ΥΣ03 Σχεδίαση Ψηφιακών Συστημάτων

Μεθοδολογία σχεδίασης επεξεργαστή RISC-V ενός κύκλου

Πασχάλης Αντώνιος, Βασιλόπουλος Διονύσης

ΠΡΟΠΤΥΧΙΑΚΟΙ ΦΟΙΤΗΤΕΣ ΤΜΗΜΑΤΟΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Χειμερινό Εξάμηνο
2025 – 2026

Project

Πίνακας Περιεχομένων

Project

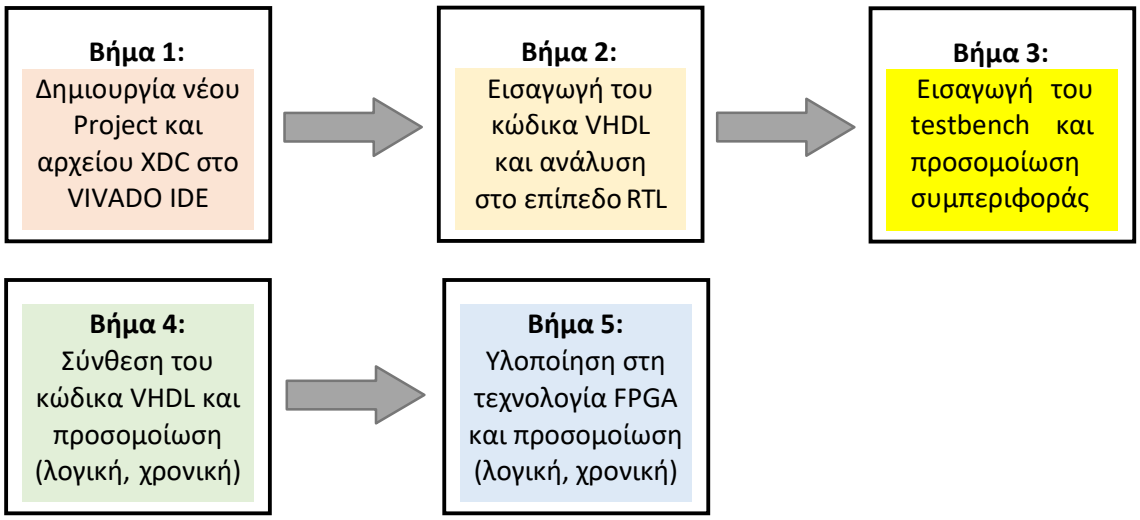
Εισαγωγικά στοιχεία

Στον παρόντα εργαστηριακό οδηγό θα περιγράψουμε αναλυτικά τη μεθοδολογία σχεδίασης της μικροαρχιτεκτονικής ενός απλοποιημένου **επεξεργαστή ενός κύκλου** αρχιτεκτονικής **RISC-V** σε τεχνολογία **FPGA** με τη χρήση του εργαλείου **Vivado IDE** (Integrated Development Environment) της Xilinx.

Αρχικά, αναλύουμε τις απαιτήσεις του επεξεργαστή που απορρέουν από την εφαρμογή της μικροαρχιτεκτονικής RISC-V, από τα 5 βήματα εκτέλεσης της εντολής στη μικροαρχιτεκτονική RISC-V (με τις αντίστοιχες λειτουργίες που εκτελούνται ανά βήμα) και από το σύνολο εντολών που πρόκειται να υλοποιηθεί. Στη συνέχεια, προσδιορίζουμε τα ψηφιακά δομικά στοιχεία της διαδρομής δεδομένων που απαιτούνται σε κάθε βήμα εκτέλεσης της εντολής και προχωράμε στη σχεδίαση της διαδρομής δεδομένων του επεξεργαστή ενός κύκλου ακολουθώντας την ιεραρχική προσέγγιση bottom-up. Μετά, προσδιορίζουμε τα ψηφιακά δομικά στοιχεία της μονάδας ελέγχου και προχωράμε στη σχεδίαση της. Τέλος, στο ανώτερο επίπεδο (top-level) του επεξεργαστή, τοποθετούμε μαζί τη διαδρομή δεδομένων και τη μονάδα ελέγχου, επαληθεύουμε την ορθή σχεδίαση με κατάλληλο πρόγραμμα σε συμβολική γλώσσα RISC-V, βρίσκουμε τη μέγιστη συχνότητα λειτουργίας και αναλύουμε τις επιδόσεις του επεξεργαστή.

0-1. **Γενική ροή σχεδίασης του επεξεργαστή**

Ακολουθούμε τα Βήματα 1-5 σε όλα τα ιεραρχικά επίπεδα της σχεδίασης του επεξεργαστή: τα δομικά στοιχεία της διαδρομής δεδομένων, τη μονάδα ελέγχου, το ανώτερο επίπεδο του επεξεργαστή.



Βήμα 1: Ανάλυση των απαιτήσεων του επεξεργαστή

1-1. **Απαιτήσεις που απορρέουν από την εφαρμογή της μικροαρχιτεκτονικής RISC-V**

1-1.1. Κάθε μικροαρχιτεκτονική RISC-V πρέπει να υλοποιεί όλους τους **αρχιτεκτονικούς καταχωρητές**, που είναι οι εξής:

- **αρχείο καταχωρητών** (register file, RF) με 32 καταχωρητές **R0-R31** των 32 bit,
- **μετρητής προγράμματος** (program counter, PC)

1-1.2. Στη μικροαρχιτεκτονική RISC-V η μνήμη είναι προσπελάσιμη **ανά byte** με λέξεις εντολών/δεδομένων των 32 bit (4 byte). Η διεύθυνση των 32 bit της λέξης ταυτίζεται με τη διεύθυνση του **λιγότερου σημαντικού byte** (least significant byte, LSB) της λέξης και είναι **ευθυγραμμισμένη** (σε πολλαπλάσια του 4). Το περισσότερο σημαντικό byte (most significant byte, **MSB**) της λέξης βρίσκεται στα **αριστερά**, ενώ το λιγότερο σημαντικό byte (least significant byte, **LSB**) της λέξης βρίσκεται στα **δεξιά**.

Για τον προσδιορισμό της διεύθυνσης της λέξης δεδομένων στη μνήμη χρησιμοποιείται ο τρόπος διευθυνσιοδότησης **μνήμης με σχετική απόσταση** (offset addressing) που χρησιμοποιεί έναν καταχωρητή βάσης (base register) που περιέχει τη διεύθυνση βάσης, και μια σχετική απόσταση (offset) ως **μη προσημασμένος ακέραιος των 12 bit** που είναι άμεσα αποθηκευμένος στην ίδια την εντολή. Για να σχηματιστεί η διεύθυνση μνήμης, προσθέτουμε/ αφαιρούμε τη σχετική απόσταση (που μπορεί να είναι 0) στο/από το περιεχόμενο του καταχωρητή βάσης.

1-1.3. Η μικροαρχιτεκτονική RISC-V υποστηρίζει εντολές που **ενεργοποιούν σημαίες συνθήκης** (*condition flags*) ανάλογα με το αποτέλεσμα μίας πράξης στη μονάδα ALU. Οι εντολές, που έπονται της εντολής που ενεργοποιεί σημαίες συνθήκης, εκτελούνται **υπό συνθήκη**, ανάλογα με τις τιμές των σημαιών.

