ΣΧΕΤΙΚΑ ΑΠΛΗ ΚΕΝΤΡΙΚΗ ΜΟΝΑΔΑ ΕΠΕΞΕΡΓΑΣΙΑΣ (ΜΚΕ)

5

ΟΝΟΜΑΤΕΠΩΝΥΜΟ : ΣΤΑΜΑΤΑΚΗΣ ΦΩΤΗΣ

ΑΜ:21048

Σκοπός

Με αφορμή την σχεδίαση και την εξομοίωση με διάφορους τρόπους, απλών ψηφιακών κυκλωμάτων θα κατακτηθεί το αντικείμενο της άσκησης αυτής που είναι η τελική σύνθεση της σχετικά απλής ΚΜΕ.

Η εσωτερική δομή της ΚΜΕ περιλαμβάνει τρία βασικά τμήματα:

- Το τμήμα των καταχωρητών (Register Section) στο οποίο, όπως υποδηλώνει και η ονομασία του, περιλαμβάνει μία σειρά από καταχωρητές και ένα μηχανισμό επικοινωνίας μεταξύ τους (εσωτερικός δίαυλος δεδομένων).
- Το τμήμα της αριθμητικής και λογικής μονάδας (Arithmetic Logic Unit - ALU), το οποίο εκτελεί αριθμητικές πράξεις όπως πρόσθεση και λογικές διεργασίες όπως AND,OR κτλ. Λαμβάνει τελεστές από το τμήμα καταχωρητών της ΚΜΕ, εκτελεί τις ανάλογες λογικές ή αριθμητικές πράξεις και αποθηκεύει τα αποτελέσματα στο τμήμα καταχωρητών.
- Το τμήμα της μονάδας ελέγχου (Control Unit) το οποίο είναι υπεύθυνο για τον έλεγχο κάθε λειτουργίας του επεξεργαστή. Η μονάδα ελέγχου λαμβάνει κάποιες τιμές δεδομένων από το τμήμα των καταχωρητών, τις οποίες χρησιμοποιεί για να δημιουργήσει τα σήματα ελέγχου. Ο ρόλος της μονάδας ελέγχου είναι να τοποθετήσει αυτά τα σήματα ελέγχου στη σωστή σειρά, ώστε η CPU αλλά και το υπόλοιπο του υπολογιστή να εκτελέσουν τις διεργασίες που απαιτούνται για την σωστή επεξεργασία των εντολών και διακοπών.

