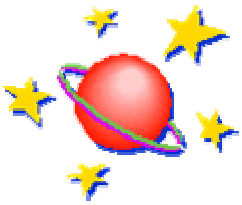
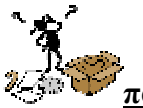


Ενότητα 4η : *Η Αρχιτεκτονική του υπολογιστή DLX*

Σκοπός **Ο σκοπός της ενότητας αυτής είναι να παρουσιάσει την αρχιτεκτονική του υπολογιστή DLX.**

Προσδοκώμενα Αποτελέσματα **Όταν θα έχετε μελετήσει την ενότητα, θα είστε σε θέση να:**

αναφέρετε τους καταχωρητές και τους τύπους δεδομένων του DLX,



παρουσιάζετε τους τρόπους διευθυνσιοδότησης που χρησιμοποιούνται στον DLX,

αναφέρετε τους τρόπους κωδικοποίησης των εντολών του DLX,

απαριθμείτε τις εντολές που υποστηρίζει ο DLX,

να εξηγείτε την εκτέλεση των εντολών που υποστηρίζει ο DLX.

Καταχωρητές και τύποι δεδομένων στον DLX

Ο DLX διαθέτει τους εξής καταχωρητές:

- 32 καταχωρητές γενικής χρήσης (**GPRs**) των 32 bits που ονομάζονται R0,R1,…,R31.

  Το περιεχόμενο του R0 είναι πάντοτε 0.

- Ένα σύνολο καταχωρητών κινητής υποδιαστολής (**FPRs**), που μπορούν να χρησιμοποιηθούν είτε ως 32 καταχωρητές των 32 bits απλής ακρίβειας είτε ως 16 ζεύγη των 64 bits που περιέχουν τιμές διπλής ακρίβειας και ονομάζονται F0, F2,…,F28, F30 και προκύπτουν από δύο καταχωρητές διπλής ακρίβειας (άρτιος-περιττός). Έτσι, για παράδειγμα ο καταχωρητής F2 αποτελείται από τους καταχωρητές διπλής ακρίβειας: F2 και F3

Οι τύποι δεδομένων του DLX είναι:

- 1 byte (8 bits), 2 bytes (16 bit – μισή λέξη) και 4 bytes (32 bits - λέξη) για ακέραια δεδομένα.
- 4 bytes (32 bits) απλής ακρίβειας και 8 bytes (64 bits) διπλής ακρίβειας για κλασματικούς αριθμούς (κινητής υποδιαστολής).

Τρόποι διευθυνσιοδότησης στον DLX

Οι τρόποι διευθυνσιοδότησης δεδομένων στον DLX είναι δύο:

- **Απευθείας** (με πεδίο 16-bits για τη τιμή του απευθείας δεδομένου)
- **Με Μετατόπιση** (με πεδίο 16-bits για τη μετατόπιση)

Οι δύο παραπάνω τρόποι διευθυνσιοδότησης εφαρμόζονται ως εξής:

- Ο **απευθείας τρόπος διευθυνσιοδότησης** σε αριθμητικές και λογικές εντολές, όπου ο ένας πηγαίος τελεστέος βρίσκεται σε καταχωρητή και ο δεύτερος είναι απευθείας δεδομένο. Το αποτέλεσμα της πράξης της εντολής καταχωρείται σε έναν καταχωρητή. Ο τρόπος διευθυνσιοδότησης στην περίπτωση αυτή προσδιορίζεται από τον κωδικό λειτουργίας (opcode) της εντολής.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 1

α) Ποιος τρόπος διευθυνσιοδότησης χρησιμοποιείται στην παρακάτω εντολή του υπολογιστή DLX; Να αιτιολογήσετε την απάντησή σας.

ADDI R5, R3, #100

β) Να περιγράψετε την εκτέλεση της εντολής. Ποιο θα είναι το αποτέλεσμα αν το περιεχόμενο του καταχωρητή R3 είναι 20;

γ) Να εκτελέσετε την παραπάνω εντολή χρησιμοποιώντας το λογισμικό που βρίσκεται στην ιστοσελίδα:

http://hermes.di.uoa.gr/DLXSim/instuctions/instr.html

δ) Να συγκρίνετε το αποτέλεσμα της εκτέλεση της εντολής από το λογισμικό, με την απάντηση που δώσατε.

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 1

α) Η εντολή αυτή είναι αριθμητική / λογική εντολή, όπου ο ένας πηγαίος τελεστέος βρίσκεται σε καταχωρητή (στον R3), ο δεύτερος είναι απευθείας δεδομένο (#100) και το αποτέλεσμα της πράξης της εντολής καταχωρείται στον καταχωρητή προορισμού (R5). Επιπλέον, ο τρόπος διευθυνσιοδότησης προσδιορίζεται από τον κωδικό λειτουργίας της εντολής (I). Επομένως χρησιμοποιείται ο απευθείας τρόπος διευθυνσιοδότησης.

β) Το περιεχόμενο του καταχωρητή R3 (20) θα προστεθεί με το 100 και το αποτέλεσμα (120) θα αποθηκευτεί στον καταχωρητή R5. Η εκτέλεση της εντολής μπορεί να παρασταθεί συμβολικά ως εξής:

$$\text{Regs[R5]} \leftarrow \text{Regs[R3]} + 100$$

Σε όλες τις αριθμητικές και λογικές εντολές στις οποίες δεν χρησιμοποιείται ο απευθείας τρόπος διευθυνσιοδότησης, στον κωδικό λειτουργίας βρίσκεται κωδικοποιημένη η πληροφορία ότι η εντολή είναι μια αριθμητική / λογική εντολή και υπονοείται ότι χρησιμοποιείται ο τρόπος διευθυνσιοδότησης άμεσος καταχωρητή. Δηλαδή τόσο οι πηγαίοι τελεστέοι όσο και ο τελεστέος προορισμού είναι περιεχόμενα καταχωρητή.

- Ο **τρόπος διευθυνσιοδότησης με μετατόπιση** χρησιμοποιείται μόνο στις εντολές φόρτωσης και στις εντολές αποθήκευσης.

Παράδειγμα

Στην εντολή *LB R5, 100(R1)* η διεύθυνση του πηγαίου τελεστέου υπολογίζεται προσθέτοντας τη τιμή 100 με το περιεχόμενο του καταχωρητή R1. Το αποτέλεσμα αυτής της πρόσθεσης είναι η διεύθυνση μιας θέσης στη μνήμη. Γίνεται ανάγνωση του περιεχομένου της θέσης μνήμης και στη συνέχεια καταχωρείται στον καταχωρητή R5.

Ο τρόπος διευθυνσιοδότησης έμμεσος καταχωρητή προκύπτει χρησιμοποιώντας ως τιμή μετατόπισης την τιμή 0. Έτσι για παράδειγμα στην εντολή *LB R3, 0(R5)* παρατηρούμε ότι η τιμή της μετατόπισης είναι 0. Στην εντολή αυτή, η διεύθυνση του πηγαίου τελεστέου υπολογίζεται προσθέτοντας την τιμή 0 στο περιεχόμενο του καταχωρητή R5. Δηλαδή, η διεύθυνση του πηγαίου τελεστέου είναι το περιεχόμενο του καταχωρητή R5 (έμμεσος καταχωρητή). Το περιεχόμενο της θέσης μνήμης που έχει τη διεύθυνση αυτή διαβάζεται και στη συνέχεια καταχωρείται στον καταχωρητή R3.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2

Ο τρόπος διευθυνσιοδότησης έμμεσος καταχωρητή προκύπτει από τον τρόπο διευθυνσιοδότησης με μετατόπιση. Μπορείτε να βρείτε έναν άλλο τρόπο διευθυνσιοδότησης που να προκύπτει με ανάλογο τρόπο από τους τρόπους διευθυνσιοδότησης που χρησιμοποιεί ο DLX; Να δώσετε ένα παράδειγμα εντολής του DLX που χρησιμοποιεί τον τρόπο διευθυνσιοδότησης που προτείνατε. Στην περίπτωση που δυσκολεύεστε, κρίνεται σκόπιμο να ανατρέξετε στον πίνακα του σχήματος 2.3.2 της 3ης ενότητας.

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 2

Ο απόλυτος τρόπος διευθυνσιοδότησης προκύπτει χρησιμοποιώντας ως τιμή μετατόπισης τη διεύθυνση μνήμης του δεδομένου που θέλουμε να προσπελάσουμε και ως καταχωρητή βάσης τον R0 (έχει πάντα την τιμή 0). Για παράδειγμα, στην εντολή *LB R3, 1000(R0)* παρατηρούμε ότι η τιμή της μετατόπισης είναι 1000 και ο R0 είναι ο καταχωρητής βάσης. Στην εντολή αυτή η διεύθυνση του πηγαίου τελεστέου υπολογίζεται προσθέτοντας την τιμή 1000 με το περιεχόμενο του καταχωρητή R0 (που είναι 0), με άλλα λόγια η διεύθυνση του πηγαίου τελεστέου είναι η τιμή 1000 (απόλυτος). Γίνεται ανάγνωση στο περιεχόμενο της θέσης μνήμης που έχει τη διεύθυνση αυτή και στη συνέχεια καταχωρείται στον καταχωρητή R3.

Παρατηρούμε ότι στον DLX μπορούμε να χρησιμοποιήσουμε τέσσερις τρόπους διευθυνσιοδότησης, παρά το γεγονός ότι ο DLX υποστηρίζει μόνο δύο.

Πώς ερμηνεύονται οι διευθύνσεις μνήμης στον DLX

- Οι διευθύνσεις μνήμης στον DLX έχουν μέγεθος 32 bits.
- Η διευθυνσιοδότηση της μνήμης γίνεται κατά byte και ακολουθεί το μοντέλο Big Endian.
- Η επικοινωνία με τη μνήμη γίνεται μέσω των εντολών φόρτωσης και αποθήκευσης.
- Όλες οι προσβάσεις στη μνήμη πρέπει να ευθυγραμμίζονται.

Πρόσβαση στη μνήμη υπάρχει μόνο σε εντολές φόρτωσης - αποθήκευσης. Τα δεδομένα που προσπελάζονται με τις εντολές αυτές είναι μεγέθους:

- byte : 8 bits (B)
- μισή λέξη : 16 bits (H)
- λέξη : 32 bits (W)
- μεγάλη λέξη : 64 bits (L)

Το μέγεθος των δεδομένων στα οποία γίνεται προσπέλαση, βρίσκεται κωδικοποιημένο στον κωδικό λειτουργίας της εντολής.

Παράδειγμα

Να εξηγήσετε τις παρακάτω εντολές φόρτωσης του DLX:
A. LB R1, 50(R2)
B. LH R1, 50(R2)
Θεωρήστε ότι στον καταχωρητή R2 είναι καταχωρημένη η τιμή 50.

Απάντηση:

A. Με την εντολή αυτή φορτώνεται στον καταχωρητή R1 το περιεχόμενο της θέσης μνήμης, η διεύθυνση της οποίας υπολογίζεται προσθέτοντας στην τιμή 50 την τιμή του καταχωρητή R2 (50). Έτσι η διεύθυνση μνήμης που βρίσκονται τα δεδομένα είναι η 100.

Παρατηρούμε ότι το μέγεθος των δεδομένων που θα διαβαστούν από τη μνήμη είναι 1 byte (λόγω του γράμματος B στην εντολή L_).

1 byte

100	Δεδομένα μεγέθους 1 byte

B. Με την εντολή αυτή φορτώνεται στον καταχωρητή R1 το περιεχόμενο της θέσης μνήμης η διεύθυνση της οποίας υπολογίζεται προσθέτοντας στην τιμή 50 την τιμή

του καταχωρητή R2 (50). Έτσι η διεύθυνση μνήμης που βρίσκονται τα δεδομένα είναι η 100.
Παρατηρούμε ότι το μέγεθος των δεδομένων που θα διαβαστούν από τη μνήμη είναι 2 bytes (λόγω του γράμματος H στην εντολή L_). Έτσι τα δεδομένα βρίσκονται σε δύο διαδοχικά bytes της μνήμης, στα bytes με διευθύνσεις 100 και 101. Στο παρακάτω σχήμα μπορούμε να δούμε πώς τα δεδομένα είναι αποθηκευμένα στη μνήμη.

1 byte

100	1° byte δεδομένων
101	2° byte δεδομένων

Από τη στιγμή που ανακτηθούν τα bytes των δεδομένων από τη μνήμη, πρέπει τα bytes αυτά να τοποθετηθούν στη σωστή θέση ώστε να ανακτηθούν όλα τα δεδομένα μεγέθους μισής λέξης (half-word). Λόγω του ότι η διευθυνσιοδότηση της μνήμης ακολουθεί το μοντέλο Big Endian, το byte που βρισκόταν στη θέση μνήμης με διεύθυνση 100 θα τοποθετηθεί στη σημαντικότερη θέση των δεδομένων, ενώ το byte που βρισκόταν στη θέση μνήμης με διεύθυνση 101 θα τοποθετηθεί στη λιγότερο σημαντική θέση των δεδομένων. Στο παρακάτω σχήμα μπορούμε να δούμε ποια μορφή θα έχουν τα ανακτημένα δεδομένα μεγέθους 2 bytes.