1-1.4. Η μικροαρχιτεκτονική RISC-V του επεξεργαστή ενός κύκλου απαρτίζεται από δύο διακριτές μονάδες που αλληλοεπιδρούν μεταξύ τους:

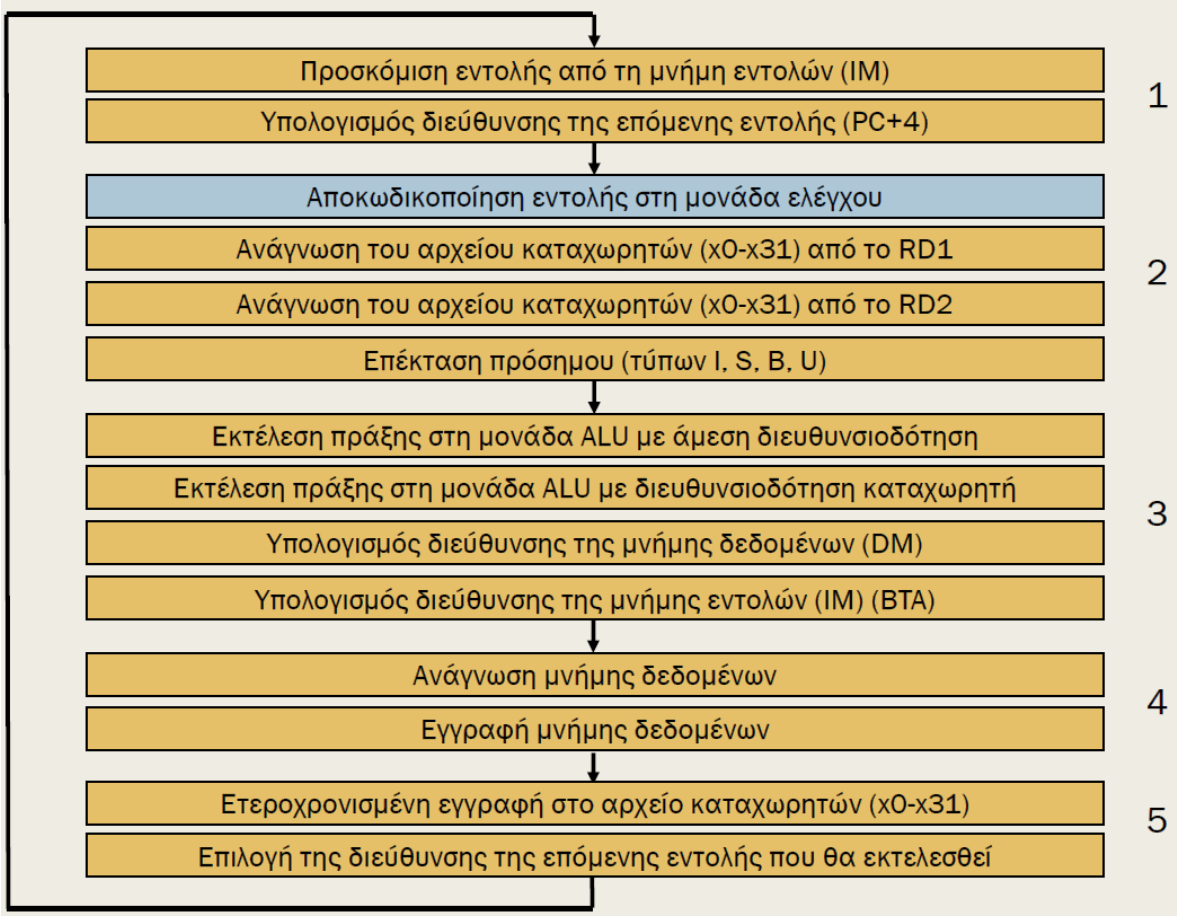
- τη **διαδρομή δεδομένων** (datapath) των 32 bit, που εκτελεί λειτουργίες με λέξεις δεδομένων και περιέχει υπομονάδες όπως: διακριτές μνήμες εντολών (ROM) και δεδομένων (RAM) μεγέθους 32 bit, αρχιτεκτονικούς καταχωρητές, μονάδα ALU για την εκτέλεση των πράξεων που απορρέουν από το σύνολο εντολών που πρόκειται να υλοποιηθεί, πολυπλέκτες για τον έλεγχο της ροής δεδομένων, επιπλέον συνδυαστική,
- τη **μονάδα ελέγχου** (control unit), ως συνδυαστική λογική, που αποκωδικοποιεί την εντολή και για κάθε εντολή υπό συνθήκη παράγει τα κατάλληλα σήματα ελέγχου των υπομονάδων της διαδρομής δεδομένων, λαμβάνοντας υπόψη τις τιμές των σημαιών (N, Z, C, V).

1-1.5. Στη μικροαρχιτεκτονική RISC-V του επεξεργαστή ενός κύκλου ολόκληρη η εντολή εκτελείται σε **έναν κύκλο ρολογιού** (CLK), όπου η περίοδος του CLK προσδιορίζεται από την εντολή **LW** που έχει τη μεγαλύτερη καθυστέρηση διάδοσης από όλες τις εντολές του επεξεργαστή.

1-2. **Τα 5 βήματα εκτέλεσης της εντολής στη μικροαρχιτεκτονική RISC-V**

1. Προσκόμιση εντολής και υπολογισμός της επόμενης διεύθυνσης (PC+4)
2. Αποκωδικοποίηση εντολής στη μονάδα ελέγχου, ανάγνωση 2 καταχωρητών από το αρχείο καταχωρητών και επέκταση προσήμου/μηδενός.
3. Εκτέλεση πράξεων στη μονάδα ALU
4. Ανάγνωση ή εγγραφή στη μνήμη δεδομένων
5. Ετεροχρονισμένη εγγραφή ενός καταχωρητή στο αρχείο καταχωρητών και επιλογή της διεύθυνσης της επόμενης εντολής που πρόκειται να εκτελεσθεί.

Στο επόμενο σχήμα φαίνονται οι 14 λειτουργίες που εκτελούνται ανά βήμα.



1-3. **Το σύνολο εντολών που πρόκειται να υλοποιηθεί**

1-3.1. Η εντολή **LW**

Κώδικας συμβολικής γλώσσας και περιγραφή στο επίπεδο RTL:

- LW Rd, imm(Rs1) Rd = mem[Rs1+S_Ext(imm)]

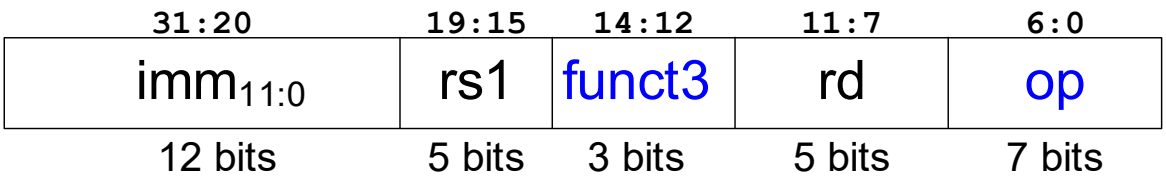
Τελεστέοι που χρησιμοποιούνται κατά την εκτέλεση της εντολής:

- ο τελεστέος στον καταχωρητή προορισμού Rd,
- ο τελεστέος στον καταχωρητή προέλευσης Rs1,
- ο προσημασμένος άμεσος τελεστέος των 12 bit με επέκταση προσήμου στα 32 bit.

