ΣΧΕΤΙΚΑ ΑΠΛΗ ΚΕΝΤΡΙΚΗ ΜΟΝΑΔΑ ΕΠΕΞΕΡΓΑΣΙΑΣ (ΜΚΕ)

5

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:

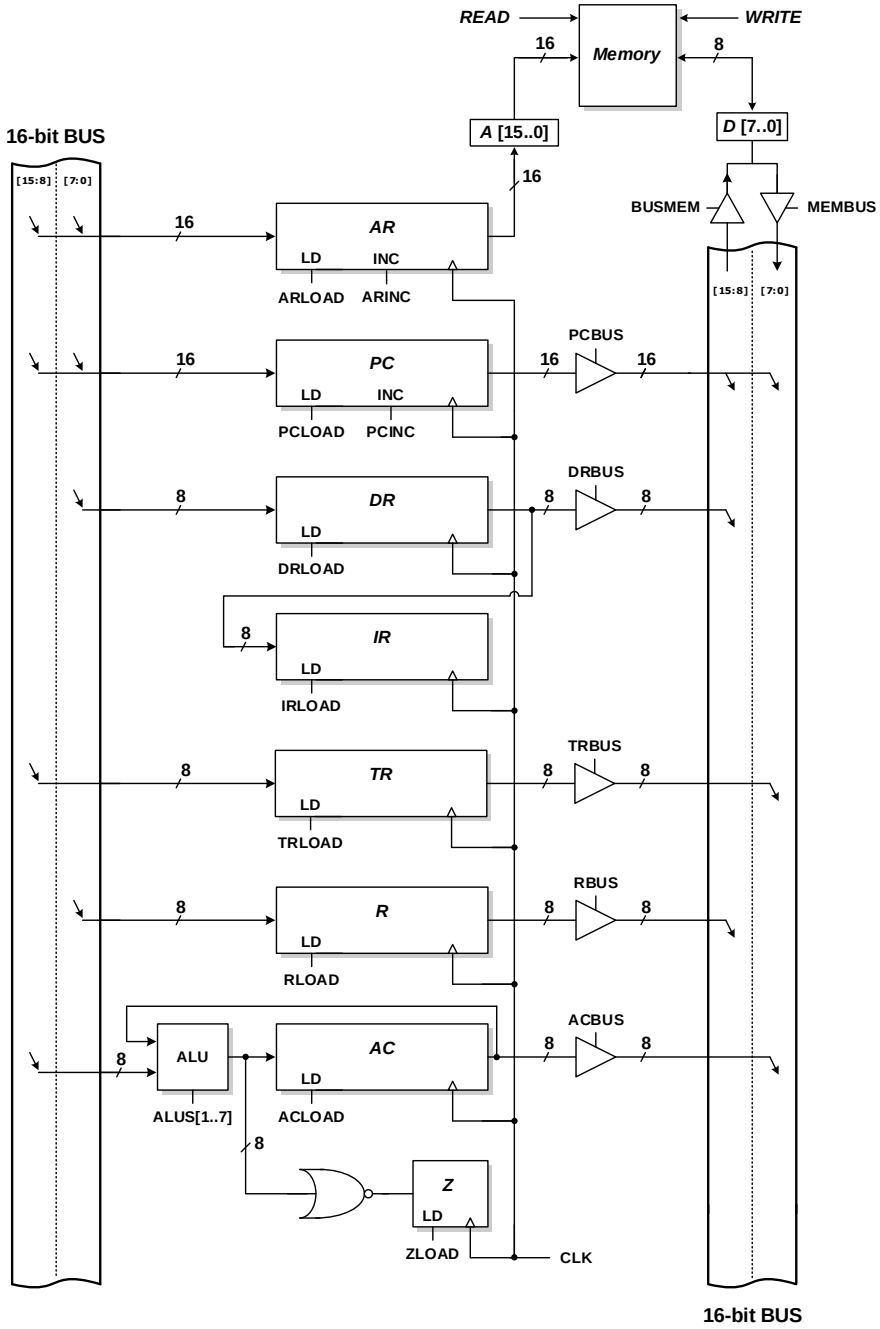
- Το τμήμα των καταχωρητών (Register Section) στο οποίο, όπως υποδηλώνει και η ονομασία του, περιλαμβάνει μία σειρά από καταχωρητές και ένα μηχανισμό επικοινωνίας μεταξύ τους (εσωτερικός δίαυλος δεδομένων).
- Το τμήμα της αριθμητικής και λογικής μονάδας (Arithmetic Logic Unit - ALU), το οποίο εκτελεί 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 (
  --bus to register (Inputs from Registers outputs)
  pc_out : IN  STD_LOGIC_VECTOR(15 DOWNTO 0);  --PC=16-bit
  dr_out : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);   --DR=8-bit
  tr_out : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);   --TR=8-bit
  r_out  : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);   --R=8-bit
  ac_out : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);   --AC=8-bit
  mem_out: IN  STD_LOGIC_VECTOR(7 DOWNTO 0);   --MEMory data=8-bit

  --buffers (Control Signals)
  pcbus   : IN  STD_LOGIC;--PC=1-bit
  drbus   : IN  STD_LOGIC;--DR=1-bit
  trbus   : IN  STD_LOGIC;--TR=1-bit
  rbus    : IN  STD_LOGIC;--R=1-bit
  acbus   : IN  STD_LOGIC;--AC=1-bit
  membus  : IN  STD_LOGIC;--MEMory buffer=1-bit
  busmem  : IN  STD_LOGIC; --control signal for memory write(Buffer BUS->MEM)

  --outputs
  dbus       : OUT STD_LOGIC_VECTOR(15 DOWNTO 0); --common buffer(bus)=16-bit
  mem_data_in : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)   --data to mem in
);
END data_bus;
```

```
ARCHITECTURE bhv OF data_bus IS
   SIGNAL internal_bus : STD_LOGIC_VECTOR(15 DOWNTO 0); --for bus reading
BEGIN

   PROCESS (pc_out, dr_out, tr_out, r_out, ac_out, mem_out,
         pcbus, drbus, trbus, rbus, acbus, membus)
   BEGIN

      internal_bus <= "ZZZZZZZZZZZZZZZZ"; --for no action(no btn press)--> high impedance(Z)->16-
bit

      --always assign the bottom 8-bits, otherwise the value changes

      IF (pcbus = '1') THEN --if pc-->needs all 16-bit bus
         internal_bus <= pc_out;



      ELSIF (drbus = '1') THEN --if dr--->since its 8-bit it needs only the bottom 8-bits
         internal_bus(15 DOWNTO 8) <= "00000000";--make the top 8-bits zero(0)
         internal_bus(7 DOWNTO 0)  <= dr_out;--bottoms take dr value



      ELSIF (trbus = '1') THEN  --same for tr
         internal_bus(15 DOWNTO 8) <= "00000000";
         internal_bus(7 DOWNTO 0)  <= tr_out;



      ELSIF (rbus = '1') THEN --same for r
         internal_bus(15 DOWNTO 8) <= "00000000";
```

```
        internal_bus(7 DOWNTO 0)  <= r_out;

      ELSIF (acbus = '1') THEN --same for ac
        internal_bus(15 DOWNTO 8) <= "00000000";
        internal_bus(7 DOWNTO 0)  <= ac_out;

      ELSIF (membus = '1') THEN --same for mem
        internal_bus(15 DOWNTO 8) <= "00000000";
        internal_bus(7 DOWNTO 0)  <= mem_out;

      END IF;

   END PROCESS;


   dbus <= internal_bus;
   mem_data_in <= internal_bus(7 DOWNTO 0) WHEN busmem = '1' ELSE (others => '0');

END bhv;
```

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:

```
-- megafunction wizard: %RAM: 1-PORT%
-- GENERATION: STANDARD
-- VERSION: WM1.0
-- MODULE: altsyncram


-- ============================================================
-- File Name: RAM.vhd
-- Megafunction Name(s):
--                         altsyncram
--
-- Simulation Library Files(s):
--
-- ============================================================
-- ************************************************************
-- THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
--
-- 20.1.1 Build 720 11/11/2020 SJ Lite Edition
-- ************************************************************



--Copyright (C) 2020  Intel Corporation. All rights reserved.
--Your use of Intel Corporation's design tools, logic functions
--and other software and tools, and any partner logic
--functions, and any output files from any of the foregoing
--(including device programming or simulation files), and any
--associated documentation or information are expressly subject
--to the terms and conditions of the Intel Program License
--Subscription Agreement, the Intel Quartus Prime License Agreement,
```

```
--the Intel FPGA IP License Agreement, or other applicable license
--agreement, including, without limitation, that your use is for
--the sole purpose of programming logic devices manufactured by
--Intel and sold by Intel or its authorized distributors.  Please
--refer to the applicable agreement for further details, at
--https://fpgasoftware.intel.com/eula.


LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY altera_mf;
USE altera_mf.altera_mf_components.all;

ENTITY RAM 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 RAM;


ARCHITECTURE SYN OF ram 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 => "UNREGISTERED",
		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;

-- ==

-- CNX file retrieval info

-- ==

-- Retrieval info: PRIVATE: ADDRESSSTALL_A NUMERIC "0"

-- Retrieval info: PRIVATE: AclrAddr NUMERIC "0"

-- Retrieval info: PRIVATE: AclrByte NUMERIC "0"

-- Retrieval info: PRIVATE: AclrData NUMERIC "0"

-- Retrieval info: PRIVATE: AclrOutput NUMERIC "0"

-- Retrieval info: PRIVATE: BYTE_ENABLE NUMERIC "0"

