Παντελεήμων Πρώιος

ice 18390023

4ο ‘Εξάμηνο

Μηχανικών Πληροφορικής και Υπολογιστών

ice18390023@uniwa.gr

ΤΕΛΙΚΗ ΕΡΓΑΣΙΑ ΕΞΑΜΗΝΟΥ

ΣΧΕΔΙΑΣΗ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

PART 6

ΥΠΕΥΘΗΝΟΙ ΚΑΘΗΓΗΤΕΣ

ΠΑΝΑΓΙΩΤΗΣ ΚΑΡΚΑΖΗΣ

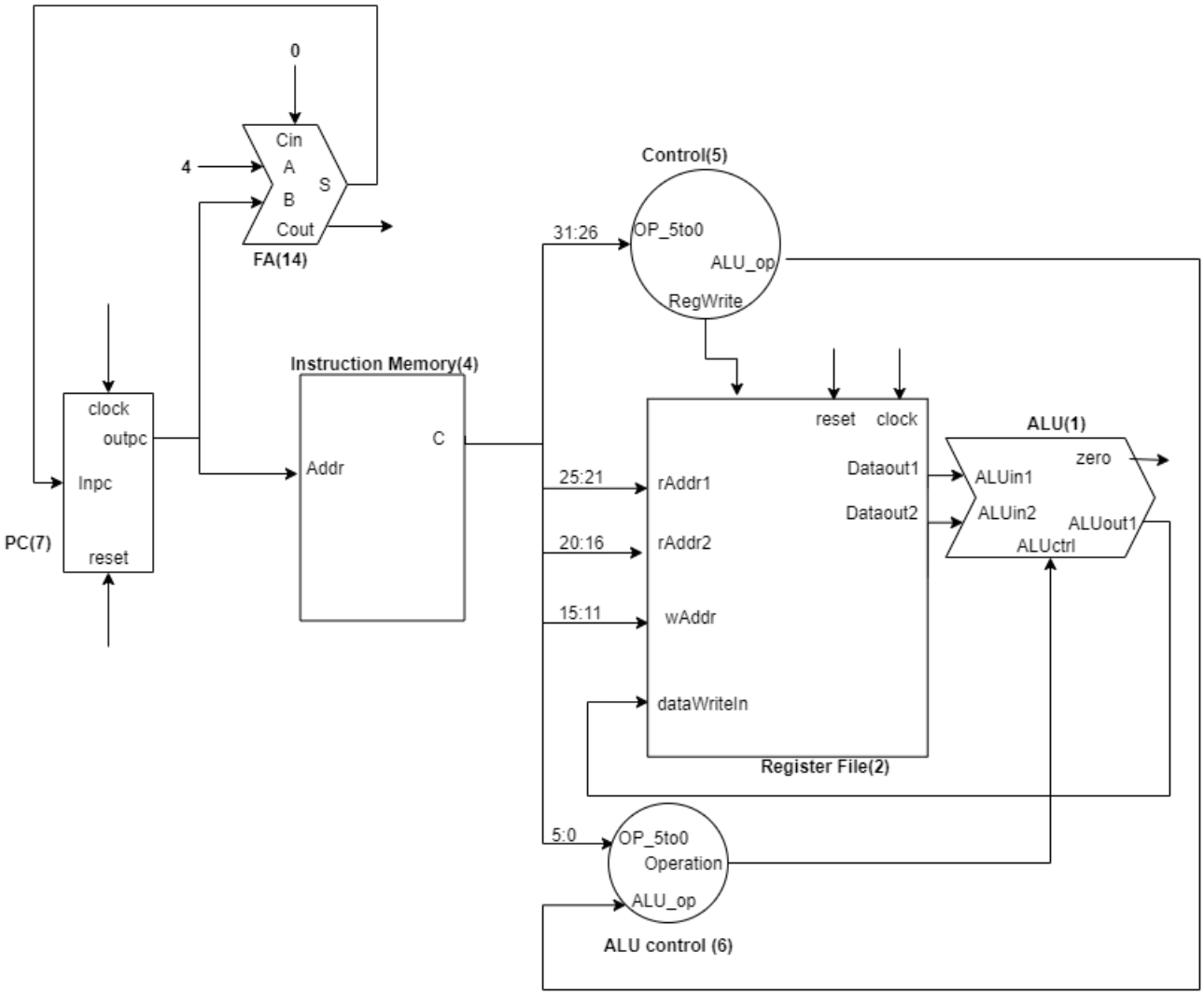
ΙΩΑΝΝΗΣ ΒΟΓΙΑΤΖΗΣ

ΑΘΑΝΑΣΙΟΣ ΜΗΛΙΔΩΝΗΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

MIPS

Στην μονάδα mips έχουμε βάλει έξτρα components τα οποία δεν χρειάζονται και θα μπορούσαμε να τα αφαιρέσουμε (είναι με το σχόλιο no need). Όπως φαίνεται στο σχήμα παρακάτω, ο component FullAdder αυξάνει ανά 4 την έξοδο του ProgramCounter component και επιστρέφει το αποτέλεσμα του αθροίσματος στην είσοδο του ProgramCounter. Ο ProgramCounter έχει ως είσοδο την διεύθυνση και την στέλνει στην είσοδο όταν το clock είναι στην ανερχόμενη ακμή και το reset δεν είναι 1. Αν το reset είναι 1 τότε επειδή είναι ασύγχρονο οποιαδήποτε χρονική στιγμή ότι τιμή και να έχει το clock, η έξοδος θα είναι η μηδενική διεύθυνση μείων 4. Στην συγκεκριμένα περίπτωση επειδή έχουμε 16 words στην instruction memory δηλαδή 16*4 = 64 bytes άρα και τόσες διεύθυνσης θα είναι 6 bits και τότε το reset θα έχει ως έξοδο το δεκαεξαδικό 