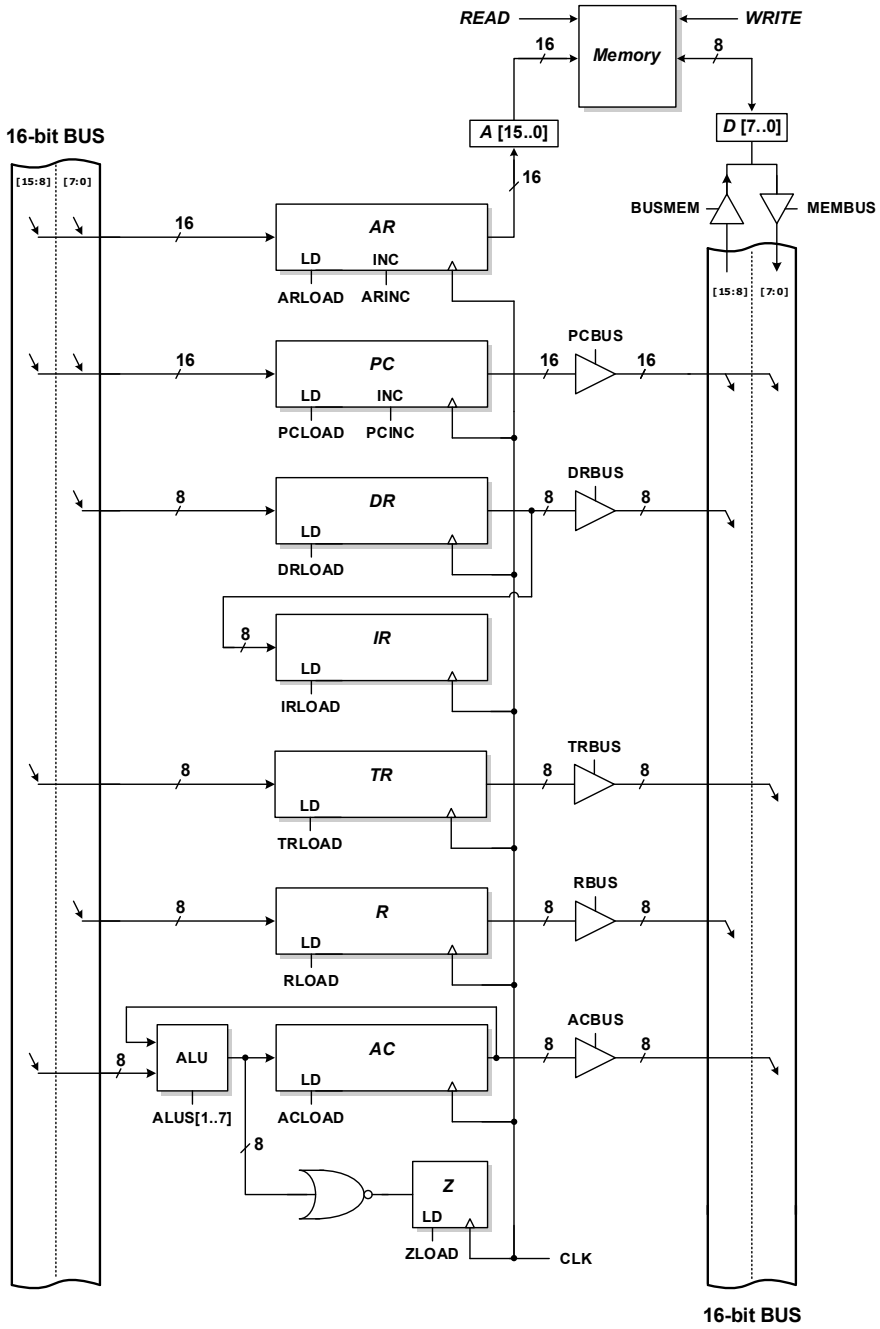
Το σύνολο των καταχωρητών αποτελεί μέρος της ISA μιας ΚΜΕ, αφού βάσει αυτών καθορίζεται ο τρόπος εκτέλεσης κάθε εντολής. Για την περίπτωση της σχετικά απλής ΚΜΕ, οι διαθέσιμοι για τη σχεδίαση και υλοποίησή της καταχωρητές είναι οι εξής :

- Ένας καταχωρητής διεύθυνσης των16-bits (Address Register - AR)
- Ένας απαριθμητής προγράμματος των16-bits (Program Counter - PC)
- Ένας καταχωρητής δεδομένων των 8-bits (Data Register – DR)
- Ένας καταχωρητής εντολών των 8-bits (Instruction Register - IR)
- Ένας συσσωρευτής των 8-bits (Accumulator - AC)
- Ένας προσωρινός καταχωρητής των 8-bits (Temporary Register - TR)
- Ένας καταχωρητής σημαίας του 1-bit (Flag register - Z)
- Ένας καταχωρητής γενικού σκοπού των 8-bits (Register - R)

Ο καταχωρητής διεύθυνσης (AR) είναι εύρους 16-bits και είναι αυτός που παρέχει τη διεύθυνση μνήμης από την οποία θα γίνει η ανάγνωση μιας εντολής ή ενός δεδομένου. Ομοίως ο απαριθμητής προγράμματος (PC) είναι εύρους 16-bits και είναι αυτός που παρέχει την διεύθυνση της επόμενης προς εκτέλεση εντολής στον καταχωρητή διεύθυνσης. Ο καταχωρητής δεδομένων (DR) έχει εύρος 8-bits και εκτός από το να δέχεται δεδομένα από τη μνήμη (DR<-M), μπορεί και να γράψει δεδομένα σε αυτήν (M<-DR). Επιπρόσθετα, ο καταχωρητής εντολών (IR) έχει εύρος 8-bits, όπως και ο συσσωρευτής (AC), όπου ο μεν πρώτος χρησιμοποιείται για την αποθήκευση του τμήματος της

εντολής που αφορά την επιλογή ενός από τους δεκαέξι (16) διαθέσιμους κύκλους εκτέλεσης (execution cycle), ενώ ο δεύτερος χρησιμοποιείται όπως είναι γνωστό για την καταχώρηση του αποτελέσματος μιας διεργασίας στην ALU.

