



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

**ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2022-2023**

**Φάση 1 - Σχεδίαση Επεξεργαστή ενός κύκλου**

**Σχεδίαση μονάδας Αριθμητικών και Λογικών Πράξεων και ενός αρχείου καταχωρητών**

**Σκοπός**

Είναι η σχεδίαση σε Γλώσσα Περιγραφής Υλικού (Hardware Description Language) VHDL μιας μονάδας αριθμητικών και λογικών πράξεων και ενός αρχείου καταχωρητών, και η προσομοίωση της με τα εργαλεία της Xilinx.

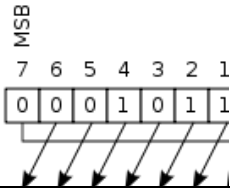
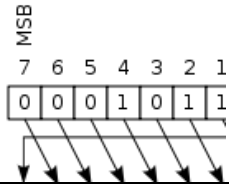
**Διεξαγωγή**

**A) Σχεδίαση της μονάδας αριθμητικών και λογικών πράξεων**

Η μονάδα έχει τις εξής εισόδους και εξόδους:

Σήμα	Είδος -Πλάτος	Λειτουργία
<b>A</b>	είσοδος (32 bits)	Πρώτος τελεσταίος σε συμπλήρωμα ως προς 2
<b>B</b>	είσοδος (32 bits)	Δεύτερος τελεσταίος σε συμπλήρωμα ως προς 2
<b>Op</b>	είσοδος (4 bits)	Κωδικός πράξης
<b>Out</b>	έξοδος (32 bits)	Αποτέλεσμα σε συμπλήρωμα ως προς 2
<b>Zero</b>	έξοδος (1 bit)	Ενεργοποιημένη αν το αποτέλεσμα είναι μηδέν
<b>Cout</b>	έξοδος (1 bit)	Ενεργοποιημένη αν υπήρξε κρατούμενο εξόδου (Carry Out)
<b>Ovf</b>	έξοδος (1 bit)	Ενεργοποιημένη αν υπήρξε υπερχείλιση

Η συμπεριφορά της ALU είναι η εξής:

Κωδικός	Πράξη	Αποτέλεσμα
Op = 0000	Πρόσθεση	Out = A + B
Op = 0001	Αφαίρεση	Out = A - B
Op = 0010	Λογικό “ΚΑΙ”	Out = A & B
Op = 0011	Λογικό “Η”	Out = A   B
Op = 0100	Αντιστροφή του A	Out = ! A
Op = 1000	Αριθμητική ολίσθηση δεξιά κατά 1 θέση (MSB ← (παλιό MSB))	Out= (int) A >> 1 Αποτέλεσμα = {A[31], A[31], ... A[1]}
Op = 1001	Λογική ολίσθηση δεξιά κατά 1 θέση (MSB ← ‘0’)	Out= (unsigned int) A >> 1 Αποτέλεσμα = {0, A[31], ... A[1]}
Op = 1010	Λογική ολίσθηση αριστερά κατά 1 θέση (LSB ← 0)	Out= A << 1 Αποτέλεσμα = {A[30], A[29],... A[0],0}
Op = 1100	Κυκλικό ολίσθηση (rotate) αριστερά το A κατά 1 θέση	
Op = 1101	Κυκλικό ολίσθηση (rotate) δεξιά το A κατά 1 θέση	

1) Γράψτε τον κώδικα που υλοποιεί την ALU σε VHDL. Δώστε προσοχή στις εξόδους Zero, Cout και Onf. Ποίες είναι οι συνθήκες που ορίζουν αυτές τις εξόδους.

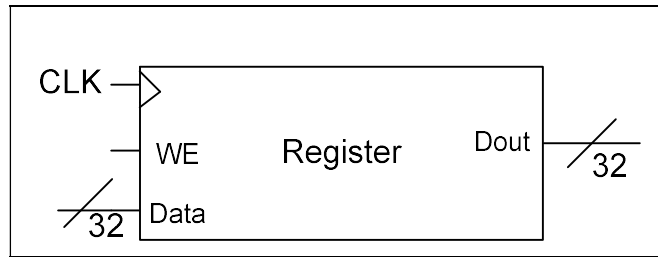
2) Προσομοιώστε την ALU με συστηματικό τρόπο.