Πράξεις που εκτελούνται στη μονάδα ALU:

- πρόσθεση

Για επέκταση προσήμου τύπου I τα σήματα ελέγχου είναι: **ImmScr(1:0) = 00**



1-3.2. Η εντολή **SW**

Κώδικας συμβολικής γλώσσας και περιγραφή στο επίπεδο RTL:

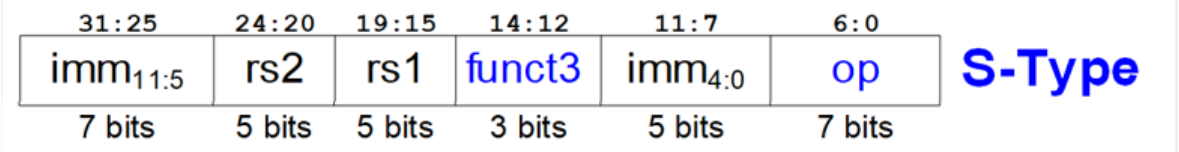
- SW Rs2, imm(Rs1) mem[Rs1+S_Ext(imm)] = Rs2

Τελεστέοι που χρησιμοποιούνται κατά την εκτέλεση της εντολής:

- ο τελεστέος στον καταχωρητή προέλευσης Rs1,
- ο τελεστέος στον καταχωρητή προέλευσης Rs2,
- ο προσημασμένος άμεσος τελεστέος των 12 bit με επέκταση προσήμου στα 32 bit (επέκταση προσήμου τύπου S).

Πράξεις που εκτελούνται στη μονάδα ALU:

- πρόσθεση



1-3.3. Οι εντολές επεξεργασίας δεδομένων **ALU** με διευθυνσιοδότηση καταχωρητή

Κώδικας συμβολικής γλώσσας και περιγραφή στο επίπεδο RTL:

ADD Rd, Rs1, Rs2 Rd = Rs1 + Rs2

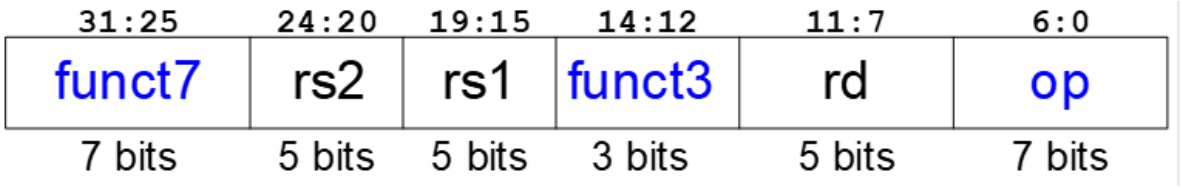
SUB Rd, Rs1, Rs2 Rd = Rs1 – Rs2
AND Rd, Rs1, Rs2 Rd = Rs1 and Rs2
OR Rd, Rs1, Rs2 Rd = Rs1 or Rs2
XOR Rd, Rs1, Rs2 Rd = Rs1 xor Rs2

Τελεστέοι που χρησιμοποιούνται κατά την εκτέλεση της εντολής:

- ο τελεστέος στον καταχωρητή προέλευσης Rs1,
- ο τελεστέος στον καταχωρητή προέλευσης Rs2,
- ο τελεστέος στον καταχωρητή προορισμού Rd,

Πράξεις που εκτελούνται στη μονάδα ALU:

- αριθμητικές πράξεις (πρόσθεση, αφαίρεση),
- λογικές πράξεις (and, or, xor)



1-3.4. Οι εντολές επεξεργασίας δεδομένων με άμεση διευθυνσιοδότηση **ALUI**

Κώδικας συμβολικής γλώσσας και περιγραφή στο επίπεδο RTL:

ADDI Rd, Rs1, imm Rd = Rs1 + S_Ext(imm)
ANDI Rd, Rs1, imm Rd = Rs1 and S_Ext(imm)
ORI Rd, Rs1, imm Rd = Rs1 or S_Ext(imm)
XORI Rd, Rs1, imm Rd = Rs1 xor S_Ext(imm)

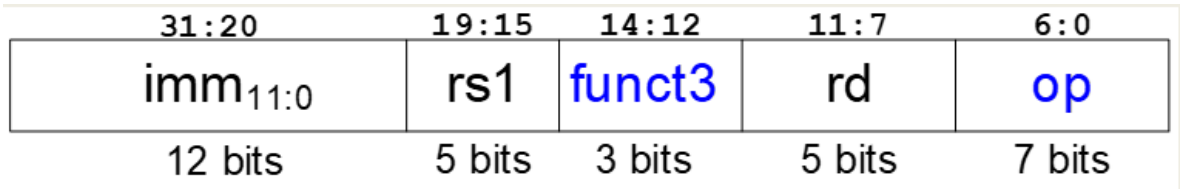
- Τελεστέοι που χρησιμοποιούνται κατά την εκτέλεση της

εντολής:

- ο τελεστέος στον καταχωρητή προέλευσης Rs1,
- ο τελεστέος στον καταχωρητή προορισμού Rd.
- ο προσημασμένος άμεσος τελεστέος των 12 bit με επέκταση προσήμου στα 32 bit (επέκταση προσήμου τύπου I).

Πράξεις που εκτελούνται στη μονάδα ALU:

- αριθμητικές πράξεις (πρόσθεση, αφαίρεση),
- λογικές πράξεις (and, or, xor)



Βήμα 2: Σχεδίαση των ψηφιακών δομικών στοιχείων της διαδρομής δεδομένων

Στο κατώτερο ιεραρχικά επίπεδο της σχεδίασης περιγράφεται στη γλώσσα VHDL η συμπεριφορά των ψηφιακών δομικών στοιχείων που απαιτούνται σε κάθε βήμα εκτέλεσης της εντολής. Το μέγεθος των αρτηριών των λειτουργικών μονάδων και των καταχωρητών, καθώς και το μέγεθος των μνημών παραμετροποιείται με τη δήλωση της εντολής generic.

2-1. **Προσκόμιση εντολής και υπολογισμός της επόμενης διεύθυνσης (PC+4)**

2-1.1. Παραμετροποιημένος καταχωρητής των N bit (με αρχική τιμή N = 32) με RESET για να χρησιμοποιηθεί ως μετρητής προγράμματος (**program counter, PC**).

2-1.2. Παραμετροποιημένη διάταξη μνήμης ROM με 2^N λέξεις μεγέθους M bit (με αρχικές τιμές N = 6 και M = 32) για να χρησιμοποιηθεί ως μνήμη εντολών (**instruction memory, IM**). Προσοχή! A[N-1:0] = PC[N+1:2].

2-1.3. Παραμετροποιημένος αθροιστής κατά 4 με έξοδο PCPlus4 για τον υπολογισμό της διεύθυνσης της αμέσως επόμενης εντολής PC + 4 (**incrementer by 4, INC4**).

2-2. **Αποκωδικοποίηση εντολής και ανάγνωση αρχείου καταχωρητών**

2-2.1. Παραμετροποιημένο αρχείο καταχωρητών των 2^N καταχωρητών μεγέθους M bit (με αρχικές τιμές N = 4 και M = 32) με δυνατότητα ασύγχρονου διαβάσματος δύο καταχωρητών και σύγχρονης εγγραφής ενός καταχωρητή, όταν RegWrite = 1 (WE3).

2-2.2. Μονάδα επέκτασης μηδενός από τα 13 bit στα 32 Bit (για ImmSrc(1:0) = “10”, εντολές branch) ή προσήμου από τα 12 bit στα 32 bit (για επέκταση τύπου I ImmSrc(1:0) = “00”, για επέκταση τύπου S ImmSrc(1:0) = “01”) (**Extend**).

2-3. **Εκτέλεση πράξεων στη μονάδα ALU**

2-3.1. Παραμετροποιημένος πολυπλέκτης 2 σε 1 των N bit (με αρχική τιμή N = 32) για να χρησιμοποιηθεί στον προσδιορισμό των δεδομένων της εισόδου SrcB της μονάδας ALU (**mux2to1**). Επιλέγει μεταξύ της θύρας ανάγνωσης A2/RD2 του αρχείου καταχωρητών (ALUSrc = 0) και της εξόδου ExtImm της μονάδας επέκτασης προσήμου-μηδενός (ALUSrc = 1). Η έξοδος συνδέεται στην είσοδο SrcB της μονάδας ALU.

2-3.2. Παραμετροποιημένη μονάδα ALU μεγέθους N bit (με αρχική τιμή N = 32) (**ALU**). Η μονάδα ALU έχει δύο εισόδους SrcA και SrcB, μία έξοδο ALUResult και συμπεριλαμβάνει αθροιστή/αφαιρέτη των N bit, μονάδα λογικών πράξεων των N bit, πύλη NOR των N bit για τη σημαία Z και τους απαραίτητους πολυπλέκτες 2 σε 1 των N bit. Η μονάδα ALU παράγει τις τιμές των σημαιών N, Z, C, V. Οι σημαίες έχουν όλες την τιμή ‘0’ όταν εκτελούνται λογικές πράξεις. Το μέγεθος του σήματος ελέγχου ALUControl εξαρτάται από το πλήθος των πράξεων που εκτελούνται στη μονάδα ALU.

2-4. **Ανάγνωση ή εγγραφή στη μνήμη δεδομένων**

2-4.1. Παραμετροποιημένη διάταξη μνήμης RAM με 2^N λέξεις μεγέθους M bit (με αρχικές τιμές N = 5 και M = 32) για να χρησιμοποιηθεί ως μνήμη δεδομένων (**data memory, DM**). Προσοχή! A[N-1:0] = ALUResult[N+1:2]. Λόγω του μικρού μεγέθους υλοποιείται ως distributed RAM. Το διάβασμα γίνεται ασύγχρονα, ενώ η εγγραφή γίνεται

σύγχρονα, όταν MemWrite = '1' (WE='1').

2-5. **Ετεροχρονισμένη εγγραφή ενός καταχωρητή στο αρχείο καταχωρητών και επιλογή της διεύθυνσης της επόμενης εντολής που πρόκειται να εκτελεσθεί.**

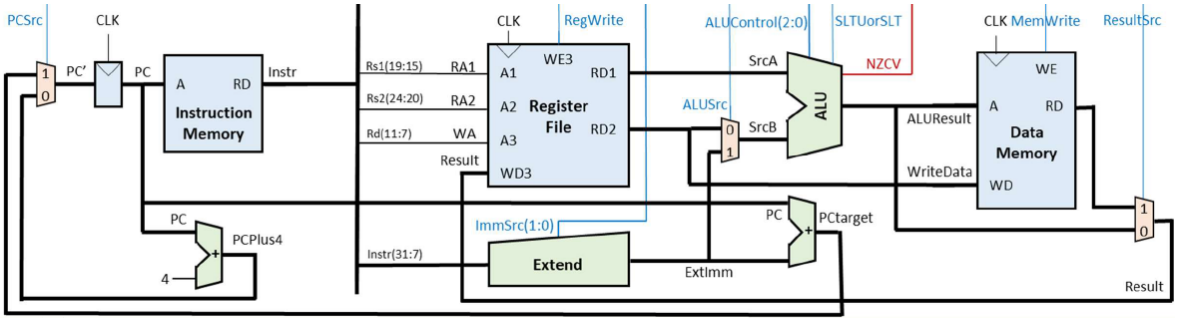
2-5.1. Παραμετροποιημένος πολυπλέκτης 2 σε 1 των N bit (με αρχική τιμή N = 32) για να χρησιμοποιηθεί στην επιλογή των δεδομένων που εγγράφονται ετεροχρονισμένα στον καταχωρητή προορισμού Rd του αρχείου καταχωρητών (**mux2to1**). Επιλέγει μεταξύ του αποτελέσματος ALUResult της πράξης που εκτελείται στη μονάδα ALU (ResultSrc = 0) και της θύρας ανάγνωσης RD της μνήμης δεδομένων (ResultSrc = 1)

2-5.2. Παραμετροποιημένος αθροιστής κατά 4 με έξοδο PCtarget για τον υπολογισμό της διεύθυνσης της αμέσως επόμενης εντολής PC + Extimm (Μόνο σε εντολές Branch).

2-5.3. Παραμετροποιημένος πολυπλέκτης 2 σε 1 των N bit (με αρχική τιμή N = 32) για να χρησιμοποιηθεί στην επιλογή της διεύθυνσης της επόμενης εντολής που πρόκειται να εκτελεσθεί (**mux2to1**). Επιλέγει μεταξύ της τιμής PC + 4 (PCSrc = 0) και του αποτελέσματος PCtarget (PCSrc = 1). Η έξοδος συνδέεται στην είσοδο δεδομένων του μετρητή προγράμματος PC. Η τιμή PC + 4 επιλέγεται ως διεύθυνση της επόμενης εντολής που πρόκειται να εκτελεσθεί, όταν εκτελούνται εντολές που δεν αλλάζουν τη ροή του προγράμματος και όταν δεν ικανοποιείται η συνθήκη κατά την εκτέλεση εντολών υπό συνθήκη. Το αποτέλεσμα PCtarget επιλέγεται ως διεύθυνση της επόμενης εντολής που πρόκειται να εκτελεσθεί όταν εκτελείται εντολή διακλάδωσης.

Βήμα 3: Σχεδίαση της διαδρομής δεδομένων (datapath)

Η σχεδίαση της διαδρομής δεδομένων (**datapath**) του επεξεργαστή ενός κύκλου ολοκληρώνεται με τη χρήση περιγραφής δομής στη γλώσσα VHDL ακολουθώντας την ιεραρχική προσέγγιση bottom-up, όπως αυτή αποτυπώνεται στο ακόλουθο σχηματικό διάγραμμα.



Βήμα 4: Σχεδίαση της μονάδας ελέγχου

Αρχικά, στο κατώτερο ιεραρχικά επίπεδο της σχεδίασης περιγράφεται στη γλώσσα VHDL η συμπεριφορά των υπομονάδων που απαρτίζουν τη μονάδα ελέγχου. Στη συνέχεια, η σχεδίαση της μονάδας ελέγχου (**control**) του επεξεργαστή ενός κύκλου ολοκληρώνεται με τη χρήση περιγραφής δομής στη γλώσσα VHDL ακολουθώντας την ιεραρχική προσέγγιση bottom-up.

4-1. **Σχεδίαση του αποκωδικοποιητή εντολής (InstrDec).**

4-1.1. Συμπλήρωση του πίνακα αλήθειας του αποκωδικοποιητή εντολής (**InstrDec**).

Είσοδος το πεδίο op (Instr(6:2)) της εντολής. Έξοδοι τα σήματα ελέγχου ResultSrc, MemWrite, ALUSrc, ImmSrc(1:0), και RegWrite, καθώς και το εσωτερικό σήμα rop(2:0).

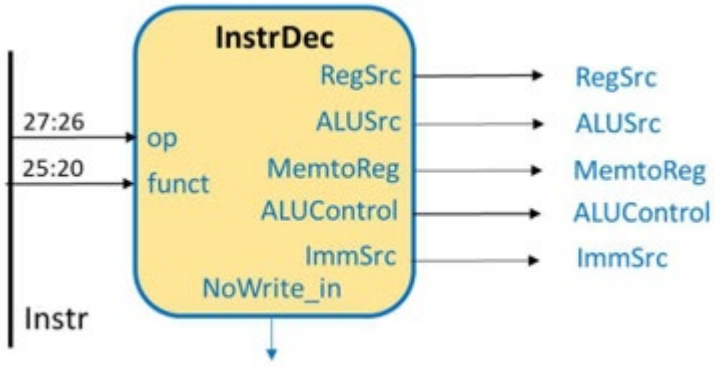
Δίδεται ως παράδειγμα ο πίνακας αλήθειας για τις εντολές LW, SW, ALU και ALUI.

Εντολή	op(6:2)	rop(2:0)	Result Src	Mem Write	ALU Src	Imm Src(1:0)	Reg Write
LW	00000	000	1	0	1	00	1
ALUI	00100	001	0	0	1	00	1
SW	01000	010	X	1	1	01	0
ALU	01100	011	0	0	0	XX	1

Τα σήματα MemWrite και RegWrite έχουν την τιμή 0 όταν δεν χρησιμοποιούνται η μνήμη δεδομένων και το αρχείο καταχωρητών.

4-1.2. Με βάση τον πίνακα αλήθειας γίνεται η σχεδίαση του αποκωδικοποιητή εντολής (**InstrDec**) σε γλώσσα VHDL (περιγραφή συμπεριφοράς με τη χρήση της εντολής case). Προσοχή! Η αξιοποίηση των αδιάφορων τιμών X στην έξοδο οδηγούν σε απλοποίηση του συνδυαστικού κυκλώματος.

4-1.3. Το σχηματικό διάγραμμα του αποκωδικοποιητή εντολής (**InstrDec**) φαίνεται στη συνέχεια.



4-2. **Σχεδίαση του αποκωδικοποιητή μονάδας ALU (ALUDec).**

4-2.1. Συμπλήρωση του πίνακα αλήθειας του αποκωδικοποιητή της μονάδας ALU (**ALUDec**).

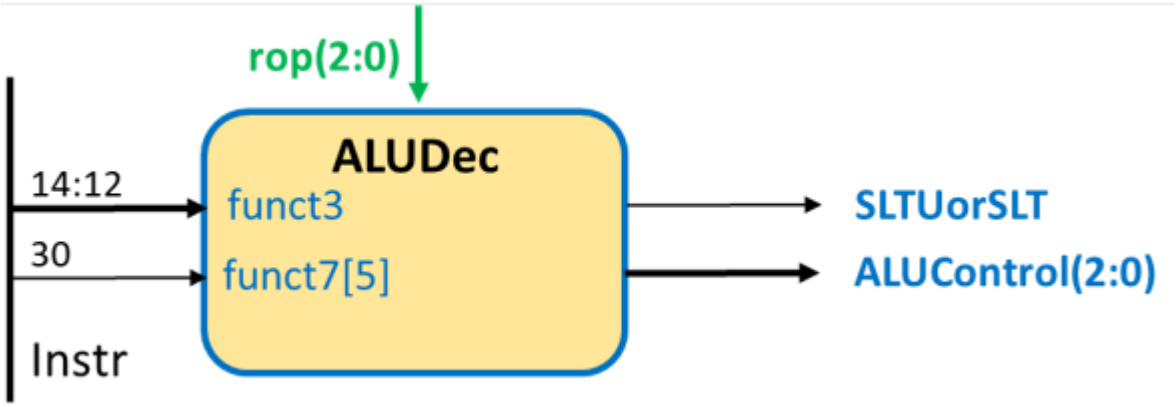
Είσοδοι, τα πεδία funct3 (Instr(14:12)) και funct7(5)(Instr(30)) της εντολής, καθώς και το εσωτερικό σήμα rop(2:0). Έξοδοι, τα εσωτερικά σήματα ελέγχου της μονάδας ALU: ALUControl(2:0) και SLTUorSLT.

Δίδεται ως παράδειγμα ο πίνακας αλήθειας για εντολές επεξεργασίας δεδομένων και μνήμης.

Εντολή	op(6:0)	Rop(2:0)	funct3	Funct7 (5)	ALU πράξη	ALUControl(2:0)	SLTUorSLT
ADD	**0110011**	**010**	**000**	**0**	**add**	**000**	**X**
SUB	**0110011**	**010**	**000**	**1**	**sub**	**001**	**X**
AND	**0110011**	**010**	**111**	**0**	**and**	**100**	**X**
OR	**0110011**	**010**	**110**	**0**	**or**	**101**	**X**
XOR	**0110011**	**010**	**100**	**0**	**xol**	**11X**	**X**
ADDI	**0010011**	**011**	**000**	**X**	**add**	TBD	**X**
ANDI	**0010011**	**011**	**111**	**X**	**and**	**TBD**	**X**
ORI	**0010011**	**011**	**110**	**X**	**or**	TBD	**X**
XORI	**0010011**	**011**	**100**	**X**	**xol**	**01X**	**0**
LW	**0000011**	**000**	**010**	**X**	**add**	**01X**	**1**
SW	**0100011**	**001**	**010**	**X**	**add**	**000**	**X**

4-2.2. Με βάση τον πίνακα αλήθειας γίνεται η σχεδίαση του αποκωδικοποιητή της μονάδας ALU (**ALUDec**) σε γλώσσα VHDL (περιγραφή συμπεριφοράς με τη χρήση της εντολής case).

4-2.3. Το σχηματικό διάγραμμα του αποκωδικοποιητή της μονάδας ALU (**ALUDec**) φαίνεται στη συνέχεια.



4-3. **Σχεδίαση της λογικής επιλογής διεύθυνσης επόμενης εντολής (PCLogic).**

4-3.1. Συμπλήρωση του πίνακα αλήθειας της λογικής επιλογής διεύθυνσης επόμενης εντολής (**PCLogic**) που ενεργοποιεί το εσωτερικό σήμα PCSrc όταν εκτελείται εντολή διακλάδωσης.

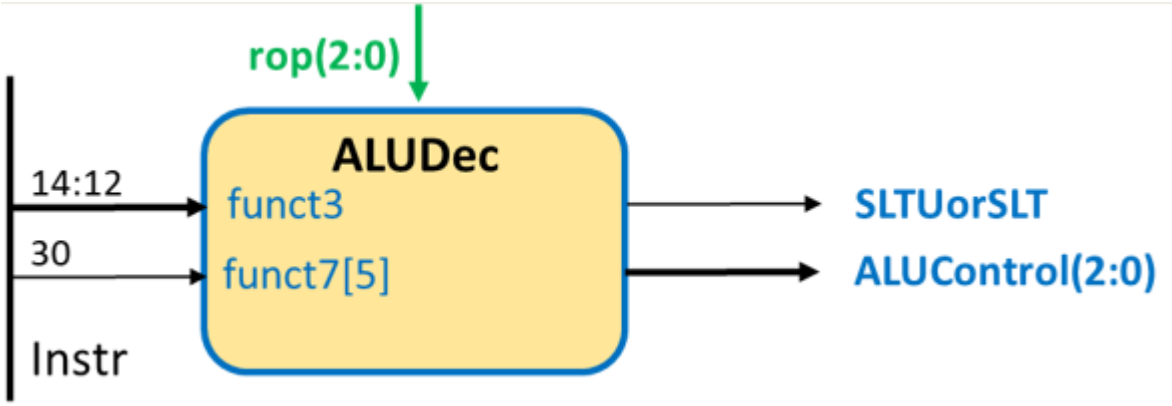
Είσοδοι, το πεδίο funct3 (Instr(14:12)) και το εσωτερικό σήμα rop(2:0) και εξετάζει εάν ικανοποιείται ή όχι η συνθήκη στις εντολές Branch με βάση τις τιμές των ALU και παράγει το σήμα ελέγχου PCSrc.

Δίδεται ως παράδειγμα ο πίνακας αλήθειας για εντολές επεξεργασίας δεδομένων, μνήμης και διακλάδωσης.

Εντολή	op(6:0)	rop(2:0)	funct3	NZCV	PCSrc
LW	**00000**	**000**	**XXX**	**XXXX**	**0**
SW	**01000**	**001**	**XXX**	**XXXX**	**0**
ALU	**01100**	**010**	**XXX**	**XXXX**	**0**
ALUI	**00100**	**011**	**XXX**	**XXXX**	**0**

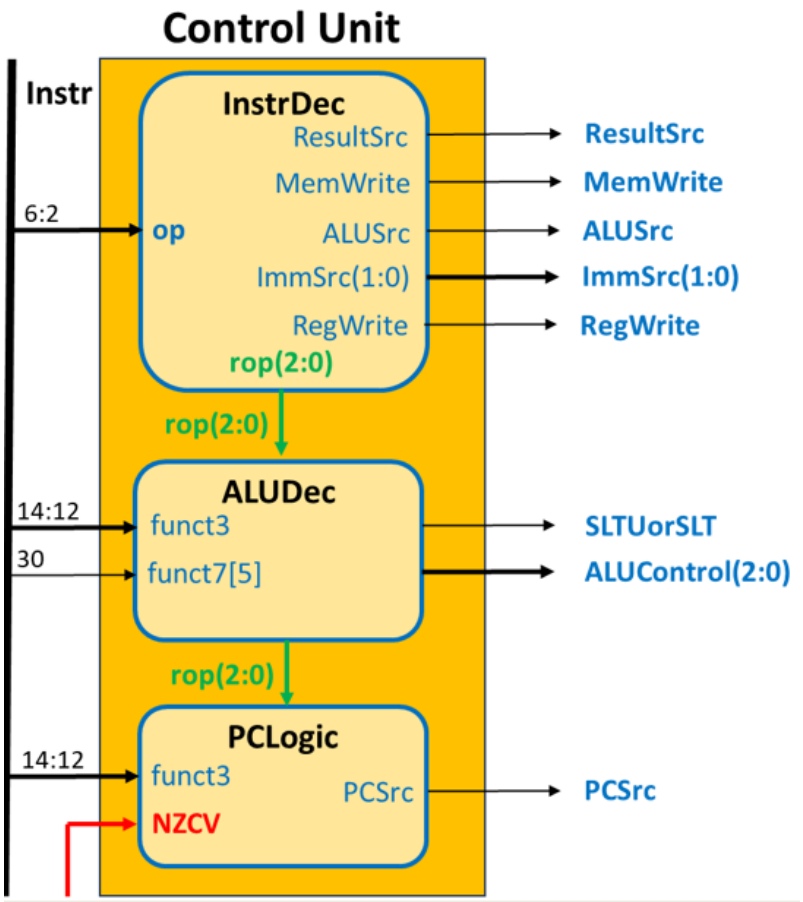
4-3.2. Με βάση τον πίνακα αλήθειας γίνεται η σχεδίαση της λογικής επιλογής διεύθυνσης επόμενης εντολής (**PCLogic**) σε γλώσσα VHDL (περιγραφή συμπεριφοράς με τη χρήση της εντολής case).

4-3.3. Το σχηματικό διάγραμμα της λογικής επιλογής διεύθυνσης επόμενης εντολής (**PCLogic**) φαίνεται στη συνέχεια.



4-4. **Σχεδίαση της μονάδας ελέγχου (control)**

Η σχεδίαση της μονάδας ελέγχου (**control**) του επεξεργαστή ενός κύκλου ολοκληρώνεται με τη χρήση περιγραφής δομής στη γλώσσα VHDL ακολουθώντας την ιεραρχική προσέγγιση bottom-up, όπως αυτή αποτυπώνεται στο ακόλουθο σχηματικό διάγραμμα.

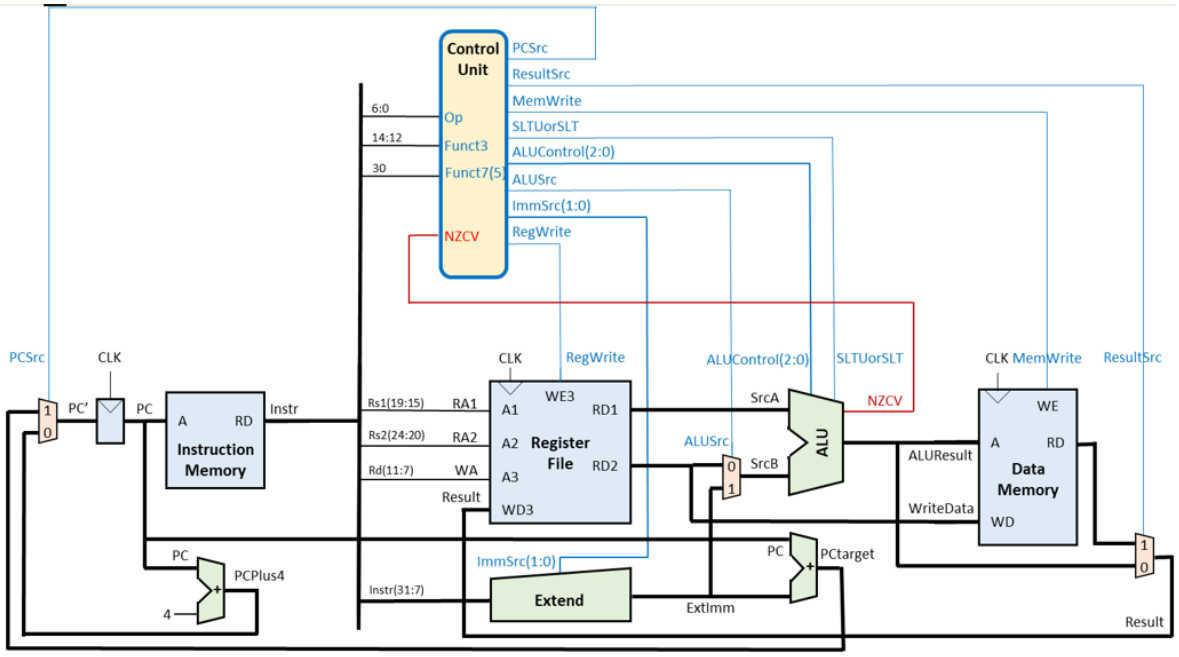


Βήμα 5: Σχεδίαση του επεξεργαστή (processor)

Στο υψηλότερο ιεραρχικά επίπεδο σχεδιάζεται ο επεξεργαστής (**processor**) με τη χρήση περιγραφής δομής στη γλώσσα VHDL ακολουθώντας την ιεραρχική προσέγγιση bottom-up, όπως αυτή αποτυπώνεται στο ακόλουθο σχηματικό διάγραμμα.

Στο πλαίσιο του Project 1, επιλέγεται ο επεξεργαστής να έχει ως εξόδους τις αρτηρίες **PC** και **Instr**, που σχετίζονται με τη μνήμη εντολών, και τις αρτηρίες **ALUResult**, **WriteData** και **Result**, που σχετίζονται με τη μνήμη δεδομένων και τη μονάδα ALU. Οι είσοδοι του επεξεργαστή είναι τα σήματα CLK και RESET. Πριν την υλοποίηση θα πρέπει να φορτωθεί το απαραίτητο πρόγραμμα στη μνήμη ROM που να συμπεριλαμβάνει όλες τις εντολές που υλοποιούνται. Προσοχή! Θα πρέπει να μελετηθούν προσεκτικά οποιεσδήποτε ανεπιθύμητες απλοποιήσεις που ενδεχομένως να γίνουν στο στάδιο της υλοποίησης.

Η διασύνδεση της διαδρομής δεδομένων (**datapath**) και της μονάδας ελέγχου (**control**) φαίνεται στο επόμενο σχηματικό διάγραμμα στο επίπεδο του επεξεργαστή (**processor**).



Βήμα 6: Επαλήθευση της ορθής σχεδίασης του επεξεργαστή (processor)

Η επαλήθευση της ορθής σχεδίασης του επεξεργαστή (**processor**) που βασίζεται στην προσομοίωση (simulation based verification) εφαρμόζεται σε επιλεγμένες υπομονάδες της διαδρομής δεδομένων, όπως η μονάδα ALU (**ALU**) και το αρχείο καταχωρητών (**RF**), στη μονάδα ελέγχου (**control**) και στον επεξεργαστή (**processor**) σαν ολότητα.