3C. Η διεύθυνση αυτή αποτελεί είσοδο στον component InstrMem αλλά μόνο τα 4 MSBs του. Επειδή τα αποθηκεύουμε ως WORDS στο array του και άρα του ζητάμε μία εντολή δηλαδή μία λέξη δηλαδή τέσσερα bytes δηλαδή τέσσερις διεύθυνσης. π.χ. απο 000000 θα πάει 000100,001000,001100,...,111100 οπότε κατευθύνονται μόνο τα 4 υπογραμμισμένα bits δηλαδή 0,1,2,3,...,F επειδή ζητάμε words. Η instrMem θα επιστρέφει μια λέξη δηλαδή 32 bits η οποία είναι η εντολή όπου τα πρώτα 6 MSBs είναι ο opcode και θα είναι η είσοδος του component Control τα επόμενα 5 θα είναι η είσοδος στον component RegFile στην είσοδο rAddr1 επίσης τα επόμενα 5 θα είναι στον ίδιο component η είσοδος στην rAddr2 θύρα και τα επόμενα 5 θα είναι η είσοδος στην wAddr. Τα 5 LSBs θα είναι η είσοδος στον component ALU_control και θα αποτελούν το πεδίο function της εντολής αν είναι R type αλλιώς μας είναι αδιάφορο (ο μικρό επεξεργαστής δουλεύει μόνο με R type). Στον component RegFile η είσοδος dataWriteIn είναι η έξοδος του component ALU της θύρας ALUout1, οι έξοδοι Dataout1, Dataout2 είναι οι είσοδοι ALUin1, ALUin2 αντίστοιχα στον component ALU. Επίσης είσοδο της ALU στην θύρα ALUctrl αποτελεί η έξοδος του component ALU_control.