## **B) Σχεδίαση του Αρχείου Καταχωρητών**

Το αρχείο καταχωρητών είναι ένα συνδυασμός από καταχωρητές και συνδυαστική λογική.

### **B1. Σχεδίαση Register**

Αρχικά υλοποιήστε έναν καταχωρητή 32 bits σε VHDL. Ο καταχωρητής θα πρέπει να έχει τα εξής σήματα: **ρολόι (CLK - 1 bit)**, **Δεδομένα εισόδου (Data - 32 bits)**, **Δεδομένα εξόδου (Dout - 32 bits)** και ένα σήμα για **Write Enable (WE - 1 bit)**. Η διεπαφή για τον register φαίνεται στην Εικόνα 1.



Εικόνα 1: Καταχωρητής 32 bits

## B2. Σχεδίαση Register File

Για την παραγωγή ενός αρχείου καταχωρητών (Register File-RF) θα χρησιμοποιήσετε 32 registers, όπως αυτούς που υλοποιήσατε στο βήμα B1. Στην συνέχεια κάνοντας την συνδεσμολογία που παρουσιάζεται στην Εικόνα 2 θα υλοποιήσετε ένα αρχείο καταχωρητών με 32 καταχωρητές με τρεις θύρες(δύο ανάγνωσης και μία εγγραφής). Η διεπαφή του αρχείου καταχωρητών έχει τις εξής εισόδους και εξόδους:

