



ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ & ΥΛΙΚΟΥ
ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ ΓΙΑ ΤΟ ΜΑΘΗΜΑ: ΗΡΥ 312
ΟΡΓΑΝΩΣΗ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2018-2019

Εργαστήριο 2

**Σχεδίαση Βασικών Βαθμίδων του Datapath ενός
Απλού Επεξεργαστή**

Σκοπός του Εργαστηρίου:

1. Ο ορισμός της αρχιτεκτονικής συνόλου εντολών που θα χρησιμοποιήσετε και στα επόμενα παραδοτέα
2. Η σχεδίαση της βαθμίδας ανάκλησης εντολών
3. Η σχεδίαση της βαθμίδας αποκωδικοποίησης εντολών
4. Η σχεδίαση της βαθμίδας Εκτέλεσης Εντολών (EX)
5. Η σχεδίαση της βαθμίδας Πρόσβασης Μνήμης (MEM)

Αρχιτεκτονική Συνόλου Εντολών

Καλείστε να υλοποιήσετε διάφορα τμήματα ενός non-pipelined επεξεργαστή βασισμένου σε υποσύνολο της αρχιτεκτονικής συνόλου εντολών CHARIS-4 (CHAnia Risc Instruction Set, Έκδοση 4) το οποίο αποτελείται από τα εξής στοιχεία:

1. 32 καταχωρητές των 32 bits. Ο καταχωρητής R0 είναι πάντα μηδέν
2. 32 bit πλάτος εντολών με μέγεθος και θέση πεδίων που περιγράφονται παρακάτω
3. Εντολές αριθμητικών και λογικών πράξεων: add, sub, and, not, or, shr, shl, sla, rol, ror, li, addi, andi, ori
4. Εντολές διακλάδωσης: b, beq, bneq
5. Εντολές μνήμης: lb, sb, lw, sw.

Οι παραπάνω εντολές έχουν δύο τύπους format:

6-bits	5-bits	5-bits	5-bits	5-bits	6-bits
Opcode	rs	rd	rt	not-used	func

6-bits	5-bits	5-bits	16-bits
Opcode	rs	rd	Immediate

Η διευθυνσιοδότηση της μνήμης γίνεται με διευθύνσεις byte, και οι εντολές και τα δεδομένα (των εντολών lb, sb, lw και sw) πρέπει να είναι ευθυγραμμισμένα σε πολλαπλάσια των 4 bytes.

Η κωδικοποίηση των εντολών γίνεται σύμφωνα με τον ακόλουθο πίνακα:

Opcode	FUNC	ΕΝΤΟΛΗ	ΠΡΑΞΗ
100000	110000	add	$RF[rd] \leftarrow RF[rs] + RF[rt]$
100000	110001	sub	$RF[rd] \leftarrow RF[rs] - RF[rt]$
100000	110010	nand	$RF[rd] \leftarrow RF[rs] \text{ NAND } RF[rt]$
100000	110100	not	$RF[rd] \leftarrow \neg RF[rs]$
100000	110011	or	$RF[rd] \leftarrow RF[rs] \mid RF[rt]$
100000	111000	sra	$RF[rd] \leftarrow RF[rs] \gg 1$
100000	111001	sll	$RF[rd] \leftarrow RF[rs] \ll 1$ (Logical, zero fill LSB)
100000	111010	srl	$RF[rd] \leftarrow RF[rs] \gg 1$ (Logical, zero fill MSB)
100000	111100	rol	$RF[rd] \leftarrow \text{Rotate left}(RF[rs])$
100000	111101	ror	$RF[rd] \leftarrow \text{Rotate right}(RF[rs])$
111000	-	li	$RF[rd] \leftarrow \text{SignExtend}(Imm)$
111001	-	lui	$RF[rd] \leftarrow Imm \ll 16$ (zero-fill)
110000	-	addi	$RF[rd] \leftarrow RF[rs] + \text{SignExtend}(Imm)$
110010	-	nandi	$RF[rd] \leftarrow RF[rs] \text{ NAND } \text{ZeroFill}(Imm)$
110011	-	ori	$RF[rd] \leftarrow RF[rs] \mid \text{ZeroFill}(Imm)$
111111	-	b	$PC \leftarrow PC + 4 + (\text{SignExtend}(Imm) \ll 2)$
000000	-	beq	if ($RF[rs] == RF[rd]$) $PC \leftarrow PC + 4 + (\text{SignExtend}(Imm) \ll 2)$ else $PC \leftarrow PC + 4$
000001	-	bne	if ($RF[rs] \neq RF[rd]$) $PC \leftarrow PC + 4 + (\text{SignExtend}(Imm) \ll 2)$ else $PC \leftarrow PC + 4$
000011	-	lb	$RF[rd] \leftarrow \text{ZeroFill}(31 \text{ downto } 8) \& \text{MEM}[RF[rs] + \text{SignExtend}(Imm)](7 \text{ downto } 0)$
000111	-	sb	$\text{MEM}[RF[rs] + \text{SignExtend}(Imm)] \leftarrow \text{ZeroFill}(31 \text{ downto } 8) \& RF[rd] (7 \text{ downto } 0)$
001111	-	lw	$RF[rd] \leftarrow \text{MEM}[RF[rs] + \text{SignExtend}(Imm)]$
011111	-	sw	$\text{MEM}[RF[rs] + \text{SignExtend}(Imm)] \leftarrow RF[rd]$