1 byte	**1 byte**
(Σημαντικότερο byte) 1° byte δεδομένου	(Λιγότερο σημαντικό byte) 2° byte δεδομένου

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 3

α) Να περιγράψετε την εκτέλεση της παρακάτω εντολής του υπολογιστή DLX. Ποιο θα είναι το αποτέλεσμα αν το περιεχόμενο του καταχωρητή R4 είναι 100;

LW R2, 150(R4)

β) Με ποιο τρόπο αποθηκεύονται τα δεδομένα στη μνήμη ανάλογα με το μέγεθός τους;

γ) Να εκτελέσετε την παραπάνω εντολή χρησιμοποιώντας το λογισμικό που βρίσκεται στην ιστοσελίδα:

http://hermes.di.uoa.gr/DLXSim/instuctions/instr.html

δ) Να συγκρίνετε το αποτέλεσμα της εκτέλεση της εντολής από το λογισμικό, με την απάντηση που δώσατε.

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 3

α) Με την εντολή αυτή φορτώνεται στον καταχωρητή R2 το περιεχόμενο της θέσης μνήμης η διεύθυνση της οποίας υπολογίζεται προσθέτοντας στην τιμή 150 την τιμή του καταχωρητή R4 (100). Έτσι η διεύθυνση μνήμης που βρίσκεται το δεδομένο είναι η 250.

β) Παρατηρούμε ότι το μέγεθος των δεδομένων που θα διαβαστούν από τη μνήμη είναι 4 bytes (λόγω του γράμματος W στην εντολή L_). Έτσι τα δεδομένα βρίσκονται σε τέσσερα διαδοχικά bytes της μνήμης, στα bytes με διευθύνσεις 250, 251, 252 και 253. Στο παρακάτω σχήμα μπορούμε να δούμε πώς τα δεδομένα είναι αποθηκευμένα στη μνήμη.

	1 byte
250	1^{ο} byte δεδομένου
251	2^{ο} byte δεδομένου
252	3^{ο} byte δεδομένου
253	4^{ο} byte δεδομένου

Από τη στιγμή που ανακτηθούν τα bytes των δεδομένων από τη μνήμη θα πρέπει τα bytes αυτά να τοποθετηθούν στη σωστή θέση ώστε να ανακτηθούν όλα τα δεδομένα μεγέθους λέξης (word). Λόγω του ότι η διευθυνσιοδότηση της μνήμης ακολουθεί το μοντέλο Big Endian το byte που βρισκόταν στη θέση μνήμης με διεύθυνση 250 θα τοποθετηθεί στη σημαντικότερη θέση των δεδομένων, το byte που βρισκόταν στη θέση μνήμης με διεύθυνση 251 θα τοποθετηθεί στην αμέσως επόμενη θέση των δεδομένων, ενώ το byte που βρισκόταν στη θέση μνήμης με διεύθυνση 253 θα τοποθετηθεί στη λιγότερο σημαντική θέση των δεδομένων. Στο παρακάτω σχήμα μπορούμε να δούμε ποια μορφή θα έχουν τα δεδομένα μεγέθους 4 bytes, που έχουν ανακτηθεί.

1 byte	**1 byte**	**1 byte**	**1 byte**
1^{ο} byte δεδομένου (Σημαντικότερο byte)	2^{ο} byte δεδομένου	3^{ο} byte δεδομένου	4^{ο} byte δεδομένου (Λιγότερο σημαντικό byte)

Κωδικοποίηση εντολών του DLX

Η κωδικοποίηση των εντολών του υπολογιστή DLX ανήκει στο σταθερό τρόπο κωδικοποίησης. Όλες οι εντολές του DLX μετά την κωδικοποίηση έχουν σταθερό μέγεθος 32 bits, με τα 6 πρώτα bits να περιέχουν τον κωδικό της εντολής ή πληροφορίες για την εντολή αυτή. Αφού οι τρόποι διευθυνσιοδότησης στον DLX είναι μόνο δύο, μπορούν να κωδικοποιηθούν στον κωδικό λειτουργίας της εντολής.

Οι εντολές του DLX κωδικοποιούνται σε έναν από τους παρακάτω τρεις τύπους:

- **Εντολή τύπου - I**
- **Εντολή τύπου - R**
- **Εντολή τύπου - J**

Εντολή τύπου - I

6	5	5	16	bits
Κωδικός λειτουργίας εντολής	Rs1	Rd	Immediate (απευθείας)	

Οι εντολές που κωδικοποιούνται με αυτό τον τρόπο είναι:

- Εντολές φόρτωσης και αποθήκευσης δεδομένων μεγέθους: byte, μισής λέξης, λέξης.
- Πράξεις με τιμές που δίνονται με τον απευθείας τρόπο διευθυνσιοδότησης: (Rd ← Rs1 op immediate)*.
- Εντολές διακλάδωσης με συνθήκη (χρησιμοποιείται ο Rs1 ενώ ο Rd δεν χρησιμοποιείται).
- Μεταπήδησης, μεταπήδησης και σύνδεσης με καταχωρητή (rd = 0, rs = προορισμός, απευθείας = 0).

Παράδειγμα

Ποια μορφή θα έχει η παρακάτω εντολή πρόσθεσης του υπολογιστή DLX; Να εξηγήσετε τον τρόπο με τον οποίο εκτελείται η συγκεκριμένη εντολή.

ADDI R1,R2,#53

* Οι συμβολισμοί Rs και Rd αναφέρονται στον πηγαίο καταχωρητή (source register) και τον καταχωρητή προορισμού (destination register) αντίστοιχα.

Απάντηση:

Μετά την κωδικοποίηση η εντολή έχει τη παρακάτω μορφή:

6	5	5	16	bits
ADDI	R2	R1	53	

Στα 6 πρώτα bits (0..5) περιέχεται ο κωδικός της εντολής. Στα επόμενα πέντε (6..10) ο καταχωρητής που περιέχει τον πηγαίο τελεστέο, στα bits 11..15 ο καταχωρητής στον οποίο θα καταχωρηθεί ο τελεστέος προορισμού και στα 16 τελευταία bits περιέχεται η απευθείας τιμή.
Με την εντολή αυτή προστίθεται η τιμή 53 με το περιεχόμενο του καταχωρητή R2 και το αποτέλεσμα καταχωρείται στον καταχωρητή R2. Έτσι αν για παράδειγμα ο R2 είχε την τιμή 47 μετά την εκτέλεση της εντολής ο καταχωρητής R1 θα έχει την τιμή 100.

Παράδειγμα

Ποια μορφή θα έχει η παρακάτω εντολή μεταπήδησης και σύνδεσης του υπολογιστή DLX; Να εξηγήσετε τον τρόπο με τον οποίο εκτελείται η συγκεκριμένη εντολή.

