

ΣΧΕΤΙΚΑ ΑΠΛΗ ΚΕΝΤΡΙΚΗ ΜΟΝΑΔΑ ΕΠΕΞΕΡΓΑΣΙΑΣ (ΚΜΕ)

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.

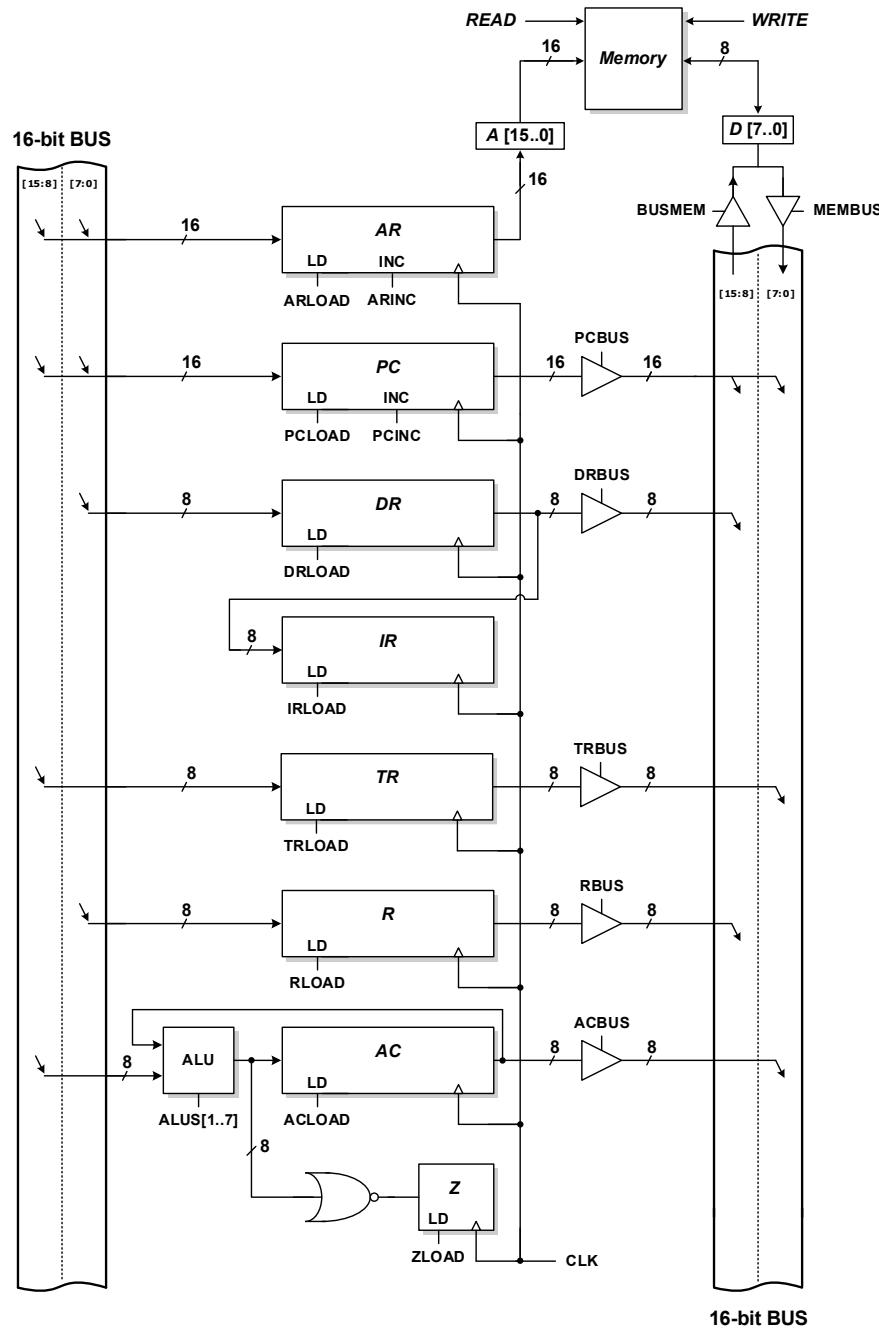
Το σύνολο των καταχωρητών αποτελεί μέρος της ISA μιας ΚΜΕ, αφού βάσει αυτών καθορίζεται ο τρόπος εκτέλεσης κάθε εντολής. Για την περίπτωση της σχετικά απλής ΚΜΕ, οι διαθέσιμοι για τη σχεδίαση και υλοποίησή της καταχωρητές είναι οι εξής :

- Ένας καταχωρητής διεύθυνσης των 16-bits (Address Register - AR)
- Ένας απαριθμητής προγράμματος των 16-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 MKE.