Διεξαγωγή

A. Μελετήστε την κωδικοποίηση των εντολών του CHARIS

Μελετήστε την κωδικοποίηση των εντολών. Παρατηρήστε την ομαδοποίηση τους, έχοντας στο μυαλό σας τους γενικούς στόχους της αποκωδικοποίησης: η παραγωγή όλων των απαραίτητων σημάτων ελέγχου για την εκτέλεση της εντολής. Τέτοια σήματα είναι ο κωδικός πράξης της ALU, οι διευθύνσεις ανάγνωσης και τυχόν εγγραφής στην Register File, η επιλογή δύο καταχωρητών ή καταχωρητή και Immediate για μια πράξη, κ.α.

Η ανάθεση κωδικών έγινε έτσι ώστε η αποκωδικοποίηση των εντολών να είναι μια σχετικά απλή διαδικασία. Βρείτε τις υπάρχουσες συμμετρίες ώστε να απλοποιήσετε την λογική αποκωδικοποίησης.

B. Σχεδιασμός και Υλοποίηση Κύριας Μνήμης 2048x32

Σε αυτό το ερώτημα καλείστε να υλοποιήσετε την κύρια μνήμη του CHARIS. Η μνήμη αυτή περιέχει τις εντολές και τα δεδομένα του εκάστοτε προγράμματος που εκτελείται. Στα πλαίσια του εργαστηρίου αυτού θα θεωρήσουμε ότι έχουμε δύο segments στη μνήμη μας. Το text segment το οποίο ξεκινάει από τη διεύθυνση 0x000, και το data segment το οποίο ξεκινάει από τη διεύθυνση 0x400. Για περισσότερες πληροφορίες αναφορικά με την κατάτμηση της μνήμης μπορείτε να ανατρέξετε στις γνώσεις σας από το μάθημα των Ψηφιακών Υπολογιστών.

Χρησιμοποιήστε τον κώδικα που περιγράφεται στο Εικόνα 1 για να φτιάξετε μία μνήμη RAM 2048 θέσεων των 32 bits. Η μνήμη πρέπει να έχει μία θύρα ανάγνωσης για τις εντολές και μία θύρα ανάγνωσης/εγγραφής για τα δεδομένα. Η διεπαφή της μνήμης πρέπει να έχει τα εξής σήματα:

Σήμα	Πλάτος	Λειτουργία
inst_addr	11 bits	Διεύθυνση εντολής
inst_dout	32 bits	Εντολή που διαβάστηκε από την μνήμη
data_we	1 bit	Σημαία ενεργοποίησης εγγραφής στη μνήμη
data_addr	11 bits	Διεύθυνση για ανάγνωση/εγγραφή δεδομένων
data_din	32 bits	Δεδομένα που θα εγγραφούν στη μνήμη
data_dout	32 bits	Δεδομένα που διαβάστηκαν από τη μνήμη
clk	1bit	Ρολόι

Προσοχή!! Για να αρχικοποιήσετε την μνήμη χρειάζεται ο κώδικας που είναι με έντονα γράμματα στην παραπάνω εικόνα. Ο συγκεκριμένος κώδικας διαβάζει το αρχείο rom.data (το οποίο θα πρέπει να βρίσκεται στον ίδιο φάκελο με το project σας) και φορτώνει κάθε γραμμή του αρχείου σε μία νέα διεύθυνση.