-- Retrieval info: PRIVATE: BYTE_SIZE NUMERIC "8"

-- Retrieval info: PRIVATE: BlankMemory NUMERIC "0"

-- Retrieval info: PRIVATE: CLOCK_ENABLE_INPUT_A NUMERIC "0"

-- Retrieval info: PRIVATE: CLOCK_ENABLE_OUTPUT_A NUMERIC "0"

-- Retrieval info: PRIVATE: Clken NUMERIC "0"

-- Retrieval info: PRIVATE: DataBusSeparated NUMERIC "1"

-- Retrieval info: PRIVATE: IMPLEMENT_IN_LES NUMERIC "0"

-- Retrieval info: PRIVATE: INIT_FILE_LAYOUT STRING "PORT_A"

-- Retrieval info: PRIVATE: INIT_TO_SIM_X NUMERIC "0"

-- Retrieval info: PRIVATE: INTENDED_DEVICE_FAMILY STRING "MAX 10"

-- Retrieval info: PRIVATE: JTAG_ENABLED NUMERIC "0"

-- Retrieval info: PRIVATE: JTAG_ID STRING "NONE"

-- Retrieval info: PRIVATE: MAXIMUM_DEPTH NUMERIC "0"

-- Retrieval info: PRIVATE: MIFfilename STRING "../../Users/btria/Documents/FPGA/LAB_5/extRAM.mif"

-- Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "256"

-- Retrieval info: PRIVATE: RAM_BLOCK_TYPE NUMERIC "0"

-- Retrieval info: PRIVATE: READ_DURING_WRITE_MODE_PORT_A NUMERIC "3"

-- Retrieval info: PRIVATE: RegAddr NUMERIC "1"

-- Retrieval info: PRIVATE: RegData NUMERIC "1"

-- Retrieval info: PRIVATE: RegOutput NUMERIC "1"

-- Retrieval info: PRIVATE: SYNTH_WRAPPER_GEN_POSTFIX STRING "0"

-- Retrieval info: PRIVATE: SingleClock NUMERIC "1"

-- Retrieval info: PRIVATE: UseDQRAM NUMERIC "1"

-- Retrieval info: PRIVATE: WRCONTROL_ACLR_A NUMERIC "0"

-- Retrieval info: PRIVATE: WidthAddr NUMERIC "8"

-- Retrieval info: PRIVATE: WidthData NUMERIC "8"

-- Retrieval info: PRIVATE: rden NUMERIC "0"

-- Retrieval info: LIBRARY: altera_mf altera_mf.altera_mf_components.all

-- Retrieval info: CONSTANT: CLOCK_ENABLE_INPUT_A STRING "BYPASS"

-- Retrieval info: CONSTANT: CLOCK_ENABLE_OUTPUT_A STRING "BYPASS"

-- Retrieval info: CONSTANT: INIT_FILE STRING "../../Users/btria/Documents/FPGA/LAB_5/extRAM.mif"

-- Retrieval info: CONSTANT: INTENDED_DEVICE_FAMILY STRING "MAX 10"

-- Retrieval info: CONSTANT: LPM_HINT STRING "ENABLE_RUNTIME_MOD=NO"

-- Retrieval info: CONSTANT: LPM_TYPE STRING "altsyncram"

-- Retrieval info: CONSTANT: NUMWORDS_A NUMERIC "256"

-- Retrieval info: CONSTANT: OPERATION_MODE STRING "SINGLE_PORT"

-- Retrieval info: CONSTANT: OUTDATA_ACLR_A STRING "NONE"

-- Retrieval info: CONSTANT: OUTDATA_REG_A STRING "CLOCK0"

-- Retrieval info: CONSTANT: POWER_UP_UNINITIALIZED STRING "FALSE"

-- Retrieval info: CONSTANT: READ_DURING_WRITE_MODE_PORT_A STRING "NEW_DATA_NO_NBE_READ"

-- Retrieval info: CONSTANT: WIDTHAD_A NUMERIC "8"

-- Retrieval info: CONSTANT: WIDTH_A NUMERIC "8"

-- Retrieval info: CONSTANT: WIDTH_BYTEENA_A NUMERIC "1"

-- Retrieval info: USED_PORT: address 0 0 8 0 INPUT NODEFVAL "address[7..0]"

-- Retrieval info: USED_PORT: clock 0 0 0 0 INPUT VCC "clock"

-- Retrieval info: USED_PORT: data 0 0 8 0 INPUT NODEFVAL "data[7..0]"

-- Retrieval info: USED_PORT: q 0 0 8 0 OUTPUT NODEFVAL "q[7..0]"

-- Retrieval info: USED_PORT: wren 0 0 0 0 INPUT NODEFVAL "wren"