Ο κώδικας της υλοποίησης του MIPS του ανωτέρου σχήματος είναι ο εξής:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY MIPS IS
PORT (
    reset   : IN STD_LOGIC;
    clock   : IN STD_LOGIC);
END MIPS;

ARCHITECTURE structural 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 regfileExtra
GENERIC (
    dw   : natural := 32; -- 32 bits of data
    size : natural := 32; -- 32 Registers
    adrw : natural := 5); -- 2^5 32Registers
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;

-- no need
COMPONENT dataMemory
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));
END COMPONENT;

COMPONENT instrMemory
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;

COMPONENT ProgramCounter
```

```
PORT (
    InPC     : IN STD_LOGIC_VECTOR(5 DOWNTO 0);
    clk      : IN STD_LOGIC;
    reset    : IN STD_LOGIC;
    OutPC    : OUT STD_LOGIC_VECTOR(5 DOWNTO 0));
END COMPONENT;

-- no need
COMPONENT mux2tolgen
GENERIC (
    dw : natural := 5);
PORT (
    a : IN STD_LOGIC_VECTOR(dw-1 DOWNTO 0);
    b : IN STD_LOGIC_VECTOR(dw-1 DOWNTO 0);
    s : IN STD_LOGIC;
    c : OUT STD_LOGIC_VECTOR(dw-1 DOWNTO 0));
END COMPONENT;

-- no need
COMPONENT SIGN_Extension
PORT (
    Instr_15to0     : IN  STD_LOGIC_VECTOR (15 DOWNTO 0);
    Sign_extended   : OUT STD_LOGIC_VECTOR (31 DOWNTO 0));
END COMPONENT;

-- no need
COMPONENT shiftleft2
PORT (
    In1 : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    d   : OUT STD_LOGIC_VECTOR(31 DOWNTO 0));
END COMPONENT;

COMPONENT FullAdder
PORT (
    cin     : IN  STD_LOGIC_VECTOR(0 DOWNTO 0);
    a,b     : IN  STD_LOGIC_VECTOR(5 DOWNTO 0);
    s       : OUT STD_LOGIC_VECTOR(5 DOWNTO 0);
    cout    : OUT STD_LOGIC);
END COMPONENT;

-- Έχουμε μόνο 16 words δηλαδή 16*4 bytes άρα χριαζόμαστε
-- 64 διευθύνσης 2^6 (5 downto 0)
SIGNAL PC           : STD_LOGIC_VECTOR( 5 DOWNTO 0) := b"11" & x"C";
SIGNAL addByFour    : STD_LOGIC_VECTOR( 5 DOWNTO 0) := b"00" & x"4";
SIGNAL newPC        : STD_LOGIC_VECTOR( 5 DOWNTO 0);
SIGNAL instructions : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL DataWriteIn  : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL RegOut1      : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL RegOut2      : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL RegWrite     : STD_LOGIC;
SIGNAL RegDst       : STD_LOGIC;
SIGNAL ALUSrc       : STD_LOGIC;
SIGNAL Branch       : STD_LOGIC;
```

```
SIGNAL MemRead      : STD_LOGIC;
SIGNAL MemWrite     : STD_LOGIC;
SIGNAL MemtoReg     : STD_LOGIC;
SIGNAL ALU_op       : STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL Operation    : STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL zero         : STD_LOGIC;

BEGIN

    -- 14
    FA1: FullAdder PORT MAP(
            a   => PC,
            b   => addByFour,
            cin => "0",
            s   => newPC); -- goes to 12,15

    -- 7
    PC1: ProgramCounter PORT MAP(
            InPC    => newPC,
            clk     => clock,
            reset   => reset,
            OutPC   => PC);

    -- 4
    IM: instrMemory PORT MAP(
            -- ο PC αυξάνει ανά 4 δηλαδή 000000, 000100, 001000,
            -- 001100
            -- οι διευθύνσης στο instrMemory είναι array 16 των 32 bit
            -- επειδή είναι WORDS είναι (0000)[00] τα (bits) και
            -- όχι τα [bits]
            Addr    => PC(5 DOWNTO 2),
            -- Read Only Memory of 16 words 2^4=16 address of words
            C       => instructions); --32 bits instruction data

    --5
    CTRL: Control PORT MAP(
            OP_5to0     => instructions(31 DOWNTO 26), -- opcode
            RegDst      => RegDst,
            RegWrite    => RegWrite,
            ALUSrc      => ALUSrc,
            Branch      => Branch,
            MemRead     => MemRead,
            MemWrite    => MemWrite,
            MemtoReg    => MemtoReg,
            ALU_op      => ALU_op);

    -- 2
    REG: regfileExtra PORT MAP(
            Datain   => DataWriteIn,
            rAddr1   => instructions(25 DOWNTO 21),
            rAddr2   => instructions(20 DOWNTO 16),
            wAddr    => instructions(15 DOWNTO 11),
            we       => RegWrite,
```

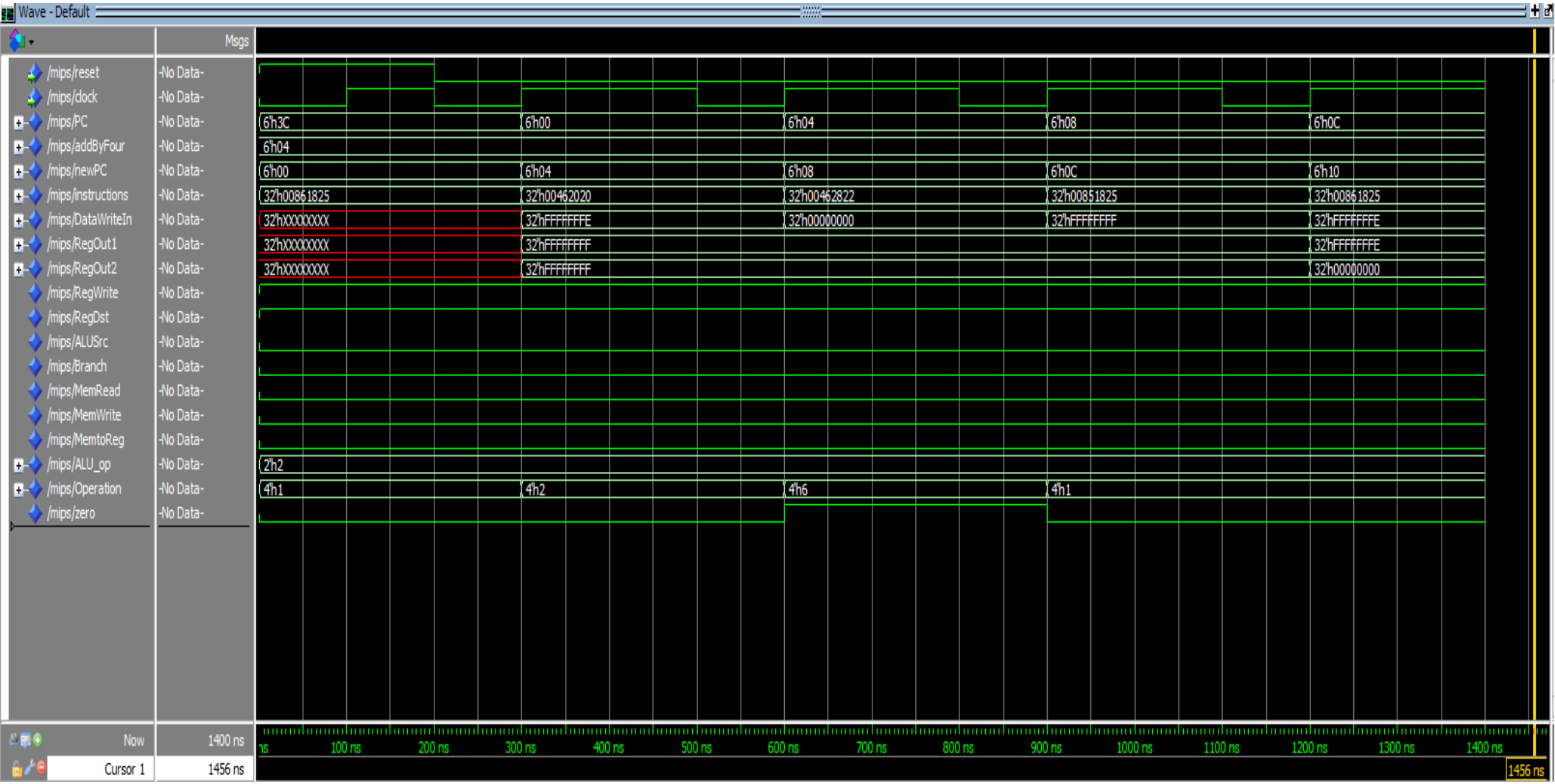
```
            clk      => clock,
            reset    => reset,
            Dataout1 => RegOut1,
            Dataout2 => RegOut2);

