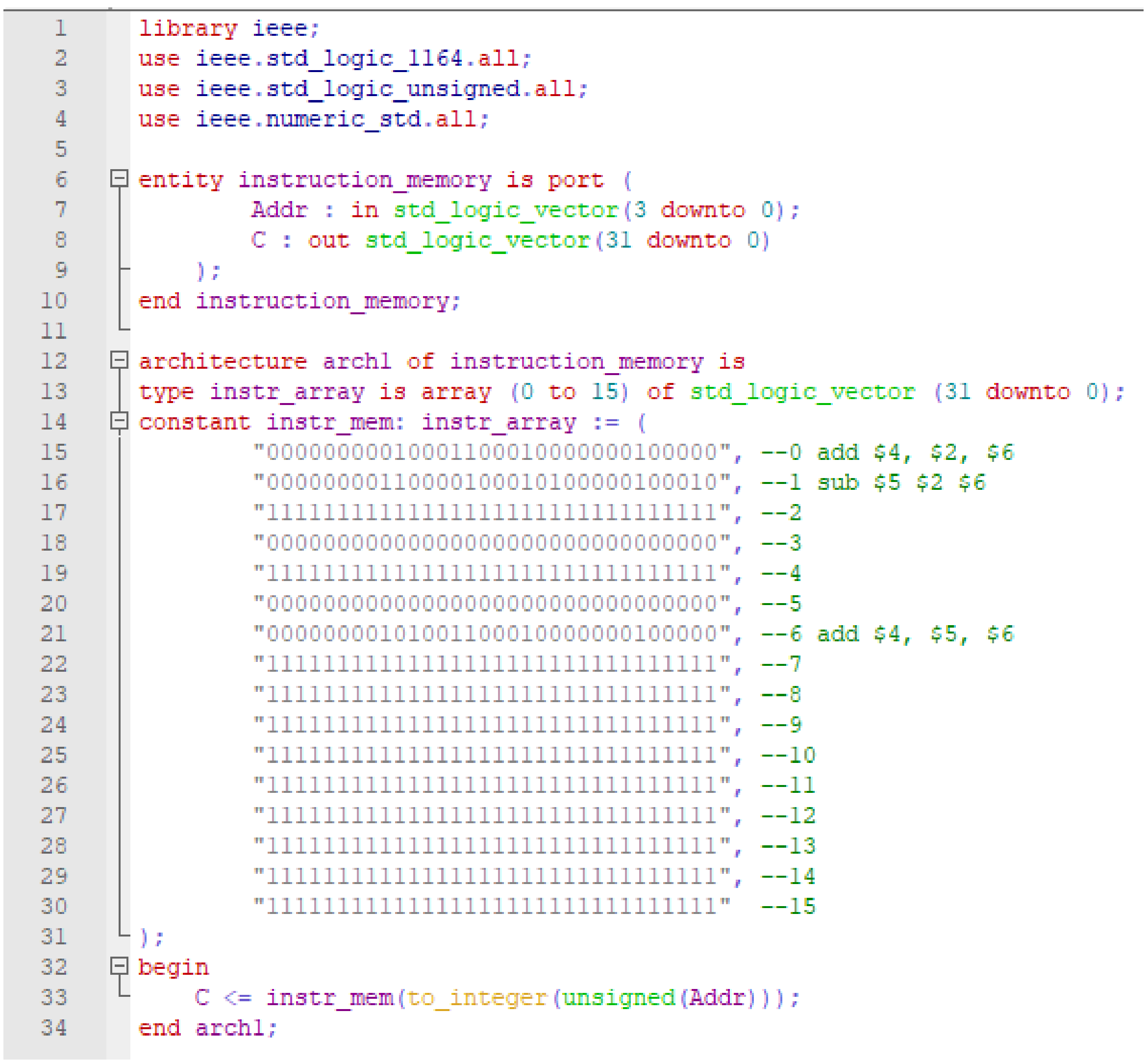
Εικ6.1.2: VHDL testbench κώδικας για το MIPS entity.

6.4.1

Τοποθετούμε τις ακόλουθες εντολές στις θέσεις 0, 1 της μνήμης.