-- Retrieval info: CONNECT: @address_a 0 0 8 0 address 0 0 8 0

-- Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0

-- Retrieval info: CONNECT: @data_a 0 0 8 0 data 0 0 8 0

-- Retrieval info: CONNECT: @wren_a 0 0 0 0 wren 0 0 0 0

-- Retrieval info: CONNECT: q 0 0 8 0 @q_a 0 0 8 0

-- Retrieval info: GEN_FILE: TYPE_NORMAL RAM.vhd TRUE

-- Retrieval info: GEN_FILE: TYPE_NORMAL RAM.inc FALSE

-- Retrieval info: GEN_FILE: TYPE_NORMAL RAM.cmp TRUE

-- Retrieval info: GEN_FILE: TYPE_NORMAL RAM.bsf FALSE

-- Retrieval info: GEN_FILE: TYPE_NORMAL RAM_inst.vhd FALSE

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:

Χρησιμοποιώ την microprocessed λογική για την συγκεκριμένη υλοποίηση CPU(προσπάθησα με hardwired αλλά μετά από αρκετό debugging δεν κατάφερα να την υλοποιήσω), τοποθετούμε στην

βιβλιοθήκη μόνο τα γενικά component, τα sub-component βρίσκονται ακόμη στις αντίστοιχες βιβλιοθήκες τους.

```
-- megafunction wizard: %RAM: 1-PORT%
-- GENERATION: STANDARD
-- VERSION: WM1.0
-- MODULE: altsyncram

-- ============================================================
-- File Name: RAM.vhd
-- Megafunction Name(s):
--                         altsyncram
--
-- Simulation Library Files(s):
--
-- ============================================================
-- ************************************************************
-- THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
--
-- 20.1.1 Build 720 11/11/2020 SJ Lite Edition
-- ************************************************************


--Copyright (C) 2020  Intel Corporation. All rights reserved.
--Your use of Intel Corporation's design tools, logic functions
--and other software and tools, and any partner logic
--functions, and any output files from any of the foregoing
--(including device programming or simulation files), and any
--associated documentation or information are expressly subject
--to the terms and conditions of the Intel Program License
```

```
--Subscription Agreement, the Intel Quartus Prime License Agreement,
--the Intel FPGA IP License Agreement, or other applicable license
--agreement, including, without limitation, that your use is for
--the sole purpose of programming logic devices manufactured by
--Intel and sold by Intel or its authorized distributors.  Please
--refer to the applicable agreement for further details, at
--https://fpgasoftware.intel.com/eula.


LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY altera_mf;
USE altera_mf.altera_mf_components.all;

ENTITY RAM 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 RAM;


ARCHITECTURE SYN OF ram 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 => "UNREGISTERED",
		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;

-- ============================================================
-- CNX file retrieval info
-- ============================================================
-- Retrieval info: PRIVATE: ADDRESSSTALL_A NUMERIC "0"
-- Retrieval info: PRIVATE: AclrAddr NUMERIC "0"
-- Retrieval info: PRIVATE: AclrByte NUMERIC "0"
-- Retrieval info: PRIVATE: AclrData NUMERIC "0"
-- Retrieval info: PRIVATE: AclrOutput NUMERIC "0"
-- Retrieval info: PRIVATE: BYTE_ENABLE NUMERIC "0"
-- Retrieval info: PRIVATE: BYTE_SIZE NUMERIC "8"
-- Retrieval info: PRIVATE: BlankMemory NUMERIC "0"
-- Retrieval info: PRIVATE: CLOCK_ENABLE_INPUT_A NUMERIC "0"
-- Retrieval info: PRIVATE: CLOCK_ENABLE_OUTPUT_A NUMERIC "0"
-- Retrieval info: PRIVATE: Clken NUMERIC "0"
-- Retrieval info: PRIVATE: DataBusSeparated NUMERIC "1"
-- Retrieval info: PRIVATE: IMPLEMENT_IN_LES NUMERIC "0"
-- Retrieval info: PRIVATE: INIT_FILE_LAYOUT STRING "PORT_A"
-- Retrieval info: PRIVATE: INIT_TO_SIM_X NUMERIC "0"
-- Retrieval info: PRIVATE: INTENDED_DEVICE_FAMILY STRING "MAX 10"
-- Retrieval info: PRIVATE: JTAG_ENABLED NUMERIC "0"
-- Retrieval info: PRIVATE: JTAG_ID STRING "NONE"
-- Retrieval info: PRIVATE: MAXIMUM_DEPTH NUMERIC "0"
-- Retrieval info: PRIVATE: MIFfilename STRING "../../Users/btria/Documents/FPGA/LAB_5/extRAM.mif"
```

-- Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "256"

-- Retrieval info: PRIVATE: RAM_BLOCK_TYPE NUMERIC "0"

-- Retrieval info: PRIVATE: READ_DURING_WRITE_MODE_PORT_A NUMERIC "3"

-- Retrieval info: PRIVATE: RegAddr NUMERIC "1"

-- Retrieval info: PRIVATE: RegData NUMERIC "1"

-- Retrieval info: PRIVATE: RegOutput NUMERIC "1"

-- Retrieval info: PRIVATE: SYNTH_WRAPPER_GEN_POSTFIX STRING "0"

-- Retrieval info: PRIVATE: SingleClock NUMERIC "1"

-- Retrieval info: PRIVATE: UseDQRAM NUMERIC "1"

-- Retrieval info: PRIVATE: WRCONTROL_ACLR_A NUMERIC "0"

-- Retrieval info: PRIVATE: WidthAddr NUMERIC "8"

-- Retrieval info: PRIVATE: WidthData NUMERIC "8"

-- Retrieval info: PRIVATE: rden NUMERIC "0"

-- Retrieval info: LIBRARY: altera_mf altera_mf.altera_mf_components.all

-- Retrieval info: CONSTANT: CLOCK_ENABLE_INPUT_A STRING "BYPASS"

-- Retrieval info: CONSTANT: CLOCK_ENABLE_OUTPUT_A STRING "BYPASS"

-- Retrieval info: CONSTANT: INIT_FILE STRING "../../Users/btria/Documents/FPGA/LAB_5/extRAM.mif"

-- Retrieval info: CONSTANT: INTENDED_DEVICE_FAMILY STRING "MAX 10"

-- Retrieval info: CONSTANT: LPM_HINT STRING "ENABLE_RUNTIME_MOD=NO"

-- Retrieval info: CONSTANT: LPM_TYPE STRING "altsyncram"

-- Retrieval info: CONSTANT: NUMWORDS_A NUMERIC "256"

-- Retrieval info: CONSTANT: OPERATION_MODE STRING "SINGLE_PORT"

-- Retrieval info: CONSTANT: OUTDATA_ACLR_A STRING "NONE"

-- Retrieval info: CONSTANT: OUTDATA_REG_A STRING "CLOCK0"

-- Retrieval info: CONSTANT: POWER_UP_UNINITIALIZED STRING "FALSE"

-- Retrieval info: CONSTANT: READ_DURING_WRITE_MODE_PORT_A STRING "NEW_DATA_NO_NBE_READ"

-- Retrieval info: CONSTANT: WIDTHAD_A NUMERIC "8"

-- Retrieval info: CONSTANT: WIDTH_A NUMERIC "8"

-- Retrieval info: CONSTANT: WIDTH_BYTEENA_A NUMERIC "1"

-- Retrieval info: USED_PORT: address 0 0 8 0 INPUT NODEFVAL "address[7..0]"

-- Retrieval info: USED_PORT: clock 0 0 0 0 INPUT VCC "clock"

-- Retrieval info: USED_PORT: data 0 0 8 0 INPUT NODEFVAL "data[7..0]"

-- Retrieval info: USED_PORT: q 0 0 8 0 OUTPUT NODEFVAL "q[7..0]"

-- Retrieval info: USED_PORT: wren 0 0 0 0 INPUT NODEFVAL "wren"

-- Retrieval info: CONNECT: @address_a 0 0 8 0 address 0 0 8 0

-- Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0

-- Retrieval info: CONNECT: @data_a 0 0 8 0 data 0 0 8 0

-- Retrieval info: CONNECT: @wren_a 0 0 0 0 wren 0 0 0 0

-- Retrieval info: CONNECT: q 0 0 8 0 @q_a 0 0 8 0

-- Retrieval info: GEN_FILE: TYPE_NORMAL RAM.vhd TRUE

-- Retrieval info: GEN_FILE: TYPE_NORMAL RAM.inc FALSE

-- Retrieval info: GEN_FILE: TYPE_NORMAL RAM.cmp TRUE

-- Retrieval info: GEN_FILE: TYPE_NORMAL RAM.bsf FALSE

-- Retrieval info: GEN_FILE: TYPE_NORMAL RAM_inst.vhd FALSE

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 internal_bus : std_logic_vector(15 downto 0);
   signal ram_q_out      : std_logic_vector(7 downto 0);
   signal mem_data_to_ram : std_logic_vector(7 downto 0);
   signal alu_result : std_logic_vector(7 downto 0);
   signal alu_z_flag : std_logic;
```

```
    signal alus_ctrl  : std_logic_vector(6 downto 0);

begin

    -- 1. CONTROL UNIT
    CONTROL_UNIT: mseq
        PORT MAP (
            ir       => IRdata(3 downto 0),
            clock    => clock,
            reset    => reset,
            z        => ZRdata,
            code     => open,
            debug_reg => open,
            mOPs     => mOP
        );

    -- 2. ALU CONTROL
    --με βάση τον πίνακα στην εργασία 3
    ALU_CONTROLLER: alus
        PORT MAP (
            andop  => mOP(7),
            orop   => mOP(6),
            xorop  => mOP(5),
            notop  => mOP(4),
            acinc  => mOP(3),
            aczero => mOP(2),
            plus   => mOP(1),
            minus  => mOP(0),
```