    -- 6
    AC: ALU_Control PORT MAP(
            OP_5to0     => instructions(5 DOWNTO 0), -- function
            ALU_op      => ALU_op,
            Operation   => Operation);


    -- 1
    ALU: ALU32 PORT MAP(
            ALUin1  => RegOut1,
            ALUin2  => RegOut2,
            ALUctrl => Operation,
            ALUout1 => DataWriteIn,
            zero    => zero);

END structural;
```

Στην παρακάτω εικόνα βλέπουμε την κυματομορφή του MIPS στην αρχή έχουμε το reset ίσο με 1 για ένα παλμό ρολογίου. Στον επόμενο παλμό έχουμε το reset ίσο με 0. Η πρώτη εντολή φαίνεται στο σήμα Instructions και είναι ο δεκαεξαδικός αριθμός 00462020 ή αλλιώς η εντολή add $4, $2, $6 η επόμενη εντολή στον επόμενο παλμό ρολογίου είναι ο δεκαεξαδικός αριθμός 00462822 ή αλλιώς sub $5, $2, $6. Η επόμενη εντολή είναι η or $3, $4, $5 για να δούμε τα περιεχόμενα των καταχωρητών 4 και 5 στο σήμα RegOut1 και RegOut2 το οποίο συμβαίνει στον επόμενο παλμό ρολογίου και έχουμε ως έξοδο τα σωστά αποτελέσματα (θα μπορούσαμε να βάλουμε οποιαδήποτε απο τις τέσσερις εντολές αρκεί να μην αλλάζαμε τους καταχωρητές 4 και 5).


Wave - Default
Msgs
/mips/reset
/mips/clock
/mips/PC
/mips/addByFour
/mips/newPC
/mips/instructions
/mips/DataWriteIn
/mips/RegOut1
/mips/RegOut2
/mips/RegWrite
/mips/RegDst
/mips/ALUSrc
/mips/Branch
/mips/MemRead
/mips/MemWrite
/mips/MemtoReg
/mips/ALU_op
/mips/Operation
/mips/zero
-No Data-
6'h3C
6'h00
6'h04
6'h08
6'h0C
6'h04
6'h00
6'h04
6'h08
6'h0C
6'h10
32'h00861825
32'h00462020
32'h00462822
32'h00851825
32'h00861825
32'hXXXXXXXX
32'hFFFFFFFE
32'h00000000
32'hFFFFFFFF
32'hXXXXXXXX
32'hFFFFFFFE
32'hXXXXXXXX
32'hFFFFFFFF
32'h00000000
2'h2
4'h1
4'h2
4'h6
4'h1
Now
1400 ns
Cursor 1
1456 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
900 ns
1000 ns
1100 ns
1200 ns
1300 ns
1400 ns
1456 ns