JALR R2

Απάντηση:

Μετά την κωδικοποίηση η εντολή έχει τη παρακάτω μορφή:

6	5	5	16	bits
JALR	R2	0	0	

Ο καταχωρητής που καθορίζει τον προορισμό βρίσκεται κωδικοποιημένος στα bits 6..10. Με την εντολή αυτή, το περιεχόμενο του καταχωρητή R2 φορτώνεται στον απαριθμητή προγράμματος (PC). Έτσι στη συνέχεια θα εκτελεστεί η εντολή που βρίσκεται στη θέση μνήμης η διεύθυνση της οποίας καθορίζεται από το περιεχόμενο του PC. Έτσι αν για παράδειγμα ο R2 είχε την τιμή 500 τότε η επόμενη εντολή που θα εκτελεστεί είναι εκείνη που βρίσκεται στη διεύθυνση 500 της μνήμης.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 4

α) Ποιος τρόπος διευθυνσιοδότησης χρησιμοποιείται στην παρακάτω εντολή φόρτωσης του DLX;

LB R1,(R2)

β) Με ποιο τρόπο κωδικοποιείται η συγκεκριμένη εντολή;

γ) Ποιο θα είναι το αποτέλεσμα μετά την εκτέλεση της εντολής, αν ο καταχωρητής R2 έχει τιμή 1000 και η διεύθυνση της θέσης μνήμης 1000 περιέχει την τιμή 500;

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 4

α) Στην εντολή αυτή χρησιμοποιείται ο τρόπος διευθυνσιοδότησης έμμεσος με καταχωρητή.

β) Η εντολή μετά την κωδικοποίηση γίνεται:

6	5	5	16	bits
LB	R2	R1	0	

Ο έμμεσος καταχωρητή προκύπτει χρησιμοποιώντας ως τιμή μετατόπισης την τιμή 0, έτσι στο πεδίο των 16-bits της κωδικοποιημένης εντολής υπάρχει η τιμή 0.

γ) Με την εντολή αυτή καταχωρείται στον καταχωρητή R1 το περιεχόμενο της θέσης μνήμης, η διεύθυνση της οποίας καθορίζεται από το περιεχόμενο του καταχωρητή R2. Έτσι αν ο καταχωρητής έχει την τιμή 1000 και στη θέση μνήμης 1000 βρίσκεται η τιμή 500, τότε μετά την εκτέλεση της εντολής ο καταχωρητής R1 έχει την τιμή 500.

Εντολή τύπου - R

6	5	5	5	11	bits
Κωδικός λειτουργίας εντολής	Rs1	Rs2	Rd	Func (Πράξη)	

Κωδικοποιεί αριθμητικές και λογικές εντολές: Rd ← Rs1 func Rs2.

Στον κωδικό λειτουργίας της εντολής περιέχεται η πληροφορία ότι η εντολή είναι μια αριθμητική / λογική εντολή στην οποία χρησιμοποιούνται μόνο καταχωρητές (υπονοείται ότι ο τρόπος διευθυνσιοδότησης των τελεστέων είναι άμεσος καταχωρητή).

Στο πεδίο Func βρίσκεται κωδικοποιημένη η πράξη που θα εκτελεστεί μεταξύ των περιεχομένων του καταχωρητή Rs1 ($1^{ος}$ πηγαίος καταχωρητής) και Rs2 ($2^{ος}$ πηγαίος καταχωρητής).

Παράδειγμα

Πως κωδικοποιείται η παρακάτω εντολή πρόσθεσης του DLX:
ADD R1, R2, R3

Απάντηση:

Η εντολή μετά την κωδικοποίηση έχει την παρακάτω μορφή:

6	5	5	5	11	bits
Κωδικός λειτουργίας εντολής	R2	R3	R1	ADD	

Οι καταχωρητές που περιέχουν τους πηγαίους τελεστέους βρίσκονται κωδικοποιημένοι στα bits 6..10 και 11..15 , ενώ ο καταχωρητής στον οποίο θα καταχωρηθεί ο τελεστέος προορισμού βρίσκεται κωδικοποιημένος στα bits 16..20. Στα τελευταία 11 bits βρίσκεται κωδικοποιημένη η πράξη που θα γίνει μεταξύ των δύο πηγαίων τελεστέων (στο παράδειγμά μας είναι η πρόσθεση ADD).

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 5

- **Να περιγράψετε την εκτέλεση της παρακάτω εντολής του υπολογιστή DLX.**
  ADD R1, R2, R3
- **Ποιο θα είναι το αποτέλεσμα αν το περιεχόμενο του καταχωρητή R2 είναι 43 και το περιεχόμενο του καταχωρητή R3 είναι 57;**
- **Να εκτελέσετε την παραπάνω εντολή χρησιμοποιώντας το λογισμικό που βρίσκεται στην ιστοσελίδα:**
  http://hermes.di.uoa.gr/DLXSim/instuctions/instr.html
- **γ) Να συγκρίνετε το αποτέλεσμα της εκτέλεση της εντολής από το λογισμικό, με την απάντηση που δώσατε.**

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 5

Το περιεχόμενο του καταχωρητή R2 (43) προστίθεται στο περιεχόμενο του καταχωρητή R3 (57) και το αποτέλεσμα (100) αποθηκεύεται στον καταχωρητή R1.

Εντολή τύπου - J

6	26 bits
Κωδικός λειτουργίας εντολής	Offset που προστίθεται στον PC

Κωδικοποιεί εντολές: Μεταπήδησης, μεταπήδησης με σύνδεση και παγίδες.

Παράδειγμα

Εξηγήστε την παρακάτω εντολή μεταπήδησης του DLX;
J name (το name συμβολίζει μια διεύθυνση μνήμης)

Απάντηση:

Η εντολή μετά την κωδικοποίηση έχει την παρακάτω μορφή:

6	26 bits
J	offset

Στο πεδίο των 26 bits βρίσκεται η τιμή (offset) που θα προστεθεί στον PC ώστε να δώσει το στόχο της μεταπήδησης (name). Έστω ότι η εντολή μεταπήδησης βρίσκεται στη θέση μνήμης με διεύθυνση 1000 και το name έχει την τιμή 1050. Ο PC μετά την αποκωδικοποίηση της εντολής έχει αυξηθεί κατά 4, δηλαδή ο PC έχει την τιμή 1004. Έτσι το offset που θα προστεθεί στον PC ώστε να δώσει τιμή 1050 είναι 46. Δηλαδή: PC=1004, offset=46, name=1050, name=PC+offset.

Σύνολο εντολών του DLX

Ο DLX υποστηρίζει τέσσερις κατηγορίες εντολών:

- Μεταφορά δεδομένων
- Αριθμητικές και λογικές εντολές
- Εντολές ελέγχου (Διακλαδώσεις με συνθήκη και μεταπηδήσεις)
- Πράξεις μεταξύ κλασματικών αριθμών (κινητής υποδιαστολής)

Εντολές μεταφοράς δεδομένων

Δεν μπορεί να γίνει μεταφορά δεδομένων από μια θέση μνήμης σε κάποια άλλη με μια μόνο εντολή. Σε αυτή την περίπτωση είναι απαραίτητη η χρήση ενός καταχωρητή για την προσωρινή αποθήκευση των δεδομένων που θα πάρουμε από τη θέση μνήμης.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 6

Για ποιο λόγο δεν είναι εφικτή η μεταφορά δεδομένων από μια θέση μνήμης σε μια άλλη στον υπολογιστή DLX; Στην περίπτωση που δυσκολεύεστε να απαντήσετε στην ερώτηση, καλό θα ήταν να ανατρέξετε στην 1η ενότητα και συγκεκριμένα στην υποενότητα «Λειτουργίες που υποστηρίζουν οι υπολογιστές RISC και CISC».

Παράδειγμα

Ποια είναι η λειτουργία της παρακάτω εντολής φόρτωσης του DLX:
LBU R1, 40(R3) (Load Byte Unsigned)

Απάντηση:

Η εντολή αυτή χρησιμοποιείται για να φορτωθεί ένα byte στον καταχωρητή R1 (32-bit) και τα υπόλοιπα 24 bits του καταχωρητή να γεμίσουν με μηδενικά (*unsigned).* Μπορούμε να περιγράψουμε την εντολή συμβολικά ως εξής:

$$Regs[R1] \leftarrow_{32} 0^{24} \,\#\#\, Mem[40+Regs[R3]]$$

Το 0^{24} σημαίνει ότι δημιουργείται ένα πεδίο μεγέθους 24 bits που περιέχει μόνο μηδενικά. Στη συνέχεια συνδέεται στο πεδίο αυτό το byte που βρίσκεται στη θέση μνήμης η διεύθυνση της οποίας υπολογίζεται προσθέτοντας το περιεχόμενο του καταχωρητή R3 με την τιμή 40 (Mem[40+Regs[R3]]). Για την εύρεση δηλαδή της διεύθυνσης στη μνήμη που βρίσκεται ο πηγαίος τελεστέος, χρησιμοποιείται ο τρόπος διευθυνσιοδότησης με μετατόπιση. Η σύνδεση αυτή συμβολίζεται με το σύμβολο ##.

Παράδειγμα

Ποια η λειτουργία της παρακάτω εντολής αποθήκευσης του DLX:

SH 502(R2), R3 (Store Half word)

Απάντηση:

Χρησιμοποιώντας διευθυνσιοδότηση με μετατόπιση μπορεί να υπολογιστεί η διεύθυνση μνήμης στην οποία θα αποθηκευτούν τα δύο λιγότερο σημαντικά bytes του καταχωρητή R3. Η διεύθυνση μνήμης υπολογίζεται από το άθροισμα του περιεχομένου του καταχωρητή R2 με την τιμή 502. Μπορούμε να περιγράψουμε την εντολή συμβολικά, ως εξής:

$$\text{MEM[502+Regs[R2]]} \leftarrow_{16} \text{Regs[R3]}_{16..31}$$

Το σύμβολο $\leftarrow_{16}$ σημαίνει ότι μεταφέρονται 16 bits στη μνήμη και το σύμβολο $Regs[R3]_{16..31}$ σημαίνει ότι παίρνουμε τα δύο λιγότερο σημαντικά bytes (μισή λέξη - lower half) του καταχωρητή R3.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 7

Να εξηγήσετε τον τρόπο με τον οποίο εκτελούνται οι παρακάτω εντολές του υπολογιστή DLX, χρησιμοποιώντας συμβολισμούς.

Εντολή	Όνομα εντολής
LW R1,30(R2)	Φόρτωση λέξης
LW R1, 1000(R0)	Φόρτωση λέξης
LB R1, 40(R3)	Φόρτωση byte
LBU R1,40(R3)	Φόρτωση μη προσημασμένου byte
LH R1, 40(R3)	Φόρτωση μισής λέξης
LF F0, 50(R3)	Φόρτωση μεταβλητής κινητής υποδιαστολής
LD F0, 50(R2)	Φόρτωση διπλής λέξης
SW 500(R4), R3	Αποθήκευση λέξης
SF 40(R3), F0	Αποθήκευση μεταβλητής κινητής υποδιαστολής
SD 40(R3), F0	Αποθήκευση διπλής λέξης
SH 502(R2), R3	Αποθήκευσης μισής λέξης
SB 41(R3), R2	Αποθήκευση byte

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 7

Εντολή	Επεξήγηση εντολής
LW R1,30(R2)	Regs[R1] $\leftarrow_{32}$ Mem[30+Regs[R2]]
LW R1, 1000(R0)	Regs[R1] $\leftarrow_{32}$ Mem[1000+Regs[R0]]
LB R1, 40(R3)	Regs[R1] $\leftarrow_{32}$ $(\text{Mem[40+Regs[R3]]}_0)^{24}$ ## Mem[40+Regs[R3]]
LBU R1,40(R3)	Regs[R1] $\leftarrow_{32}$ 0^{24} ## Mem[40+Regs[R3]]
LH R1, 40(R3)	Regs[R1] $\leftarrow_{32}$ $(\text{Mem[40+Regs[R3]]}_0)^{16}$ ## Mem[40+Regs[R3]]##Mem[41+Regs[R3]]
LF F0, 50(R3)	Regs[F0] $\leftarrow_{32}$ Mem[50+Regs[R3]]
LD F0, 50(R2)	Regs[F0] ## Regs[F1] $\leftarrow_{64}$ Mem[50+Regs[R2]]
SW 500(R4), R3	Mem[500+Regs[R4]]] $\leftarrow_{32}$ Regs[R3]
SF 40(R3), F0	Mem[40+Regs[R3]] $\leftarrow_{32}$ Regs[F0]
SD 40(R3), F0	Mem[40+Regs[R3]] $\leftarrow_{32}$ Regs[F0] Mem[44+Regs[R3]] $\leftarrow_{32}$ Regs[F1]
SH 502(R2), R3	Mem[502+Regs[R2]] $\leftarrow_{16}$ $\text{Regs[R31]}_{16...31}$
SB 41(R3), R2	Mem[41+Regs[R3]] $\leftarrow_{8}$ $\text{Regs[R2]}_{24...31}$

Αριθμητικές και λογικές εντολές

Σε όλες τις αριθμητικές και λογικές εντολές στις οποίες οι πηγαίοι τελεστέοι και οι τελεστέοι προορισμού βρίσκονται σε καταχωρητές, υπονοείται ότι ο τρόπος διευθυνσιοδότησης των τελεστέων ο άμεσος καταχωρητή. Οι εντολές αυτού του τύπου περιλαμβάνουν:

1. Αριθμητικές και λογικές πράξεις (πρόσθεση, αφαίρεση, λογικό AND, λογικό OR, λογικό XOR, συγκρίσεις, ολισθήσεις). Ένας πηγαίος τελεστέος μπορεί να είναι απευθείας δεδομένο των 16 bits με επέκταση προσήμου. Η λειτουργία LHI (Load High Immediate), φορτώνει στο πάνω μισό ενός καταχωρητή την απευθείας τιμή που δίνεται, ενώ τοποθετεί 0 στο κάτω μισό.
2. Εντολές σύγκρισης (=, ≠ , <, >, ≤ , ≥). Τοποθετούν 1 ή 0 στον καταχωρητή προορισμού ανάλογα με το αποτέλεσμα της σύγκρισης. Οι κωδικοί των εντολών είναι set-equal, set-not-equal, set-less-than κ.ο.κ.

Παράδειγμα

Ποια είναι η σημασία της παρακάτω λογικής εντολής του DLX:
SLT R1, R2, R3 (Set Less Than)

Απάντηση:

Με την εντολή αυτή συγκρίνονται τα περιεχόμενα του R2 και R3 και ο καταχωρητής R1 παίρνει την τιμή 1 ή την τιμή 0, ανάλογα με το αποτέλεσμα της σύγκρισης. Έτσι αν το περιεχόμενο του R2 είναι μικρότερο απ'αυτό του R3 τότε ο καταχωρητής R1 τίθεται 1, διαφορετικά τίθεται 0. Μπορούμε να περιγράψουμε την εντολή αυτή χρησιμοποιώντας συμβολισμούς, ως εξής:

```
If (Regs[R2] < Regs[R3])
   Regs[R1] ← 1
else
   Regs[R1] ← 0
```

Παράδειγμα

Ποια είναι η σημασία της παρακάτω λογικής εντολής του DLX:
SGTI R1,R2,#50 (Set Greater Than Immediate)

Απάντηση:

Παρατηρούμε ότι στον κωδικό της εντολής υπάρχει το γράμμα I που δηλώνει ότι ένας πηγαίος τελεστέος είναι απευθείας δεδομένο (στο παράδειγμά μας είναι η τιμή 50). Με την εντολή αυτή συγκρίνονται το περιεχόμενο του καταχωρητή R2 με την τιμή 50 και ο καταχωρητής R1 παίρνει την τιμή 1 ή την τιμή 0, ανάλογα με το αποτέλεσμα της σύγκρισης. Έτσι αν το περιεχόμενο του R2 είναι μεγαλύτερο από την τιμή 50 τότε ο καταχωρητής R1 τίθεται 1, διαφορετικά τίθεται 0. Μπορούμε να περιγράψουμε την εντολή αυτή χρησιμοποιώντας συμβολισμούς, ως εξής:

```
If (Regs[R2] > 50)
   Regs[R1] ← 1
else
   Regs[R1] ← 0
```

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 8

Να εξηγήσετε τον τρόπο με τον οποίο εκτελούνται οι παρακάτω εντολές του υπολογιστή DLX, χρησιμοποιώντας συμβολισμούς.

Εντολή	Όνομα εντολής
ADD R1, R2, R3	**Πρόσθεση**
ADDI R1, R2, #3	**Απευθείας πρόσθεση**
LHI R1, #42	**Απευθείας φόρτωση στα δύο πιο σημαντικά bytes**
SLLI R1, R2, #5	**Απευθείας αριστερή λογική ολίσθηση**
STL R1, R2, R3	**Θέσε μικρότερο από (set less than)**

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 8

Να εξηγήσετε τον τρόπο με τον οποίο εκτελούνται οι παρακάτω εντολές του υπολογιστή DLX, χρησιμοποιώντας συμβισμούς.

Εντολή	Επεξήγηση εντολής
ADD R1, R2, R3	**Regs[R1]← Regs[R2]+Regs[R3]**
ADDI R1, R2, #3	**Regs[R1] ← Regs[R2]+ 3**
LHI R1, #42	**Regs[R1] ← 42# # 0^{16}**
SLLI R1, R2, #5	**Regs[R1] ← Regs[R2] << 5**
STL R1, R2, R3	**if (Regs[R2]<Regs[R3]) Regs[R1]← 1 else Regs[R1] ← 0**

Εντολές ελέγχου (Διακλαδώσεις με συνθήκη και μεταπηδήσεις)

Τα είδη των μεταπηδήσεων είναι δύο:

- Απλή μεταπήδηση
- Μεταπήδηση με σύνδεσμο, όπου η διεύθυνση της επόμενης εντολής τοποθετείται στον R31.

Οι διακλαδώσεις είναι όλες υπό συνθήκη. Η συνθήκη διακλάδωσης ελέγχει αν το περιεχόμενο του πηγαίου καταχωρητή είναι ίσο ή διάφορο του μηδενός. Οι εντολές αυτές είναι: BNEZ (Branch if Not Equal Zero) και BEZ (Branch if Equal Zero).

Παράδειγμα

Ποια η λειτουργία της παρακάτω εντολής ελέγχου ροής προγράμματος του DLX:
JALR R2 (Jump And Link Register)

Απάντηση:

Η εντολή αυτή είναι μια εντολή μεταπήδησης με σύνδεσμο. Η διεύθυνση της επόμενης εντολής τοποθετείται στον καταχωρητή R31, ενώ η διεύθυνση του στόχου είναι το περιεχόμενο του καταχωρητή R2. Μπορούμε να περιγράψουμε την εντολή αυτή χρησιμοποιώντας συμβολισμούς, ως εξής:

Regs[R31] ← PC + 4;
PC ← Regs[R2]

Παράδειγμα

Ποια η λειτουργία της παρακάτω εντολής ελέγχου ροής του προγράμματος του DLX:

BEZ R2, name (Branch if Equal Zero)
(το name συμβολίζει μια διεύθυνση μνήμης **)**

Απάντηση:

Η εντολή αυτή είναι μια εντολή διακλάδωσης με συνθήκη. Αν το περιεχόμενο του καταχωρητή R2 είναι ίσο με το 0, τότε στον απαριθμητή προγράμματος (PC) φορτώνεται το name, με άλλα λόγια η επόμενη εντολή που θα εκτελεστεί είναι η εντολή που βρίσκεται στη διεύθυνση μνήμης name. Μπορούμε να περιγράψουμε την εντολή αυτή χρησιμοποιώντας συμβολισμούς, ως εξής:

```
If   (Regs[R2] = 0)
  then
    PC ← name;
```

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 9

Να εξηγήσετε τον τρόπο με τον οποίο εκτελούνται οι παρακάτω εντολές του υπολογιστή DLX, χρησιμοποιώντας συμβισμούς.

Εντολή	Όνομα εντολής
J name	Μεταπήδηση
JAL name	Μεταπήδηση και σύνδεση
JALR R2	Μεταπήδηση και σύνδεση καταχωρητή
JR R3	Μεταπήδηση καταχωρητή
BEQZ R4, name	Διακλάδωση ίση με μηδέν
BNEZ R4, name	Διακλάδωση άνιση με μηδέν

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 9

Εντολή	Όνομα εντολής
J name	PC← name; $((PC+4) - 2^{25}) \leq name < ((PC+4) + 2^{25})$
JAL name	R31← PC+4; PC← name; $((PC+4) - 2^{25}) \leq name < ((PC+4) + 2^{25})$
JALR R2	Regs[R31] ← PC+4; PC← Regs[R2]
JR R3	PC← Regs[R3]
BEQZ R4, name	if (Regs[R4]== 0) PC← name; $((PC+4) - 2^{15}) \leq name < ((PC+4) + 2^{15})$
BNEZ R4, name	if (Regs[R4]!= 0) PC← name; $((PC+4) - 2^{15}) \leq name < ((PC+4) + 2^{15})$

Εντολές κλασματικών αριθμών (κινητής υποδιαστολής)

Οι εντολές κινητής υποδιαστολής χρησιμοποιούν τους καταχωρητές κινητής υποδιαστολής και καθορίζουν το αν μία πράξη που θα πραγματοποιηθεί είναι απλής ή διπλής ακρίβειας. Οι εντολές αυτές είναι:

- MOVD, MOVF: μετακινούν το περιεχόμενο ενός καταχωρητή κινητής υποδιαστολής απλής ή διπλής ακρίβειας αντίστοιχα σε έναν καταχωρητή του ίδιου τύπου.
- MOVFP2I, MOVI2FP: μετακινούν δεδομένα ανάμεσα σε έναν καταχωρητή κινητής υποδιαστολής απλής ακρίβειας και σε έναν καταχωρητή ακεραίου.
- ADDD, ADDF, SUBD, SUBF, MULTD, MULTF, DIVD, DIVF: προσθέτουν, αφαιρούν, πολλαπλασιάζουν και διαιρούν αριθμούς κινητής υποδιαστολής απλής και διπλής ακρίβειας αντίστοιχα.

Παρακάτω, παρουσιάζεται ο συγκεντρωτικός πίνακας που περιλαμβάνει όλους του τύπους εντολών του DLX καθώς και τη σημασία της κάθε εντολής.

Τύπος εντολής/ κωδικός πράξης	**Σημασία εντολής**
Μεταφορές δεδομένων	**Μετακίνηση δεδομένων μεταξύ καταχωρητών και μνήμης, ή μεταξύ του ακεραίου και FP ή ειδικούς καταχωρητές. Μόνος τρόπος διευθυνσιοδότησης είναι με μετατόπιση (16-bit μετατόπιση + περιεχόμενα ενός καταχωρητή γενικής χρήσης).**
LB,LBU, SB	Φόρτωση byte, φόρτωση byte χωρίς πρόσημο, αποθήκευση byte.
LH,LHU,SH	Φόρτωση μισής λέξης, φόρτωση μισής λέξης χωρίς πρόσημο, αποθήκευση μισής λέξης.
LW, SW	Φόρτωση λέξης, αποθήκευση λέξης (σε/από καταχωρητή ακεραίων).
LF, LD, SF, SD	Φόρτωση καταχωρητή κινητής υποδιαστολής απλής ακρίβειας, φόρτωση καταχωρητή κινητής υποδιαστολής διπλής ακρίβειας, αποθήκευση καταχωρητή κινητής υποδιαστολής απλής ακρίβειας, αποθήκευση καταχωρητή κινητής υποδιαστολής διπλής ακρίβειας.
MOVI2S, MOVS2I	Μετακίνηση από /σε καταχωρητή γενικής χρήσης σε /από έναν ειδικό καταχωρητή
MOVF, MOVD	Αντιγραφή του περιεχομένου ενός καταχωρητή κινητής υποδιαστολής ή ενός ζεύγους καταχωρητών γενικής χρήσης σε άλλον καταχωρητή ή ζεύγος.
MOVFP2I, MOVI2FP	Μετακίνηση 32 bits από /σε καταχωρητές κινητής υποδιαστολής σε /από ακέραιους καταχωρητές.
Αριθμητικές/λογικές	**Πράξεις σε ακέραια ή λογικά δεδομένα που βρίσκονται σε καταχωρητές γενικής χρήσης. Προσημασμένη αριθμητική Παγίδα και υπερχείλιση.**

ADD, ADDI, ADDU, ADDUI	Πρόσθεση, απευθείας πρόσθεση (όλα τα απευθείας δεδομένα είναι 16 bits) με πρόσημο και χωρίς.
SUB, SUBI, SUBU, SUBUI	Αφαίρεση, απευθείας αφαίρεση προσημασμένη και μη.
MULT, MULTU, DIV,DIVU	Πολλαπλασιασμός και διαίρεση, προσημασμένα και μη. Οι τελεστέοι πρέπει να βρίσκονται σε καταχωρητές κινητής υποδιαστολής. Όλες οι πράξεις παίρνουν και δίνουν τιμές 32-bit.
AND, ANDI	And, απευθείας and.
OR, ORI, XOR, XORI	Or, απευθείας or, αποκλειστικό or, απευθείας αποκλειστικό or.
LHI	Φόρτωση στα δύο πιο σημαντικά bytes ενός καταχωρητή μια απευθείας τιμή των 16 bits.
SLL, SRL, SRA, SLLI, SRLI, SRAI	Ολισθήσεις: απευθείας (S_I) και μεταβλητή (S_) μορφή. Οι ολισθήσεις είναι αριστερές λογικές, δεξιές λογικές, δεξιές αριθμητικές.
S_, S_I	Η συνθήκη: "_" μπορεί να είναι LT,GT, LE, GE, EQ, NE
Έλεγχος	**Υπό συνθήκη διακλαδώσεις και μεταπηδήσεις. PC-σχετικές ή μέσω καταχωρητών**
BEQZ, BNEZ	Διακλάδωση αν το περιεχόμενο του καταχωρητή είναι ίσο/άνισο με το μηδέν. 16-bit offset από PC+4
J, JR	Μεταπηδήσεις: 26-bit offset από PC+4 (J) ή στόχος σε καταχωρητή (JR).
JAL, JALR	Μεταπήδηση και σύνδεση: σώσε PC+4 στον R31,στόχος είναι PC-σχετικός(JAL) ή ένας καταχωρητής (JALR).
TRAP	Μεταφορά στο λειτουργικό σύστημα σε ένα διάνυσμα διεύθυνσης
RFE	Επιστροφή στον χρησιμοποιούμενο κώδικα από μία εξαίρεση. Επαναφύλαξη του Τρόπου του χρήστη.

Κινητής υποδιαστολής	**FP πράξεις σε DP και SP τύπους**
ADDD, ADDF	Πρόσθεση αριθμών κινητής υποδιαστολής
SUBD, SUBF	Αφαίρεση αριθμών κινητής υποδιαστολής
MULTD, MULTF	Πολλαπλασιασμός αριθμών κινητής υποδιαστολής
DIVD, DIVF	Διαίρεση αριθμών κινητής υποδιαστολής
CVTF2D, CVTF2I CVTD2F, CTD2I CVTI2F, CVTI2D	Εντολές μετατροπής: CVT_X2y μετατρέπει από τον τύπο x στον τύπο y, όπου x και y είναι I (ακέραιος), D (διπλής ακρίβειας), ή F (απλής ακρίβειας). Και οι δύο Τελεστέοι βρίσκονται σε καταχωρητές κινητής υποδιαστολής

Παράδειγμα

Φανταστείτε το παρακάτω τμήμα κώδικα σε C:

```
for (i=0; i<=100; i++)
{A[i] = B[i] + C ;}
```

Υποθέστε ότι τα A και B είναι πίνακες από ακέραιους 32-bit. Υποθέστε ότι όλες οι τιμές δεδομένων και οι διευθύνσεις τους φυλάσσονται στη μνήμη (σε διευθύνσεις 0, 5000, 1500, και 2000 για τα A, B, C, και i αντίστοιχα) εκτός και αν συμμετέχουν σε πράξεις. Υποθέστε ότι καμία τιμή δεν φυλάσσεται σε καταχωρητές κατά τη διάρκεια των βρόχων.
Γράψτε τον κώδικα για τον DLX. Πόσες εντολές χρειάζονται δυναμικά ; Πόσες αναφορές δεδομένων στη μνήμη θα εκτελεστούν ; Ποιό είναι το μέγεθος κώδικα σε bytes;

Απάντηση:

Ο κώδικας για τον DLX είναι ο ακόλουθος:

```
 Program:
   SW 2000(R0),  R0   ;   αποθήκευση της τιμής  0
                      ;  στη διεύθυνση που βρίσκεται ο i
Loop:
  LW R1, 2000(R0)      ;  πάρε την τιμή του i
```

```
MULT R2, R1, #4       ;  R2=word offset του πίνακα B
ADDI  R3, R2, #5000   ;  πρόσθεσε την διεύθυνση βάσης στον R2
LW R4, 0(R3)          ;  πάρε την τιμή B[i]
LW R5, 1500(R0)       ;  πάρε την τιμή του C
ADD R6, R4, R5        ;  B[i] + C
LW R1, 2000(R0)       ; πάρε την τιμή του i
MULT R2, R1,  #4      ;  R2=word offset του A
ADDI R7, R2, #0       ; πρόσθεσε τη διεύθυνση βάσης στον R2
SW 0(R7), R6          ;  A[i] ← B[i] + C
LW R1, 2000(R0)       ;  πάρε την τιμή του i
ADDI R1, R1, #1       ; αύξησε τον i
SW 2000(R0), R1       ; αποθήκευσε τον i
LW R1, 2000(R0)       ; πάρε την τιμή του i
ADDI R8, R1, #-101    ; είναι ο μετρητής 101?
BNEZ R8, Loop         ; αν όχι, επανέλαβε
```

- Οι εντολές που απαιτούνται δυναμικά είναι:
  1 + (16 x 101) = 1617
  Οι εντολές στο Loop, που στο σύνολό τους είναι 16 επαναλαμβάνονται 101 φορές.
- Οι εντολές με αναφορές στη μνήμη που θα εκτελεστούν είναι:
  1 + (8 x 101) = 809
- Το μέγεθος του κώδικα σε bytes είναι:
  4 x 17 = 68 αφού κάθε εντολή είναι μεγέθους 4 bytes.

Θα πιστεύετε ίσως, ότι μία αρχιτεκτονική σαν αυτή του DLX με απλούς τύπους εντολών, απλούς τρόπους διευθυνσιοδότησης, και απλές πράξεις, θα ήταν αργή κυρίως επειδή θα έπρεπε να εκτελέσει περισσότερες εντολές (αντί για λιγότερες και πιο πολύπλοκες). Για να δοθεί απάντηση στο ερώτημα αυτό, χρειάστηκε να συγκρίνουμε έναν υπολογιστή που μοιάζει με τον DLX (επιλέχτηκε ο MIPS M2000) με κάποιον άλλο με πολυπλοκότερη αρχιτεκτονική συνόλου εντολών (επιλέχτηκε ο VAX 8700). Όπως αποδείχθηκε, ο MIPS εκτελούσε σχεδόν τις διπλάσιες εντολές σε σχέση με τον VAX, ενώ ο CPI για τον VAX ήταν περίπου έξι φορές μεγαλύτερος από ότι αυτός για τον MIPS. Το τελικό συμπέρασμα ήταν ότι ο MIPS M2000 είχε περίπου τρεις φορές μεγαλύτερη απόδοση από αυτήν του VAX 8700. Αυτό συμβαίνει γιατί οι υπολογιστές όπως ο DLX επιτρέπουν σε διαφορετικά τμήματα διαφορετικών εντολών να εκτελούνται ταυτόχρονα (σωλήνωση).

Ανακεφαλαιώνοντας λοιπόν …

Ο DLX υποστηρίζει όλους τους συνήθεις τύπους δεδομένων, ενώ έχει στη διάθεσή του τόσο καταχωρητές γενικής χρήσης, όσο και καταχωρητές για πράξεις με αριθμούς κινητής υποδιαστολής.

Οι τρόποι διευθυνσιοδότησης των δεδομένων στον DLX είναι: απευθείας και με μετατόπιση, ενώ η μνήμη είναι διευθυνσιοδοτημένη κατά byte με οργάνωση Big Endian.

Οι εντολές στον DLX κωδικοποιούνται σε έναν από τους παρακάτω τύπους: -I, -R, -J, και είναι αρκετοί για την εκτέλεση όλων των απαιτούμενων λειτουργιών του υπολογιστή.

Οι λειτουργίες που υποστηρίζονται από τον DLX είναι: λειτουργίες μεταφοράς δεδομένων, αριθμητικές και λογικές λειτουργίες, λειτουργίες ελέγχου ροής προγράμματος (με εντολές μεταπήδησης και διακλάδωσης), καθώς και όλες οι αντίστοιχες λειτουργίες για αριθμούς κινητής υποδιαστολής.