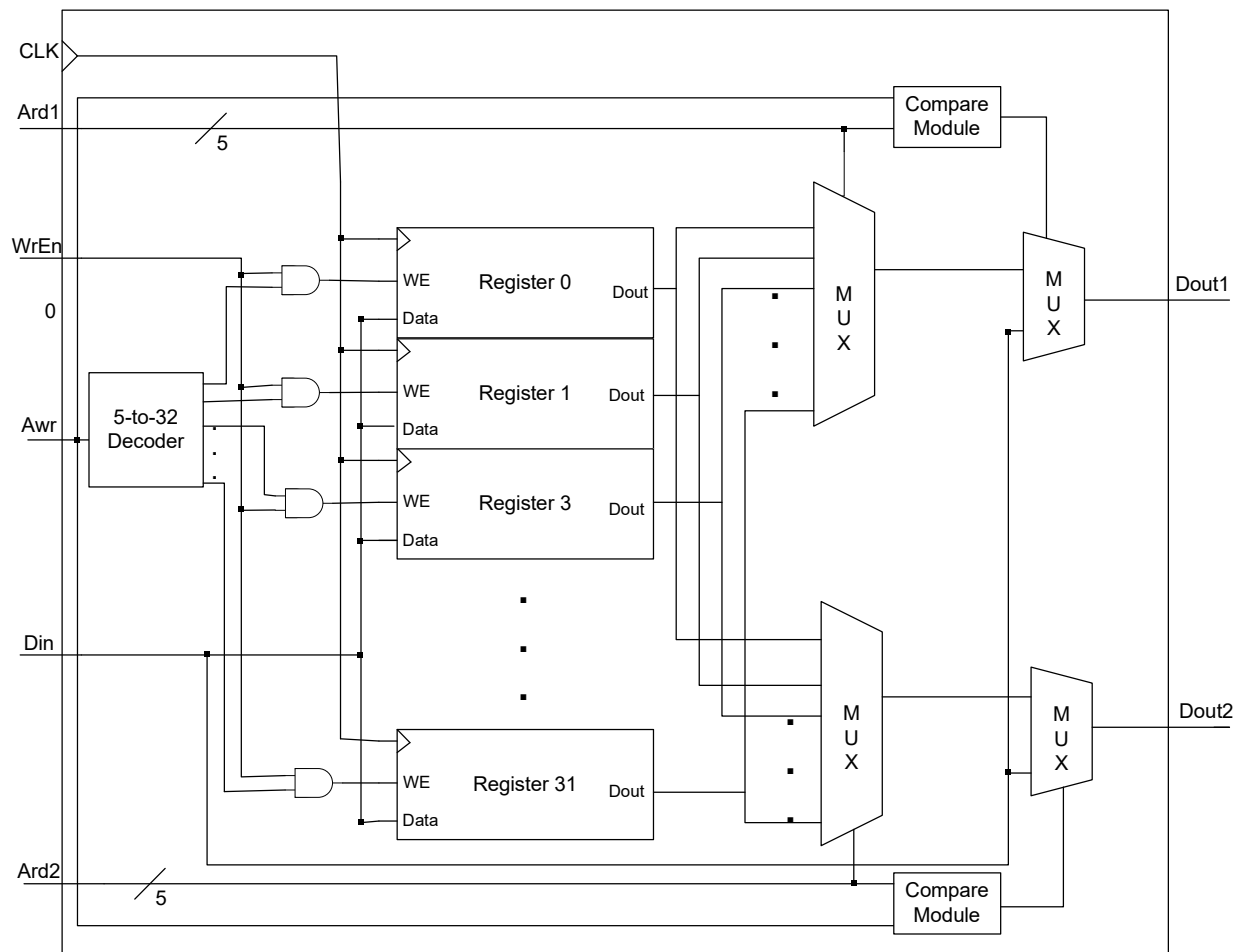
Σήμα	Είδος -Πλάτος	Λειτουργία
Ard1	Είσοδος (5 bits)	Διεύθυνση πρώτου καταχωρητή για ανάγνωση
Ard2	Είσοδος (5 bits)	Διεύθυνση δεύτερου καταχωρητή για ανάγνωση
Awr	Είσοδος (5 bits)	Διεύθυνση καταχωρητή για εγγραφή
Dout1	Έξοδος (32 bits)	Δεδομένα πρώτου καταχωρητή
Dout2	Έξοδος (32 bits)	Δεδομένα δεύτερου καταχωρητή
Din	Είσοδος (32 bits)	Δεδομένα για εγγραφή
WrEn	Είσοδος (1 bit)	Ενεργοποίηση Εγγραφής καταχωρητή
Clk	Είσοδος (1 bit)	Ρολόι

### Παρατηρήσεις:

- 1) Στην διεπαφή δεν υπάρχει είσοδος ενεργοποίησης ανάγνωσης, το οποίο σημαίνει ότι το αρχείο καταχωρητών διαβάζει πάντα και από τις δύο θέσεις που υποδεικνύουν οι διευθύνσεις ανάγνωσης.
- 2) Όλα τα modules που απεικονίζονται στην Εικόνα 2 είναι ασύγχρονα (εκτός προφανώς τους registers).
- 3) Τι χρειάζεται να αλλάξετε στην συνδεσμολογία της Εικόνας 2 έτσι ώστε να μην αλλάζει ποτέ η τιμή του R0; (Ως γνωστό η τιμή του R0 παραμένει πάντα σταθερά 0).
- 4) Το **Compare Module** είναι ένα ασύγχρονο κύκλωμα που ελέγχει αν οι δύο καταχωρητές (εγγραφής και ανάγνωσης) είναι ίδιοι όταν ενεργοποιηθεί το σήμα WrEn. Αν αυτό ισχύει, τότε στην έξοδο της Register File θα πρέπει να έρχεται η νέα τιμή του καταχωρητή και όχι αυτή που είναι αποθηκευμένη στον αντίστοιχο καταχωρητή.