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use std.textio.all;
use ieee.std_logic_textio.all;

entity RAM is
    port (
        clk          : in std_logic;
        inst_addr     : in std_logic_vector(10 downto 0);
        inst_dout     : out std_logic_vector(31 downto 0);
        data_we       : in std_logic;
        data_addr     : in std_logic_vector(10 downto 0);
        data_din      : in std_logic_vector(31 downto 0);
        data_dout     : out std_logic_vector(31 downto 0));
end RAM;

architecture syn of RAM is
    type ram_type is array (2047 downto 0) of std_logic_vector (31 downto 0);

    impure function InitRamFromFile (RamFileName : in string) return ram_type is
        FILE ramfile : text is in RamFileName;
        variable RamFileLine : line;
        variable ram : ram_type;
    begin
        for i in 0 to 1023 loop
            readline(ramfile, RamFileLine);
            read (RamFileLine, ram(i));
        end loop;
        for i in 1024 to 2047 loop
            ram(i) := x"00000000";
        end loop;
        return ram;
    end function;

    signal RAM: ram_type := InitRamFromFile("ram.data");

begin
    process (clk)
    begin
        if clk'event and clk = '1' then
            if data_we = '1' then
                RAM(conv_integer(data_addr)) <= data_din;
            end if;
        end if;
    end process;
    data_dout <= RAM(conv_integer(data_addr)) after 10ns;
    inst_dout <= RAM(conv_integer(inst_addr)) after 10ns;
end syn;

```

Εικόνα 1: Κώδικας μνήμης RAM

Γ. Σχεδιασμός και υλοποίηση βαθμίδας ανάκλησης εντολών (IF)

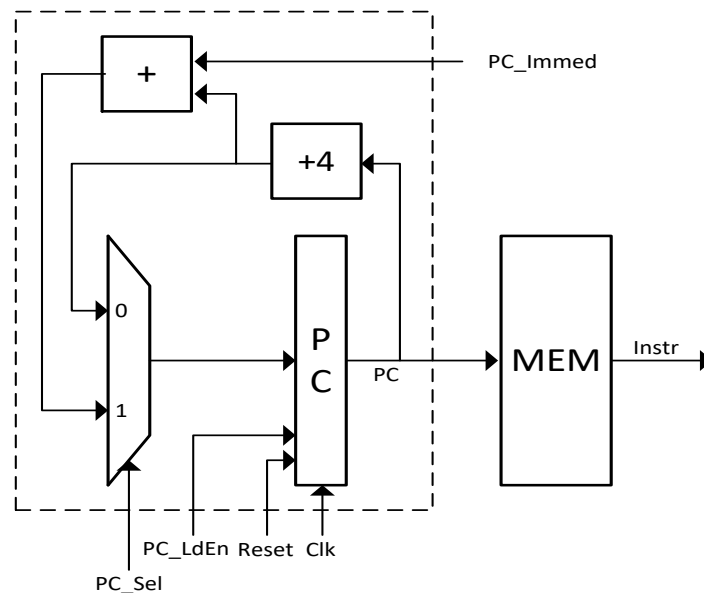
Χρησιμοποιώντας την κατάλληλη πόρτα της κύριας μνήμης και άλλη λογική, σχεδιάστε και υλοποιήστε μια βαθμίδα ανάκλησης εντολών. Η βαθμίδα είναι απλή και αποτελείται από τα παρακάτω δομικά στοιχεία:

1. Τον καταχωρητή PC (32 bits)

2. Την μνήμη που σχεδιάσατε παραπάνω. Στην περίπτωση των εντολών, λόγω του ότι το segment τους ξεκινάει από τη διεύθυνση 0x000, δεν χρειάζεται να προστεθεί κάποιο offset στη διεύθυνση εντολών
3. Έναν αθροιστή (ή ακριβέστερα αυξητή/incrementor κατά 4 ο οποίος υπολογίζει την τιμή $PC + 4$)
4. Έναν αθροιστή ο οποίος υπολογίζει την τιμή $(PC + 4) + Immediate*4$ για εντολές διακλάδωσης
5. Ένα πολυπλέκτη 2 σε 1 ο οποίος διαλέγει μία από τις 2 δυνατές τιμές ($PC+4$, $PC+4+Immediate*4$) για να ενημερωθεί ο PC.

Η διεπαφή της βαθμίδας αυτής συνοψίζεται ως εξής:

Σήμα	Πλάτος	Είδος	Περιγραφή
PC_Immed	32 bits	Είσοδος	Τιμή Immediate για εντολές b, beqz, bnez
PC_sel	1 bit	Είσοδος	Επιλογή εισόδου για ενημέρωση του PC: 0 → $PC+4$ 1 → $PC+4 + Immediate*4$
PC_LdEn	1 bit	Είσοδος	Ενεργοποίηση εγγραφής στον PC
Reset	1 bit	Είσοδος	Είσοδος Reset (active-high) του καταχωρητή PC
Clk	1 bit	Είσοδος	Ρολόι
PC	32 bits	Εξοδος	Διεύθυνση εντολής στην μνήμη



Εικόνα 2: Σχηματικό διάγραμμα βαθμίδας ανάκλησης εντολών

Εκτέλεση

1. Παρατηρήστε το διάγραμμα της βαθμίδας στο σχήμα που ακολουθεί ώστε να κατανοήσετε την λειτουργία της.
2. Γράψτε τον κώδικα VHDL που να υλοποιεί τα επιμέρους τμήματα της βαθμίδας και κάντε τις κατάλληλες εσωτερικές συνδέσεις για να υλοποιήσετε την βαθμίδα

- ανάκλησης εντολών χρησιμοποιώντας τη παραπάνω μνήμη, πολυπλέκτες, καταχωρητές και ό,τι άλλη λογική χρειάζεστε. Ονομάστε το αρχείο σας IFSTAGE.vhd
3. Προσομοιώστε και επιβεβαιώσετε την βαθμίδα ανάκλησης εντολών. Χρησιμοποιήστε το αρχείο rom.data για να αρχικοποιήσετε τη μνήμη και ελέγξτε την λειτουργία της βαθμίδας ανάκλησης σε ακολουθιακές προσβάσεις αλλά και ενεργοποιώντας τις άλλες τιμές για εγγραφή στον PC.

Δ. Σχεδιασμός και υλοποίηση βαθμίδας αποκωδικοποίησης εντολών (DECODE)

