UNIVERSITY OF
PATRAS
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών & Πληροφορικής

Διπλωματική Εργασία

Υλοποίηση του RISC-V με την τεχνική των μερικώς επικαλυπτόμενων λειτουργιών

Γιαννάκης Εμμανουήλ Δημήτριος

Α.Μ. : 1067491

Επιβλέπων

Νικολός Δημήτριος, Καθηγητής

Μέλη Εξεταστικής Επιτροπής

Βέργος Χαρίδημος, Καθηγητής

Ζερβάκης Γεώργιος, Επίκουρος Καθηγητής

Ευχαριστίες

Περίληψη

Θα γραφεί κατά την ολοκλήρωση του κειμένου.

Abstract

Θα γραφεί κατά την ολοκλήρωση του κειμένου.

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ

1.1 Ο επεξεργαστής

Οι επεξεργαστές αποτελούν τον πυρήνα των υπολογιστικών συστημάτων, εκτελούν ποικίλες εργασίες με μεγάλη ταχύτητα και αποτελεσματικότητα. Αναλαμβάνουν την επεξεργασία δεδομένων, την εκτέλεση εντολών και τον συντονισμό των λειτουργιών του υπολογιστικού συστήματος. Οι εξελίξεις στην τεχνολογία των επεξεργαστών έχουν οδηγήσει σε συνεχή αύξηση της ταχύτητας, της απόδοσης και της ενεργειακής αποδοτικότητας. Η σχεδίαση των επεξεργαστών γίνεται σύμφωνα με διάφορες αρχιτεκτονικές.

1.2 Αρχιτεκτονική συνόλου εντολών

Η αρχιτεκτονική συνόλου εντολών (Instruction Set Architecture - ISA) είναι μέρος του abstract μοντέλου ενός υπολογιστή που καθορίζει τον τρόπο με τον οποίο η CPU ελέγχεται από το λογισμικό. Η ISA αποτελεί τη σύνδεση μεταξύ του υλικού και του λογισμικού, καθορίζοντας το τι είναι ικανός να κάνει ο επεξεργαστής, αλλά και τον τρόπο με τον οποίο γίνεται αυτό. Μπορεί να θεωρηθεί ως το εγχειρίδιο του προγραμματιστή, επειδή είναι το τμήμα της μηχανής που είναι ορατό στον προγραμματιστή. Η ISA ορίζει τους υποστηριζόμενους τύπους δεδομένων, τους καταχωρητές, τον τρόπο με τον οποίο το υλικό διαχειρίζεται την μνήμη, ποιες εντολές μπορεί να εκτελέσει ένας επεξεργαστής. Τα αποτελέσματα της εκτέλεσης των εντολών μπορούν να είναι αριθμητικά αποτελέσματα, αλλαγή της ροής του προγράμματος με βάση μια συνθήκη ή κατάσταση, αλλαγή της κατάστασης που βρίσκεται ο επεξεργαστής. Οι κατηγορίες για τις αρχιτεκτονικές συνόλου εντολών είναι οι εξής:

1. Reduced Instruction Set Computer (RISC)
   Το πλήθος των εντολών είναι περιορισμένο και οι εντολές αυτές εκτελούν βασικές λειτουργίες. Κάθε εντολή εκτελεί μια απλή λειτουργία γενικού σκοπού. Υπάρχουν ξεχωριστές εντολές για την επεξεργασία δεδομένων και την προσπέλαση της μνήμης.

2. Complex Instruction Set Computer (CISC)
   Το πλήθος των εντολών είναι μεγάλο και οι εντολές αυτές εκτελούν λειτουργίες ειδικού σκοπού. Κάθε εντολή εκτελεί μια εξειδικευμένη λειτουργία.

1.3 Αρχιτεκτονική RISC-V

Η αρχιτεκτονική RISC-V ανήκει στην αρχιτεκτονική RISC και είναι η πρώτη open-source αρχιτεκτονική συνόλου εντολών. Δημιουργήθηκε το 2010 στο Πανεπιστήμιο της Καλιφόρνια, Berkeley από τους Krste Asanović, Andrew Waterman, David Patterson και άλλους. Από τη δημιουργία της έως και σήμερα πολλά άτομα έχουν συμβάλει στην ανάπτυξή της [1]. Για τον σχεδιασμό της RISC-V δόθηκε έμφαση στην απλότητα και στην επεκτασιμότητα. Όσον αφορά την απλότητα, η RISC-V στη βάση της, ορίζεται ως μια μικρή αρχιτεκτονική συνόλου εντολών που αφορά ακέραιους αριθμούς και οι εντολές αυτές πρέπει να είναι παρούσες σε όλες τις υλοποιήσεις. Κάθε σύνολο εντολών για ακέραιους αριθμούς χαρακτηρίζεται από το μήκος των καταχωρητών και το μήκος των διευθύνσεων. Οι δύο βασικές παραλλαγές της RISC-V ISA είναι οι RV32I και RV64I οι οποίες αναφέρονται σε μήκος διεύθυνσης 32-bit και 64-bit αντίστοιχα. Όσον αφορά την επεκτασιμότητα, υπάρχουν επεκτάσεις (“extensions”) του συνόλου εντολών, όπως για παράδειγμα η «F» η οποία αφορά πράξεις με αριθμούς κινητής υποδιαστολής. Οι επεκτάσεις χωρίζονται σε standard και non-standard. Οι standard θεωρούνται εκείνες που είναι γενικά χρήσιμες και είναι

σχεδιασμένες έτσι ώστε να διαχωρίζονται με σαφήνεια από τις υπόλοιπες standard. Οι non-standard θεωρούνται εκείνες που είναι υψηλά εξιδεικευμένες και ίσως να μην διαχωρίζονται με σαφήνεια από τις υπόλοιπες standard και non-standard [2]. Παρακάτω παρουσιάζονται κάποιες standard επεκτάσεις:

RV32I/RV32E/RV64I/RV128I Base Integer Instruction Set

Το βασικό σύνολο εντολών για 32, 64, 128 bit αντίστοιχα. Περιλαμβάνει εντολές για αριθμητικές και λογικές πράξεις ακέραιων αριθμών, εντολές για ανάγνωση και προσπέλαση μνήμης.

"M" extension Multiplication - Division

Το σύνολο αυτό περιλαμβάνει εντολές για πράξεις πολλαπλασιασμού και διαίρεσης. Γίνεται διαχωρισμός από το Base Integer Instruction Set για να απλουστευθούν οι low-end υλοποιήσεις που δεν προβλέπουν τέτοιες πράξεις.

"A" extension Atomic Instructions

Το σύνολο αυτό περιλαμβάνει εντολές για ανάγνωση, τροποποίηση, εγγραφή της μνήμης για τον συγχρονισμό των πυρήνων ενός RISC-V επεξεργαστή που μοιράζονται την ίδια μνήμη.

"F" extension Single-Precision Floating-Point

Το σύνολο αυτό περιλαμβάνει εντολές για πράξεις με αριθμούς κινητής υποδιαστολής. Η επέκταση αυτή είναι συμβατή με το αριθμητικό πρότυπο IEEE 754-2008. Γίνεται διαχωρισμός από το Base Integer Instruction Set για να απλουστευθούν οι low-end υλοποιήσεις που δεν προβλέπουν τέτοιες πράξεις.

"D" extension Double-Precision Floating-Point

Το σύνολο αυτό περιλαμβάνει εντολές για πράξεις με αριθμούς κινητής υποδιαστολής διπλής ακρίβειας. Βασίζεται στην επέκταση «F».

"Q" extension Quad-Precision Floating-Point

Το σύνολο αυτό περιλαμβάνει εντολές για πράξεις με αριθμούς κινητής υποδιαστολής τετραπλής ακρίβειας. Βασίζεται στις επεκτάσεις «F» και «D».

"C" extension Compressed Instructions

Το σύνολο αυτό περιλαμβάνει συμπιεσμένες εντολές κωδικοποιημένες στα 16-bit οι οποίες αφορούν βασικές λειτουργίες. Αυτό έχει ως αποτέλεσμα τη μείωση του μεγέθους του κώδικα.

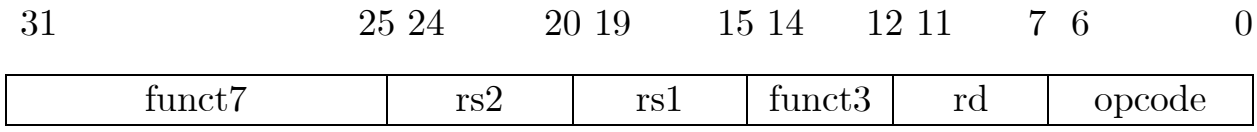
2. ΣΥΝΟΛΟ ΕΝΤΟΛΩΝ RV32I

Το σύνολο RV32I είναι το βασικό σύνολο εντολών της αρχιτεκτονικής RISC-V, όπου το 32 αντιπροσωπεύει το μέγεθος των διανυσμάτων των εντολών. Παρακάτω θα αναλυθούν οι εντολές του συνόλου, καθώς και τύποι δυαδικής κωδικοποίησης αυτών των εντολών.

2.1 Τύποι κωδικοποίησης εντολών

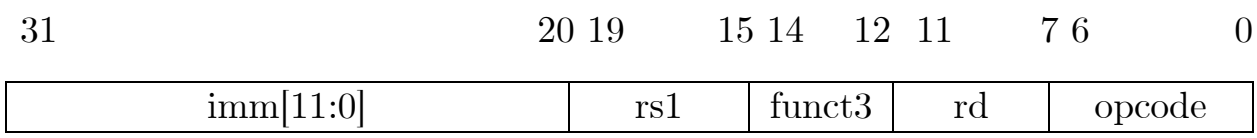
Στο σύνολο εντολών RV32I εμφανίζονται έξι διαφορετικοί τύποι κωδικοποίησης και καθένας προορίζεται για διαφορετική ομάδα εντολών. Πιο αναλυτικά:

R-type:



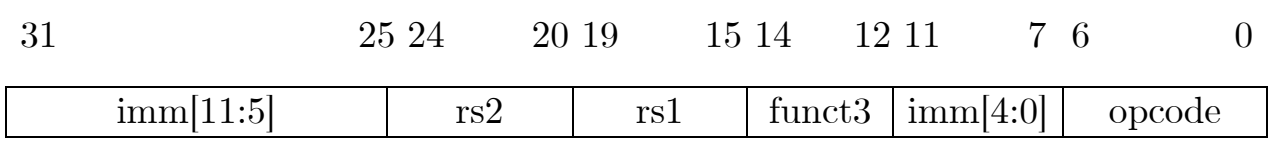
Σε αυτό τον τύπο κωδικοποίησης αξιοποιούνται τρία πεδία καταχωρητών rs1, rs2 και rd. Παρατηρούνται επίσης και τα πεδία funct7 και funct3. Η ομάδα εντολών στην οποία συναντάται αυτός ο τύπος είναι οι εντολές πράξεων μεταξύ δυο καταχωρητών (Register – Register).

I-type:



Σε αυτό τον τύπο κωδικοποίησης αξιοποιούνται δύο πεδία καταχωρητών rs1 και rd. Παρατηρούνται επίσης το πεδίο άμεσου δεδομένου imm και το funct3. Η ομάδα εντολών στην οποία συναντάται αυτός ο τύπος είναι οι εντολές πράξεων μεταξύ ενός καταχωρητή κι ενός άμεσου δεδομένου (Register - Immediate) καθώς και από τις εντολές ανάγνωσης της μνήμης δεδομένων (Load).

S-type:



Σε αυτό τον τύπο κωδικοποίησης αξιοποιούνται δύο πεδία καταχωρητών rs1 και rs2. Παρατηρούνται επίσης δύο πεδία για το άμεσο δεδομένο imm και το funct3. Η ομάδα εντολών στην οποία συναντάται αυτός ο τύπος είναι οι εντολές εγγραφής στην μνήμη δεδομένων (Store).

B-type:

imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode

Σε αυτό τον τύπο κωδικοποίησης αξιοποιούνται δύο πεδία καταχωρητών rs1 και rs2. Παρατηρούνται επίσης δύο πεδία για το άμεσο δεδομένο imm και το funct3. Η ομάδα εντολών στην οποία συναντάται αυτός ο τύπος είναι οι εντολές διακλάδωσης υπό συνθήκη (Branch).

U-type:

31	12	11	7	6	0

imm[31:12]	rd	opcode

Σε αυτό τον τύπο κωδικοποίησης αξιοποιείται ένα πεδίο για τον καταχωρητή rd. Παρατηρείται επίσης ένα πεδίο για το άμεσο δεδομένο imm. Η ομάδα εντολών στην οποία συναντάται αυτός ο τύπος είναι οι εντολές LUI (Load Upper Immediate) και AUIPC.

J-type:

31	12	11	7	6	0

imm[20|10:1|11|19:12]	rd	opcode

Σε αυτό τον τύπο κωδικοποίησης αξιοποιείται ένα πεδίο για τον καταχωρητή rd. Παρατηρείται επίσης ένα πεδίο για το άμεσο δεδομένο imm. Η ομάδα εντολών στην οποία συναντάται αυτός ο τύπος είναι οι εντολές άλματος (Unconditional Jumps).

Παρατηρείται πως σε καθεμιά κατηγορία υπάρχει διαφορετικός αριθμός πεδίων που έχουν διάφορες χρήσεις. Όμως, τα κοινά πεδία μεταξύ των

εντολών είναι στην ίδια θέση, αυτό το χαρακτηριστικό προσφέρει απλότητα στον σχεδιασμό και κάνει την παραγωγή των δυαδικών κωδικοποιήσεων των εντολών πιο εύκολη [3]. Παρακάτω ακολουθεί πίνακας που επεξηγεί κάθε ξεχωριστό πεδίο από τους παραπάνω τύπους κωδικοποίησης.

Πεδίο	Μέγεθος	Περιγραφή	Θέση
rs1	5 bit	Ο πρώτος καταχωρητής που θα γίνει ανάγνωσή του από το αρχείο καταχωρητών	19 – 15 bits
rs2	5 bit	Ο δεύτερος καταχωρητής που θα γίνει ανάγνωσή του από το αρχείο καταχωρητών	24 – 20 bits
rd	5 bit	Ο καταχωρητής στον οποίο αποθηκεύεται το αποτέλεσμα	11 – 7 bits
opcode	7 bit	Ο μοναδικός κωδικός εκτέλεσης για κάθε ομάδα εντολών	0 – 6 bits
funct3	3 bit	Εξειδικεύει περισσότερο τη λειτουργία της εντολής	14 – 12 bits
funct7	7 bit	Εξειδικεύει περισσότερο τη λειτουργία της εντολής	31 – 25 bits
imm	Ανάλογα με τον τύπο κωδικοποίησης	Άμεσο δεδομένο που χρησιμοποιείται στην εκτέλεση της εντολής	Ανάλογα με τον τύπο κωδικοποίησης

Πίνακας 1: Πεδία της κωδικοποίησης των εντολών του συνόλου RV32I

2.2 Εντολές αριθμητικών και λογικών πράξεων

2.2.1 Εντολές για πράξεις μεταξύ δυο καταχωρητών

Η κατηγορία αυτών των εντολών περιλαμβάνει όλες τις εντολές οι οποίες εκτελούν μια μαθηματική ή λογική πράξη μεταξύ των δεδομένων που περιέχουν δυο καταχωρητές (Register - Register). Ακολουθεί πίνακας με αυτές τις εντολές:

Διευκρινήσεις:

1) Όπου εμφανίζεται ο συμβολισμός c(x0), συμβολίζει το περιεχόμενο του καταχωρητή x0. Το ίδιο ισχύει και για τους υπόλοιπους καταχωρητές.
2) Οι καταχωρητές που χρησιμοποιούνται έχουν επιλεχθεί τυχαία.

Εντολή	Πράξη	Λειτουργία
ADD x0, x1, x2	x0 ‹– x1 + x2	Αριθμητική πρόσθεση των c(x1), c(x2). Αποθήκευση του αποτελέσματος στον x0.
SUB x0, x1, x2	x0 ‹– x1 - x2	Αριθμητική αφαίρεση των c(x1), c(x2). Αποθήκευση του αποτελέσματος στον x0.

SLL x0, x1, x2	x0 ‹– x1 << x2[4:0]	Λογική ολίσθηση του c(x1) κατά X θέσεις αριστερά. Η τιμή X ισούται με τα 5 λιγότερο σημαντικά bit της τιμής c(x2). Αποθήκευση του αποτελέσματος στον καταχωρητή x0.
SLT x0, x1, x2	(x1 < x2) ? x0 ‹– 1 : 0	Το c(x0) παίρνει την τιμή 1 αν c(x1) < c(x2), αλλιώς παίρνει την τιμή 0.
SLTU x0, x1, x2	(x1 < x2) ? x0 ‹– 1 : 0	Ίδια λειτουργία με SLT. Οι αριθμοί θεωρούνται μη προσημασμένοι.
XOR x0, x1, x2	x0 ‹– x1 ^ x2	Λογική πράξη XOR ανά bit μεταξύ των c(x1), c(x2).
SRL x0, x1, x2	x0 ‹– x1 >> x2[4:0]	Λογική ολίσθηση του c(x1) κατά X θέσεις δεξιά. Η τιμή X ισούται με τα 5 λιγότερο σημαντικά bit της τιμής c(x2). Αποθήκευση του αποτελέσματος στον x0.
SRA x0, x1, x2	x0 ‹– x1 >>> x2[4:0]	Αριθμητική ολίσθηση του c(x1) κατά X θέσεις δεξιά. Η τιμή X ορίζεται ως τα 5 λιγότερο σημαντικά bit της τιμής c(x2). Αποθήκευση του αποτελέσματος στον x0.
OR x0, x1, x2	x0 ‹– x1 | x2	Λογική πράξη OR ανά bit μεταξύ των c(x1), c(x2).
AND x0, x1, x2	x0 ‹– x1 & x2	Λογική πράξη AND ανά bit μεταξύ των c(x1), c(x2).

Πίνακας 2: Εντολές Register – Register

2.2.2 Εντολές για πράξεις μεταξύ καταχωρητή – άμεσου δεδομένου

Η κατηγορία αυτών των εντολών περιλαμβάνει όλες τις εντολές οι οποίες εκτελούν μια μαθηματική ή λογική πράξη μεταξύ του περιεχομένου ενός καταχωρητή και ενός άμεσου δεδομένου (Register - Immediate). Ακολουθεί πίνακας με αυτές τις εντολές:

Διευκρινήσεις:

1) Όπου εμφανίζεται ο συμβολισμός c(x0), συμβολίζει το περιεχόμενο του καταχωρητή x0. Το ίδιο ισχύει και για τους υπόλοιπους καταχωρητές.
2) Οι καταχωρητές που χρησιμοποιούνται έχουν επιλεχθεί τυχαία.

Εντολή	Πράξη	Λειτουργία
ADDI x0, x1, imm	x0 ‹– x1 + x2	Αριθμητική πρόσθεση των c(x1), imm. Αποθήκευση του αποτελέσματος στον x0.
SLTI x0, x1, imm	(x1 < imm) ? x0 ‹– 1 : 0	Το c(x0) παίρνει την τιμή 1 αν c(x1) < imm, αλλιώς παίρνει την τιμή 0.
SLTIU x0, x1, imm	(x1 < imm) ? x0 ‹– 1 : 0	Ίδια λειτουργία με SLTI. Οι αριθμοί θεωρούνται μη προσημασμένοι.

XORI x0, x1, imm	x0 ‹– x1 ^ imm	Λογική πράξη XOR ανά bit μεταξύ των c(x1), imm.
ORI x0, x1, imm	x0 ‹– x1 ^ imm	Λογική πράξη OR ανά bit μεταξύ των c(x1), imm.
ANDI x0, x1, imm	x0 ‹– x1 ^ imm	Λογική πράξη AND ανά bit μεταξύ των c(x1), imm.
SLLI x0, x1, imm	x0 ‹– x1 << imm[4:0]	Λογική ολίσθηση του c(x1) κατά X θέσεις αριστερά. Η τιμή X ισούται με τα 5 λιγότερο σημαντικά bit του imm. Αποθήκευση του αποτελέσματος στον καταχωρητή x0.
SRLI x0, x1, imm	x0 ‹– x1 >> imm[4:0]	Λογική ολίσθηση του c(x1) κατά X θέσεις δεξιά. Η τιμή X ισούται με τα 5 λιγότερο σημαντικά bit του imm. Αποθήκευση του αποτελέσματος στον καταχωρητή x0.
SRAI x0, x1, imm	x0 ‹– x1 >>> imm[4:0]	Αριθμητική ολίσθηση του c(x1) κατά X θέσεις δεξιά. Η τιμή X ισούται με τα 5 λιγότερο σημαντικά bit της τιμής imm. Αποθήκευση του αποτελέσματος στον x0.

Πίνακας 3: Εντολές Register – Immediate

2.3 Εντολές προσπέλασης της μνήμης

Στην αρχιτεκτονική RV32I μόνο οι εντολές load και store έχουν πρόσβαση στη μνήμη [2].

2.3.1 Εντολές εγγραφής στη μνήμη (Store)

Η κατηγορία αυτών των εντολών περιλαμβάνει όλες τις εντολές οι οποίες εκτελούν μεταφορά δεδομένων από τους καταχωρητές στην μνήμη ώστε να αποθηκευτούν. Ακολουθεί πίνακας με αυτές τις εντολές:

Διευκρινήσεις:

1) Όπου εμφανίζεται ο συμβολισμός c(x0), συμβολίζει το περιεχόμενο του καταχωρητή x0. Το ίδιο ισχύει και για τους υπόλοιπους καταχωρητές.
2) Όπου εμφανίζεται ο συμβολισμός M(x), συμβολίζει το περιεχόμενο της θέσης μνήμης με διεύθυνση x και από αυτή την διεύθυνση ξεκινάει η αποθήκευση.
3) Οι καταχωρητές που χρησιμοποιούνται έχουν επιλεχθεί τυχαία.
4) Η μνήμη είναι οργανωμένη σε κελία των 8-bit ανά διεύθυνση και έχει επιλεχθεί Little Endian υλοποίηση. Αυτό σημαίνει ότι τα πιο σημαντικά byte ενός word αποθηκεύονται στις θέσεις μνήμης με μεγαλύτερη διεύθυνση.

Εντολή	Πράξη	Λειτουργία
SB x4, imm(x1)	M(c(x1) + imm) ‹– c(x4)[7:0]	Αποθήκευση του λιγότερου σημαντικού byte του c(x4) στην θέση μνήμης με διεύθυνση (c(x1) + imm).
SH x4, imm(x1)	M(c(x1) + imm) ‹– c(x4)[15:0]	Αποθήκευση του λιγότερου σημαντικού byte του c(x4) και του αμέσως επόμενου λιγότερο σημαντικού, στις θέσεις μνήμης με διεύθυνση (c(x1) + imm), (c(x1) + imm + 1) αντίστοιχα.
SW x4, imm(x1)	M(c(x1) + imm) ‹– c(x4)[31:0]	Αποθήκευση του c(x4) ξεκινώντας από το λιγότερο σημαντικό byte, στις θέσεις μνήμης με διεύθυνση (c(x1) + imm), (c(x1) + imm + 1),

		(c(x1) + imm + 2), (c(x1) + imm + 3)

Πίνακας 4: Εντολές Store

2.3.2 Εντολές ανάγνωσης της μνήμης (Load)

Η κατηγορία αυτών των εντολών περιλαμβάνει όλες τις εντολές για την μεταφορά δεδομένων από την μνήμη στους καταχωρητές. Ακολουθεί πίνακας με αυτές τις εντολές:

Διευκρινήσεις:

1) Όπου εμφανίζεται ο συμβολισμός c(x0), συμβολίζει το περιεχόμενο του καταχωρητή x0. Το ίδιο ισχύει και για τους υπόλοιπους καταχωρητές.
2) Όπου εμφανίζεται ο συμβολισμός M(x), συμβολίζει το περιεχόμενο της θέσης μνήμης με διεύθυνση x και από αυτή την διεύθυνση ξεκινάει η ανάγνωση.
3) Οι καταχωρητές που χρησιμοποιούνται έχουν επιλεχθεί τυχαία.
4) Η μνήμη είναι οργανωμένη σε κελία των 8-bit ανά διεύθυνση και έχει επιλεχθεί Little Endian υλοποίηση. Αυτό σημαίνει ότι τα πιο σημαντικά byte ενός word αποθηκεύονται στις θέσεις μνήμης με μεγαλύτερη διεύθυνση.

Εντολή	Πράξη	Λειτουργία
LB x4, imm(x1)	x4 ‹– M(c(x1) + imm)	Ανάγνωση του περιεχομένου της θέσης μνήμης με διεύθυνση (c(x1) + imm) και αποθήκευση στον x4

		αφού εφαρμοστεί επέκταση προσήμου.
LH x4, imm(x1)	x4 ‹– M(c(x1) + imm)	Ανάγνωση του περιεχομένου των θέσεων μνήμης με διεύθυνση (c(x1) + imm), (c(x1) + imm + 1), και αποθήκευση στον x4 ξεκινώντας από το λιγότερο σημαντικό byte. Επίσης εφαρμόζεται επέκταση προσήμου πριν την αποθήκευση.
LW x4, imm(x1)	x4 ‹– M(c(x1) + imm)	Ανάγνωση του περιεχομένου των θέσεων μνήμης με διεύθυνση (c(x1) + imm), (c(x1) + imm + 1), (c(x1) + imm + 2), (c(x1) + imm + 3), και αποθήκευση στον x4 ξεκινώντας από το λιγότερο σημαντικό byte. Επίσης εφαρμόζεται επέκταση προσήμου πριν την αποθήκευση.
LBU x4, imm(x1)	x4 ‹– M(c(x1) + imm)	Ίδια λειτουργία με LB χωρίς την επέκταση προσήμου.
LHU x4, imm(x1)	x4 ‹– M(c(x1) + imm)	Ίδια λειτουργία με LH χωρίς την επέκταση προσήμου.

Πίνακας 5: Εντολές Load

2.4 Εντολές διακλάδωσης

Η κατηγορία αυτών των εντολών περιλαμβάνει όλες τις εντολές που μπορούν να αλλάξουν την ροή του προγράμματος.

2.4.1 Εντολές διακλάδωσης υπό συνθήκη (Conditional Branch)

Στις εντολές διακλάδωσης υπό συνθήκη γίνεται κατάλληλη σύγκριση του περιεχομένου των δύο καταχωρητών που υποδεικνύονται από την εκάστοτε εντολή. Αν η συνθήκη ικανοποιείται τότε πραγματοποιείται η διακλάδωση, αλλιώς συνεχίζεται η κανονική ροή του προγράμματος. Ακολουθεί πίνακας με αυτές τις εντολές:

Διευκρινήσεις:

1) Όπου εμφανίζεται ο συμβολισμός c(x0), συμβολίζει το περιεχόμενο του καταχωρητή x0. Το ίδιο ισχύει και για τους υπόλοιπους καταχωρητές.
2) Όπου εμφανίζεται ο συμβολισμός PC, συμβολίζει το Μετρητή Προγράμματος.

Εντολή	Πράξη	Λειτουργία

BEQ x1, x2, offset	c(x1) == c(x2) ? PC += offset : PC += 4	Αν c(x1) ίσο με c(x2) τότε προστίθεται στην παρούσα τιμή του PC το offset.
BNE x1, x2, offset	c(x1) != c(x2) ? PC += offset : PC += 4	Αν c(x1) διαφορετικό του c(x2) τότε προστίθεται στην παρούσα τιμή του PC το offset.
BLT x1, x2, offset	c(x1) < c(x2) ? PC += offset : PC += 4	Αν c(x1) μικρότερο του c(x2) τότε προστίθεται στην παρούσα τιμή του PC το offset.
BGE x1, x2, offset	c(x1) >= c(x2) ? PC += offset : PC += 4	Αν c(x1) μεγαλύτερο ή ίσο του c(x2) τότε προστίθεται στην παρούσα τιμή του PC το offset.
BLTU x1, x2, offset	c(x1) < c(x2) ? PC += offset : PC += 4	Αντίστοιχο της BLT, όμως η σύγκριση γίνεται θεωρώντας πως οι αριθμοί είναι μη προσημασμένοι.
BGEU x1, x2, offset	c(x1) >= c(x2) ? PC += offset : PC += 4	Αντίστοιχο της BGE, όμως η σύγκριση γίνεται θεωρώντας πως οι αριθμοί είναι μη προσημασμένοι.

Πίνακας 6: Εντολές Branch

2.4.2 Εντολές άλματος (Jump)

Η κατηγορία αυτών των εντολών περιλαμβάνει όλες τις εντολές που αλλάζουν την ροή του προγράμματος χωρίς την ικανοποίηση κάποιας συνθήκης. Ακολουθεί πίνακας με αυτές τις εντολές:

Διευκρινήσεις:

1) Όπου εμφανίζεται ο συμβολισμός c(x0), συμβολίζει το περιεχόμενο του καταχωρητή x0. Το ίδιο ισχύει και για τους υπόλοιπους καταχωρητές.
2) Όπου εμφανίζεται ο συμβολισμός PC, συμβολίζει το Μετρητή Προγράμματος.

Εντολή	Πράξη	Λειτουργία
JAL x1, offset	x1 ‹– PC + 4, PC += offset	Αποθηκεύεται η διεύθυνση της επόμενης εντολής στον x1 και προστίθεται στην παρούσα τιμή του PC το offset.
JALR x1, offset(x2)	x1 ‹– PC + 4, PC += c(x2) + offset	Αποθηκεύεται η διεύθυνση της επόμενης εντολής στον x1 και προστίθεται στην παρούσα τιμή του PC το (offset + c(x2)) και το λιγότερο σημαντικό bit του

		αποτελέσματος τίθεται ίσο με 0.

Πίνακας 7: Εντολές Jump

2.5 Εντολές ειδικών δεδομένων

Η κατηγορία αυτών των εντολών περιλαμβάνει δυο εντολές που χρησιμοποιούν την κωδικοποίηση U-type. Ακολουθεί πίνακας με αυτές τις εντολές:

Διευκρινήσεις:

1) Όπου εμφανίζεται ο συμβολισμός PC, συμβολίζει το Μετρητή Προγράμματος.

Εντολή	Πράξη	Λειτουργία
LUI x1, imm	x1 ‹– (imm << 12)	Αποθηκεύεται το 20-bit άμεσο δεδομένο στα 20 πιο σημαντικά bit του x1 και τα 12 λιγότερο σημαντικά bit συμπληρώνονται με μηδέν.
AUIPC x1, imm	x1 ‹– PC + (imm << 12)	Δημιουργείται μια σταθερά 32-bit από το 20-bit imm και συμπλήρωση των 12 λιγότερο σημαντικών bit με μηδέν. Η σταθερά αυτή προστίθεται στην παρούσα τιμή του PC και το αποτέλεσμα αποθηκεύεται

		στον x1.

3. ΕΠΕΞΕΡΓΑΣΤΗΣ

Σε αυτή την ενότητα θα παρουσιαστούν οι σχεδιαστικές αποφάσεις που λήφθηκαν κατά την σχεδίαση του επεξεργαστή. Οι αποφάσεις πάρθηκαν με βάση την ανάλυση των εντολών του συνόλου RV32I που έγινε στην προηγούμενη ενότητα (Ενότητα 2).

3.1 Τεχνική μερικώς επικαλυπτόμενων λειτουργιών (Pipelining)

Είναι γνωστό ότι κατά την εκτέλεση μιας εντολής, ακολουθούνται κάποια βήματα που εκτελούνται το ένα μετά το άλλο. Η τεχνική των μερικώς επικαλυπτόμενων λειτουργιών ορίζει διαδοχικές βαθμίδες, οι οποίες εκτελούν η κάθε μια ένα από αυτά τα βήματα. Το σημαντικό χαρακτηριστικό της τεχνικής αυτής είναι ότι προσφέρει τη δυνατότητα να εκκινήσει η εκτέλεση μιας εντολής χωρίς να έχει ολοκληρωθεί η εκτέλεση μιας παλαιότερης, δηλαδή εκτελούνται παράλληλα τα στάδια διαφορετικών εντολών. Η τεχνική αυτή, έχει εφαρμοστεί σε επεξεργαστές RISC και CISC αρχιτεκτονικής.

Η τεχνική των μερικώς επικαλυπτόμενων λειτουργιών στην αρχιτεκτονική RISC ορίζει τις παρακάτω βαθμίδες [4]:

1) IF – Instruction Fetching (Προσκόμιση εντολής):
   Στο στάδιο αυτό προσκομίζεται από τη μνήμη εντολών η εντολή που τίθεται προς εκτέλεση. Ο μετρητής προγράμματος αυξάνεται

κατάλληλα ώστε να υποδείξει τη διεύθυνση της επόμενης προς εκτέλεση εντολής.

2) ID – Instruction Decoding (Αποκωδικοποίηση εντολής):
   Στο στάδιο αυτό γίνεται ανάγνωση των καταχωρητών και παράγονται τα ανάλογα σήματα ελέγχου για κάθε εντολή.

3) EX – Execution (Εκτέλεση πράξεων):
   Στο στάδιο αυτό γίνεται η πράξη που έχει οριστεί από την εντολή. Το αποτέλεσμα της πράξης μπορεί να είναι η διεύθυνση προσπέλασης της μνήμης δεδομένων που βρίσκεται στο αμέσως επόμενο στάδιο (MEM) ή ένα αριθμητικό αποτέλεσμα το οποίο αποθηκεύεται σε κάποιο καταχωρητή σε μεταγενέστερο στάδιο (WB).

4) MEM – Memory Access (Προσπέλαση Μνήμης):
   Στο στάδιο αυτό γίνεται ανάγνωση δεδομένων από τη μνήμη ή αποθήκευση δεδομένων σε αυτή.

5) WB – Write Back (Αποθήκευση αποτελεσμάτων):
   Στο στάδιο αυτό που είναι και το τελευταίο, γίνεται αποθήκευση αποτελέσματος σε κάποιον από τους καταχωρητές.

Είναι αρκετά δύσκολο να σχεδιαστεί ένας επεξεργαστής με την τεχνική των μερικώς επικαλυπτόμενων λειτουργιών δίχως την ύπαρξη καταχωρητών μεταξύ των βαθμίδων που αναφέρθηκαν πριν. Αυτό οφείλεται στην ανάγκη κάθε βαθμίδα να έχει τον ίδιο χρόνο καθυστέρησης με τις υπόλοιπες. Η ανάγκη αυτή λοιπόν οδηγεί στη χρήση καταχωρητών μεταξύ των βαθμίδων οι οποίοι ονομάζονται ενδιάμεσοι καταχωρητές. Οι καταχωρητές λαμβάνουν το ίδιο σήμα χρονισμού (ρολόι) και η περίοδος αυτού του σήματος πρέπει να έχει διάρκεια που να επιτρέπει την εκτέλεση όλων των λειτουργιών που έχει αναλάβει κάθε βαθμίδα. Οι ενδιάμεσοι καταχωρητές που ορίστηκαν κατά τον

σχεδιασμό του παρόντος επεξεργαστή είναι οι IF_ID, ID_EX, EX_MEM, MEM_WB. Η ονομασία κάθε καταχωρητή προέκυψε από τις βαθμίδες στις οποίες βρίσκεται ανάμεσα ο εκάστοτε καταχωρητής.

Όπως αναφέρθηκε πρωτύτερα, σε ένα μηχανισμό μερικώς επικαλυπτόμενων λειτουργιών τα διαφορετικά στάδια διαφορετικών εντολών εκτελούνται παράλληλα μέσα στο μηχανισμό. Αυτό το γεγονός προκαλεί καταστάσεις στις οποίες εμποδίζεται κάποιο στάδιο μιας εντολής να εκτελεστεί την χρονική περίοδο που έχει οριστεί. Αυτές οι καταστάσεις ονομάζονται εξαρτήσεις (hazards). Οι εξαρτήσεις που εγείρονται κατατάσσονται σε τρεις κατηγορίες οι οποίες αναφέρονται παρακάτω [4]:

1) Structural Hazards (Δομικές εξαρτήσεις):
   Αυτό το είδος εξάρτησης προκύπτει όταν κάποιες εντολές που έχουν εισαχθεί για εκτέλεση στον μηχανισμό μερικώς επικαλυπτόμενων λειτουργιών, απαιτούν την ταυτόχρονη χρήση μιας βαθμίδας του μηχανισμού. #Να αναφέρω ότι δεν υπάρχουν στην δική μου περίπτωση.

2) Data Hazards (Εξαρτήσεις από δεδομένα):
   Αυτό το είδος εξάρτησης προκύπτει όταν μεταξύ δυο εντολών που είναι κοντά στη σειρά εκτέλεσης ώστε η επικάλυψη των λειτουργιών τους να οδηγεί σε λανθασμένη προσπέλαση κάποιων δεδομένων. Για παράδειγμα αναφέρεται ένα ζεύγος διαδοχικών εντολών όπου η δεύτερη χρησιμοποιεί το αποτέλεσμα της εκτέλεσης της πρώτης.

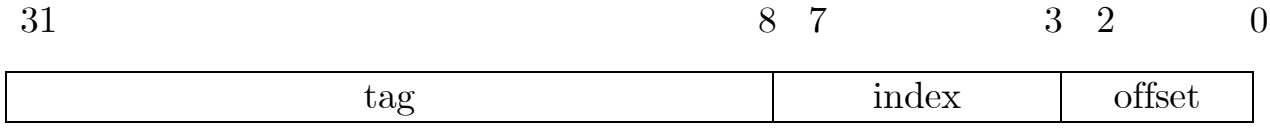
3) Control Hazards (Διαδικασιακές εξαρτήσεις):
   Αυτό το είδος εξάρτησης προκύπτει όταν η εκτέλεση μιας εντολής διακλάδωσης (2.4) προκαλεί αλλαγή στη ροή του προγράμματος.

3.2 Λειτουργικές μονάδες του επεξεργαστή

Ως λειτουργικές μονάδες του επεξεργαστή θεωρούνται οι μονάδες που υπάρχουν μέσα στον επεξεργαστή και υλοποιούνται ως ακολουθιακά κυκλώματα. Η θύρα εισόδου του σήματος ρολογιού (clk) καθώς και η θύρα εισόδου για το σήμα reset παραλείπεται από τις περιγραφές.

3.2.1 Κρυφή Μνήμη Εντολών (Instruction Cache Memory)

Η Κρυφή Μνήμη Εντολών (ΚΜΕ) είναι μία κρυφή μνήμη με οργάνωση μονοσήμαντης απεικόνισης (Direct Mapping). Λαμβάνει διευθύνσεις από τον Μετρητή Προγράμματος (Programm Counter) και με αυτό τον τρόπο κάνει ανάγνωση της εντολής η οποία τίθεται προς εκτέλεση. Ο τρόπος χρήσης των διευθύνσεων που λαμβάνει η ΚΜΕ είναι ο εξής:



Το διάστημα offset καταλαμβάνει τα 3 λιγότερο σημαντικά bits της διεύθυνσης και προσδιορίζει τη διεύθυνση μιας λέξης μέσα στο πλαίσιο της κρυφής μνήμης. Στην συγκεκριμένη περίπτωση το πλαίσιο αποτελείται από 8 λέξεις. Το διάστημα index καταλαμβάνει τα 5 αμέσως επόμενα bits της διεύθυνσης και προσδιορίζει τη διεύθυνση του πλαισίου μέσα στην κρυφή μνήμη, η κρυφή μνήμη αποτελείται από 32 πλαίσια. Το διάστημα tag

καταλαμβάνει τα υπόλοιπα 24 bits τα οποία αποτελούν την ετικέτα κάθε διεύθυνσης.

Η KME διαθέτει την θύρα εισόδου:

- addr: Θύρα εισόδου σήματος για την διεύθυνση προσπέλασης της μνήμης, με μέγεθος 32 bits.

Η KME διαθέτει την θύρα εξόδου:

- inst: Θύρα εξόδου σήματος για την εντολή που διαβάζεται από την μνήμη, με μέγεθος 32 bits.

3.2.2 Αρχείο Καταχωρητών (Register File)

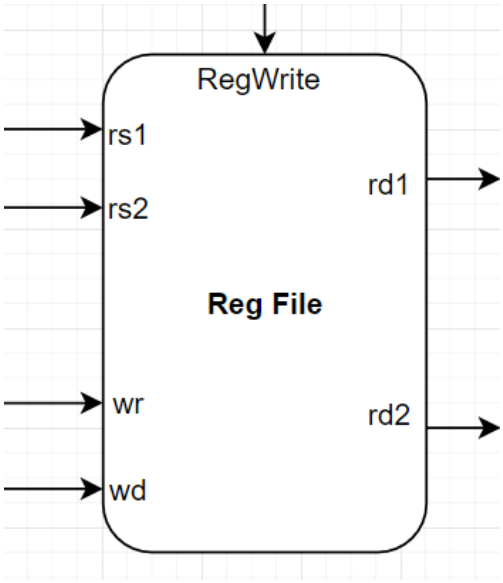
Το Αρχείο Καταχωρητών (ΑΚ) ορίζεται ως το σύνολο των καταχωρητών του επεξεργαστή και χρησιμεύουν στην προσωρινή αποθήκευση δεδομένων. Η αρχιτεκτονική συνόλου εντολών RV32I προδιαγράφει το πλήθος των καταχωρητών στο ΑΚ να είναι 32. Άλλο ένα χαρακτηριστικό είναι ότι ο καταχωρητής 0 δηλαδή ο πρώτος, πρέπει πάντα το περιεχόμενό του να είναι ίσο με μηδέν. Επιπλέον, με την τεχνική των μερικώς επικαλυπτόμενων λειτουργιών επιβάλλεται η ταυτόχρονη εγγραφή και ανάγνωση του ΑΚ κάτι το οποίο μπορεί να προκαλέσει δομική εξάρτηση. Η δομική εξάρτηση προκαλείται όταν μια μονάδα χρησιμοποιείται σε παραπάνω από μία βαθμίδες. Στην προκειμένη περίπτωση αυτές οι βαθμίδες είναι οι WB (εγγραφή σε καταχωρητή του ΑΚ) και ID (ανάγνωση καταχωρητή του ΑΚ). Το πρόβλημα αντιμετωπίζεται με την υλοποίηση του ΑΚ η οποία επιτρέπει στον ίδιο κύκλο ρολογιού την ταυτόχρονη εγγραφή και ανάγνωση στους καταχωρητές. Στην πιο ειδική περίπτωση όπου η εγγραφή και ανάγνωση αφορά τον ίδιο καταχωρητή, η εγγραφή πραγματοποιείται στο πρώτο μισό του σήματος ρολογιού και η ανάγνωση στο δεύτερο μισό.

Το ΑΚ διαθέτει τις παρακάτω θύρες εισόδου:

- rs1, rs2: Θύρες εισόδου για τη διεύθυνση των καταχωρητών προς ανάγνωση, με μέγεθος 5 bits.
- wr: Θύρα εισόδου για τη διεύθυνση του καταχωρητή προς εγγραφή, με μέγεθος 5 bits.
- wd: Θύρα εισόδου για τα δεδομένα προς εγγραφή στον καταχωρητή που υποδεικνύει η διεύθυνση wr, με μέγεθος 32 bits.
- RegWrite: Θύρα εισόδου με μέγεθος 1 bit που λειτουργεί ως είσοδος επίτρεψης για την εγγραφή του καταχωρητή που υποδεικνύει η διεύθυνση wr. Αν πάρει την τιμή 1 τότε γίνεται εγγραφή.

Το ΑΚ διαθέτει τις παρακάτω θύρες εξόδου:

- rd1, rd2: Θύρες εξόδου για τα δεδομένα που είναι αποθηκευμένα στους καταχωρητές που υποδεικνύονται από τις διευθύνσεις rs1, rs2 αντίστοιχα, με μέγεθος 32 bits.



Εικόνα 1: Αρχείο Καταχωρητών

3.2.3 Μονάδα Παραγωγής Άμεσων Δεδομένων (Immediate Generator)

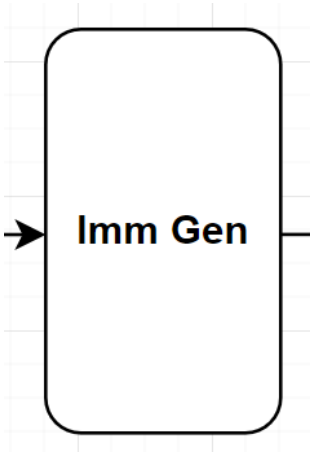
Η Μονάδα Παραγωγής Άμεσων Δεδομένων (ΜΠΑΔ) έχει ως σκοπό την αποκωδικοποίηση των άμεσων δεδομένων που είναι ενσωματωμένα στις εντολές. Άμεσα δεδομένα περιέχουν οι εντολές τύπου I, S, B, U, J όπως αναφέρθηκε στην 2.1 . Ο κάθε τύπος κωδικοποιεί το δεδομένο με διαφορετικό τρόπο μέσα στην εντολή. Έτσι, ανάλογα με τον opcode της εντολής η μονάδα μετατρέπει το δεδομένο σε κατάλληλη μορφή ώστε να μεταφερθεί στην επόμενη βαθμίδα για περαιτέρω επεξεργασία. Το δεδομένο που παράγεται πρέπει να οδηγηθεί στο στάδιο EX, άρα πρέπει η ΜΠΑΔ να τοποθετηθεί πριν από αυτό. Επίσης, η ΜΠΑΔ αντικαθιστά την ανάγνωση κάποιου καταχωρητή με την παραγωγή ενός άμεσου δεδομένου. Επομένως, η ΜΠΑΔ τοποθετείται στο στάδιο ID.

Η ΜΠΑΔ διαθέτει την θύρα εισόδου:

- Instruction: Θύρα εισόδου διανύσματος εντολής, με μέγεθος 32 bits.

Η ΜΠΑΔ διαθέτει την θύρα εξόδου:

- Immediate: Θύρα εξόδου για το αποτέλεσμα της αποκωδικοποίησης του άμεσου δεδομένου, με μέγεθος 32 bits.



Εικόνα 2: Μονάδα Παραγωγής Άμεσων Δεδομένων

3.2.4 Μονάδα Ελέγχου (Control Unit)

Η Μονάδα Ελέγχου (ΜΕ) είναι η μονάδα που αναλαμβάνει την αποκωδικοποίηση της εντολής που προσκομίστηκε στον προηγούμενο κύκλο ρολογιού από την Μνήμη Εντολών, με αποτέλεσμα να τοποθετηθεί στο στάδιο ID. Η αποκωδικοποίηση βασίζεται αποκλειστικά στον κωδικό κάθε εντολής (opcode). Είναι ένα εξαιρετικά σημαντικό κύκλωμα διότι παράγει όλα τα σήματα που κρίνονται απαραίτητα για την εκτέλεση της εκάστοτε εντολής.

Η ΜΕ διαθέτει την θύρα εισόδου:

- Opcode: Θύρα εισόδου του κωδικού της εντολής, με μέγεθος 7 bits.

Η ΜΕ διαθέτει τις θύρες εξόδου:

- LUIorAUIPC: Θύρα εξόδου σήματος με μέγεθος 1 bit το οποίο μέσω των ενδιάμεσων καταχωρητών διαδίδεται μέχρι το στάδιο EX όπου εκεί καθορίζει τη ροή της λειτουργίας σύμφωνα με το αν εντολή που εκτελείται είναι η LUI ή η AUIPC.
- Jump: Θύρα εξόδου σήματος με μέγεθος 2 bits το οποίο μέσω των ενδιάμεσων καταχωρητών διαδίδεται μέχρι το στάδιο EX όπου εκεί μπαίνει ως είσοδος στη Μονάδας Διαχείρισης Διακλαδώσεων.
- ALUop: Θύρα εξόδου σήματος με μέγεθος 2 bits το οποίο μέσω των ενδιάμεσων καταχωρητών διαδίδεται μέχρι το στάδιο EX όπου εκεί μπαίνει ως είσοδος στη Μονάδας Ελέγχου ΑΛΜ.
- ALUsrc: Θύρα εξόδου σήματος με μέγεθος 1 bit το οποίο μέσω των ενδιάμεσων καταχωρητών διαδίδεται μέχρι το στάδιο EX όπου εκεί καθορίζει αν η είσοδος 1 (ALU1) της ΑΛΜ θα

τροφοδοτηθεί από ένα άμεσο δεδομένο ή από το περιεχόμενο κάποιου καταχωρητή.

- MemRead: Θύρα εξόδου σήματος με μέγεθος 1 bit το οποίο μέσω των ενδιάμεσων καταχωρητών διαδίδεται μέχρι το στάδιο MEM όπου εκεί καθορίζει αν θα γίνει ανάγνωση στην Μνήμη Δεδομένων.
- MemWrite: Θύρα εξόδου σήματος με μέγεθος 1 bit το οποίο μέσω των ενδιάμεσων καταχωρητών διαδίδεται μέχρι το στάδιο MEM όπου εκεί καθορίζει αν θα γίνει εγγραφή στην Μνήμη Δεδομένων.
- RegWrite: Θύρα εξόδου σήματος με μέγεθος 1 bit το οποίο μέσω των ενδιάμεσων καταχωρητών διαδίδεται μέχρι το στάδιο WB όπου εκεί καθορίζει αν θα γίνει εγγραφή σε καταχωρητή του Αρχείου Καταχωρητών.
- MemToReg: Θύρα εξόδου σήματος με μέγεθος 2 bits το οποίο μέσω των ενδιάμεσων καταχωρητών διαδίδεται μέχρι το στάδιο WB όπου εκεί καθορίζει την έξοδο ενός πολυπλέκτη.

3.2.5 Μονάδα Ανίχνευσης Εξαρτήσεων (Hazard Detect Unit)

Η Μονάδα Ανίχνευσης Εξαρτήσεων (MAE) είναι η μονάδα που αναλαμβάνει να ανιχνεύσει μια ειδική περίπτωση εξάρτησης από δεδομένα. Αυτή η εξάρτηση προκύπτει από μια συγκεκριμένη ακολουθία εντολών, πιο συγκεκριμένα όταν μια εντολή ανάγνωσης της μνήμης (Load) ακολουθείται από μια εντολή η οποία χρησιμοποιεί το περιεχόμενο του καταχωρητή που ανανεώνει η Load που αναφέρθηκε (ακολουθία εντολών Load-Use). Το επιθυμητό δεδομένο θα είναι διαθέσιμο στο στάδιο εκτέλεσης MEM της εντολής Load, αν δεν ληφθεί κάποιο μέτρο τότε η εντολή Use θα φτάσει στο στάδιο EX έχοντας διαβάσει λανθασμένο περιεχόμενο για τον καταχωρητή. Για αυτό κρίνεται απαραίτητη η καθυστέρηση (pipeline stall) της εντολής Use

κατά ένα κύκλο ρολογιού όταν αυτή βρίσκεται στο στάδιο ID, επομένως όταν η Load βρίσκεται στο στάδιο EX. Επομένως, η MAE τοποθετείται στο στάδιο ID.

Η MAE διαθέτει τις θύρες εισόδου:

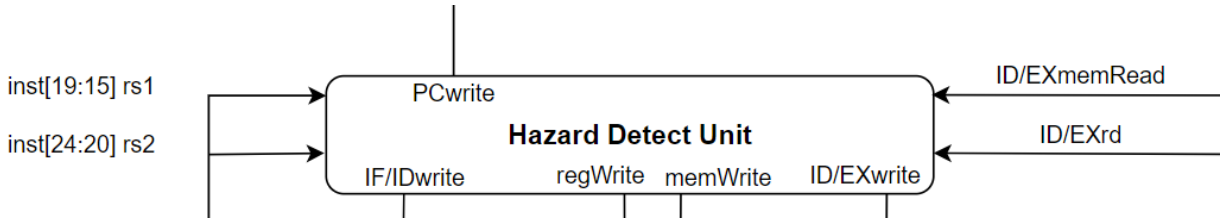
- ID_EXmemRead: Θύρα εισόδου σήματος με μέγεθος 1 bit το οποίο μαρτυρά αν η εντολή που βρίσκεται στο στάδιο EX είναι εντολή Load.
- ID_EXrd: Θύρα εισόδου σήματος με μέγεθος 5 bits το οποίο πληροφορεί για τον καταχωρητή στον οποίο πρόκειται να αποθηκεύσει η εντολή που βρίσκεται στο στάδιο EX.
- IF_IDrs1, IF_IDrs2: Θύρες εισόδου σημάτων με μέγεθος 5 bits τα οποία πληροφορούν για τους καταχωρητές που θέλει να κάνει ανάγνωση η εντολή που βρίσκεται στο στάδιο ID.

Η MAE διαθέτει τις θύρες εξόδου:

- PCwrite: Θύρα εξόδου σήματος με μέγεθος 1 bit το οποίο όταν είναι 1 τότε ο Μετρητής Προγράμματος (ΜΠ) θα ανανεώσει κατάλληλα την τιμή του στον επόμενο κύκλο ρολογιού, αλλιώς αν είναι 0 τότε ο ΜΠ θα διατηρήσει την τιμή που είχε στον προηγούμενο κύκλο ρολογιού.
- IF_IDwrite: Θύρα εξόδου σήματος με μέγεθος 1 bit το οποίο όταν είναι 1 τότε ο ενδιάμεσος καταχωρητής IF_ID θα ανανεώσει κατάλληλα την τιμή του στον επόμενο κύκλο ρολογιού, αλλιώς αν είναι 0 τότε ο IF_ID θα διατηρήσει την τιμή που είχε στον προηγούμενο κύκλο ρολογιού.
- ID_EXwrite: Θύρα εξόδου σήματος με μέγεθος 1 bit το οποίο όταν είναι 1 τότε ο ενδιάμεσος καταχωρητής ID_EX θα ανανεώσει κατάλληλα την τιμή του στον επόμενο κύκλο ρολογιού, αλλιώς αν είναι 0 τότε ο ID_EX θα διατηρήσει την τιμή που είχε στον προηγούμενο κύκλο ρολογιού

- regWrite: Θύρα εξόδου σήματος με μέγεθος 1 bit το οποίο οδηγεί ένα πολυπλέκτη. Αν το σήμα έχει τιμή 1 τότε ο πολυπλέκτης θα διαδώσει την τιμή του σήματος RegWrite που παράγει η Μονάδα Ελέγχου και αναφέρθηκε στο 3.2.4 . Αλλιώς, αν έχει τιμή 0 ο πολυπλέκτης θα διαδώσει την τιμή 0.
- memWrite: Θύρα εξόδου σήματος με μέγεθος 1 bit το οποίο οδηγεί ένα πολυπλέκτη. Αν το σήμα έχει τιμή 1 τότε ο πολυπλέκτης θα διαδώσει την τιμή του σήματος MemWrite που παράγει η Μονάδα Ελέγχου και αναφέρθηκε στο 3.2.4 . Αλλιώς, αν έχει τιμή 0 ο πολυπλέκτης θα διαδώσει την τιμή 0.

Οι τιμές των σημάτων στις θύρες εξόδου πρέπει να είναι αρχικοποιημένες στο 1, όταν αρχίζει η εκτέλεση ενός προγράμματος.



Εικόνα 3: Μονάδα Ανίχνευσης Εξαρτήσεων

3.2.6 Αριθμητική και Λογική Μονάδα (Arithmetic Logic Unit - ALU)

Η Αριθμητική και Λογική Μονάδα (ΑΛΜ) είναι η μονάδα που εκτελούνται οι πράξεις που απαιτούνται από εντολές τύπου R, I, S, B. Τοποθετείται στο στάδιο EX. Στις εντολές τύπου R εφαρμόζεται μία πράξη σε 2 δεδομένα που προκύπτουν από την ανάγνωση 2 καταχωρητών. Στις

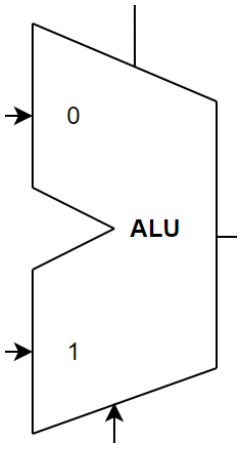
εντολές τύπου I γίνεται μία πράξη σε 2 δεδομένα που προκύπτουν από την ανάγνωση ενός καταχωρητή και από ένα άμεσο δεδομένο. Επιπλέον, στις εντολές τύπου I κατατάσσονται και οι εντολές Load για τις οποίες η ΑΛΜ υπολογίζει τη διεύθυνση προσπέλασης της Μνήμης Δεδομένων. Οι εντολές τύπου S είναι οι εντολές Store για τις οποίες η ΑΛΜ υπολογίζει επίσης τη διεύθυνση προσπέλασης της Μνήμης Δεδομένων. Τέλος, για τις εντολές τύπου B, δηλαδή για τις εντολές διακλάδωσης υπό συνθήκη (2.4.1) η ΑΛΜ εφαρμόζει τη σύγκριση που απαιτείται από την εκάστοτε εντολή.

Η ΑΛΜ διαθέτει τις θύρες εισόδου:

- data0, data1: Θύρες εισόδου σημάτων με μέγεθος 32 bits, τα οποία συμμετέχουν στην εκτέλεση της πράξης.
- ctrl: Θύρα εισόδου σήματος με μέγεθος 4 bits το οποίο προέρχεται από την Μονάδα Ελέγχου ΑΛΜ, το σήμα αυτό υποδεικνύει την πράξη ή την σύγκριση που χρειάζεται να εκτελέσει η ΑΛΜ.

Η ΑΛΜ διαθέτει τις θύρες εξόδου:

- result: Θύρα εξόδου σήματος με μέγεθος 32 bits το οποίο δίνει το αποτέλεσμα της πράξης που εκτελεί η ΑΛΜ.
- branch: Θύρα εξόδου σήματος με μέγεθος 1 bit το οποίο οδηγείται στην Μονάδα Διαχείρισης Διακλαδώσεων. Αν το σήμα είναι 1 τότε αυτό σημαίνει πως ικανοποιείται η συνθήκη της εντολής διακλάδωσης.



Εικόνα 4: Αριθμητική και Λογική Μονάδα

3.2.7 Μονάδα Ελέγχου ΑΛΜ (ALU Control)

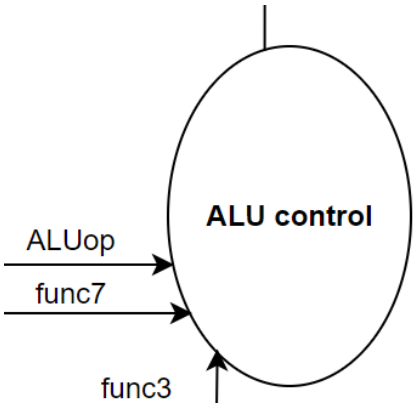
Η Μονάδα Ελέγχου ΑΛΜ (ΜΕΑΛΜ) είναι η μονάδα που υποδεικνύει στην Αριθμητική και Λογική Μονάδα την πράξη ή την σύγκριση που πρέπει να εκτελέσει. Η ΜΕΑΛΜ εκμεταλλεύεται συγκεκριμένα πεδία κάθε εντολής.

Η ΜΕΑΛΜ διαθέτει τις θύρες εισόδου:

- ALUctrl_f7: Θύρα εισόδου σήματος με μέγεθος 7 bits το οποίο αντιπροσωπεύει το πεδίο funct7 που αναφέρθηκε στον Πίνακα 1.
- ALUctrl_f3: Θύρα εισόδου σήματος με μέγεθος 3 bits το οποίο αντιπροσωπεύει το πεδίο funct3 που αναφέρθηκε στον Πίνακα 1.
- ALUop: Θύρα εισόδου σήματος με μέγεθος 2 bits το οποίο προέρχεται από την Μονάδα Ελέγχου (3.2.4).

Η ΜΕΑΛΜ διαθέτει τις θύρες εξόδου:

- ALUctrl_lines: Θύρα εξόδου σήματος με μέγεθος 4 bits το οποίο οδηγείται στην Αριθμητική και Λογική Μονάδα.



Εικόνα 5: Μονάδα Ελέγχου ΑΛΜ

3.2.8 Μονάδα Διαχείρισης Διακλαδώσεων (Branch Unit)

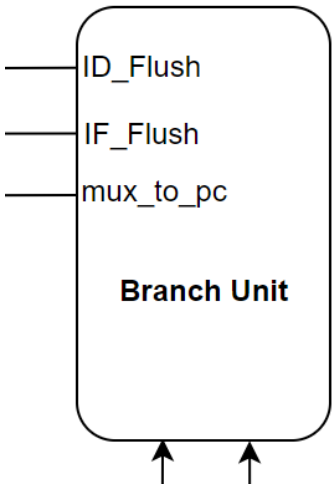
Η Μονάδα Διαχείρισης Διακλαδώσεων (ΜΔΔ) είναι η μονάδα που καθορίζει τις διαδικασίες που πρέπει να γίνουν κατά την εκτέλεση μιας εντολής διακλάδωσης (2.4). Κάθε εντολή διακλάδωσης πρέπει να φτάσει στο στάδιο ΕΧ διότι χρειάζονται τιμές από το αρχείο καταχωρητών για την εκτέλεση τους. Ειδικότερα στην περίπτωση των εντολών διακλάδωσης υπό συνθήκη, πρέπει να γίνει και σύγκριση μεταξύ των περιεχομένων των καταχωρητών. Με βάση τα παραπάνω η ΜΔΔ τοποθετείται στο στάδιο ΕΧ.

Η ΜΔΔ διαθέτει τις θύρες εισόδου:

- branch: Θύρα εισόδου σήματος με μέγεθος 1 bit το οποίο προέρχεται από την Αριθμητική και Λογική Μονάδα. Αν το σήμα είναι ίσο με 1, αυτό σημαίνει πως η συνθήκη της εντολής διακλάδωσης ικανοποιείται.
- jump: Θύρα εισόδου σήματος με μέγεθος 2 bit το οποίο προέρχεται από την Μονάδα Ελέγχου. Αν είναι ίσο με 00 τότε αυτό σημαίνει πως η εντολή που βρίσκεται στο στάδιο ΕΧ δεν είναι εντολή διακλάδωσης επομένως η ΜΔΔ δεν θα επηρεάσει τη ροή του προγράμματος.

Η ΜΔΔ διαθέτει τις θύρες εξόδου:

- IF_Flush: Θύρα εξόδου σήματος με μέγεθος 1 bit το οποίο ισούται με 1 σε 2 περιπτώσεις. Πρώτον, όταν ικανοποιείται η συνθήκη της εντολής διακλάδωσης και δεύτερον όταν η εντολή είναι εντολή άλματος. Όταν λοιπόν ισούται με 1 τότε το περιεχόμενο του ενδιάμεσου καταχωρητή IF_ID δεν διαδίδεται περαιτέρω καθώς περιέχει πληροφορίες για την εκτέλεση μιας εντολής (η δεύτερη μετά την εντολή διακλάδωσης) που δεν πρέπει να εκτελεστεί αφού η ροή του προγράμματος θα αλλάξει.
- ID_Flush: Θύρα εξόδου σήματος με μέγεθος 1-bit το οποίο ισούται με 1 σε 2 περιπτώσεις. Πρώτον, όταν ικανοποιείται η συνθήκη της εντολής διακλάδωσης και δεύτερον όταν η εντολή είναι εντολή άλματος. Όταν λοιπόν ισούται με 1 τότε το περιεχόμενο του ενδιάμεσου καταχωρητή ID_EX δεν διαδίδεται περαιτέρω καθώς περιέχει πληροφορίες για την εκτέλεση μιας εντολής (η πρώτη μετά την εντολή διακλάδωσης) που δεν πρέπει να εκτελεστεί αφού η ροή του προγράμματος θα αλλάξει.
- mux_to_pc: Θύρα εξόδου σήματος με μέγεθος 2 bits το οποίο οδηγεί τον πολυπλέκτη που τοποθετείται πριν τον Μετρητή Προγράμματος και καθορίζει το δεδομένο με το οποίο θα τροφοδοτηθεί ο ΜΠ.



Εικόνα 6: Μονάδα Διαχείρισης Διακλαδώσεων

3.2.9 Μονάδα Παροχέτευσης (Forwarding Unit)

Η Μονάδα Παροχέτευσης (ΜΠαρ) είναι η μονάδα που αναλαμβάνει να αντιμετωπίσει τις εξαρτήσεις από δεδομένα. Στην τεχνική των μερικώς επικαλυπτόμενων λειτουργιών, εξαρτήσεις από δεδομένα δημιουργούνται όταν μία εντολή αποθηκεύει ένα αποτέλεσμα σε ένα καταχωρητή και στη συνέχεια οι εντολές που ακολουθούν αναγιγνώσκουν το περιεχόμενο του ίδιου καταχωρητή. Αυτό συμβαίνει διότι η εγγραφή ενός καταχωρητή γίνεται στο στάδιο WB. Όταν η εντολή που αποθηκεύει σε ένα καταχωρητή βρίσκεται σε προγενέστερο στάδιο από το WB αυτό έχει ως αποτέλεσμα οι εντολές που ακολουθούν και επιθυμούν να κάνουν ανάγνωση του περιεχομένου του ίδιου καταχωρητή να μην διαβάσουν το νέο και έγκυρο περιεχόμενο διότι δεν θα έχει αποθηκευτεί ακόμη. Η λύση στο πρόβλημα δίνεται από την τεχνική της παροχέτευσης.

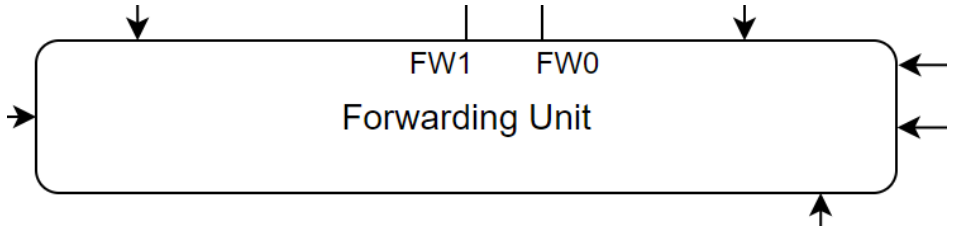
Η ΜΠαρ διαθέτει τις θύρες εισόδου:

- EX_MEMregWrite: Θύρα εισόδου σήματος μεγέθους 1 bit που αντιπροσωπεύει το σήμα RegWrite από τη Μονάδα Ελέγχου για την εντολή που βρίσκεται στο στάδιο MEM.
- MEM_WBregWrite: Θύρα εισόδου σήματος μεγέθους 1 bit που αντιπροσωπεύει το σήμα RegWrite από τη Μονάδα Ελέγχου για την εντολή που βρίσκεται στο στάδιο WB.
- ID_EXrs1, ID_EXrs2: Θύρες εισόδου σημάτων μεγέθους 5 bits που αντιπροσωπεύουν το περιεχόμενο των καταχωρητών rs1, rs2 τους οποίους έχει διαβάσει η εντολή που βρίσκεται στο στάδιο EX.

- EX_MEMrd: Θύρα εισόδου σήματος μεγέθους 5 bits που αντιπροσωπεύει τον καταχωρητή που στοχεύει να εγγράψει η εντολή που βρίσκεται στο στάδιο MEM.
- MEM_WBrd: Θύρα εισόδου σήματος μεγέθους 5 bits που αντιπροσωπεύει τον καταχωρητή που στοχεύει να εγγράψει η εντολή που βρίσκεται στο στάδιο WB.

Η ΜΠαρ διαθέτει τις θύρες εξόδου:

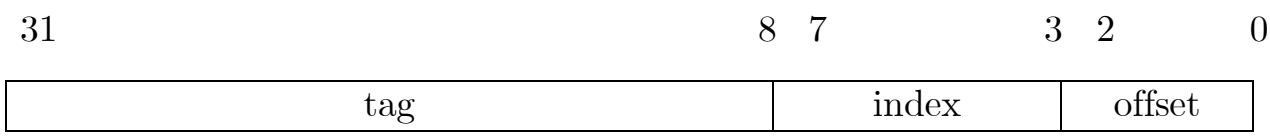
- FW0: Θύρα εξόδου σήματος μεγέθους 2 bits το οποίο οδηγεί τον πολυπλέκτη FW0 ο οποίος καθορίζει από ποιά βαθμίδα θα παρθεί το αποτέλεσμα που πρόκειται να εγγραφεί σε καταχωρητή. Διότι αυτό το αποτέλεσμα θα είναι το έγκυρο.
- FW1: Θύρα εξόδου σήματος μεγέθους 2 bits το οποίο οδηγεί τον πολυπλέκτη FW1 ο οποίος καθορίζει από ποιά βαθμίδα θα παρθεί το αποτέλεσμα που πρόκειται να εγγραφεί σε καταχωρητή. Διότι αυτό το αποτέλεσμα θα είναι το έγκυρο.



Εικόνα 7: Μονάδα Παροχέτευσης

3.2.10 Κρυφή Μνήμη Δεδομένων (Data Cache Memory)

Η Κρυφή Μνήμη Δεδομένων (ΔΜΕ) είναι μία κρυφή μνήμη με οργάνωση μονοσήμαντης απεικόνισης (Direct Mapping). Σε αυτή τη μνήμη γίνεται είτε εγγραφή είτε ανάγνωση δεδομένων. Ο τρόπος χρήσης των διευθύνσεων που λαμβάνει η ΚΜΔ είναι ο εξής:



Το διάστημα offset καταλαμβάνει τα 3 λιγότερο σημαντικά bits της διεύθυνσης και προσδιορίζει τη διεύθυνση μιας λέξης μέσα στο πλαίσιο της κρυφής μνήμης. Στην συγκεκριμένη περίπτωση το πλαίσιο αποτελείται από 8 λέξεις. Το διάστημα index καταλαμβάνει τα 5 αμέσως επόμενα bits της διεύθυνσης και προσδιορίζει τη διεύθυνση του πλαισίου μέσα στην κρυφή μνήμη, η κρυφή μνήμη αποτελείται από 32 πλαίσια. Το διάστημα tag καταλαμβάνει τα υπόλοιπα 24 bits τα οποία αποτελούν την ετικέτα κάθε διεύθυνσης.

Η ΚΜΔ διαθέτει τις παρακάτω θύρες εισόδου:

- addr: Θύρα εισόδου σήματος για την διεύθυνση προσπέλασης της μνήμης, με μέγεθος 32 bits.
- MemWrite: Θύρα εισόδου σήματος με μέγεθος 1 bit το οποίο όταν ισούται με 1 τότε θα γίνει εγγραφή της τιμής του σήματος WriteData στη θέση που υποδεικνύει το σήμα addr.
- MemRead: Θύρα εισόδου σήματος με μέγεθος 1 bit το οποίο όταν ισούται με 1 τότε θα γίνει ανάγνωση του περιεχομένου της θέσης που υποδεικνύει το σήμα addr.
- WriteData: Θύρα εισόδου σήματος με μέγεθος 32 bits το οποίο δίνει την τιμή που εγγραφεί στην διεύθυνση της μνήμης που έχει ορίσει το σήμα addr.
- funct3: Θύρα εισόδου σήματος με μέγεθος 3 bits το οποίο αντιπροσωπεύει το πεδίο funct3 που αναφέρθηκε στον Πίνακα 1. Το

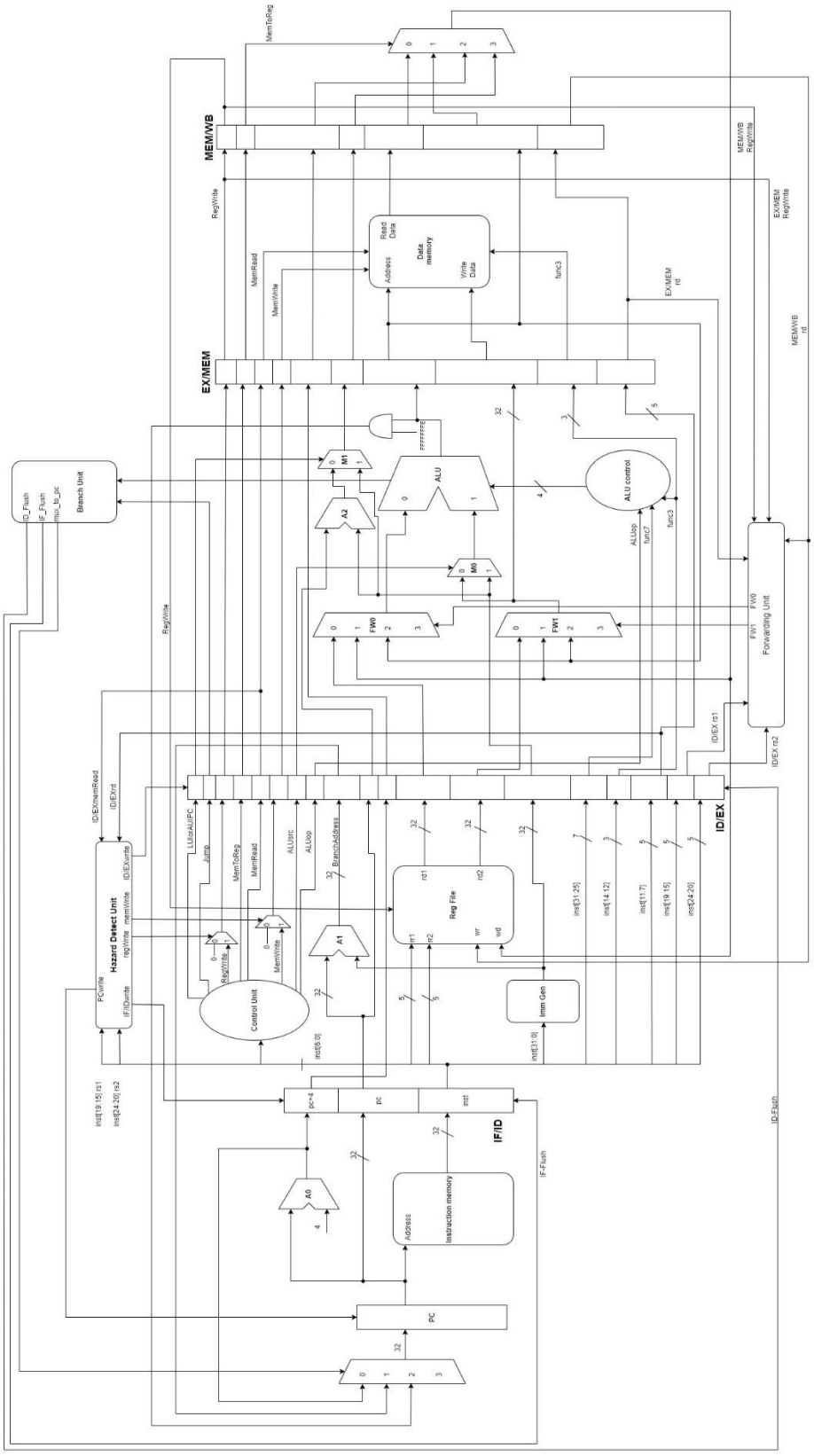
σήμα αυτό χρησιμοποιείται για να καθοριστεί η παραλλαγή των εντολών τύπου load ή store (π.χ. load byte, store halfword).

Η ΚΜΕ διαθέτει την θύρα εξόδου:

- output_data: Θύρα εξόδου σήματος για τα δεδομένα που διαβάζονται από την μνήμη, με μέγεθος 32 bits.

3.3 Τελικός σχεδιασμός

Ακολουθώντας την τεχνική των μερικώς επικαλυπτόμενων λειτουργιών, σε συνδυασμό με τις μονάδες που αναφέρθηκαν στην ενότητα 3.2 και τις εντολές του συνόλου RV32I καταλήγουμε σε ένα τελικό σχεδιασμό. Για να είναι ολοκληρωμένος και λειτουργικός ο σχεδιασμός χρειάζονται επιπλέον στοιχεία λογικού σχεδιασμού. Στην επόμενη σελίδα απεικονίζεται ο τελικός σχεδιασμός.



Εικόνα 8: Τελικός σχεδιασμός

4. ΕΠΑΛΗΘΕΥΣΗ ΣΧΕΔΙΑΣΜΟΥ

Σε αυτό το κεφάλαιο θα γίνει επιβεβαίωση της σωστής λειτουργίας του επεξεργαστή μέσω μικρών προγραμμάτων που περιέχουν πολλών ειδών εξαρτήσεις.

4.1 Εξαρτήσεις από δεδομένα

4.1.1 Ανάγνωση μετά από εγγραφή

Παρακάτω ακολουθεί το πρόγραμμα που θα εκτελεστεί. Ο καταχωρητής x1 περιέχει την δεκαδική τιμή 2, ο x3 την δεκαδική τιμή 1 και ο x5 την τιμή 4. Όλοι οι υπόλοιποι έχουν αρχικοποιηθεί στην τιμή 0.

Πρόγραμμα	Κωδικοποίηση (hex)
sub x2, x1, x3	40308133
and x12, x2, x5	00517633
or x13, x6, x2	002366b3
add x14, x2, x2	00210733

Η πρώτη εξάρτηση εντοπίζεται μεταξύ της εντολής sub και της and αφού η sub αποθηκεύει το αποτέλεσμα της στον x2 και η and χρησιμοποιεί αυτό το αποτέλεσμα. Η δεύτερη εξάρτηση εντοπίζεται μεταξύ της εντολής sub και της or για τον ίδιο λόγο με παραπάνω. Οι εξαρτήσεις αυτές προκύπτουν διότι οι and και or φθάνουν στο στάδιο ID όπου γίνεται ανάγνωση των καταχωρητών προτού η sub φτάσει στο στάδιο WB. Μεταξύ των εντολών sub και add δεν υπάρχει εξάρτηση διότι η sub βρίσκεται στο στάδιο WB όταν η add βρίσκεται

στο ID. Δεν προκύπτει εξάρτηση διότι στο Αρχείο καταχωρητών η εγγραφή των καταχωρητών προηγείται της ανάγνωσης αυτών.

Αποτελέσματα εξομοίωσης:

```
time =        50, pc_out = 00000000, inst = 0000000f, fw0 = 00, fw1 = 00, data0 = xxxxxxxx, data1 = xxxxxxxx, Alu_res = xxxxxxxx

time =       100, pc_out = 00000000, inst = 40308133, fw0 = 00, fw1 = 00, data0 = xxxxxxxx, data1 = xxxxxxxx, Alu_res = xxxxxxxx
time =       150, pc_out = 00000001, inst = 40308133, fw0 = 00, fw1 = 00, data0 = 00000000, data1 = 00000000, Alu_res = 00000000

time =       200, pc_out = 00000001, inst = 00517633, fw0 = 00, fw1 = 00, data0 = 00000000, data1 = 00000000, Alu_res = 00000000
time =       250, pc_out = 00000002, inst = 00517633, fw0 = 00, fw1 = 00, data0 = 00000002, data1 = 00000001, Alu_res = 00000001

time =       300, pc_out = 00000002, inst = 002366b3, fw0 = 00, fw1 = 00, data0 = 00000002, data1 = 00000001, Alu_res = 00000001
time =       350, pc_out = 00000003, inst = 002366b3, fw0 = 10, fw1 = 00, data0 = 00000001, data1 = 00000004, Alu_res = 00000000

time =       400, pc_out = 00000003, inst = 00210733, fw0 = 10, fw1 = 00, data0 = 00000001, data1 = 00000004, Alu_res = 00000000
time =       450, pc_out = 00000004, inst = 00210733, fw0 = 00, fw1 = 01, data0 = 00000000, data1 = 00000001, Alu_res = 00000001

time =       500, pc_out = 00000004, inst = 00000000, fw0 = 00, fw1 = 01, data0 = 00000000, data1 = 00000001, Alu_res = 00000001
time =       550, pc_out = 00000005, inst = 00000000, fw0 = 00, fw1 = 00, data0 = 00000001, data1 = 00000001, Alu_res = 00000002
```

Το σήμα time οδηγείται από το ρολόι. Τις χρονικές στιγμές 100, 200 έως και 500 το ρολόι βρίσκεται στην θετική του ακμή. Ενώ τις 50, 150 έως και 550 βρίσκεται στην αρνητική ακμή. Ισχύει ότι στην αρνητική ακμή του ρολογιού έχει επιλεγεί ο Μετρητής Προγράμματος και οι ενδιάμεσοι καταχωρητές να διαδίδουν το περιεχόμενό τους στην επόμενη βαθμίδα.

Ανάλυση αποτελεσμάτων εξομοίωσης:

Αρχικά, τη χρονική στιγμή 100 διαβάζεται η κωδικοποίηση της εντολής sub x2, x1, x3 επομένως η sub βρίσκεται στο στάδιο IF. Την 150 η sub περνάει στο στάδιο ID και την 250 στο στάδιο EX, όπου εκεί γίνεται η εκτέλεση της πράξης. Το αποτέλεσμα της πράξης φαίνεται στο σήμα Alu_res το οποίο στην περίπτωση της sub είναι το αναμενόμενο ($2 - 1 = 1$). Τη χρονική στιγμή 350 η εντολή and περνάει στο στάδιο EX και όπως φαίνεται το σήμα data0 λαμβάνει την σωστή τιμή χάρη στη λειτουργία της Μονάδας Παροχέτευσης η οποία θέτει το σήμα fw0 στο 10. Τη χρονική στιγμή 450 η

εντολή or περνάει στο στάδιο EX και όπως φαίνεται το σήμα data1 λαμβάνει την σωστή τιμή χάρη στη λειτουργία της Μονάδας Παροχέτευσης η οποία θέτει το σήμα fw1 στο 01. Τη χρονική στιγμή 550 η εντολή add περνάει στο στάδιο EX και όπως φαίνεται τα σήματα data0, data1 λαμβάνουν τη σωστή τιμή η οποία διαβάζεται από το Αρχείο καταχωρητών.

4.1.2 Πράξεις αριθμών σε ένα καταχωρητή

Παρακάτω ακολουθεί το πρόγραμμα που θα εκτελεστεί. Ο καταχωρητής x1 περιέχει την δεκαδική τιμή 2, ο x2 την δεκαδική τιμή 6, ο x3 την δεκαδική τιμή 1 και ο x5 την τιμή 4. Όλοι οι υπόλοιποι έχουν αρχικοποιηθεί στην τιμή 0.

Πρόγραμμα	Κωδικοποίηση (hex)
1.add x1, x1, x2	002080b3
2.add x1, x1, x3	003080b3
3.add x1, x1, x5	005080b3

Σε αυτή την περίπτωση στον καταχωρητή x1 γράφουν οι εντολές 1 και 2, ενώ η εντολή 3 διαβάζει από αυτόν. Επομένως, κρίνεται αναγκαίο η Μονάδα Παροχέτευσης όταν η εντολή 3 φτάσει στο στάδιο EX, να διοχετεύσει την τιμή, που προορίζεται να αποθηκευτεί στον x1, από το στάδιο MEM όπου βρίσκεται η εντολή 2 και όχι από το στάδιο WB όπου βρίσκεται η εντολή 1.

Αποτελέσματα εξομοίωσης:

```
time =          50, pc_out = 00000000, inst = 0000000f, fw0 = 00, fw1 = 00, data0 = xxxxxxxx, data1 = xxxxxxxx, Alu_res = xxxxxxxx

time =         100, pc_out = 00000000, inst = 002080b3, fw0 = 00, fw1 = 00, data0 = xxxxxxxx, data1 = xxxxxxxx, Alu_res = xxxxxxxx
time =         150, pc_out = 00000001, inst = 002080b3, fw0 = 00, fw1 = 00, data0 = 00000000, data1 = 00000000, Alu_res = 00000000

time =         200, pc_out = 00000001, inst = 003080b3, fw0 = 00, fw1 = 00, data0 = 00000000, data1 = 00000000, Alu_res = 00000000
time =         250, pc_out = 00000002, inst = 003080b3, fw0 = 00, fw1 = 00, data0 = 00000002, data1 = 00000006, Alu_res = 00000008

time =         300, pc_out = 00000002, inst = 005080b3, fw0 = 00, fw1 = 00, data0 = 00000002, data1 = 00000006, Alu_res = 00000008
time =         350, pc_out = 00000003, inst = 005080b3, fw0 = 10, fw1 = 00, data0 = 00000008, data1 = 00000001, Alu_res = 00000009

time =         400, pc_out = 00000003, inst = 00000000, fw0 = 10, fw1 = 00, data0 = 00000008, data1 = 00000001, Alu_res = 00000009
time =         450, pc_out = 00000004, inst = 00000000, fw0 = 10, fw1 = 00, data0 = 00000009, data1 = 00000004, Alu_res = 0000000d
```

Το σήμα time οδηγείται από το ρολόι. Τις χρονικές στιγμές 100, 200 έως και 500 το ρολόι βρίσκεται στην θετική του ακμή. Ενώ τις 50, 150 έως και 550 βρίσκεται στην αρνητική ακμή. Ισχύει ότι στην αρνητική ακμή του ρολογιού έχει επιλεγεί ο Μετρητής Προγράμματος και οι ενδιάμεσοι καταχωρητές να διαδίδουν το περιεχόμενό τους στην επόμενη βαθμίδα.

Ανάλυση αποτελεσμάτων εξομοίωσης:

Αρχικά, τη χρονική στιγμή 100 διαβάζεται η κωδικοποίηση της εντολής 1 επομένως βρίσκεται στο στάδιο IF. Την χρονική στιγμή 150 η 1 περνάει στο στάδιο ID και την 250 στο στάδιο EX, όπου εκεί γίνεται η εκτέλεση της πράξης. Το αποτέλεσμα της πράξης φαίνεται στο σήμα Alu_res το οποίο στην περίπτωση της 1 είναι το αναμενόμενο (0x2 + 0x6 = 0x8). Τη χρονική στιγμή 350 η εντολή 2 περνάει στο στάδιο EX και όπως φαίνεται το σήμα data0 λαμβάνει την σωστή τιμή χάρη στη λειτουργία της Μονάδας Παροχέτευσης η οποία θέτει το σήμα fw0 στο 10. Το αποτέλεσμα της πράξης φαίνεται στο σήμα Alu_res το οποίο στην περίπτωση της 2 είναι το αναμενόμενο (0x8 + 0x1 = 9). Τη χρονική στιγμή 450 η εντολή 3 περνάει στο στάδιο EX και όπως φαίνεται το σήμα data0 λαμβάνει την σωστή τιμή χάρη στη λειτουργία της Μονάδας Παροχέτευσης η οποία θέτει το σήμα fw0 στο 10. Το

αποτέλεσμα της πράξης φαίνεται στο σήμα Alu_res το οποίο στην περίπτωση της 3 είναι το αναμενόμενο (0x9 + 0x4 = 0xd).

4.1.3 Ακολουθία εντολών load-use

Παρακάτω ακολουθεί το πρόγραμμα που θα εκτελεστεί. Ο καταχωρητής x1 περιέχει την δεκαδική τιμή 2, ο x2 την δεκαδική τιμή 6, ο x3 την δεκαδική τιμή 1 και ο x5 την τιμή 4. Όλοι οι υπόλοιποι έχουν αρχικοποιηθεί στην τιμή 0.

Πρόγραμμα	Κωδικοποίηση (hex)
lb x3, 0(x8)	00040183
ori x10, x3, 2	0021e513
slt x9, x2, x1	001124b3

Σε αυτή την περίπτωση η εντολή lb x3, 0(x8) αποθηκεύει στον καταχωρητή x3 και η εντολή ori x10, x3, 2 διαβάζει από αυτόν, με αποτέλεσμα να προκύπτει μία εξάρτηση δεδομένων. Η τιμή που θα αποθηκευτεί στον x3 εμφανίζεται πρώτη φορά τη στιγμή που η εντολή lb βρίσκεται στο στάδιο MEM διότι τότε γίνεται ανάγνωση της μνήμης δεδομένων, όμως αυτή η τιμή χρειάζεται να γίνει διαθέσιμη την ίδια στιγμή στο στάδιο EX όπου εκεί βρίσκεται η εντολή ori. Επομένως, είναι αναγκαίο μόλις αναγνωριστεί αυτή η εξάρτηση να «παγώσει» η εκτέλεση της εντολής ori για ένα κύκλο.

Αποτελέσματα εξομοίωσης:

```
time =        50, PCwrite = 1, pc_out = 00000000, inst = 0000000f, IF_IDwrite = 1, ID_EXwrite = 1, fw0 = 00, fw1 = 00, data0 = xxxxxxxx, data1 = xxxxxxxx, Alu_res = xxxxxxxx, DataMem_out = xxxxxxxx

time =       100, PCwrite = 1, pc_out = 00000000, inst = 00040183, IF_IDwrite = 1, ID_EXwrite = 1, fw0 = 00, fw1 = 00, data0 = xxxxxxxx, data1 = xxxxxxxx, Alu_res = xxxxxxxx, DataMem_out = xxxxxxxx
time =       150, PCwrite = 1, pc_out = 00000001, inst = 00040183, IF_IDwrite = 1, ID_EXwrite = 1, fw0 = 00, fw1 = 00, data0 = 00000000, data1 = 00000000, Alu_res = 00000000, DataMem_out = xxxxxxxx

time =       200, PCwrite = 1, pc_out = 00000001, inst = 0021e513, IF_IDwrite = 1, ID_EXwrite = 1, fw0 = 00, fw1 = 00, data0 = 00000000, data1 = 00000000, Alu_res = 00000000, DataMem_out = xxxxxxxx
time =       250, PCwrite = 0, pc_out = 00000002, inst = 0021e513, IF_IDwrite = 0, ID_EXwrite = 0, fw0 = 00, fw1 = 00, data0 = 00000000, data1 = 00000000, Alu_res = 00000000, DataMem_out = fffffffa

time =       300, PCwrite = 0, pc_out = 00000002, inst = 001124b3, IF_IDwrite = 0, ID_EXwrite = 0, fw0 = 00, fw1 = 00, data0 = 00000000, data1 = 00000000, Alu_res = 00000000, DataMem_out = fffffffa
time =       350, PCwrite = 1, pc_out = 00000002, inst = 001124b3, IF_IDwrite = 1, ID_EXwrite = 1, fw0 = 00, fw1 = 00, data0 = 00000000, data1 = 00000000, Alu_res = 00000000, DataMem_out = fffffffa

time =       400, PCwrite = 1, pc_out = 00000002, inst = 001124b3, IF_IDwrite = 1, ID_EXwrite = 1, fw0 = 00, fw1 = 00, data0 = 00000000, data1 = 00000000, Alu_res = 00000000, DataMem_out = 00000004
time =       450, PCwrite = 1, pc_out = 00000003, inst = 001124b3, IF_IDwrite = 1, ID_EXwrite = 1, fw0 = 01, fw1 = 00, data0 = 00000004, data1 = 00000002, Alu_res = 00000006, DataMem_out = 00000004

time =       500, PCwrite = 1, pc_out = 00000003, inst = 00000000, IF_IDwrite = 1, ID_EXwrite = 1, fw0 = 01, fw1 = 00, data0 = 00000004, data1 = 00000002, Alu_res = 00000006, DataMem_out = fffffffa
time =       550, PCwrite = 1, pc_out = 00000004, inst = 00000000, IF_IDwrite = 1, ID_EXwrite = 1, fw0 = 00, fw1 = 00, data0 = 00000006, data1 = 00000002, Alu_res = 00000000, DataMem_out = 00000000
```

Το σήμα time οδηγείται από το ρολόι. Τις χρονικές στιγμές 100, 200 έως και 500 το ρολόι βρίσκεται στην θετική του ακμή. Ενώ τις 50, 150 έως και 550 βρίσκεται στην αρνητική ακμή. Ισχύει ότι στην αρνητική ακμή του ρολογιού έχει επιλεγεί ο Μετρητής Προγράμματος και οι ενδιάμεσοι καταχωρητές να διαδίδουν το περιεχόμενό τους στην επόμενη βαθμίδα.

Ανάλυση αποτελεσμάτων εξομοίωσης:

Αρχικά, τη χρονική στιγμή 100 η εντολή lb βρίσκεται στο στάδιο IF διότι τότε διαβάζεται από την κρυφή μνήμη δεδομένων. Τη χρονική στιγμή 250 η εντολή lb περνάει στο στάδιο EX και η εντολή ori στο στάδιο ID. Σε αυτή τη στιγμή η Μονάδα Ανίχνευσης Εξαρτήσεων ανιχνεύει τη συγκεκριμένη εξάρτηση με αποτέλεσμα να οδηγήσει τα σήματα PCwrite, IF_IDwrite, ID_EXwrite στο 0 για ένα κύκλο ρολογιού. Το οποίο σημαίνει ότι θα σταματήσει η εκτέλεση των εντολών που ακολουθούν μετά την lb.

6. ΥΛΟΠΟΙΗΣΗ

6.1 Αρχεία Verilog που περιγράφουν του Υλικό

6.1.1 Περιγραφή Κρυφής Μνήμης Εντολών – InstCache.v

```
1  module InstCache (clk, addr, reset, inst);
2      input [31:0] addr;
3      input clk, reset;
4
5      output reg [31:0] inst;
6
7      wire [255:0] block;
8
9      reg [23:0] tags [31:0];
10     reg [255:0] data [31:0];
11     reg valid_bits [31:0];
12
13     initial begin
14         $readmemh("tags.mem", tags);
15         $readmemh("data.mem", data);
16         $readmemh("valid_bits.mem", valid_bits);
17     end
18
19     assign block = data[addr[7:3]];
20
21     always @ (posedge clk) begin
22         if ((valid_bits[addr[7:3]] == 1) && (tags[addr[7:3]] == addr[31:8])) begin
23             //block <= {data[addr[7:3]]};
24             case(addr[2:0])
25                 3'b000: begin
26                     inst <= block[255:224];
27                 end
28                 3'b001: begin
29                     inst <= block[223:192];
30                 end
31                 3'b010: begin
32                     inst <= block[191:160];
33                 end
```

```
34                3'b011: begin
35                    inst <= block[159:128];
36                end
37                3'b100: begin
38                    inst <= block[127:96];
39                end
40                3'b101: begin
41                    inst <= block[95:64];
42                end
43                3'b110: begin
44                    inst <= block[63:32];
45                end
46                3'b111: begin
47                    inst <= block[31:0];
48                end
49            endcase
50        end
51        else begin
52            inst <= 32'hF;
53        end
54    end
55 endmodule
```

6.1.2 Περιγραφή Ενδιάμεσου Καταχωρητή IF_ID – IF_ID.v

```
56 module IF_ID (clk, hazDetect_IF_ID, IF_Flush, pcPlusFour_i, pc_i, inst_i, pcPlusFour_o, pc_o,
   inst_o);
57
58 input clk, hazDetect_IF_ID, IF_Flush;
59 input [31:0] pcPlusFour_i, pc_i, inst_i;
60 output reg [31:0] pcPlusFour_o, pc_o, inst_o;
61
62 always @(negedge clk) begin
63     if(IF_Flush == 0) begin
64         if(hazDetect_IF_ID) begin
65             pcPlusFour_o <= pcPlusFour_i;
66             pc_o <= pc_i;
67             inst_o <= inst_i;
68         end
69     end
70     else begin
71         pcPlusFour_o <= 0;
72         pc_o <= 0;
```

```
73          inst_o <= 0;
74      end
75  end
76  endmodule
```

6.1.3 Περιγραφή Αρχείου Καταχωρητών – RegFile.v

6.1.4 Περιγραφή Μονάδας Ανίχνευσης Εξαρτήσεων – HazardDetectUnit.v

6.1.1 Περιγραφή Κρυφής Μνήμης Εντολών – DataCache.v

6.1.1 Περιγραφή Κρυφής Μνήμης Εντολών – DataCache.v

6.1.1 Περιγραφή Κρυφής Μνήμης Εντολών – DataCache.v

6.2 Testbenches αρχεία για την εξομοίωση του σχεδιασμού

7. ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Harris, S., & Harris, D. (2021). *Digital design and computer architecture: RISC-V Edition.* Morgan Kaufmann.

[2] "The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 2.2", Editors Andrew Waterman and Krste Asanovic, RISC-V Foundation, May 2017.

[3] Patterson, D. A., & Hennessy, J. L. (n.d.). *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*. Morgan Kaufmann.

[4] Dimitrios Nikolos. *Computer Architecture*. Pan. Papakonstantinou, 2017. ISBN: 978-618-83197-0-7.