### Υλοποίηση:

- 1) Γράψτε τον κώδικα VHDL που υλοποιεί το αρχείο καταχωρητών χρησιμοποιώντας τον καταχωρητή που υλοποιήσατε στο **B1**.
- 2) Προσομοιώστε την RF στο περιβάλλον Xilinx.
- 3) Δώστε αρκετές διαφορετικές εισόδους στην προσομοίωση ώστε να ελέγξετε όλες τις (ενδιαφέρουσες) περιπτώσεις των σημάτων εισόδου-εξόδου.



Εικόνα 2: Αρχείο καταχωρητών (Register File)

## Φάση 2 - Σχεδίαση Βασικών Βαθμίδων του Datapath ενός Απλού Επεξεργαστή

### Σκοπός:

1. Ο ορισμός της αρχιτεκτονικής συνόλου εντολών που θα χρησιμοποιήσετε και στις επόμενες φάσεις.
2. Η σχεδίαση της βαθμίδας ανάκλησης εντολών.
3. Η σχεδίαση της βαθμίδας αποκωδικοποίησης εντολών.
4. Η σχεδίαση της βαθμίδας Εκτέλεσης Εντολών.
5. Η σχεδίαση της βαθμίδας Πρόσβασης Μνήμης.

### Αρχιτεκτονική Συνόλου Εντολών

Καλείστε να υλοποιήσετε διάφορα τμήματα ενός non-pipelined επεξεργαστή βασισμένου σε υποσύνολο της αρχιτεκτονικής συνόλου εντολών CHARIS (CHAnia Risc Instruction Set) το οποία 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, srl, sll, 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	and	$RF[rd] \leftarrow RF[rs] \& RF[rt]$
100000	110100	not	$RF[rd] \leftarrow ! RF[rs]$
100000	110011	or	$RF[rd] \leftarrow RF[rs]   RF[rt]$
100000	111000	srl	$RF[rd] \leftarrow RF[rs] \gg 1$
100000	111001	sll	$RF[rd] \leftarrow RF[rs] \ll 1$
100000	111010	sla	$RF[rd] \leftarrow RF[rs] \gg 1$ (Logical, zero fill MS bit)
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	-	andi	$RF[rd] \leftarrow RF[rs] \& \text{ZeroFill}(Imm)$
110011	-	ori	$RF[rd] \leftarrow RF[rs]   \text{ZeroFill}(Imm)$
111111	-	b	$PC \leftarrow PC + 4 + (\text{SignExtend}(Imm) \ll 2)$
010000	-	beq	if ( $RF[rs] == RF[rd]$ ) $PC \leftarrow PC + 4 + (\text{SignExtend}(Imm) \ll 2)$ else $PC \leftarrow PC + 4$
010001	-	bne	if ( $RF[rs] != RF[rd]$ ) $PC \leftarrow PC + 4 + (\text{SignExtend}(Imm) \ll 2)$ else $PC \leftarrow PC + 4$
000011	-	lb	$RF[rd] \leftarrow \text{ZeroFill}(31 \text{ downto } 8) \& MEM[RF[rs] + \text{SignExtend}(Imm)](7 \text{ downto } 0)$
000111	-	sb	$MEM[RF[rs] + \text{SignExtend}(Imm)] \leftarrow \text{ZeroFill}(31 \text{ downto } 8) \& RF[rd](7 \text{ downto } 0)$
001111	-	lw	$RF[rd] \leftarrow MEM[RF[rs] + \text{SignExtend}(Imm)]$
011111	-	sw	$MEM[RF[rs] + \text{SignExtend}(Imm)] \leftarrow RF[rd]$

## 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 βαθμίδας ανάκλησης εντολών (IF)**

### **Παραγωγή Μνήμης 1024x32**

Χρησιμοποιήστε το εργαλείο core generator για να φτιάξετε μία μνήμη ROM 1024 θέσεων των 32 bits. Η μνήμη πρέπει να έχει μία θύρα ανάγνωσης. Η διεπαφή της μνήμης πρέπει να έχει τα εξής σήματα: διεύθυνση ανάγνωσης, δεδομένα που διαβάστηκαν και ρολόι. Η διεπαφή της μνήμης συνοψίζεται στον ακόλουθο πίνακα:

Σήμα	Πλάτος	Λειτουργία
Addr	10 bits	Διεύθυνση για ανάγνωση εντολής
Dout	32 bits	Δεδομένα που διαβάστηκαν από την μνήμη
Clk	1bit	Ρολόι

### **B1. Σχεδίαση των επιμέρους τμημάτων και υλοποίηση της βαθμίδας**

Χρησιμοποιώντας μια νέα αλλά ίδια μνήμη και άλλη λογική, σχεδιάστε και υλοποιήστε μια βαθμίδα ανάκλησης εντολών. Η βαθμίδα είναι απλή και αποτελείται από τα παρακάτω δομικά στοιχεία:

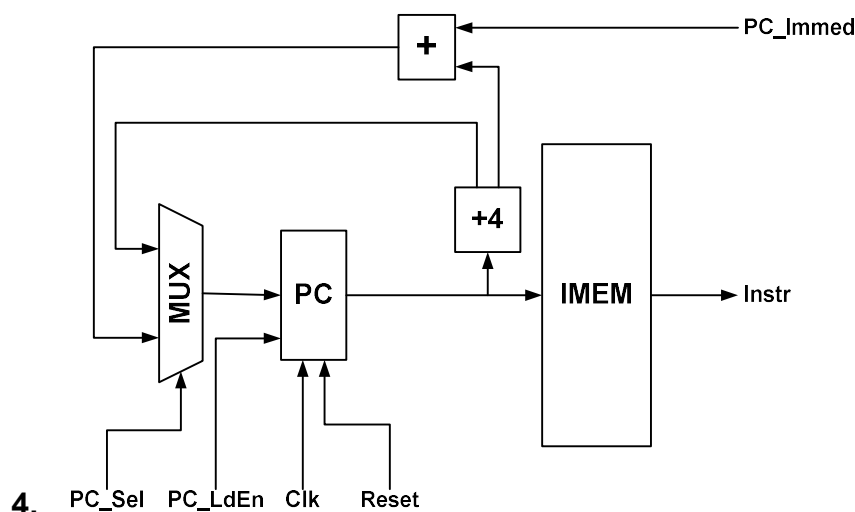
1. Τον καταχωρητή PC (32 bits).
2. Την μνήμη εντολών που σχεδιάσατε παραπάνω
3. Έναν αθροιστή (ή ακριβέστερα αυξητή/incrementor κατά 4o οποίος υπολογίζει την τιμή  $PC + 4$ ).
4. Έναν αθροιστή ο οποίος υπολογίζει την τιμή  $(PC + 4) + \text{Immediate}$  για εντολές διακλάδωσης.
5. Ένα πολυπλέκτη 2 σε 1 ο οποίος διαλέγει μία από τις 2 δυνατές τιμές ( $PC+4$ ,  $PC+4+\text{Immediate}$ ) 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: $0 \rightarrow PC+4$ , $1 \rightarrow PC+4 + \text{Immediate}$ .
PC_LdEn	1 bit	Είσοδος	Ενεργοποίηση εγγραφής στον PC
Reset	1 bit	Είσοδος	Είσοδος Reset για αρχικοποίηση του καταχωρητή PC
Clk	1 bit	Είσοδος	Ρολόι
Instr	32 bits	Εξοδος	Η εντολή που ανακλήθηκε.

## 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 IFSTAGE.vhd
3. Προσομοιώστε και επιβεβαιώσετε την βαθμίδα ανάκλησης εντολών.



Σχήμα 1: Σχηματικό διάγραμμα βαθμίδας ανάκλησης εντολών.

## Γ. Σχεδιασμός και υλοποίηση βαθμίδας αποκωδικοποίησης εντολών (DECODE)

Χρησιμοποιώντας ένα αντίγραφο του αρχείου καταχωρητών που παράγατε στη φάση 1 και άλλη λογική, σχεδιάστε και υλοποιήστε μια βαθμίδα αποκωδικοποίησης εντολών. Η βαθμίδα είναι αρκετά απλή στην λογική 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 16bits 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.

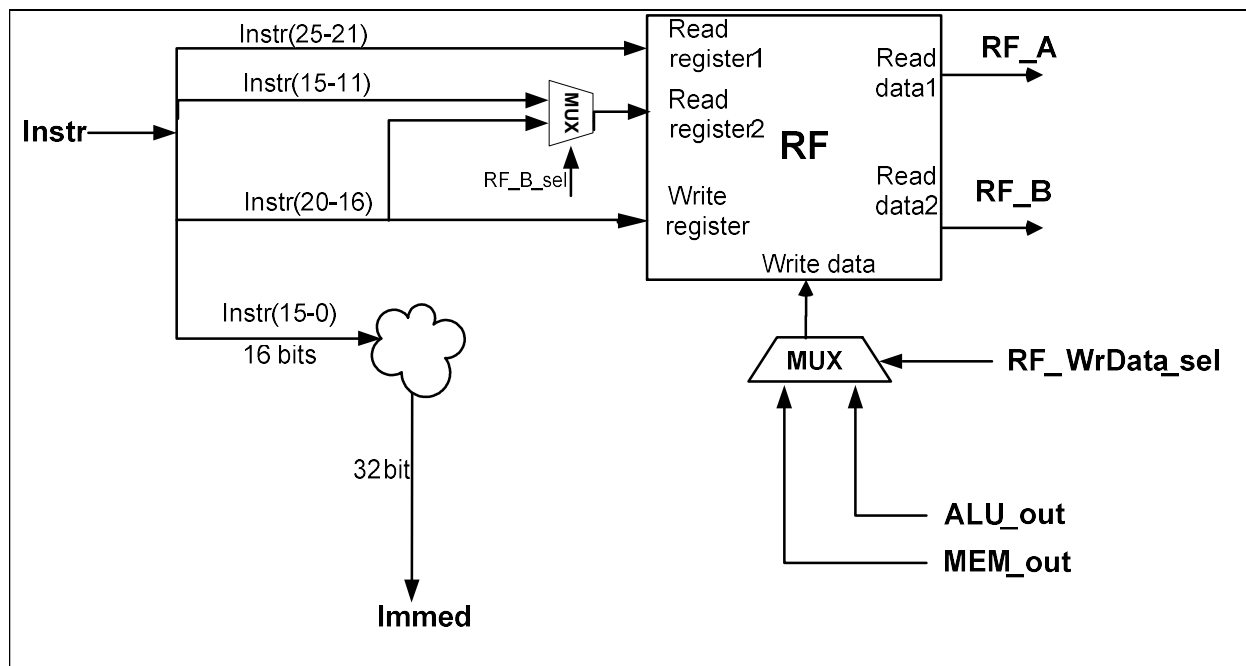


Η διεπαφή της βαθμίδας συνοψίζεται ως εξής:

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: 1 → MEM 0 → ALU
RF_B_sel	1 bit	Είσοδος	Επιλογή πεδίου που καθορίζει τον δεύτερο καταχωρητή ανάγνωσης: 0 → Instr(15-11) 1 → Instr(20-16)
Clk	1 bit	Είσοδος	Ρολόι
Immed	32 bits	Έξοδος	Immediate προς τις επόμενες βαθμίδες
RF_A	32 bits	Έξοδος	Η τιμή του 1 <sup>ου</sup> καταχωρητή
RF_B	32 bits	Έξοδος	Η τιμή του 2 <sup>ου</sup> καταχωρητή

## Εκτέλεση

1. Παρατηρήστε το διάγραμμα της βαθμίδας στο σχήμα που ακολουθεί ώστε να κατανοήσετε την λειτουργία της
2. Ο καταχωρητής R0 πρέπει να έχει πάντα την τιμή "0" (μηδέν). Πώς το υλοποιείτε αυτό;
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 RegisterFile 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.



**Σχήμα 2:** Σχηματικό διάγραμμα βαθμίδας αποκωδικοποίησης εντολών.

#### **Δ. Σχεδιασμός και υλοποίηση βαθμίδας Εκτέλεσης Εντολών (ALU)**

Χρησιμοποιώντας την ALU που σχεδιάσατε στη Φάση 1 και άλλη λογική, σχεδιάστε και υλοποιήστε μια βαθμίδα εκτέλεσης αριθμητικών και λογικών εντολών. Η βαθμίδα αποτελείται από:

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

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

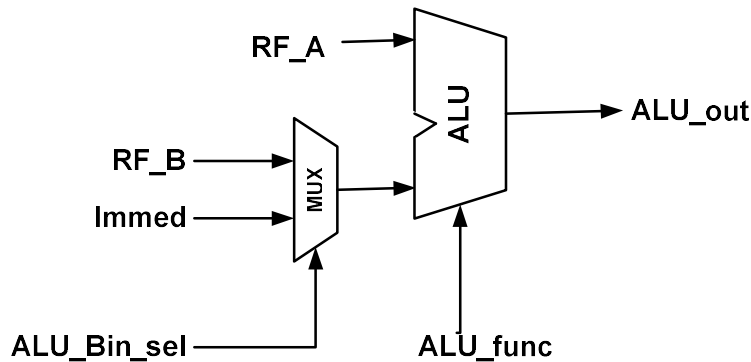
Σήμα	Πλάτος	Είδος	Περιγραφή
RF_A	32 bits	Είσοδος	RF[rs]
RF_B	32 bits	Είσοδος	RF[rt] ή RF[rd]
Immed	32 bits	Είσοδος	Immediate
ALU_Bin_sel	1 bit	Είσοδος	Επιλογή Εισόδου B της ALU από RF_B ή Immediate 1 → Immed 0 → RF_B
ALU_func	4 bit	Είσοδος	Πράξη ALU
ALU_out	32 bit	Έξοδος	Αποτέλεσμα ALU

#### **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.

3. Προσομοιώστε και επιβεβαιώστε την βαθμίδα εκτέλεσης αριθμητικών και λογικών εντολών.



**Σχήμα 3:** Σχηματικό διάγραμμα βαθμίδας εκτέλεσης εντολών.

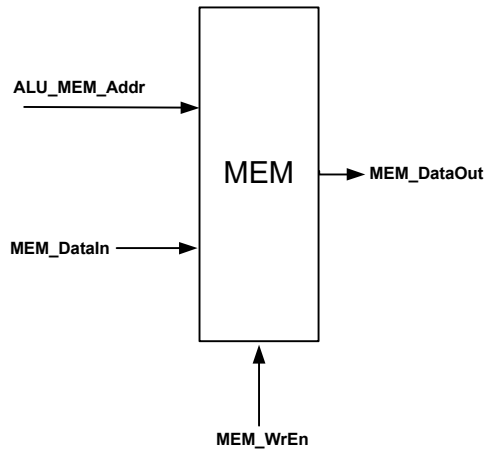
### ***E. Σχεδιασμός και υλοποίηση βαθμίδας Πρόσβασης Μνήμης (MEM)***

Με το εργαλείο core generator της Xilinx ISE να παράγετε μια μνήμη RAM1024 θέσεων των 32 bits. Η μνήμη είναι **ReadFirst** και έχει μία θύρα ανάγνωσης και εγγραφής. 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 bitss	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 χρησιμοποιώντας τη μνήμη που παράγατε παραπάνω.
3. Προσομοιώστε και επιβεβαιώστε την βαθμίδα πρόσβασης στη μνήμη.



Σχήμα 4: Σχηματικό διάγραμμα βαθμίδας πρόσβασης στη μνήμη.