Χρησιμοποιώντας ένα αντίγραφο του αρχείου καταχωρητών που παράγατε στο 1^ο Εργαστήριο και άλλη λογική, σχεδιάστε και υλοποιήστε μια βαθμίδα αποκωδικοποίησης εντολών. Η βαθμίδα είναι αρκετά απλή και αποτελείται από:

1. Το αρχείο καταχωρητών
2. Έναν πολυπλέκτη 2 σε 1 ο οποίος επιλέγει μία από τις 2 προελεύσεις των δεδομένων προς εγγραφή στο αρχείο καταχωρητών (ALU, MEM)
3. Μία μονάδα που δέχεται σαν είσοδο τα 16 bits που αποτελούν το immediate μιας εντολής και το μετατρέπει σε ένα σήμα 32 bits επιλέγοντας αν θα γίνει αριστερή ολίσθηση του immediate κατά 2 ή όχι, και επίσης αν θα γίνει zero-filling ή sign-extension του immediate προκειμένου να μετατραπεί σε 32 bit.

Η διεπαφή της βαθμίδας συνοψίζεται ως εξής:

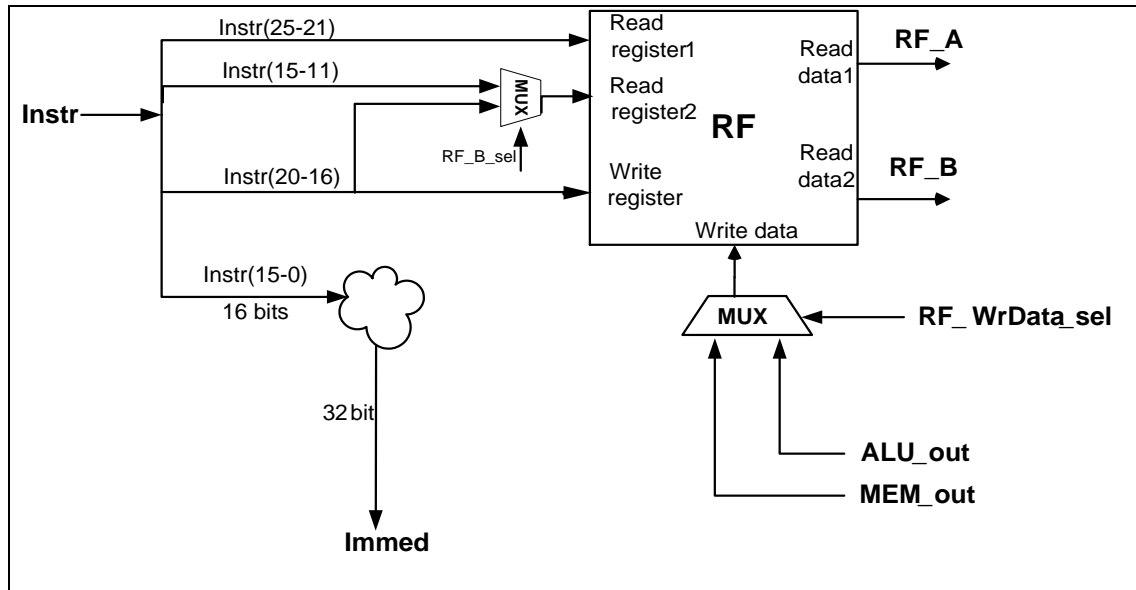
Σήμα	Πλάτος	Είδος	Περιγραφή
Instr	32 bit	Είσοδος	Η εντολή που πρέπει να αποκωδικοποιηθεί
RF_WrEn	1 bit	Είσοδος	Ενεργοποίηση εγγραφής καταχωρητή
ALU_out	32 bits	Είσοδος	Δεδομένα εγγραφής καταχωρητή προερχόμενα από την ALU
MEM_out	32 bits	Είσοδος	Δεδομένα εγγραφής καταχωρητή προερχόμενα από τη μνήμη
RF_WrData_sel	1 bit	Είσοδος	Επιλογή πεδίου που καθορίζει την προέλευση δεδομένων προς εγγραφή: 0 → ALU, 1 → MEM
RF_B_sel	1 bit	Είσοδος	Επιλογή πεδίου που καθορίζει τον δεύτερο καταχωρητή ανάγνωσης: 0 → Instr(15-11), 1 → Instr(20-16)
ImmExt	1 bit	Είσοδος	0 → γέμισμα με μηδέν, 1 → επέκταση πρόσημου
Clk	1 bit	Είσοδος	Ρολόι
Immed	32 bits	Έξοδος	Immediate προς τις επόμενες βαθμίδες
RF_A	32 bits	Έξοδος	Η τιμή του 1 ^{ου} καταχωρητή
RF_B	32 bits	Έξοδος	Η τιμή του 2 ^{ου} καταχωρητή

Εκτέλεση

1. Παρατηρήστε το διάγραμμα της βαθμίδας στο σχήμα που ακολουθεί ώστε να κατανοήσετε την λειτουργία της.
2. Ο καταχωρητής R0 θα έχει πάντα την τιμή "0" (μηδέν) όπως υλοποιήσατε στο πρώτο εργαστήριο.
3. Γράψτε τον κώδικα VHDL που να υλοποιεί τα επιμέρους τμήματα της βαθμίδας και κάντε τις κατάλληλες εσωτερικές συνδέσεις για να υλοποιήσετε την βαθμίδα ανάκλησης εντολών χρησιμοποιώντας τη αρχείο Register File που παράγατε στο