Το συνολικό data path όπου απεικονίζονται οι απαραίτητες διασυνδέσεις των καταχωρητών, της εξωτερικής μνήμης, των απομονωτών , της ALU και ο αριθμός των bits σε κάθε δίαυλο φαίνονται στο σχήμα 1. Για λόγους απλότητας έχει παραληφθεί το τμήμα της μονάδας ελέγχου η οποία παρέχει όλα τα απαραίτητα σήματα ελέγχου.



Σχήμα 1: *Συνολικό data path σχετικά απλής ΜΚΕ.*

Κύκλωμα παραγωγής σημάτων ALU

Όπως έχει αναφερθεί ο τμήμα της μονάδας ελέγχου παρέχει όλα τα απαραίτητα σήματα ελέγχου απευθείας στα επιμέρους στοιχεία της ΚΜΕ με μοναδική εξαίρεση την ALU. Για την σωστή λειτουργεία της ALU είναι απαραίτητη παραγωγή των σημάτων ελέγχου alus[0..7] από τα σήματα που παράγει η μονάδα ελέγχου. Το κύκλωμα αυτό δίνεται στο πρόγραμμα 1 που ακολουθεί.

```
library ieee ;
use ieee.std_logic_1164.all ;

entity alus IS
port(rbus,acload,zload,andop      : in std_logic;
     orop,notop,xorop,aczero      : in std_logic;
     acinc,plus,minus,drbus       : in std_logic;
     alus                         : out std_logic_vector(6 downto 0));
end alus ;

architecture arc of alus is
signal control : std_logic_vector(11 downto 0);
begin
  control <= rbus & drbus & acload & zload & andop & orop
          & notop & xorop & aczero & acinc & plus & minus ;
  process(control)
  begin
    case control is
      WHEN "101110000000" => alus <= "1000000" ; -- AND
      WHEN "101101000000" => alus <= "1100000" ; -- OR
      WHEN "001100100000" => alus <= "1110000" ; -- NOT
      WHEN "101100010000" => alus <= "1010000" ; -- XOR
      WHEN "001100001000" => alus <= "0000000" ; -- CLAC
      WHEN "001100000100" => alus <= "0001001" ; -- INAC
      WHEN "101100000010" => alus <= "0000101" ; -- ADD
      WHEN "101100000001" => alus <= "0001011" ; -- SUB
      WHEN "101100000000" => alus <= "0000100" ; -- MOVR
      WHEN "011100000000" => alus <= "0000100" ; -- LDAC5
      WHEN others => alus <= "1111111" ; -- NO OPERATION
    end case;
  end process;
end arc ;
```

Πρόγραμμα 1: *Κύκλωμα παραγωγής σημάτων ελέγχου ALU.*

Δίαυλος Δεδομένων

Ο δίαυλος δεδομένων, σε συνδυασμό με τη διάταξη των απομονωτών (buffers), παίζει σημαντικό ρόλο σε μια ΚΜΕ, αφού η λειτουργία της εξαρτάται από τη σωστή οργάνωση της διακίνησης των δεδομένων στο εσωτερικό της. Με εξαίρεση τους καταχωρητές IR και Z, όλοι οι υπόλοιποι δέχονται δεδομένα μέσω του διαύλου, ενώ πέντε (5) από αυτούς παρέχουν το περιεχόμενό τους σε αυτόν, μέσω απομονωτών. Παράλληλα, δυνατότητα αποστολής και λήψης δεδομένων μέσω του διαύλου έχει και η μνήμη.

Γράψτε τον κώδικα για τον δίαυλο δεδομένων με τους απομονωτές όπως προκύπτει από το σχήμα 1.

Γράψτε εδώ το πρόγραμμά σας:

```
library ieee;
use ieee.std_logic_1164.all;

entity data_bus is
   port (

      AC_in, DR_in, R_in, TR_in, Mem_in : in std_logic_vector(7 downto 0);
      PC_in : in std_logic_vector(15 downto 0);
      AC_en, DR_en, R_en, TR_en, Mem_en, PC_low_en, PC_high_en : in std_logic;
      bus_out : out std_logic_vector(7 downto 0)
   );
end data_bus;



architecture behavioral of data_bus is
begin

   bus_out <= AC_in when AC_en = '1' else (others => 'Z');
   bus_out <= DR_in when DR_en = '1' else (others => 'Z');
   bus_out <= R_in when R_en = '1' else (others => 'Z');
   bus_out <= TR_in when TR_en = '1' else (others => 'Z');
   bus_out <= Mem_in when Mem_en = '1' else (others => 'Z');
   bus_out <= PC_in(7 downto 0) when PC_low_en = '1' else (others => 'Z');
   bus_out <= PC_in(15 downto 8) when PC_high_en = '1' else (others => 'Z');
end behavioral;
```

Πρόγραμμα 2: *Ο δίαυλος δεδομένων.*

Η Εξωτερική Μνήμη

Το τμήμα που ολοκληρώνει την υλοποίηση της σχετικά απλής ΚΜΕ σε VHDL, είναι η εξωτερική μνήμη (external memory) ή απλά μνήμη η οποία αποτελεί την πηγή παροχής δεδομένων και εντολών για τη λειτουργία της ΜΚΕ δηλαδή περιέχει τα προγράμματα εφαρμογών .