## Φάση 3 - Ολοκλήρωση ενός επεξεργαστή ενός κύκλου

### Σκοπός

Σκοπός της φάσης αυτής είναι η ολοκλήρωση του επεξεργαστή ενός κύκλου.

### Κατασκευή ενιαίου datapath

Μελετήστε όλες τις βαθμίδες που υλοποιήσατε στα προηγούμενα εργαστήρια. Σχεδιάστε τις συνδέσεις των 4 αυτών βαθμίδων έτσι ώστε να κατασκευάσετε ένα ενιαίο datapath του επεξεργαστή. Καλό είναι στο σημείο αυτό να γίνει εξαντλητικό simulation και κατανόηση του τρόπου λειτουργίας του datapath. Οι είσοδοι στο simulation θα πρέπει να είναι κυρίως η εντολή και τα σχετικά σήματα ελέγχου όπως πρέπει να δουλεύουν σύμφωνα με την εισαχθείσα εντολή.

### Σχεδιασμός και υλοποίηση της μονάδας ελέγχου

Για να λειτουργήσει το datapath του επεξεργαστή χρειάζεται μια μονάδα ελέγχου που να γεννά τα επιθυμητά σήματα ελέγχου ανάλογα με την εντολή που εκτελείται.

Γράψτε τον κώδικα VHDL που υλοποιεί το control του επεξεργαστή το οποίο θα παράγει τα σωστά σήματα ελέγχου για την κάθε εντολή.

Ολοκληρώστε πρώτα τη σχεδίαση του Datapath και στη συνέχεια δημιουργήστε την μονάδα ελέγχου.

Η μονάδα ελέγχου υλοποιείται σαν μια μηχανή πεπερασμένων καταστάσεων η οποία έχει σαν είσοδο την εντολή "Instr" που "βγαίνει" από το IFSTAGE και σαν εξόδους όλα τα σήματα ελέγχου που περιέχονται στο Datapath. Ενδεχομένως να χρειαστείτε και κάποιους επιπλέον καταχωρητές προκειμένου να μπορείτε να κρατάτε τις τιμές για

κάποια σήματα που παράγονται από κάποια βαθμίδα και πρέπει να χρησιμοποιηθούν σε επόμενη κατάσταση σε κάποια άλλη βαθμίδα.

**ΠΡΟΣΟΧΗ!!!** Μία εντολή τύπου “00000000000000000000000000000000” θα πρέπει να θεωρείται `nop` και το `control` να την αγνοεί μη ενεργοποιώντας κανένα σήμα έλεγχου και απλά να διαβάζει την επομένη εντολή από την `IF_MEM`.

### **Εκτέλεση**

1. Γράψτε τον κώδικα VHDL που υλοποιεί το datapath του επεξεργαστή ενώνοντας τις 4 βαθμίδες που υλοποιήσατε στο 2ο Εργαστήριο.
2. Γράψτε τον κώδικα VHDL που υλοποιεί την μονάδα έλεγχου του datapath.
3. Προσθέστε ό, τι επιπλέον λογική χρειαστείτε στο datapath (καταχωρητές, κτλ).
4. Συνδέστε το datapath με το CONTROL ώστε να υλοποιήσετε την πλήρη λειτουργία ενός επεξεργαστή ενός κύκλου.
5. Επιβεβαιώστε την ορθή λειτουργία του επεξεργαστή δίνοντας πραγματικές εντολές σαν είσοδο και ελέγχοντας τις τιμές των σημάτων εξόδου σε κάθε κατάσταση της FSM.
6. Προσομοιώστε και επιβεβαιώσετε την λειτουργία του επεξεργαστή εκτελώντας εντολές μια-μια (με το χέρι)
7. Προσομοιώστε και επιβεβαιώσετε την λειτουργία του επεξεργαστή εκτελώντας μικρά δικά σας προγράμματα με τα οποία θα αρχικοποιήσετε την `IF_MEM`.