πρώτο εργαστήριο, πολυπλέκτες, καταχωρητές και ό,τι άλλη λογική χρειάζεστε. Ονομάστε το αρχείο σας: DECSTAGE.vhd

4. Προσομοιώστε την βαθμίδα αποκωδικοποίησης εντολών καλύπτοντας τις βασικές κατηγορίες εντολών ώστε να επιβεβαιώσετε την λογική αποκωδικοποίησης. Επεκτείνετε την προσομοίωση για να καλύψετε όσο το δυνατόν περισσότερες περιπτώσεις.



Εικόνα 3: Σχηματικό διάγραμμα βαθμίδας αποκωδικοποίησης εντολών

Ε. Σχεδιασμός και υλοποίηση βαθμίδας Εκτέλεσης Εντολών (EX)

Χρησιμοποιώντας την ALU που σχεδιάσατε στο 1^ο Εργαστήριο και άλλη λογική, σχεδιάστε και υλοποιήστε μια βαθμίδα εκτέλεσης αριθμητικών και λογικών εντολών. Η βαθμίδα αποτελείται από:

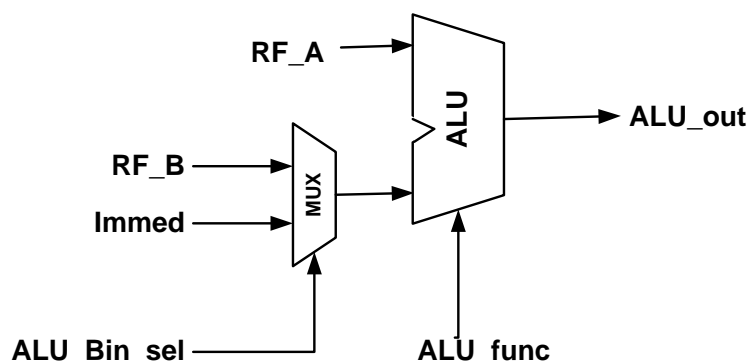
1. Την ALU
2. Ένας πολυπλέκτες που επιλέγει ποιος θα είναι ο δεύτερος τελεστής της ALU.

Συνοπτικά η διεπαφή της βαθμίδας Εκτέλεσης Εντολών (ALU) είναι η εξής:

Σήμα	Πλάτος	Είδος	Περιγραφή
RF_A	32 bits	Είσοδος	RF[rs]
RF_B	32 bits	Είσοδος	RF[rt] ή RF[rd]
Immed	32 bits	Είσοδος	Immediate
ALU_Bin_sel	1 bit	Είσοδος	Επιλογή Εισόδου B της ALU από RF_B ή Immediate 1 → Immed 0 → RF_B
ALU_func	4 bit	Είσοδος	Πράξη ALU
ALU_out	32 bits	Έξοδος	Αποτέλεσμα ALU
ALU_zero	1 bit	Έξοδος	Zero flag

Εκτέλεση

1. Παρατηρήστε το διάγραμμα της βαθμίδας στο σχήμα που ακολουθεί ώστε να κατανοήσετε την λειτουργία της
2. Γράψτε τον κώδικα VHDL που να υλοποιεί τα επιμέρους τμήματα της βαθμίδας και κάντε τις κατάλληλες εσωτερικές συνδέσεις για να υλοποιήσετε την βαθμίδα ανάκλησης εντολών χρησιμοποιώντας τη μνήμη που παράγατε παραπάνω, πολυπλέκτες, καταχωρητές και ό,τι άλλη λογική χρειάζεστε. Ονομάστε το αρχείο σας: EXSTAGE.vhd
3. Προσομοιώστε και επιβεβαιώστε την βαθμίδα εκτέλεσης αριθμητικών και λογικών εντολών.



Εικόνα 4: Σχηματικό διάγραμμα βαθμίδας εκτέλεσης εντολών

ΣΤ. Σχεδιασμός και υλοποίηση βαθμίδας Πρόσβασης Μνήμης (MEM)