6-1. **Επαλήθευση της ορθής σχεδίασης της μονάδας ALU (ALU).**

6-1.1. Ακολουθήστε όλα τα Βήματα 1-5 της ιεραρχικής σχεδίασης της οντότητας της μονάδας ALU (**ALU**), όπως αυτά περιγράφονται στον οδηγό χρήσης των βασικών λειτουργειών του **Vivado IDE** δημιουργώντας και το απαραίτητο πρόγραμμα δοκιμής (**ALU_TB**), που εκτελεί όλες τις πράξεις. Χρησιμοποιείστε ως βάση τον αθροιστή με καταχωρητές εισόδου και εξόδου.

6-2. **Επαλήθευση της ορθής σχεδίασης του επεξεργαστή (processor).**

6-2.1. Ακολουθήστε όλα τα Βήματα 1-5 της ιεραρχικής σχεδίασης της οντότητας του επεξεργαστή (**processor**), όπως αυτά περιγράφονται στον οδηγό χρήσης των βασικών λειτουργειών του Vivado IDE δημιουργώντας και το απαραίτητο πρόγραμμα δοκιμής (**processor_TB**). Προσοχή! Το πλήθος των κύκλων ρολογιού που θα τρέξει ο επεξεργαστής πρέπει να αντιστοιχεί στο πλήθος των εντολών που είναι αποθηκευμένες στη μνήμη ROM. Σε κάθε περίπτωση, η τελευταία εντολή του προγράμματός σας επαναφέρει τη συνέχεια της εκτέλεσης του προγράμματος στην πρώτη εντολή του.

6-2.2. Δημιουργήστε ένα πρόγραμμα σε συμβολική γλώσσα της αρχιτεκτονικής RISC-V, που να συμπεριλαμβάνει όλες τις εντολές που υλοποιείτε και ενεργοποιεί όλες τις πιθανές ροές δεδομένων και λειτουργίες στη διαδρομή δεδομένων.

Για τη δημιουργία του προγράμματος μπορείτε να χρησιμοποιήσετε έναν ελεύθερο συμβολομεταφραστή (assembler) της αρχιτεκτονικής RISC-V, όπως είναι ο ακόλουθος της της εταιρείας SYSPROGS (https://gnutoolchains.com/risc-v/), ο οποίος στην πραγματικότητα είναι cross assembler υπό την έννοια ότι παράγει μεν κώδικα σε γλώσσα μηχανής RISC-V, ο ίδιος όμως δεν τρέχει σε επεξεργαστή αρχιτεκτονικής RISC-V αλλά αρχιτεκτονικής X86/X64.

Κάνετε download την επιλογή

2.29	7.7.1	2.5.0	8.0	risc-v-gcc7.7.1.exe (141 MB)

και κάνετε εγκατάσταση το εκτελέσιμο. Ύστερα, έστω ότι έχετε δημιουργήσει ένα μικρό πρόγραμμα σε assembly RISC-V, στο αρχείο test_risc_v.s

test_risc_v.s

```
1   addi t0, x0, 256     # t0 = 0x100 (base address)
2   addi t1, x0, 7       # t1 = 7
3   sw   t1, 0(t0)       # MEM[0x100] = 7
4   lw   t2, 0(t0)       # t2 = 7
5   add  t3, t2, t2      # t3 = 14
6   sw   t3, 4(t0)       # MEM[0x104] = 14
7
```

Εκτελέστε την εφαρμογή του assembler από το command line:

riscv64-unknown-elf-as -march=rv32i -mabi=ilp32 test_risc_v.s -o test_risc_v.o

όπου

-march=rv32i → Βασικές εντολές ακεραίων

-mabi=ilp32 → Καταχωρητές των 32bit

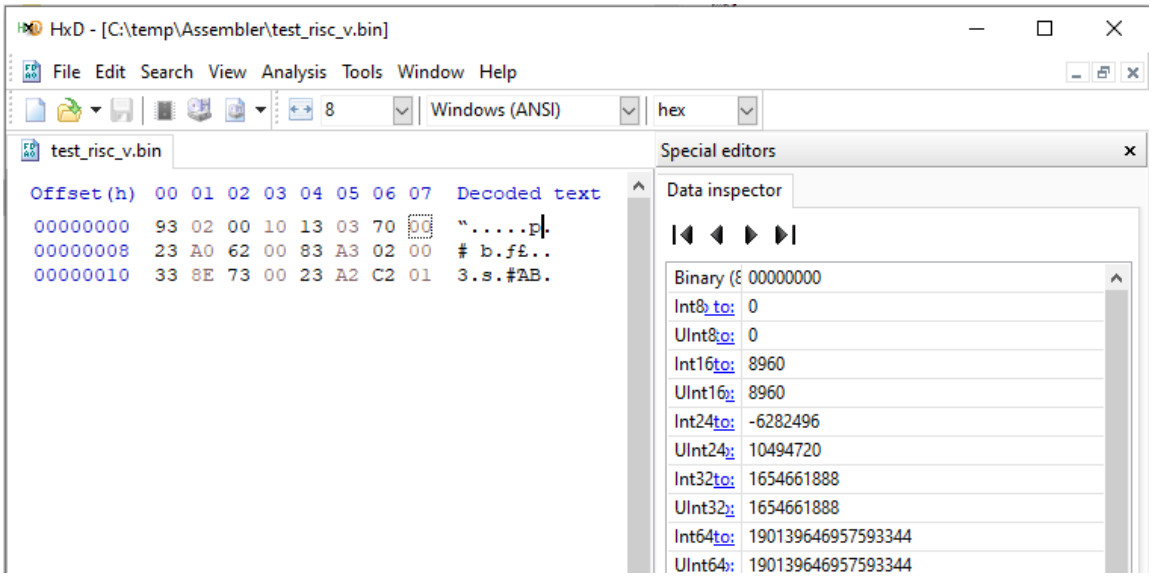
Παράγεται ένα object file (test_risc_v.o) με τις εντολές πλέον σε γλώσσα μηχανής.

Τρέχοντας κατόπιν το

riscv64-unknown-elf-objcopy -O binary test_risc_v.o test_risc_v.bin

δημιουργούμε από το .o αρχείο το αντίστοιχο .bin που περιέχει μόνο το binary κομμάτι των εντολών χωρίς επιπλέον πληροφορίες που έχει το .o file (π.χ. metadata).

6-2.3. Ανοίξτε το binary αρχείο με έναν ελεύθερο hex editor, όπως είναι ο **HxD**, αφού πρώτα τον εγκαταστήσετε στον υπολογιστή σας.



Κάθε 4 συνεχόμενα byte (1 byte = 2 δεκαεξαδικά ψηφία) αποτελούν μία εντολή, ξεκινώντας από την πρώτη κατά σειρά έως και την τελευταία, όπως αυτές εμφανίζονται στο πηγαίο αρχείο **test_risc_v.bin**.

6-2.4. Αντιγράψτε τις εντολές σε γλώσσα μηχανής στο αρχείο ROM_array.vhd που περιγράφει τη μνήμη εντολών, ως μνήμη ROM. Αντιγράψτε μία-μία τις τετράδες των byte στις κατάλληλες θέσεις του ROM_array ξεκινώντας από την πρώτη θέση. Θα χρειαστεί να αφαιρέσετε με το χέρι τα ενδιάμεσα κενά μεταξύ των διαδοχικών byte και να αντιστρέψετε τη σειρά τους, αφού το περισσότερο σημαντικό byte (most significant byte, MSB) της λέξης εντολών βρίσκεται στα αριστερά, ενώ το λιγότερο σημαντικό byte (least significant byte, LSB) της λέξης εντολών βρίσκεται στα δεξιά. Προσοχή! Στην περίπτωση που οι εντολές είναι λιγότερες από τη συνολική χωρητικότητα της μνήμης ROM θα πρέπει να γεμίσετε τις υπόλοιπες θέσεις με X”00000000”. Για παράδειγμα, για μία μνήμη ROM χωρητικότητας 16 εντολών των 32 bit (N = 4, M = 32) ο ανωτέρω κώδικας σε γλώσσα VHDL διαμορφώνεται ως εξής:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

entity ROM is

generic (

N : positive := 4; -- address length
M : positive := 32 -- data word length

);

port (

ADDR:       in STD_LOGIC_VECTOR (N-1 downto 0);
ROM_OUT:    out STD_LOGIC_VECTOR (M-1 downto 0)

);

end entity ROM;

architecture Behavioral of ROM is

type ROM_array is array (0 to 2**N-1) of STD_LOGIC_VECTOR (M-1 downto 0);

constant ROM_array_16x32 : ROM_array := (

-- ************************************************

--ALL COMMANDS


X"10000293", X"00700313", X"0062A023", X"0002A383",
X"00738E33", X"01C2A223", X"00000000", X"00000000",
X"00000000", X"00000000", X"00000000", X"00000000",
X"00000000", X"00000000", X"00000000", X"00000000");

begin

    ROM_OUT <= ROM_array_64x32(to_integer(unsigned(ADDR)));

end Behavioral;
```

6-2.5. Ολοκληρώστε την επαλήθευση της ορθής σχεδίασης του επεξεργαστή (**processor**) συγκρίνοντας για κάθε εντολή, που εκτελείται σε έναν κύκλο, τις τιμές των αρτηριών **PC** και **Instr**, που σχετίζονται με τη μνήμη εντολών, και των αρτηριών **ALUResult**, **WriteData** και **Result**, που σχετίζονται με τη μνήμη δεδομένων και τη μονάδα ALU που προκύπτουν μετά την προσομοίωση με τις αντίστοιχες αναμενόμενες τιμές που φαίνονται στον ακόλουθο πίνακα επαλήθευσης ορθής λειτουργίας του επεξεργαστή και προκύπτουν μετά από μελέτη των εντολών που έχουν αποθηκευτεί στη μνήμη εντολών και έχουν υλοποιηθεί στον επεξεργαστή. Εμφανίστε τις τιμές των καταχωρητών και των θέσεων μνήμης που επηρεάζονται, σε κάθε κύκλο. Επιπλέον μελετήστε και τιμές των σημαιών N, Z, C, V, ως εσωτερικά σήματα.

Βήμα 7: Παράδοση τεχνικής αναφοράς της σχεδίασης του επεξεργαστή

Μετά την ολοκλήρωση της σχεδίασης του επεξεργαστή (**processor**) παραδίδετε το project συνοδευόμενο από μία απλοποιημένη τεχνική αναφορά η οποία συμπεριλαμβάνει τα ακόλουθα:

7-1. **Περιγραφή των στοιχείων και της δομής του επεξεργαστή.**

7-1.1. Παραθέστε το σχηματικό διάγραμμα στο επίπεδο RTL (elaborated design) της διαδρομής δεδομένων (**datapath**) του επεξεργαστή. Περιγράψτε τα στοιχεία (components) που χρησιμοποιείτε (λειτουργία, ποια δεδομένα ανταλλάσσουν με τα component με τα οποία αλληλοεπιδρούν).

7-1.2. Παραθέστε το σχηματικό διάγραμμα στο επίπεδο RTL (elaborated design) της μονάδας ελέγχου (**control**) του επεξεργαστή. Περιγράψτε όλες τις υπομονάδες της μονάδας ελέγχου (λειτουργία, τα δεδομένα που ανταλλάσσουν με τα component με τα οποία αλληλοεπιδρούν) συμπληρώνοντας τους αντίστοιχους πίνακες αλήθειας. Εξηγήστε για κάθε εντολή πως προκύπτουν οι αντίστοιχες τιμές στου πίνακες αληθείας. Η ανάλυση θα γίνει ανά πίνακα αληθείας.

7-1.3. Περιγράψτε τη λειτουργία της ALU που έχετε σχεδιάσει (RTL διάγραμμα, περιγραφή/documentation του κώδικα, πόρους που χρησιμοποιεί στο στάδιο του implementation, μέγιστη ταχύτητα λειτουργίας,…).

7-1.4. Βρείτε τη μέγιστη συχνότητα λειτουργίας στο επίπεδο του επεξεργαστή (**processor**), προσδιορίζοντας επίσης τη χειρότερη κρίσιμη διαδρομή και τη χειρότερη σύντομη διαδρομή.

7-2. **Επαλήθευση της ορθής σχεδίασης και λειτουργίας του επεξεργαστή (processor).**

7-2.1. Δημιουργείστε κατάλληλο πρόγραμμα σε συμβολική γλώσσα αρχιτεκτονικής RISC-V, που συμπεριλαμβάνει όλες τις υλοποιημένες εντολές και ενεργοποιεί όλες (κατά το δυνατόν) τις πιθανές ροές δεδομένων και λειτουργίες στη διαδρομή δεδομένων. Τεκμηριώστε γιατί το πρόγραμμα σας σε συμβολική γλώσσα επαληθεύει την ορθή σχεδίαση και λειτουργία του επεξεργαστή (**processor**) (δηλαδή, ποιο τμήμα του προγράμματος επαληθεύει ποιες εντολές και με ποιον τρόπο).

7.2.2. Παραθέστε τα διαγράμματα χρονισμού των προσομοιώσεων της ALU (behavioral και post implementation timing) και τεκμηριώστε την ορθή του σχεδίαση και λειτουργία της ALU για κάθε πράξη που υλοποιεί.

7.2.3 Παραθέστε τα διαγράμματα χρονισμού των προσομοιώσεων του behavioral model και του post-implementation model (μόνο χρονική προσομοίωση) για τον επεξεργαστή και τεκμηριώστε την ορθή του σχεδίαση και λειτουργία.

7-3. **Ανάλυση των αποτελεσμάτων της σύνθεσης και της υλοποίησης του επεξεργαστή.**

7-3.1. Αναλύστε τα αποτελέσματα της σύνθεσης και της υλοποίησης του επεξεργαστή (**processor**) μελετώντας το project summary και το report utilization.

Οι εντολές RISC-V που πρέπει να υλοποιηθούν είναι οι:

LW, SW, ADD, SUB, OR, XOR, ADDI, ANDI, ORI, XORI

Τα datapath, control unit και γενικότερα όλος ο επεξεργαστής θα πρέπει να σχεδιαστεί ώστε να υλοποιεί μόνο τις συγκεκριμένες εντολές. Οπότε σήματα ή δομές που δεν χρειάζονται για την περίπτωσή μας (παρότι περιγράφονται στη θεωρία για τον επεξεργαστή ή και στην εκφώνηση της άσκησης για λόγους πληρότητας) δεν θα πρέπει να υλοποιηθούν.

Προσοχή. Η τεχνική αναφορά θα πρέπει να ακολουθεί πλήρως τη δομή που περιγράφεται στις παραγράφους 7.1. έως και 7.3. ΔΕΝ πρέπει να συγχωνεύσετε τις απαντήσεις σας, για κάθε ενότητα θα πρέπει να υπάρχει διακριτή ανάλυση. Αναμένουμε λοιπόν μια αναφορά 3 κεφαλαίων (χωρίς την εισαγωγή ή πιθανά παραρτήματα) και τον κώδικα υλοποίησης μαζί με τα απαραίτητα testbench.