```
      -- Βοηθητικά σήματα ALU (LOADS/BUSES που επηρεάζουν την ALU)
      rbus   => mOP(9),  -- RBUS (Bit 9)
      acload => mOP(18), -- ACLOAD (Bit 18)
      zload  => mOP(17), -- ZLOAD (Bit 17)
      drbus  => mOP(11), -- DRBUS (Bit 11)

      alus   => alus_ctrl
   );

-- 3. BUS SYSTEM
-- Αντιστοίχιση βάσει εικόνας:
-- PCBUS(12), DRBUS(11), TRBUS(10), RBUS(9), ACBUS(8)
-- MEMBUS(14 - Read Buffer), BUSMEM(13 - Write Buffer)
BUS_SYSTEM: data_bus
   PORT MAP (
      pc_out  => PCdata, dr_out  => DRdata,
      tr_out  => TRdata, r_out   => RRdata,
      ac_out  => ACdata, mem_out => ram_q_out,

      pcbus   => mOP(12),
      drbus   => mOP(11),
      trbus   => mOP(10),
      rbus    => mOP(9),
      acbus   => mOP(8),
      membus  => mOP(14),
      busmem  => mOP(13),

      dbus       => internal_bus,
      mem_data_in => mem_data_to_ram
```

```
   );

-- 4. REGISTERS

-- PC: PCLOAD(24), PCINC(23)
REG_PC: regnbit GENERIC MAP (n => 16)
   PORT MAP (clk => clock, rst => reset, ld => mOP(24), inc => mOP(23),
         din => internal_bus, dout => PCdata);

-- AR: ARLOAD(26), ARINC(25)
REG_AR: regnbit GENERIC MAP (n => 16)
   PORT MAP (clk => clock, rst => reset, ld => mOP(26), inc => mOP(25),
         din => internal_bus, dout => ARdata);

-- DR: DRLOAD(22)
REG_DR: regnbit GENERIC MAP (n => 8)
   PORT MAP (clk => clock, rst => reset, ld => mOP(22), inc => '0',
         din => internal_bus(7 downto 0), dout => DRdata);

-- IR: IRLOAD(20)
REG_IR: regnbit GENERIC MAP (n => 8)
   PORT MAP (clk => clock, rst => reset, ld => mOP(20), inc => '0',
         din => internal_bus(7 downto 0), dout => IRdata);

-- TR: TRLOAD(21)
REG_TR: regnbit GENERIC MAP (n => 8)
   PORT MAP (clk => clock, rst => reset, ld => mOP(21), inc => '0',
         din => internal_bus(7 downto 0), dout => TRdata);
```

```
-- R: RLOAD(19)
REG_R: regnbit GENERIC MAP (n => 8)
   PORT MAP (clk => clock, rst => reset, ld => mOP(19), inc => '0',
         din => internal_bus(7 downto 0), dout => RRdata);

-- AC: ACLOAD(18)
REG_AC: regnbit GENERIC MAP (n => 8)
   PORT MAP (clk => clock, rst => reset, ld => mOP(18), inc => '0',
         din => alu_result, dout => ACdata);

-- Z: ZLOAD(17)
REG_Z: regnbit GENERIC MAP (n => 1)
   PORT MAP (clk => clock, rst => reset, ld => mOP(17), inc => '0',
         din(0) => alu_z_flag, dout(0) => ZRdata);

-- 5. MEMORY
-- WRITE signal is mOP(15)
MEMORY_UNIT: RAM
   PORT MAP (
      clock   => clock,
      address => ARdata(7 downto 0),
      data    => mem_data_to_ram,
      wren    => mOP(15), -- WRITE SIGNAL
      q       => ram_q_out
   );

ALU_UNIT: alu GENERIC MAP (n => 8)
   PORT MAP (
      ac    => ACdata,
```

```
        db    => internal_bus(7 downto 0),
        alus  => alus_ctrl,
        dout  => alu_result,
        z_out => alu_z_flag
    );

  -- 6. OUTPUTS
  addressBus <= ARdata;
  dataBus    <= internal_bus(7 downto 0);