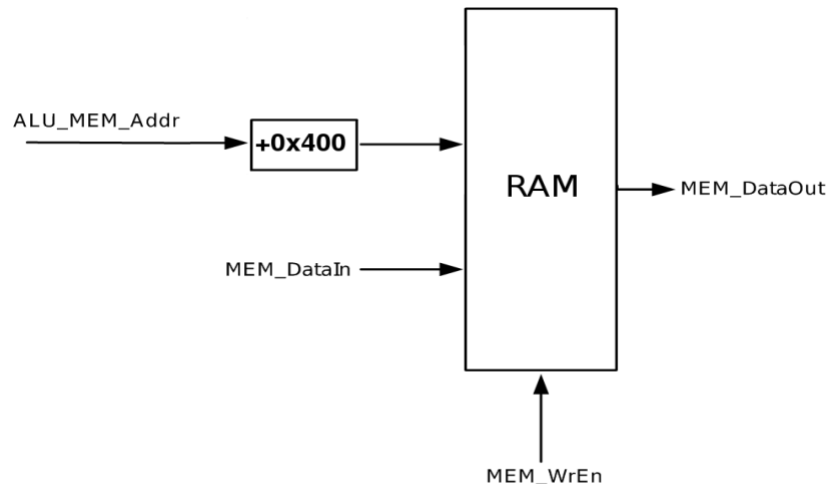
Χρησιμοποιήστε την μνήμη που περιγράφεται στην Εικόνα 1. Επειδή η μνήμη είναι κοινή για τα δεδομένα και τις εντολές, ο επεξεργαστής θα πρέπει να διαβάζει και να γράφει τα δεδομένα από το αντίστοιχο τμήμα. Για το λόγο αυτό η πραγματική διεύθυνση που φτάνει στη μνήμη, είναι η διεύθυνση από την ALU προστιθέμενη κατά 0x400.

Η διεπαφή της βαθμίδας πρέπει να έχει τις εξής εισόδους και εξόδους:

Σήμα	Πλάτος	Είδος	Περιγραφή
clk	1 bit	Είσοδος	Ρολόι
Mem_WrEn	1 bit	Είσοδος	Σημαία ενεργοποίησης εγγραφής στη μνήμη
ALU_MEM_Addr	32 bits	Είσοδος	Αποτέλεσμα ALU (βλέπε εντολές lb, sb, lw, sw)
MEM_DataIn	32 bits	Είσοδος	Αποτέλεσμα RF[rd] για αποθήκευση στη μνήμη για εντολές swap και sb, sw
MEM_DataOut	32 bits	Έξοδος	Δεδομένα που φορτώθηκαν από τη μνήμη προς εγγραφή σε καταχωρητή για εντολές lb, lw

Εκτέλεση

1. Παρατηρήστε το διάγραμμα της βαθμίδας στο σχήμα που ακολουθεί ώστε να κατανοήσετε την λειτουργία της
2. Γράψτε τον κώδικα VHDL που υλοποιεί την βαθμίδα Πρόσβασης Μνήμης και συνδέστε τη με την εξωτερική μνήμη της Εικόνας 1
3. Προσομοιώστε και επιβεβαιώστε την βαθμίδα πρόσβασης στη μνήμη.



Εικόνα 5: Σχηματικό διάγραμμα βαθμίδας πρόσβασης στη μνήμη

Παραδοτέα

1. Αρχεία κώδικα VHDL (πηγαίος)
2. Κυματομορφές προσομοίωσης (καλύψτε όσες περισσότερες περιπτώσεις μπορείτε)
3. Προαιρετικά: Σχηματικό διάγραμμα (όχι κώδικας VHDL) του ολοκληρωμένου datapath δείχνοντας τις μεταξύ των βαθμίδων συνδέσεις
4. Προαιρετικά: Μετά την ολοκλήρωση του datapath καταγράψτε τις αλλαγές που ενδεχομένως κάνατε στις επιμέρους βαθμίδες.

Αναφορά

1. Σύντομη αναφορά στη διαδικασία σχεδίασης και υλοποίησης (μαζί με σχόλια και ενδεχόμενα προβλήματα που παρατηρήθηκαν για μελλοντική του βελτίωση του εργαστηρίου).

Παρατηρήσεις / Σημειώσεις

- Στον ορισμό των modules βάλτε τις παραμέτρους με την σειρά που εμφανίζονται στους πίνακες.