Κύκλωμα παραγωγής σημάτων ALU

Όπως έχει αναφερθεί ο τμήμα της μονάδας ελέγχου παρέχει όλα τα απαραίτητα σήματα ελέγχου απευθείας στα επιμέρους στοιχεία της KME 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  
-- Ρόλος: Κωδικοποιεί τα σήματα ελέγχου της μικροεντολής σε έναν 7-bit κωδικό  
-- προς την ALU / datapath (έξοδος "alus").  
--  
-- Είσοδοι (control signals):  
-- rbus, drbus : επιλογές/ενεργοποιήσεις πηγών στο bus (π.χ. R-bus, D-bus)  
-- acload, zload : φόρτωση καταχωρητών (AC, Z)  
-- andop, orop,  
-- notop, xorop : επιλογή λογικών πράξεων  
-- aczero : καθαρισμός/μηδενισμός AC (CLAC)  
-- acinc : αύξηση AC (INAC)  
-- plus, minus : επιλογή πρόσθεσης/αφαίρεσης  
--  
-- Έξοδος:  
-- alus : 7-bit κωδικός λειτουργίας προς την ALU (ALU select)  
-- ======
```

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

```
-- -----
-- To "control" είναι ένα συμπιεσμένο vector 12-bit που ενώνει ΟΛΑ τα control
-- σήματα σε μία λέξη, για να γίνει εύκολο το case decoding.
```

```
--
```

```
-- Bit order (MSB -> LSB) ακριβώς όπως τα κάνεις concatenate:
```

```
-- control(11) = rbus
-- control(10) = drbus
-- control(9) = acload
-- control(8) = zload
-- control(7) = andop
-- control(6) = orop
-- control(5) = notop
-- control(4) = xorop
-- control(3) = aczero
-- control(2) = acinc
-- control(1) = plus
-- control(0) = minus
```

```
-- -----
signal control : std_logic_vector(11 downto 0);
```

```
begin
```

```
-- Συνένωση των σημάτων ελέγχου σε ένα 12-bit "control"
control <= rbus & drbus & acload & zload & andop & orop
& notop & xorop & aczero & acinc & plus & minus;
```

-
- Combinational process (εναίσθητο μόνο στο control):
 - Κάθε μοναδικό pattern control αντιστοιχεί σε μία λειτουργία ALU (alus code).
-

process(control)

begin

case control is

-- AND: απαιτεί συγκεκριμένο συνδυασμό (rbus=1, drbus=0, acload=1, zload=1,

-- andop=1, τα υπόλοιπα 0)

when "101110000000" => alus <= "1000000"; -- AND

-- OR

when "101101000000" => alus <= "1100000"; -- OR

-- NOT (μονοτελής τελεστής, συνήθως στο AC)

when "001100100000" => alus <= "1110000"; -- NOT

-- XOR

when "101100010000" => alus <= "1010000"; -- XOR

-- CLAC: clear accumulator (μηδενισμός AC)

when "001100001000" => alus <= "0000000"; -- CLAC

-- INAC: increment accumulator (AC <- AC + 1)

when "001100000100" => alus <= "0001001"; -- INAC

-- ADD: πρόσθεση (π.χ. AC <- AC + R ή AC <- AC + DR ανάλογα με bus select)

when "101100000010" => alus <= "0000101"; -- ADD

-- SUB: αφαίρεση

```

when "101100000001" => alus <= "0001011"; -- SUB

-- MOVR: μεταφορά/πέρασμα operand (ουσιαστικά "pass-through" στην ALU)
when "101100000000" => alus <= "0000100"; -- MOVR

-- LDAC5: ειδική περίπτωση όπου το ίδιο ALU code χρησιμοποιείται
-- (π.χ. σε συγκεκριμένο βήμα μικροακολουθίας LDAC)
when "011100000000" => alus <= "0000100"; -- LDAC5

-- Default: αν δεν ταιριάζει κανένα pattern, δίνεις "invalid"/NOP code
-- (εδώ "1111111")
when others => alus <= "1111111"; -- NO OPERATION / invalid

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 KME, 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:

Πρόγραμμα 2: Ο δίαυλος δεδομένων.

H Εξωτερική Μνήμη

Το τμήμα που ολοκληρώνει την υλοποίηση της σχετικά απλής KME σε VHDL, είναι η εξωτερική μνήμη (external memory) ή απλά μνήμη η οποία αποτελεί την πηγή παροχής δεδομένων και εντολών για τη λειτουργία της MKE 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 KME.

```
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:

Πρόγραμμα 4: Η εξωτερική μνήμη.

Συνολική Υλοποίηση KME

Έχοντας ολοκληρώσει την περιγραφή του κώδικα σε VHDL για κάθε ένα επιμέρους τμήμα της MKE, 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 KME.

Γράψτε εδώ το πρόγραμμά σας:

Πρόγραμμα 5: βιβλιοθήκη στοιχείων για την KME.

Με βάση το σκελετό που ακολουθεί (πρόγραμμα 6) γράψτε τον κώδικα περιγραφής για την KME. 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
```

begin

end arc;

Πρόγραμμα 6: Συνολική περιγραφή της KME.

Εξομοίωση της KME.

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 KME.

Τοποθετήστε εδώ τις κυματομορφές σας:

Εικόνα 1: Κυματομορφές εξομοίωσης της KME

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

A/A 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: