**Εικόνα που περιέχει κείμενο, γραμματοσειρά, λογότυπο, έμβλημα

Περιγραφή που δημιουργήθηκε αυτόματα**

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Διπλωματική Εργασία



Τίτλος



Γιαννάκης Εμμανουήλ Δημήτριος

A.M. : 1067491

Επιβλέπων

Νικολός Δημήτριος, Καθηγητής

Μέλη Εξεταστικής Επιτροπής

Βέργος Χαρίδημος, Καθηγητής

Ευχαριστίες

Περίληψη

**Abstract**

ΠΕΡΙΕΧΟΜΕΝΑ

[1. ΕΙΣΑΓΩΓΗ 6](#_Toc157769405)

[1.1 Ο επεξεργαστής 6](#_Toc157769406)

[1.2 Αρχιτεκτονική συνόλου εντολών 6](#_Toc157769407)

[1.3 Αρχιτεκτονική RISC-V 7](#_Toc157769408)

[2. ΣΥΝΟΛΟ ΕΝΤΟΛΩΝ RV32I 10](#_Toc157769409)

[2.1 Τύποι κωδικοποίησης εντολών 10](#_Toc157769410)

[2.2 Εντολές αριθμητικών και λογικών πράξεων 14](#_Toc157769411)

[2.2.1 Εντολές για πράξεις μεταξύ δυο καταχωρητών 14](#_Toc157769412)

[2.2.2 Εντολές για πράξεις μεταξύ καταχωρητή – άμεσου δεδομένου 16](#_Toc157769413)

[2.3 Εντολές προσπέλασης της μνήμης 17](#_Toc157769414)

[2.3.1 Εντολές εγγραφής στη μνήμη (Store) 18](#_Toc157769415)

[2.3.2 Εντολές ανάγνωσης της μνήμης (Load) 19](#_Toc157769416)

[2.4 Εντολές διακλάδωσης 21](#_Toc157769417)

[2.4.1 Εντολές διακλάδωσης υπό συνθήκη (Conditional Branch) 21](#_Toc157769418)

[2.4.2 Εντολές άλματος (Jump) 23](#_Toc157769419)

[2.5 Εντολές φόρτωσης ειδικών δεδομένων 24](#_Toc157769420)

[3. ΕΠΕΞΕΡΓΑΣΤΗΣ 25](#_Toc157769421)

[3.1 Διασωλήνωση (Pipelining) 25](#_Toc157769422)

[6. ΒΙΒΛΙΟΓΡΑΦΙΑ 28](#_Toc157769423)

# 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. Οι επεξεργαστές σχεδιάζονται σύμφωνα με διάφορες αρχιτεκτονικές.

## 1.2 Αρχιτεκτονική συνόλου εντολών

H 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.

1. 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](#_6._ΒΙΒΛΙΟΓΡΑΦΙΑ)]. 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](#_6._ΒΙΒΛΙΟΓΡΑΦΙΑ)].

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. Παρακάτω θα αναλυθούν όλες 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:

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| funct7 | rs2 | rs1 | funct3 | rd | opcode |

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:

31 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| imm[11:0] | rs1 | funct3 | rd | opcode |

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:

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| imm[11:5] | rs2 | rs1 | funct3 | imm[4:0] | 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 εγγραφής στην μνήμη δεδομένων (Store).

B-type:

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 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 στη ίδια θέση, αυτό το χαρακτηριστικό προσφέρει απλότητα στον σχεδιασμό και κάνει την παραγωγή των δυαδικών κωδικοποιήσεων των εντολών πιο εύκολη [[3](#_6._ΒΙΒΛΙΟΓΡΑΦΙΑ_1)]. Παρακάτω ακολουθεί πίνακας που επεξηγεί κάθε ξεχωριστό πεδίο από τους παραπάνω τύπους κωδικοποίησης.

|  |  |  |  |
| --- | --- | --- | --- |
| Όνομα | Μέγεθος | Χρήση | 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 | O καταχωρητής στον οποίο αποθηκεύεται το αποτέλεσμα | 11 – 7 bits |
| opcode | 7 bit | O 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 Χ 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 ˆ 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 Χ θέσεις δεξιά. Η τιμή Χ 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 Χ θέσεις δεξιά. Η τιμή Χ 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 ˆ imm | Λογική πράξη XOR ανά bit μεταξύ των c(x1), imm. |
| ORI x0, x1, imm | x0 ‹‒ x1 ˆ imm | Λογική πράξη OR ανά bit μεταξύ των c(x1), imm. |
| ANDI x0, x1, imm | x0 ‹‒ x1 ˆ 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 Χ 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 Χ ισούται με τα 5 λιγότερο σημαντικά bit του imm. Αποθήκευση του αποτελέσματος στον καταχωρητή x0. |
| SRAΙ x0, x1, imm | x0 ‹‒ x1 >>> imm[4:0] | Αριθμητική ολίσθηση του c(x1) κατά Χ θέσεις δεξιά. Η τιμή Χ 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 φόρτωσης ειδικών δεδομένων

Η κατηγορία αυτών των εντολών περιλαμβάνει δυο εντολές που χρησιμοποιούν την κωδικοποίηση U-type.

# 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](#_2._ΣΥΝΟΛΟ_ΕΝΤΟΛΩΝ)).

## 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 υπομονάδες, οι οποίες εκτελούν η κάθε μια ένα ξεχωριστό βήμα. Η τεχνική αυτή, έχει εφαρμοστεί σε επεξεργαστές RISC και CISC αρχιτεκτονικής.

Η τεχνική των μερικώς επικαλυπτόμενων λειτουργιών στην αρχιτεκτονική RISC ορίζει τα εξής στάδια [[4](#_6._ΒΙΒΛΙΟΓΡΑΦΙΑ_1)]:

1. ΙF – 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.

1. ID – Instruction Decoding (Αποκωδικοποίηση εντολής):

Στο στάδιο αυτό γίνεται ανάγνωση των καταχωρητών και παράγονται τα ανάλογα σήματα ελέγχου για κάθε εντολή.

1. 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).

1. MEM – Memory Access (Προσπέλαση Μνήμης):

Στο στάδιο αυτό γίνεται ανάγνωση δεδομένων από τη μνήμη ή αποθήκευση δεδομένων σε αυτή.

1. WB – Write Back (Αποθήκευση αποτελεσμάτων):

Στο στάδιο αυτό που είναι και το τελευταίο, γίνεται αποθήκευση αποτελέσματος σε κάποιον από τους καταχωρητές.

# 

# 6. 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.