add $4, $2, $6

sub $5, $2, $6

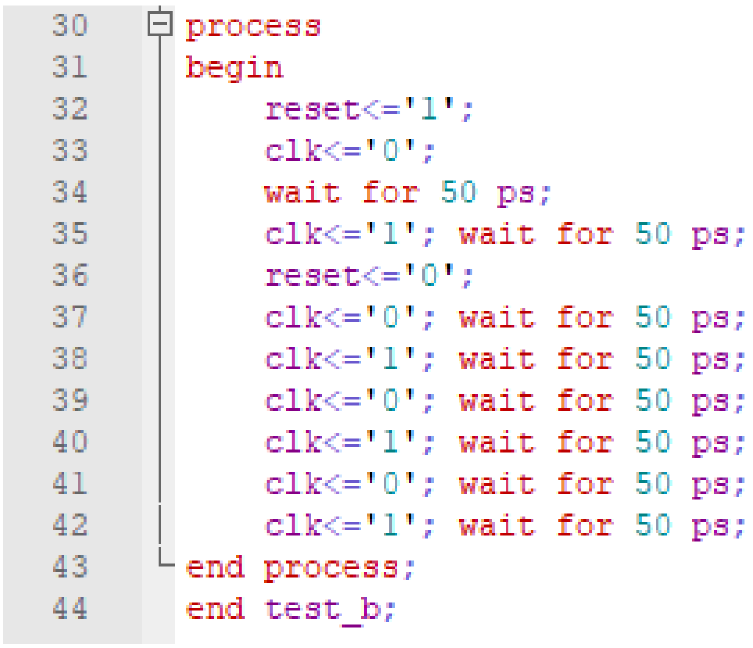


```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;

entity instruction_memory is port (
        Addr : in std_logic_vector(3 downto 0);
        C : out std_logic_vector(31 downto 0)
    );
end instruction_memory;

architecture archl of instruction_memory is
type instr_array is array (0 to 15) of std_logic_vector (31 downto 0);
constant instr_mem: instr_array := (
        "00000000010001100010000000100000", --0 add $4, $2, $6
        "00000000011000100010100000100010", --1 sub $5 $2 $6
        "11111111111111111111111111111111", --2
        "00000000000000000000000000000000", --3
        "11111111111111111111111111111111", --4
        "00000000000000000000000000000000", --5
        "00000000101001100010000000100000", --6 add $4, $5, $6
        "11111111111111111111111111111111", --7
        "11111111111111111111111111111111", --8
        "11111111111111111111111111111111", --9
        "11111111111111111111111111111111", --10
        "11111111111111111111111111111111", --11
        "11111111111111111111111111111111", --12
        "11111111111111111111111111111111", --13
        "11111111111111111111111111111111", --14
        "11111111111111111111111111111111"  --15
);
begin
    C <= instr_mem(to_integer(unsigned(Addr)));
end archl;
```

Εικ6.4.1: VHDL κώδικας μνήμης εντολών MIPS.

6.4.2

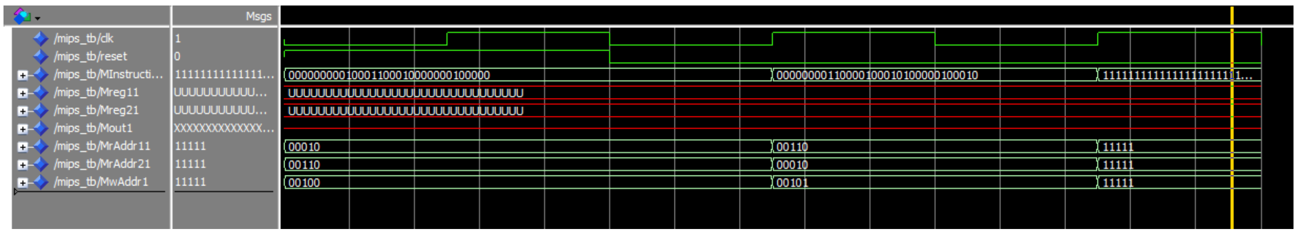
Ενεργοποιούμε το σήμα reset για ένα κύκλο του ρολογιού όπως φαίνεται στην Εικόνα 6.4.2.



```
process
begin
    reset<='1';
    clk<='0';
    wait for 50 ps;
    clk<='1'; wait for 50 ps;
    reset<='0';
    clk<='0'; wait for 50 ps;
    clk<='1'; wait for 50 ps;
    clk<='0'; wait for 50 ps;
    clk<='1'; wait for 50 ps;
    clk<='0'; wait for 50 ps;
    clk<='1'; wait for 50 ps;
end process;
end test_b;
```

Εικ6.4.2: Ενεργοποίηση reset για ένα κύκλο ρολογιού μέσω αρχείου testbench.

6.4.3



```
                   Msgs
/mips_tb/clk       1
/mips_tb/reset     0
/mips_tb/MInstructi... 11111111111111...  000000000100011000100000000100000  000000001100001000101000000100010  11111111111111111111111...
/mips_tb/Mreg11    UUUUUUUUUUU...  UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
/mips_tb/Mreg21    UUUUUUUUUUU...  UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
/mips_tb/Mout1     XXXXXXXXXXXXXX...
/mips_tb/MrAddr11  11111  00010  00110  11111
/mips_tb/MrAddr21  11111  00110  00010  11111
/mips_tb/MwAddr1   11111  00100  00101  11111
```

Εικ6.4.3: Αποτελέσματα του MIPS από τη κυματομορφή.