end arc;
```

Πρόγραμμα 6: *Συνολική περιγραφή της ΚΜΕ.*

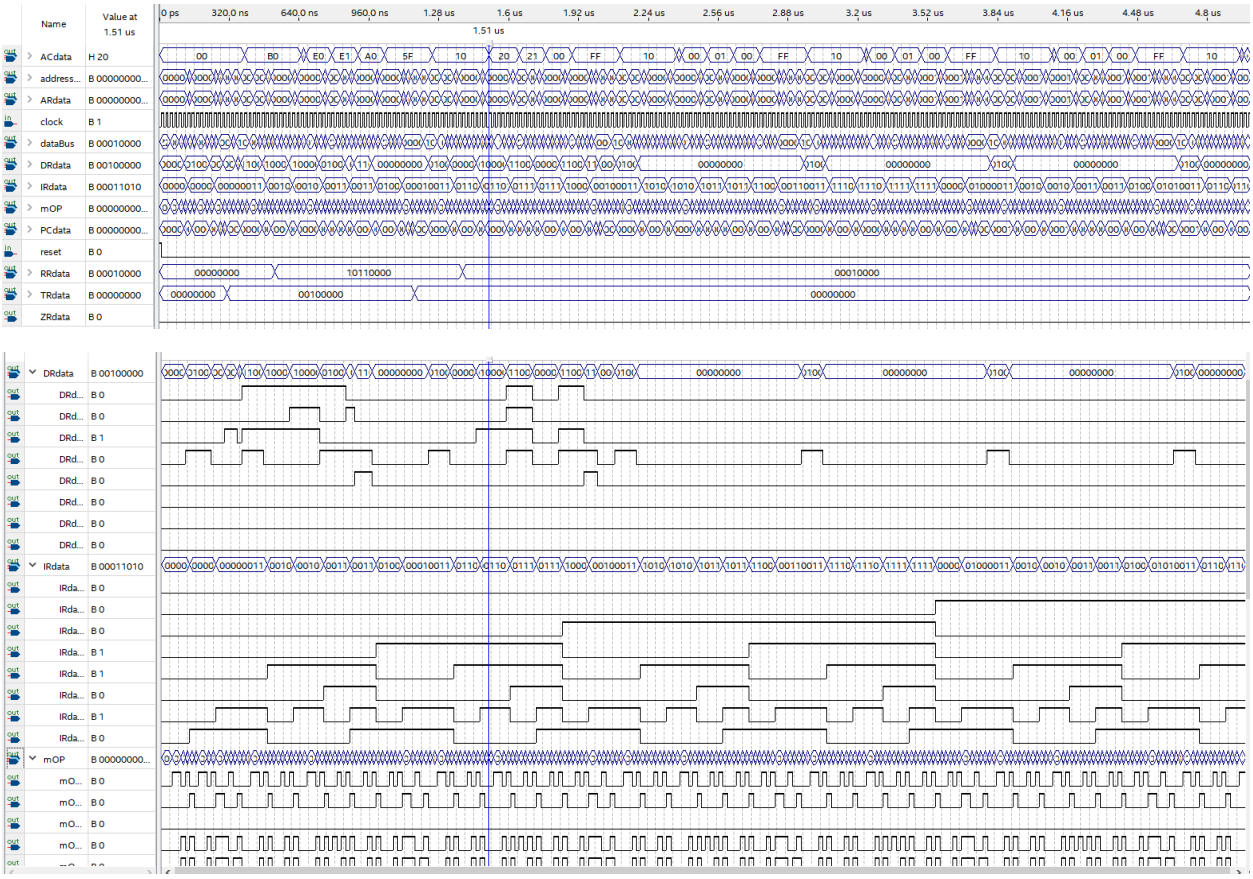
Εξομοίωση της ΚΜΕ.

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:

Η ορθή λειτουργία της CPU επιβεβαιώθηκε πλήρως μέσω της εξομοίωσης. Το βασικό αποδεικτικό στοιχείο είναι ότι ο καταχωρητής AC αλλάζει τιμές δυναμικά B0- E0- E1), ακολουθώντας ακριβώς το πρόγραμμα που φορτώσαμε.

Συγκεκριμένα:



1. **Fetch & Decode:** Η αρχική αλλαγή από 0 σε B0 δείχνει ότι η CPU διαβάζει και αναγνωρίζει σωστά τις εντολές από τη μνήμη, χωρίς να μένει κολλημένη.
2. **ALU:** Η μετάβαση E0 - E1αποδεικνύει ότι η εντολή INAC εκτελέστηκε σωστά, άρα η ALU και το κύκλωμα πρόσθεσης λειτουργούν και ενημερώνουν τον AC.
3. **Μνήμη & Δίαυλος:** Το γεγονός ότι βλέπουμε δεδομένα που υπάρχουν στο αρχείο .mif της μνήμης, σημαίνει ότι η μεταφορά δεδομένων μέσω του διαύλου (Bus) γίνεται σωστά, χωρίς συγκρούσεις.

Συμπερασματικά, η ομαλή ροή του προγράμματος επιβεβαιώνει ότι τα σήματα ελέγχου (mOPs) έχουν συνδεθεί σωστά και ο μικροκώδικας συγχρονίζει άψογα όλα τα τμήματα του επεξεργαστή.

Εικόνα 1: *Κυματομορφές εξομοίωσης της ΚΜΕ*

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

Για να συμπληρώσω τον πίνακα, έκανα τρία απλά βήματα για κάθε γραμμή:

Hex σε Binary: Μετέτρεψα τον δεκαεξαδικό αριθμό (Hex) σε δυαδικό (Binary).

Π.χ. Το 0 είναι 0000 και το 1 είναι 0001. Άρα το 01 γίνεται 00000001.

Αναγνώριση Εντολής (Opcode): Κοίταξα τον κωδικό (το Hex νούμερο) και είδα σε ποια εντολή αντιστοιχεί με βάση τη θεωρία της CPU μας:

Όπου είδα 01 έγραψα **LDAC**.

Όπου είδα 08 έγραψα **ADD**.

Όπου είδα 05 έγραψα **JUMP**, κλπ.

Εντοπισμός Δεδομένων (Operand): Κάποιες εντολές (όπως η LDAC, STAC, JUMP) πιάνουν **δύο θέσεις** μνήμης.

Η 1η θέση είναι η εντολή (π.χ. Φόρτωσε).

Η 2η θέση είναι η *διεύθυνση* (π.χ. Από πού;).

Γι' αυτό, κάτω από το 01 (LDAC), το 28 δεν είναι εντολή, αλλά **address**.

Α/Α Μνήμης	Περιεχόμενο(Hex)	Περιεχόμενο(Bin)	Εντολή	Λειτουργία
0	01	00000001	LDAC	Φόρτωσε στον AC
1	28	00101000	ADDRESS	το περιεχόμενο της δνσης 28H
40	7A	01111010	DATA	(Αριθμός 122) - Είναι δεδομένο, όχι εντολή
3	02	00000010	STAC	Αποθήκευσε τον AC
4	2B	00101011	ADDRESS	στη διεύθυνση μνήμης 2BH
6	03	00000011	MVAC	Αντίγραψε τον AC στον R
7	0A	00001010	INAC	Αύξησε τον AC κατά 1 (AC = AC + 1)
8	0F	00001111	NOT	Αντέστρεψε τα bits του AC (NOT)
9	0E	00001110	XOR	Κάνε XOR μεταξύ AC και R
10	08	00001000	ADD	Πρόσθεσε: AC = AC + R
11	09	00001001	SUB	Αφαίρεσε: AC = AC - R
12	04	00000100	MOVR	Φέρε πίσω τον αριθμό από τον R στον AC
13	05	00000101	JUMP	Πήγαινε (κάνε άλμα)
14	16	00010110	ADDRESS	στη διεύθυνση 16H
22	07	00000111	JPNZ	Πήγαινε σε άλλη εντολή αν το αποτέλεσμα ΔΕΝ είναι 0
23	20	00100000	ADDRESS	...στη διεύθυνση 20H

32	0B	00001011	CLAC	Μηδένισε τον AC (γίνεται 0)
33	06	00000110	JMPZ	Πήγαινε σε άλλη εντολή αν το αποτέλεσμα ΕΙΝΑΙ 0
34	1C	00011100	ADDRESS	στη διεύθυνση 1CH
25	01	00000001	LDAC	Φόρτωσε στον AC
28	07	00000111	DATA	(Αριθμός 7) - Αυτό διαβάζει η εντολή στη θέση 0

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:

Η εξομοίωση της σχεδίασης της CPU είναι επιτυχής, καθώς τα αποτελέσματα των κυματομορφών ταυτίζονται πλήρως με την αναμενόμενη ροή του προγράμματος μηχανής που παρατέθηκε στον πίνακα. Αρχικά, επιβεβαιώθηκε η ορθότητα του Κύκλου Εντολής (Fetch-Decode-Execute), με τον Program Counter (PC) να διαχειρίζεται σωστά τόσο τις single bit όσο και τις two bit εντολές, χωρίς σφάλματα χρονισμού. Η ακεραιότητα του Data Path και η επικοινωνία με την εξωτερική μνήμη επαληθεύτηκαν μέσω της αρχικής εντολής LDAC, η οποία φόρτωσε επιτυχώς στον Συσσωρευτή (AC) την τιμή 07 από τη διεύθυνση 28H, αποδεικνύοντας τη σωστή λειτουργία του διαύλου δεδομένων. Παράλληλα, η συνεχής μεταβολή των τιμών του AC αποδεικνύει την ορθή λειτουργία της ALU στην εκτέλεση αριθμητικών και λογικών πράξεων (όπως INAC, ADD, XOR), καθώς τα σήματα ελέγχου από τη μικροπρογραμματιζόμενη μονάδα ενεργοποιούν τα κατάλληλα κυκλώματα. Τέλος, η επιτυχής εκτέλεση των εντολών διακλάδωσης (JUMP, JPNZ) επιβεβαιώνει τη δυνατότητα της CPU να τροποποιεί τη ροή του προγράμματος βάσει συνθηκών, ολοκληρώνοντας την εικόνα ενός πλήρως λειτουργικού συστήματος. Βλέπουμε πως η microprocessed λογική αποτελεί σε εισαγωγικά μια πιο εύκολη και διαχειρήσιμη μέθοδος υλοποίησης σε σχέση με την hardwired.