Η υλοποίηση της εξωτερικής μνήμης σε VHDL, θα γίνει μέσω του δομικού στοιχείου του Quartus , RAM: 1-PORT με τη βοήθεια του Megafunction Wizard του Quartus.

Με τη βοήθεια οποιουδήποτε Editor συντάξτε και αποθηκεύστε το πρόγραμμα 3 που ακολουθεί με όνομα "extRAM.mif". Το αρχείο αυτό θα χρησιμοποιηθεί για την αρχικοποίηση της μνήμης μικροκώδικα(μεταβλητή lpm_file) που θα δημιουργήσουμε στο επόμενο βήμα και περιέχει το σύνολο των εντολών του προγράμματος που θα εκτελέσει η ΚΜΕ.

```
WIDTH=8;
DEPTH=256;

ADDRESS_RADIX=DEC;
DATA_RADIX=BIN;

CONTENT BEGIN
        0   :   00000001; -- LDAC
        1   :   00101000; -- 28H
        2   :   00000000; -- 00H
        3   :   00000010; -- STAC
        4   :   00101011; -- 2BH
        5   :   00000000; -- 00H
        6   :   00000011; -- MVAC
        7   :   00001010; -- INAC
        8   :   00001111; -- NOT
        9   :   00001110; -- XOR
        10  :   00001000; -- ADD
        11  :   00001001; -- SUB
        12  :   00000100; -- MOVR
        13  :   00000101; -- JUMP
```

```
        14  :   00011000; -- 18H
        [15..23]  :   00000000;
        24  :   00000111; -- JPNZ
        25  :   00100000; -- 20H
        26  :   00000000; -- 00H
        27  :   00001111; -- NOT
        28  :   00000111; -- JPNZ
        [29..31]  :   00000000;
        32  :   00001011; -- CLAC
        33  :   00000110; -- JMPZ
        34  :   00011000; -- 18H
        [35..39]  :   00000000;
        40  :   01111010; -- 7AH
        [41..255]  :   00000000;
END;
```

Πρόγραμμα 3: *Περιεχόμενα μνήμης*

Ακολουθώντας τη διαδικασία που έχει περιγραφεί στην άσκηση 3 δημιουργήστε μια μνήμη RAM – 1PORT με 256 θέσεις , εύρους 8-bits η κάθε μία,

Γράψτε εδώ το πρόγραμμά σας:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY altera_mf;
USE altera_mf.altera_mf_components.all;

ENTITY extRAM IS
PORT
(
address                 : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
clock           : IN STD_LOGIC  := '1';
data            : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
wren            : IN STD_LOGIC ;
q               : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
);
END extRAM;


ARCHITECTURE SYN OF extram IS

SIGNAL sub_wire0        : STD_LOGIC_VECTOR (7 DOWNTO 0);

BEGIN
q    <= sub_wire0(7 DOWNTO 0);

altsyncram_component : altsyncram
GENERIC MAP (
clock_enable_input_a => "BYPASS",
```

```
clock_enable_output_a => "BYPASS",
init_file => "extRAM.mif",
intended_device_family => "MAX 10",
lpm_hint => "ENABLE_RUNTIME_MOD=NO",
lpm_type => "altsyncram",
numwords_a => 256,
operation_mode => "SINGLE_PORT",
outdata_aclr_a => "NONE",
outdata_reg_a => "CLOCK0",
power_up_uninitialized => "FALSE",
read_during_write_mode_port_a => "NEW_DATA_NO_NBE_READ",
widthad_a => 8,
width_a => 8,
width_byteena_a => 1
)
PORT MAP (
address_a => address,
clock0 => clock,
data_a => data,
wren_a => wren,
q_a => sub_wire0
);



END SYN;
```

Πρόγραμμα 4: *Η εξωτερική μνήμη.*

Συνολική Υλοποίηση ΚΜΕ

Έχοντας ολοκληρώσει την περιγραφή του κώδικα σε VHDL για κάθε ένα επιμέρους τμήμα της ΜΚΕ, και αφού όλα συγκεντρωθούν σε μία βιβλιοθήκη, μπορεί πλέον να γίνει η διασύνδεσή τους σε ένα κεντρικό πρόγραμμα.

Γράψτε τον κώδικα για τη βιβλιοθήκη (package), με το όνομα cpulib, η οποία θα περιέχει τα επιμέρους στοιχεία που συνθέτουν την ΚΜΕ.

Γράψτε εδώ το πρόγραμμά σας:

```
library ieee;
use ieee.std_logic_1164.all;

package cpulib is



component alus
      port (
         rbus, acload, zload, andop, orop, notop, xorop, aczero, acinc, plus, minus, drbus  : in std_logic;
         alus : out std_logic_vector(6 downto 0)
      );
end component;



component extRAM
      port (
         address : in std_logic_vector(7 downto 0);
         clock   : in std_logic;
         data    : in std_logic_vector(7 downto 0);
         wren    : in std_logic;
         q       : out std_logic_vector(7 downto 0)
      );
end component;
```

```
component control_unit
      port (
         clock, reset : in std_logic;
         IR          : in std_logic_vector(7 downto 0);
         Z           : in std_logic;
         mOP         : out std_logic_vector(26 downto 0)
      );
   end component;




component data_bus
      port (
         AC_in, DR_in, R_in, TR_in, Mem_in : in std_logic_vector(7 downto 0);
         PC_in                    : in std_logic_vector(15 downto 0);
         sel                      : in std_logic_vector(6 downto 0);
         bus_out                  : out std_logic_vector(7 downto 0)
      );
end component;


end cpulib;
```

Πρόγραμμα 5: *βιβλιοθήκη στοιχείων για την ΚΜΕ.*

Με βάση το σκελετό που ακολουθεί (πρόγραμμα 6) γράψτε τον κώδικα περιγραφής για την ΚΜΕ. Σαν εξωτερικά σήματα εκτός των clock και reset, να οριστούν τα δεδομένα των καταχωρητών (ARdata, PCdata, DRdata, ACdata, IRdata, RRdata και TRdata), η τιμή της σημαίας (ZRdata), τα

δεδομένα των διαύλων διευθύνσεων και δεδομένων (addressBus, dataBus) καθώς και το τμήμα των μικρολειτουργιών μOPs (mOP) με σκοπό τον έλεγχό τους σε κάθε παλμό.

***ΣΗΜΕΙΩΣΗ**: Χρησιμοποιήστε όποια μονάδα ελέγχου (microprogrammed ή hardwired) επιθυμείτε .*

Γράψτε εδώ το πρόγραμμά σας:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use work.cpulib.all;

entity rs_cpu is
   port (
      ARdata, PCdata  : buffer std_logic_vector(15 downto 0);
      DRdata, ACdata  : buffer std_logic_vector(7 downto 0);
      IRdata, TRdata  : buffer std_logic_vector(7 downto 0);
      RRdata        : buffer std_logic_vector(7 downto 0);
      ZRdata        : buffer std_logic;
      clock, reset    : in std_logic;
      mOP           : buffer std_logic_vector(26 downto 0);
      addressBus     : buffer std_logic_vector(15 downto 0);
      dataBus       : buffer std_logic_vector(7 downto 0)
   );
end rs_cpu;



architecture arc of rs_cpu is
   signal alu_ctrl_signals : std_logic_vector(6 downto 0);
   signal memory_out       : std_logic_vector(7 downto 0);
begin

CONTROL: control_unit port map (
      clock => clock,
```

```
    reset => reset,
    IR    => IRdata,
    Z     => ZRdata,
    mOP   => mOP
);
ALU_CONTROL_UNIT: alus port map (
    rbus   => mOP(0),  -- rbus
    acload => mOP(1),  -- acload
    zload  => mOP(2),  -- zload
    andop  => mOP(3),  -- andop
    orop   => mOP(4),  -- orop
    notop  => mOP(5),  -- notop
    xorop  => mOP(6),  -- xorop
    aczero => mOP(7),  -- aczero
    acinc  => mOP(8),  -- acinc
    plus   => mOP(9),  -- plus
    minus  => mOP(10), -- minus
    drbus  => mOP(11), -- drbus
    alus   => alu_ctrl_signals
);

RAM_INST: extRAM port map (
    address => ARdata(7 downto 0),
    clock   => clock,
    data    => dataBus,
    wren    => mOP(12),
    q       => memory_out
);
```

```
    dataBus <= ACdata     when mOP(13) = '1' else (others => 'Z');
    dataBus <= DRdata     when mOP(14) = '1' else (others => 'Z');
    dataBus <= RRdata     when mOP(15) = '1' else (others => 'Z');
    dataBus <= TRdata     when mOP(16) = '1' else (others => 'Z');
    dataBus <= memory_out when mOP(17) = '1' else (others => 'Z');
    dataBus <= PCdata(7 downto 0)  when mOP(18) = '1' else (others => 'Z');
    dataBus <= PCdata(15 downto 8) when mOP(19) = '1' else (others => 'Z');
    addressBus <= ARdata;



    process(clock, reset)
    begin
       if reset = '1' then
          ARdata <= (others => '0');
          PCdata <= (others => '0');
          DRdata <= (others => '0');
          ACdata <= (others => '0');
          IRdata <= (others => '0');
          TRdata <= (others => '0');
          RRdata <= (others => '0');
          ZRdata <= '0';

elsif rising_edge(clock) then
          if mOP(20) = '1' then ARdata <= PCdata; end if;      -- AR <- PC
          if mOP(21) = '1' then IRdata <= dataBus; end if;     -- IR <- Bus
          if mOP(22) = '1' then DRdata <= dataBus; end if;     -- DR <- Bus
          if mOP(23) = '1' then PCdata(7 downto 0) <= dataBus; end if;  -- PC Low byte
          if mOP(24) = '1' then PCdata(15 downto 8) <= dataBus; end if; -- PC High byte
```

```
        if mOP(25) = '1' then PCdata <= PCdata + 1; end if;          -- epomenh entolh
        if mOP(26) = '1' then TRdata <= dataBus; end if;



        if mOP(1) = '1' then -- acload
           case alu_ctrl_signals is
              when "1000000" => ACdata <= ACdata and DRdata;
              when "1100000" => ACdata <= ACdata or DRdata;
              when "1110000" => ACdata <= not ACdata;
              when "1010000" => ACdata <= ACdata xor DRdata;
              when "0000000" => ACdata <= (others => '0');
              when "0001001" => ACdata <= ACdata + 1;
              when "0000101" => ACdata <= ACdata + DRdata;
              when "0000111" => ACdata <= ACdata - DRdata;
              when "0000100" => ACdata <= RRdata;
              when others    => null;
           end case;
        end if;

        if mOP(2) = '1' then -- zload
           if ACdata = "00000000" then ZRdata <= '1';
           else ZRdata <= '0';
           end if;
        end if;
      end if;
   end process;

end arc;
```

Πρόγραμμα 6: *Συνολική περιγραφή της ΚΜΕ.*

Εξομοίωση της ΚΜΕ.

Το επόμενο στάδιο περιλαμβάνει την εξομοίωση της μονάδας ελέγχου με τον Waveform Editor με σκοπό τον έλεγχο της λειτουργίας της. Με οδηγό τις προηγούμενες ασκήσεις, δημιουργήστε ένα καινούργιο project και εξομοιώστε τη λειτουργία της ΚΜΕ.

Τοποθετήστε εδώ τις κυματομορφές σας:

Εικόνα 1: *Κυματομορφές εξομοίωσης της ΚΜΕ*

Δώστε την ανάλυση των περιεχομένων της εξωτερικής μνήμης που χρησιμοποιήσατε (πρόγραμμα 3) συμπληρώνοντας τον ακόλουθο πίνακα:

Α/Α Μνήμης	Περιεχόμενο(Hex)	Περιεχόμενο(Bin)	Εντολή	Λειτουργία
0	01			
1	28			
40	7A			
3	02			
4	2B			
6	03			
7	0A			
8	0F			
9	0E			
10	08			
11	09			
12	04			
13	05			
14	16			
22	07			
23	20			
32	0B			
33	06			
34	1C			
25	01			
28	07			

Δώστε τα συμπερασματά σας και τις παρατηρήσεις σας σχετικά με τη λειτουργία της σχετικά απλής CPU. Είναι τα αποτελέσματα της εξομοίωσης τα αναμενόμενα σύμφωνα με τον κώδικα της εξωτερικής μνήμης

Γράξτε εδώ τα σχόλια σας: