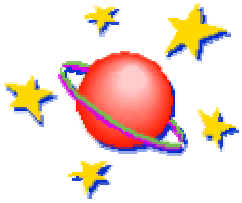


Ενότητα 2η : *Σχεδίαση και υλοποίηση της διόδου δεδομένων και της μονάδας ελέγχου του υπολογιστή MIPS*

Σκοπός **Ο σκοπός της ενότητας αυτής είναι να παρουσιάσει να τον τρόπο με τον οποίο σχεδιάζεται και υλοποιείται η δίοδος δεδομένων και η μονάδα ελέγχου του υπολογιστή MIPS.**

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

εξηγείτε πως λειτουργεί το ρολόι για την ανάγνωση ή την εγγραφή ενός σήματος,

χρησιμοποιείτε τα κατάλληλα κυκλώματα για την κατασκευή των τμημάτων της διόδου δεδομένων, για κάθε τύπο εντολής,

σχεδιάζετε τη δίοδο δεδομένων και τη βασική μονάδα ελέγχου ενός κύκλου του υπολογιστή MIPS,

σχεδιάζετε τη μονάδα ελέγχου όταν η εκτέλεση των εντολών χωρίζεται σε πολλούς κύκλους ρολογιού,

συγκρίνετε την υλοποίηση της διόδου δεδομένων του ενός κύκλου, με αυτή των πολλών κύκλων.

ακμοπυροδοτούμενη λειτουργία ρολογιού, μονάδα μνήμης δεδομένων, μονάδα επέκτασης προσήμου

Ο ρόλος του ρολογιού στη μετάδοση σημάτων.

Στην πρώτη υποενότητα θα μελετήσουμε τον τρόπο με τον οποίο μεταδίδονται τα σήματα στα στοιχεία μνήμης, σε έναν ή σε δύο κύκλους ρολογιού. Στη συνέχεια θα περιγράψουμε το κύκλωμα ανάγνωσης ή εγγραφής ενός σήματος σε έναν κύκλο ρολογιού.

Μετάδοση σημάτων σε στοιχεία μνήμης

Η λειτουργία του ρολογιού καθορίζει το πότε θα γίνει ανάγνωση ή εγγραφή ενός σήματος. Είναι σημαντικό να διαχωριστεί ο χρόνος ανάγνωσης από το χρόνο εγγραφής ενός σήματος. Στην περίπτωση που σε ένα σήμα η εγγραφή και η ανάγνωση συμβαίνουν ταυτόχρονα, τότε η τιμή του σήματος στην ανάγνωση μπορεί να είναι είτε η παλιά τιμή του σήματος, είτε η νέα τιμή της εγγραφής, ή ακόμα και μια τιμή ανάμεσα σε αυτές τις δύο. Η ακμοπυροδοτούμενη λειτουργία του ρολογιού σχεδιάστηκε για να αποφευχθούν παρόμοιες καταστάσεις.

Με την ακμοπυροδοτούμενη λειτουργία ρολογιού διαχωρίζεται ο χρόνος εγγραφής και ανάγνωσης ενός σήματος. Στην λειτουργία αυτή, τα κυκλώματα του υπολογιστή ενημερώνονται μόνο στην ακμή του ωρολογιακού παλμού. Επειδή μόνο τα στοιχεία μνήμης μπορούν να αποθηκεύσουν τις τιμές των δεδομένων, σε κάθε συνδυαστικό κύκλωμα πρέπει οι είσοδοι να προέρχονται από τα στοιχεία μνήμης και οι έξοδοι να γράφονται σε ένα σύνολο από στοιχεία μνήμης. Οι είσοδοι είναι τιμές που προέρχονται από προηγούμενο κύκλο ρολογιού, ενώ οι έξοδοι είναι οι τιμές που θα χρησιμοποιηθούν στον επόμενο κύκλο ρολογιού. Στα σχήματα 3.2.1 και 3.3.2 παρουσιάζονται δύο παραδείγματα.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 1

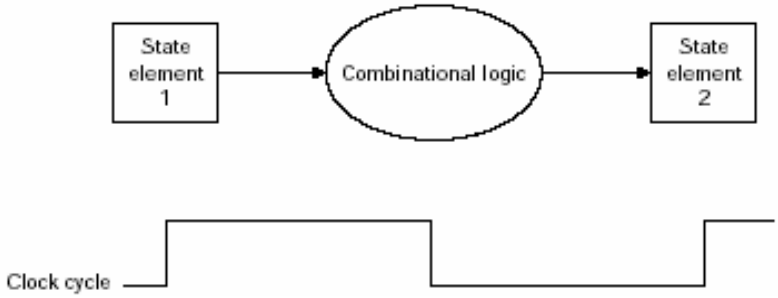
Θυμάστε τι είναι τα στοιχεία μνήμης; Για περισσότερες λεπτομέρειες θα ήταν καλό να ανατρέξετε στο βιβλίο του M. Morris Mano: "Ψηφιακή σχεδίαση" και συγκεκριμένα στο κεφάλαιο «Σύγχρονα Ακολουθιακά Κυκλώματα».

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 1

Τα στοιχεία μνήμης είναι συσκευές που μπορούν να αποθηκεύσουν δυαδικές πληροφορίες. Στην κάθε χρονική στιγμή, οι δυαδικές πληροφορίες που είναι αποθηκευμένες στα στοιχεία μνήμης ενός ακολουθιακού κυκλώματος, αποτελούν την κατάστασή του ("state"). Το απλούστερο στοιχείο μνήμης είναι το flip-flop.

Παράδειγμα

Στο σχήμα 3.2.1, το συνδυαστικό κύκλωμα λειτουργεί σε έναν κύκλο ρολογιού. Σε αυτή την περίπτωση, όλα τα σήματα πρέπει να μεταδίδονται από το στοιχείο μνήμης 1, μέσω του συνδυαστικού κυκλώματος, στο στοιχείο μνήμης 2 σε χρόνο ενός κύκλου ρολογιού και η εγγραφή στο στοιχείο μνήμης 2 γίνεται στο τέλος κάθε κύκλου ρολογιού. Ο χρόνος που απαιτείται για να φτάσουν τα σήματα στο στοιχείο μνήμης 2 καθορίζει το μήκος του κύκλου του ρολογιού.

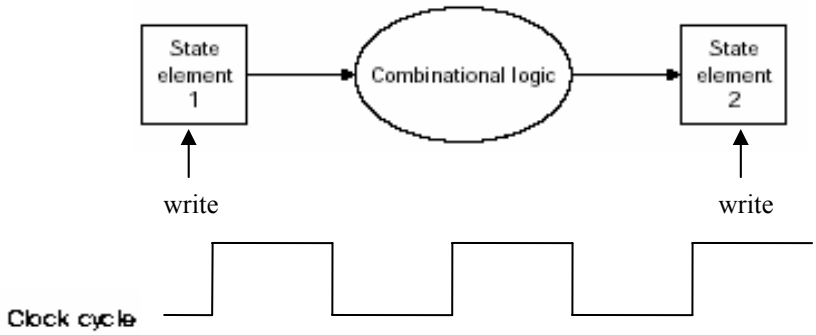


Σχήμα 3.2.1 - Το συνδυαστικό κύκλωμα, τα στοιχεία μνήμης και το ρολόι είναι στενά συνδεδεμένα: Σε ένα σύγχρονο ψηφιακό σύστημα, το ρολόι προσδιορίζει το πότε θα γίνει εγγραφή στα στοιχεία μνήμης. Η κάθε είσοδος πρέπει να φτάσει μια σταθερή τιμή (η τιμή δε θα αλλάξει πριν τον επόμενο ωρολογιακό παλμό).

Παράδειγμα

Στο δεύτερο παράδειγμα απαιτούνται δύο κύκλοι ρολογιού για να μεταδοθούν τα σήματα από το στοιχείο μνήμης 1, μέσω του συνδυαστικού κυκλώματος, στο στοιχείο μνήμης 2. Σ΄ αυτή την περίπτωση απαιτείται ένα σήμα ελέγχου εγγραφής του δεύτερου στοιχείου μνήμης, κατάλληλο, ώστε η εγγραφή σ΄ αυτό το κύκλωμα να γίνεται κατά την ακμή του δεύτερου ωρολογιακού παλμού. Έτσι τα ψηφιακά κυκλώματα ενημερώνονται στην ακμή του ωρολογιακού παλμού, μόνο αν το σήμα εγγραφής είναι ενεργό, όπως φαίνεται στο σχήμα 3.2.2. Επίσης η έξοδος του

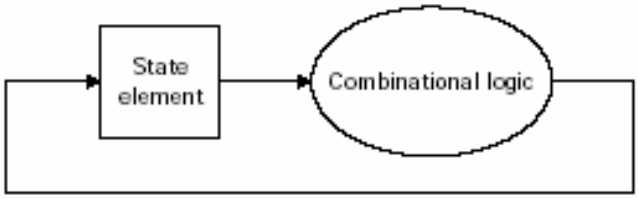
στοιχείου μνήμης 1 πρέπει να παραμένει σταθερή κατά τη διάρκεια των κύκλων του ρολογιού, που απαιτούνται για τη μετάδοση του σήματος. Για να απλοποιήσουμε το σχήμα δε δείχνουμε το σήμα εγγραφής, όταν η κατάσταση ενημερώνεται σε κάθε ακμή του ωρολογιακού παλμού. Πρέπει να τονιστεί πως όλα τα στοιχεία μνήμης έχουν σαν είσοδο το ρολόι.



Σχήμα 3.2.2 – Στο στοιχείο μνήμης 2 δε γίνεται εγγραφή σε κάθε κύκλο ρολογιού, αλλά μόνο στην ακμή του ωρολογιακού παλμού, όταν το σήμα εγγραφής είναι ενεργό: Αυτή η σχεδίαση επιτρέπει στο συνδυαστικό κύκλωμα να μεταδίδει τις εισόδους του στοιχείου μνήμης 1 σε περισσότερους κύκλους ρολογιού. Η έξοδος του κυκλώματος 1 δεν πρέπει να αλλάζει κατά τη διάρκεια που το σήμα μεταδίδεται μέσω του συνδυαστικού κυκλώματος. Επομένως το στοιχείο μνήμης 1 πρέπει να έχει ένα κατάλληλο σήμα εγγραφής. Αν για παράδειγμα απαιτούνται 2 κύκλοι ρολογιού για να μεταδοθεί το σήμα μέσω του συνδυαστικού κυκλώματος, τότε το σήμα εγγραφής για το στοιχείο μνήμης 1 πρέπει να απενεργοποιηθεί κατά τον πρώτο ωρολογιακό παλμό.

Ανάγνωση ή εγγραφή σήματος

Η ακμοπυροδοτούμενη λειτουργία μας επιτρέπει να διαβάζουμε τα περιεχόμενα ενός καταχωρητή, να στέλνουμε τις τιμές των σημάτων μέσω ενός συνδυαστικού κυκλώματος και να γράφουμε στον καταχωρητή στον ίδιο κύκλο ρολογιού, όπως φαίνεται στο σχήμα 3.2.3. Δεν έχει σημασία εάν θεωρήσουμε πως οι εγγραφές γίνονται σε θετική ή αρνητική ακμή των παλμών του ρολογιού, αφού οι είσοδοι στο συνδυαστικό κύκλωμα δεν αλλάζουν εκτός από την επιλεγόμενη ακμή.



Σχήμα 3.2.3 - Η ακμοπυροδοτούμενη λειτουργία επιτρέπει να γίνεται ανάγνωση και εγγραφή σε ένα μόνο κύκλο ρολογιού στο στοιχείο μνήμης, χωρίς να οδηγούμαστε σε ακαθόριστες τιμές των δεδομένων. Ο κύκλος του ρολογιού πρέπει να έχει κατάλληλο μήκος έτσι ώστε οι τιμές στις εισόδους να παραμένουν σταθερές όταν εμφανίζεται η ακμή του επόμενου ωρολογιακού παλμού.

 Σχεδόν όλα τα στοιχεία μνήμης και τα συνδυαστικά κυκλώματα έχουν εισόδους και εξόδους με μέγεθος 32 bits, αφού αυτό είναι το μέγεθος των δεδομένων που επεξεργάζεται ο επεξεργαστής. Τα σχήματα περιέχουν αρτηρίες (buses), οι οποίες μεταφέρουν σήματα με μέγεθος μεγαλύτερο από 1 bit. Τα βέλη δείχνουν την κατεύθυνση της ροής των δεδομένων ανάμεσα στα κυκλώματα.

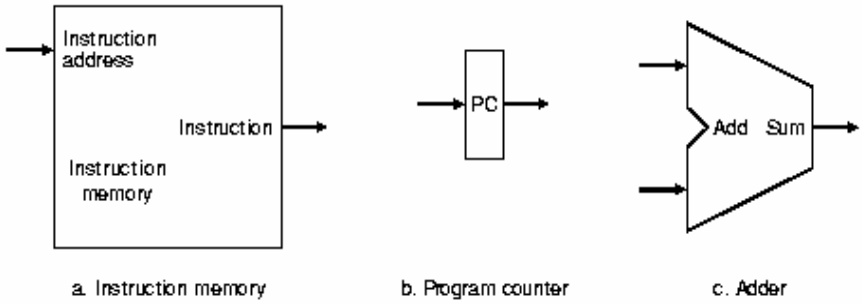
Τα τμήματα της διόδου δεδομένων του υπολογιστή MIPS για κάθε τύπο εντολής

Στην υποενότητα αυτή θα ασχοληθούμε με το σχεδιασμό της διόδου δεδομένων του υπολογιστή MIPS και με τα βασικά κυκλώματα που χρειάζονται για το σχεδιασμό αυτό. Αρχικά, θα περιγράψουμε τα κυκλώματα της διόδου δεδομένων για την ανάκληση των εντολών και την αύξηση του απαριθμητή προγράμματος. Στη συνέχεια θα εξετάσουμε για κάθε τύπο εντολής ξεχωριστά, τον τρόπο με τον οποίο εκτελούνται οι εντολές, καθώς και τα επιπλέον κυκλώματα που χρειάζονται για την υλοποίηση του τμήματος της διόδου δεδομένων της κάθε εντολής.

Δίοδος δεδομένων για ανάκληση εντολών και αύξηση του απαριθμητή προγράμματος

Για το σχεδιασμό της διόδου δεδομένων, αρχικά θα πρέπει να εξετάσουμε τα βασικά κυκλώματα που χρειάζονται για κάθε τύπο εντολής. Δηλαδή, να κατασκευάσουμε για κάθε τύπο εντολής ξεχωριστά τη δίοδο δεδομένων με τα αντίστοιχα σήματα ελέγχου.

Αρχικά, χρειαζόμαστε ένα κύκλωμα για να αποθηκεύονται οι εντολές του προγράμματος. Η μονάδα μνήμης, η οποία αποτελείται από στοιχεία μνήμης, χρησιμοποιείται για την αποθήκευση και την ανάκληση εντολών όταν δίνεται η διεύθυνσή τους, όπως φαίνεται στο σχήμα 3.2.4. Η διεύθυνση της εντολής φυλάσσεται σε έναν καταχωρητή, που ονομάζουμε απαριθμητή προγράμματος (σχήμα 3.2.4). Επίσης, χρειαζόμαστε έναν αθροιστή για να αυξάνει την τιμή του απαριθμητή προγράμματος ώστε να δείχνει τη διεύθυνση της επόμενης εντολής. Αυτός ο αθροιστής, ο οποίος είναι ένα συνδυαστικό κύκλωμα, μπορεί να δημιουργηθεί σε μια ALU προσδιορίζοντας τις γραμμές ελέγχου, έτσι ώστε να μας δίνει πάντα τη λειτουργία άθροισης. Επομένως, για το σχεδιασμό της διόδου δεδομένων που αφορά στην ανάκληση των εντολών, χρειαζόμαστε μια μονάδα μνήμης, έναν απαριθμητή προγράμματος και έναν αθροιστή.

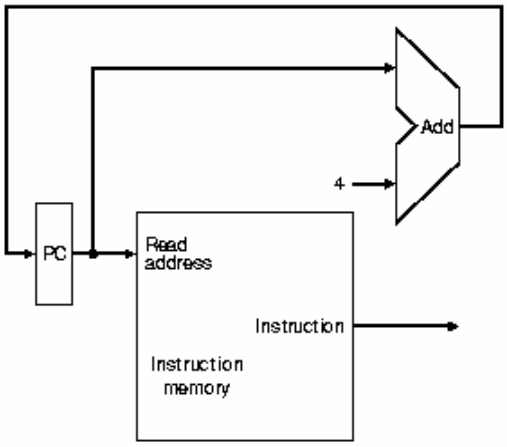


Σχήμα 3.2.4 – Τα κυκλώματα που χρειάζονται για την αποθήκευση και την ανάκληση των εντολών και ο αθροιστής για τον υπολογισμό της διεύθυνσης της επόμενης εντολής: Η μνήμη εντολών και ο απαριθμητής προγράμματος είναι στοιχεία μνήμης. Η μνήμη εντολών χρησιμοποιείται μόνο για ανάγνωση, επειδή η δίοδος δεδομένων δεν κάνει εγγραφή των εντολών, (η εγγραφή των εντολών στη μνήμη γίνεται όταν φορτώνουμε το πρόγραμμα, έτσι την παραλείπουμε για απλοποίηση του σχήματος). Επομένως, αφού στη μνήμη εντολών μπορεί να γίνει μόνο ανάγνωση, δεν θα συμπεριλάβουμε το σήμα ελέγχου εγγραφής. Ο απαριθμητής προγράμματος είναι ένας καταχωρήτης 32-bits, στον οποίο γίνεται εγγραφή κάτω από τον έλεγχο ενός σήματος εγγραφής. Ο αθροιστής είναι ένα κύκλωμα ALU καλωδιωμένο κατάλληλα, έτσι ώστε να κάνει πάντα άθροιση στις δύο εισόδους του (επίσης των 32-bits), και να δίνει το αποτέλεσμα στην έξοδό του.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2

Να περιγράψτε τη λειτουργία των κυκλωμάτων της μονάδας μνήμης, της ALU και του απαριθμητή προγράμματος. Μπορείτε να τα συνδέσετε με τέτοιο τρόπο, ώστε να σχηματιστεί το τμήμα της διόδου δεδομένων για την ανάκληση των εντολών και την αύξηση του απαριθμητή προγράμματος; Να συγκρίνετε το σχήμα της διόδου δεδομένων που δημιουργήσατε, με αυτό του σχήματος που ακολουθεί (σχήμα 3.2.5).

 Για την εκτέλεση μιας εντολής πρέπει να ανακαλέσουμε την εντολή από τη μνήμη. Για την εκτέλεση της επόμενης εντολής πρέπει να αυξήσουμε τον απαριθμητή προγράμματος κατά 4, έτσι ώστε να μας δίνει τη διεύθυνση της επόμενης εντολής. Η δίοδος δεδομένων για αυτό το βήμα φαίνεται στο σχήμα 3.2.5.



Σχήμα 3.2.5 – Το τμήμα της διόδου δεδομένων που χρησιμοποιείται για την ανάκληση των εντολών και την αύξηση του απαριθμητή προγράμματος.

Δίοδος δεδομένων για τις εντολές τύπου R

Οι εντολές τύπου R είναι εντολές που εκτελούν αριθμητικές και λογικές πράξεις. Οι πηγαίοι τελεστέοι, καθώς και οι τελεστέοι προορισμού, βρίσκονται μέσα σε καταχωρητές. Οι εντολές τύπου R διαβάζουν τους δύο καταχωρητές, εκτελούν μια λειτουργία ALU στα περιεχόμενα των καταχωρητών, και γράφουν το αποτέλεσμα σε έναν καταχωρητή.

- Οι εντολές τύπου R είναι οι: add, sub, and, or, slt.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 3

α)Θυμάστε τον τρόπο με τον οποίο συντάσσονται και εκτελούνται οι εντολές στον υπολογιστή DLX; Να γράψετε την εντολή για την πρόσθεση των καταχωρητών και να περιγράψετε τη λειτουργία της. Για περισσότερες λεπτομέρειες θα ήταν καλό να ανατρέξετε στις σημειώσεις της "Αρχιτεκτονικής Υπολογιστών" και συγκεκριμένα στο δεύτερο κεφάλαιο: "Αρχιτεκτονικές Συνόλου Εντολών", 4η ενότητα : Η αρχιτεκτονική του DLX.

β)Να χρησιμοποιήσετε το λογισμικό προσομοίωσης της εκτέλεσης των εντολών τύπου R του DLX, το οποίο παρατίθεται στην ιστοσελίδα:

http://hermes.di.uoa.gr/DLXSim/r_type/R.html

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 3

Η εντολή πρόσθεσης στον υπολογιστή DLX είναι: ADD R1, R2, R3. Σε αυτή την περίπτωση οι R2, R3 είναι πηγαίοι καταχωρητές και ο R1 είναι ο καταχωρητής προορισμού. Κατά την εκτέλεση της εντολής, γίνεται πρόσθεση στο περιεχόμενο των πηγαίων καταχωρητών (R2 και R3) και το αποτέλεσμα αποθηκεύεται στον καταχωρητή προορισμού (R1). Επίσης, υπάρχουν εντολές πρόσθεσης για κλασματικούς αριθμούς, οι οποίες είναι οι εξής: ADD F και ADD D.
(Με ανάλογο τρόπο συντάσσονται και εκτελούνται οι εντολές πρόσθεσης στον υπολογιστή MIPS.)

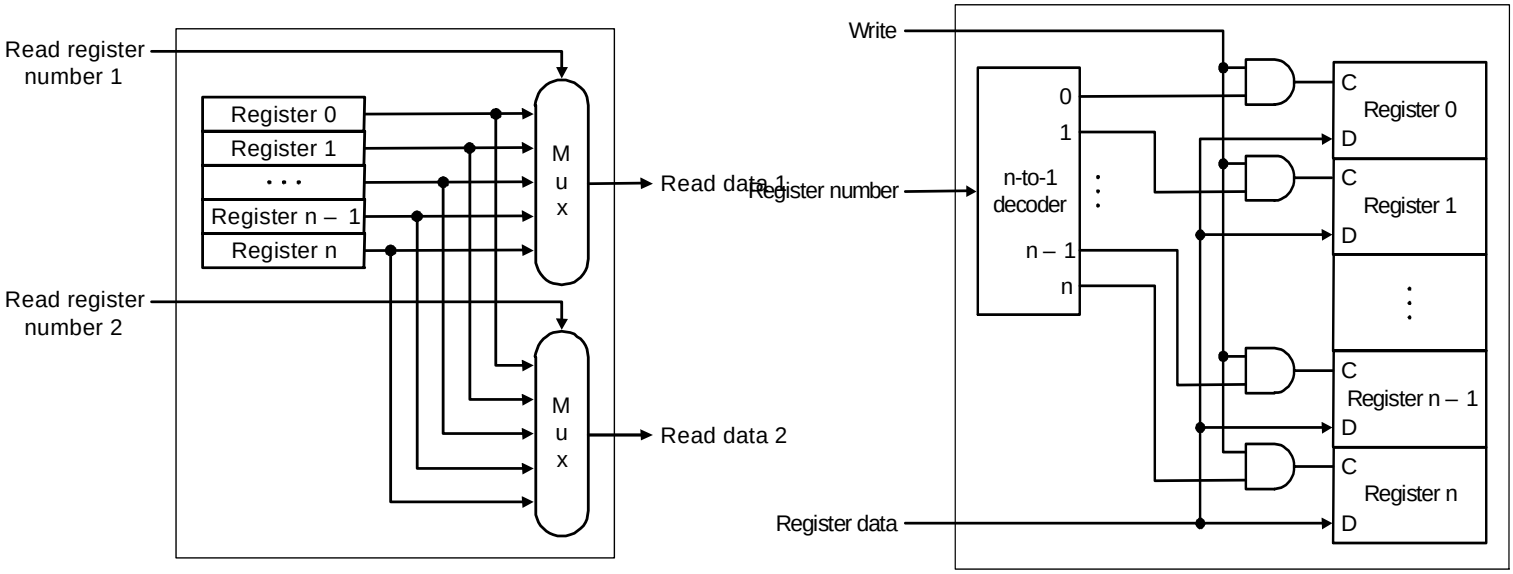
Παράδειγμα

Να εξηγήσετε το αποτέλεσμα της εντολής : add $1, $2, $3 στον υπολογιστή MIPS.

Απάντηση:

Η εντολή αυτή διαβάζει τα περιεχόμενα των καταχωρητών 2 και 3, εκτελεί την πρόσθεση και γράφει το αποτέλεσμα στον καταχωρητή 1.

Οι 32 καταχωρητές του επεξεργαστή βρίσκονται στο αρχείο καταχωρητών. Το αρχείο καταχωρητών είναι ένα σύνολο από καταχωρητές, στο οποίο οποιοσδήποτε καταχωρητής μπορεί να διαβαστεί ή να γίνει εγγραφή σε αυτόν κατά τη διάρκεια εκτέλεσης της εντολής. Για να γίνει αυτό πρέπει να καθοριστεί ο αριθμός του καταχωρητή μέσα στο αρχείο. Σε κάποια χρονική στιγμή οι καταχωρητές έχουν μια συγκεκριμένη τιμή. Η τιμή αυτή ονομάζεται κατάσταση των καταχωρητών. Το αρχείο καταχωρητών περιέχει την κατάσταση καταχωρητών του υπολογιστή. Επιπλέον θα χρειαστούμε μία ALU για να εκτελέσει τις λειτουργίες για τις τιμές που θα έχουν διαβαστεί από τους καταχωρητές. Στο παρακάτω σχήμα φαίνεται το αρχείο καταχωρητών με τα κατάλληλα κυκλώματα εγγραφής και ανάγνωσης.



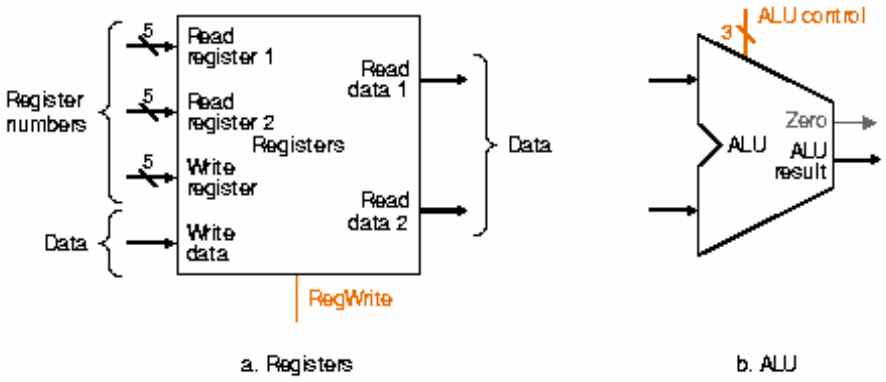
Σχήμα 3.2.6 – Οι καταχωρητές που περιέχονται στο αρχείο καταχωρητών αποτελούνται από D flip-flops: Στο σχήμα a φαίνονται τα κυκλώματα που επιτρέπουν ανάγνωση, και στο σχήμα b αυτά που επιτρέπουν εγγραφή. Τα κυκλώματα a και b περιέχονται στο αρχείο καταχωρητών του σχήματος 3.2.7.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 4

Να εξηγήσετε σε 10 γραμμές τη λειτουργία του αρχείου καταχωρητών. Από ποια κυκλώματα αποτελείται το αρχείο καταχωρητών και πώς αυτά συνδέονται μεταξύ τους; Να συγκρίνετε την απάντησή σας με την παράγραφο που μόλις διαβάσατε.

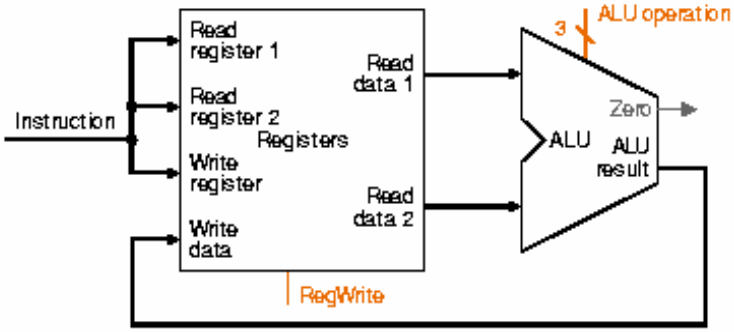
Επειδή οι εντολές τύπου R έχουν 3 τελεστέους, θα χρειαστεί να διαβάσουμε 2 τελεστέους και να γράψουμε 1 τελεστέο στο αρχείο καταχωρητών, για κάθε εντολή. Για κάθε τελεστέο που διαβάζεται από το αρχείο καταχωρητών, χρειαζόμαστε μια είσοδο στο αρχείο καταχωρητών που καθορίζει τον αριθμό του καταχωρητή στον οποίο πρόκειται να γίνει ανάγνωση, και μία έξοδο από το αρχείο καταχωρητών που θα μεταφέρει την τιμή που διαβάστηκε. Για την εγγραφή ενός τελεστέου θα χρειαστούν 2 είσοδοι: μία για να καθορίσει τον αριθμό του καταχωρητή που θα γίνει η εγγραφή, και μία για να δώσει τα δεδομένα προς εγγραφή στον καταχωρητή. Επομένως, συνολικά χρειαζόμαστε τέσσερις εισόδους (τρεις για τους αριθμούς των καταχωρητών και έναν για τα δεδομένα), και δύο εξόδους (και οι δύο για τα δεδομένα), όπως φαίνεται στο σχήμα 3.2.7. Στο αρχείο καταχωρητών στην έξοδο Read data1 εμφανίζεται το περιεχόμενο οποιουδήποτε καταχωρητή είναι είσοδος στο Read register 1. Αντίστοιχα και για την έξοδο Read data 2 και την είσοδο Read register 2. Οι εγγραφές ωστόσο, ελέγχονται από το σήμα ελέγχου εγγραφής. Οι

είσοδοι Register numbers που προσδιορίζουν τον αριθμό του καταχωρητή, έχουν μέγεθος 5 bits, δεδομένου ότι υπάρχουν 32 καταχωρητές ($32=2^5$), και η είσοδος των δεδομένων και οι δύο έξοδοι των δεδομένων έχει η κάθε μία μέγεθος 32 bits.
Για την εκτέλεση των λειτουργιών τύπου R χρειάζεται η ALU. Η ALU έχει δύο εισόδους των 32 bits και το αποτέλεσμα που παράγει είναι επίσης 32 bits. Στην ALU, η οποία φαίνεται στο σχήμα 3.2.7, γίνεται έλεγχος από ένα σήμα 3 bits (με 8 διαφορετικές τιμές, $2^3=8$), ώστε η ALU να μπορεί να εκτελέσει το πολύ 8 διαφορετικές λειτουργίες.



Σχήμα 3.2.7 - Τα κυκλώματα που χρειάζονται για την εκτέλεση των λειτουργιών της ALU στις εντολές τύπου R, είναι το αρχείο καταχωρητών και η ALU: Το αρχείο καταχωρητών περιέχει όλους τους καταχωρητές και παρέχει δύο θύρες για ανάγνωση και μία θύρα για εγγραφή. Το αρχείο καταχωρητών δίνει στην έξοδο τα περιεχόμενα των καταχωρητών που αντιστοιχούν στις εισόδους Read register. Οι εγγραφές ελέγχονται από το σήμα ελέγχου εγγραφής. Οι είσοδοι που μεταφέρουν τον αριθμό του καταχωρητή στο αρχείο καταχωρητών έχουν μέγεθος 5 bits, ενώ οι γραμμές που μεταφέρουν τις τιμές των δεδομένων έχουν μέγεθος 32 bits. Το σήμα ελέγχου της ALU είναι 3 bits. Η έξοδος Zero της ALU χρησιμοποιείται για τις εντολές διακλάδωσης.

Η δίοδος για τις εντολές τύπου R, που χρησιμοποιούν το αρχείο καταχωρητών και την ALU (του σχήματος 3.2.7), φαίνεται στο σχήμα 3.2.8. Αφού οι αριθμοί των καταχωρητών προέρχονται από πεδία της εντολής, στο σχήμα απεικονίζεται η εντολή που προέρχεται από το σχήμα 3.2.5



Σχήμα 3.2.8 - Η δίοδος δεδομένων για τις εντολές τύπου R: Η ALU παρέχει όλες τις βασικές λειτουργίες που απαιτούνται για τις εντολές τύπου R (όπως προσδιορίζεται από την είσοδο ελέγχου των 3 bits). Το αρχείο καταχωρητών στέλνει (από τις εξόδους Read data 1 και Read data 2) στην ALU τα δεδομένα που διάβασε από τους καταχωρητές. Η ALU εκτελεί την λειτουργία που προσδιορίζεται από την εντολή τύπου R και στέλνει το αποτέλεσμα στο αρχείο καταχωρητών (στην είσοδο Write data). Το αποτέλεσμα αυτό γράφεται στον καταχωρητή προορισμού.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 5

Να υπολογίσετε πόσες είσοδοι χρειάζονται, ώστε να γίνει εγγραφή ή ανάγνωση στο αρχείο καταχωρητών όταν εκτελείται η εντολή add $1, $2, $3. Να αιτιολογήστε την απάντησή σας.

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 5

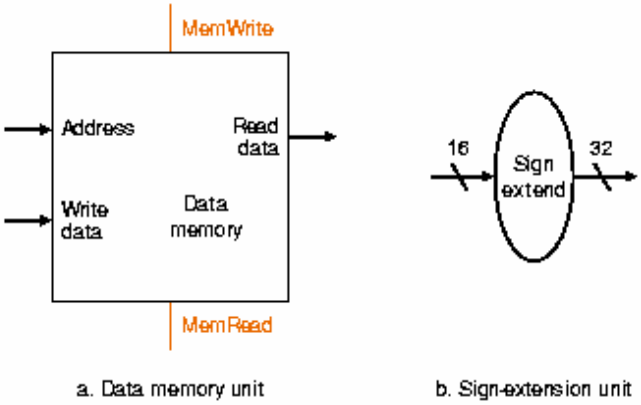
Η εντολή add $1, $2, $3 διαβάζει τα περιεχόμενα των καταχωρητών 2 και 3, εκτελεί την πρόσθεση και γράφει το αποτέλεσμα στον καταχωρητή 1. Επειδή οι εντολές τύπου R έχουν 3 τελεστέους, θα χρειαστεί να διαβάσουμε 2 τελεστέους και να γράψουμε 1 τελεστέο στο αρχείο καταχωρητών, για κάθε εντολή. Για κάθε τελεστέο που διαβάζεται από το αρχείο καταχωρητών, χρειαζόμαστε μια είσοδο στο αρχείο καταχωρητών που καθορίζει τον αριθμό του καταχωρητή στον οποίο πρόκειται να γίνει ανάγνωση, και μία έξοδο από το αρχείο καταχωρητών που θα μεταφέρει την τιμή που διαβάστηκε. Για την εγγραφή ενός τελεστέου θα χρειαστούν 2 είσοδοι: μία για να καθορίσει τον αριθμό του καταχωρητή που θα γίνει η εγγραφή, και μία για να δώσει τα δεδομένα προς εγγραφή στον καταχωρητή. Επομένως, συνολικά χρειαζόμαστε τέσσερις εισόδους (τρεις για τους αριθμούς των καταχωρητών και μία για τα δεδομένα), και δύο εξόδους (και οι δύο για τα δεδομένα).

Δίοδος δεδομένων για τις εντολές φόρτωσης και αποθήκευσης

Ας θεωρήσουμε τις εντολές φόρτωσης και αποθήκευσης (load and store instructions), που έχουν τη μορφή:

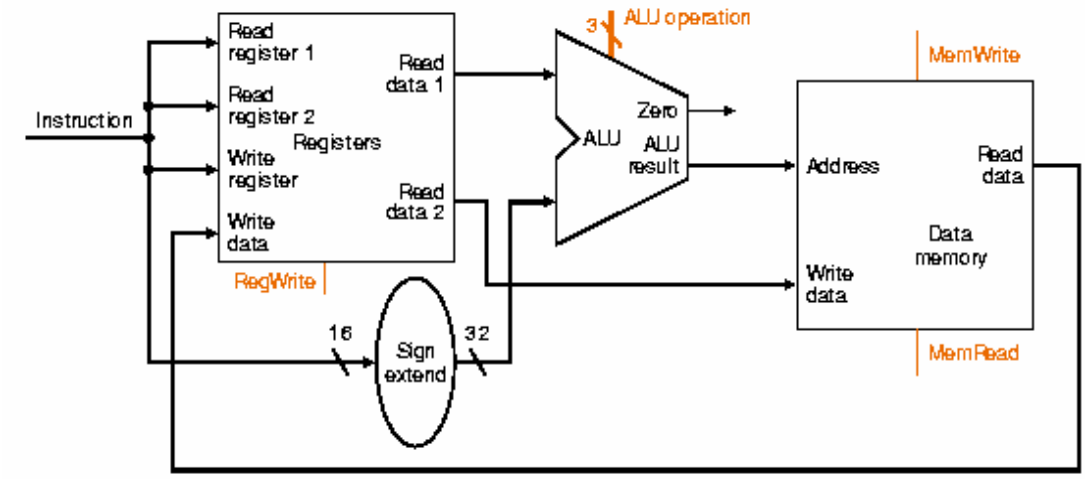
- lw $1, offset_value($2) ή sw $1, offset_value($2).

Οι εντολές αυτές υπολογίζουν τη διεύθυνση της μνήμης δεδομένων προσθέτοντας το περιεχόμενο του βασικού καταχωρητή ($2) στο πεδίο offset που βρίσκεται μέσα στην εντολή και έχει μέγεθος 16 bits. Στις εντολές αποθήκευσης, η τιμή που πρέπει να αποθηκευτεί διαβάζεται από το αρχείο καταχωρητών ($1). Στις εντολές φόρτωσης, η τιμή που διαβάζεται από τη μνήμη πρέπει να γραφεί στο αρχείο καταχωρητών και συγκεκριμένα στον καταχωρητή ($1). Επομένως χρειαζόμαστε το αρχείο καταχωρητών και την ALU όπως και στις εντολές τύπου R, όπως φαίνεται στο σχήμα 3.2.7. Επιπλέον χρειαζόμαστε μία μονάδα για την επέκταση προσήμου του πεδίου offset της εντολής από 16 bits σε 32 bits, και μία μονάδα μνήμης δεδομένων για να γίνει ανάγνωση ή εγγραφή σε αυτή. Η μνήμη δεδομένων έχει σήματα ελέγχου εγγραφής και ανάγνωσης, όπως επίσης έχει είσοδο για εγγραφή των δεδομένων στη μνήμη. Αυτά τα δύο κυκλώματα φαίνονται στο σχήμα 3.2.9.



Σχήμα 3.2.9 - Οι δύο μονάδες που χρειάζονται για την εκτέλεση των εντολών φόρτωσης και αποθήκευσης είναι η μονάδα μνήμης δεδομένων και η μονάδα επέκτασης προσήμου: Εκτός από αυτές τις μονάδες χρειάζεται το αρχείο καταχωρητών και η ALU του σχήματος 3.2.7. Η μονάδα μνήμης δεδομένων είναι ένα στοιχείο μνήμης με εισόδους: Read/Write Address και Write data, και μία μόνο έξοδο για το Read data. Υπάρχουν διαφορετικά σήματα ελέγχου για την εγγραφή και την ανάγνωση, παρ΄ όλο που μόνο ένα από αυτά είναι ενεργό σε κάθε κύκλο ρολογιού. Η μονάδα για την επέκταση του προσήμου έχει είσοδο των 16 bits, και μετά από επέκταση του προσήμου της εισόδου δίνει αποτέλεσμα των 32 bits που παρουσιάζεται στην έξοδο.


Το σχήμα 3.2.10 απεικονίζει τον τρόπο με τον οποίο συνδέονται οι μονάδες όταν κατασκευάζεται η δίοδος δεδομένων, για μια εντολή φόρτωσης ή αποθήκευσης. Οι είσοδοι που δέχονται τους αριθμούς που προσδιορίζουν τους δύο καταχωρητές του αρχείου καταχωρητών, προέρχονται από τα πεδία της εντολής. Από τα πεδία της εντολής προέρχεται και η τιμή του offset, η οποία μετά την επέκταση προσήμου γίνεται η δεύτερη είσοδος της ALU.



Σχήμα 3.2.10 - Η δίοδος δεδομένων για τις εντολές φόρτωσης ή αποθήκευσης. Τα πεδία της εντολής δίνουν τον αριθμό του καταχωρητή που θα γίνει ανάγνωση ή εγγραφή. Μετά γίνεται προσπέλαση του αρχείου καταχωρητών και η έξοδος Read data 1 γίνεται η πρώτη είσοδος της ALU. Η δεύτερη είσοδος της ALU είναι η τιμή του offset μετά από επέκταση προσήμου. Ακολουθεί ο υπολογισμός της διεύθυνσης της μνήμης δεδομένων, από την ALU. Στη συνέχεια γίνεται ανάγνωση ή εγγραφή στη μνήμη δεδομένων και μία εγγραφή στο αρχείο καταχωρητών αν έχουμε εντολή φόρτωσης.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 6

Ποια είναι τα επιπλέον κυκλώματα που χρειάζονται για την εκτέλεση των εντολών φόρτωσης και αποθήκευσης; Να συγκρίνετε την απάντησή σας με το σχήμα 3.2.10.

Δίοδος δεδομένων για τις εντολές διακλάδωσης με συνθήκη και τις εντολές μεταπήδησης

Η εντολή beq (εντολή διακλάδωσης με συνθήκη) για να εκτελεστεί χρειάζεται δύο καταχωρητές (τα περιεχόμενά τους συγκρίνονται αν είναι ίσα), και ένα πεδίο offset των 16 bits που χρησιμοποιείται για τον υπολογισμό της διεύθυνσης του στόχου διακλάδωσης. Η μορφή της εντολής είναι:

- beq $1, $2, offset.

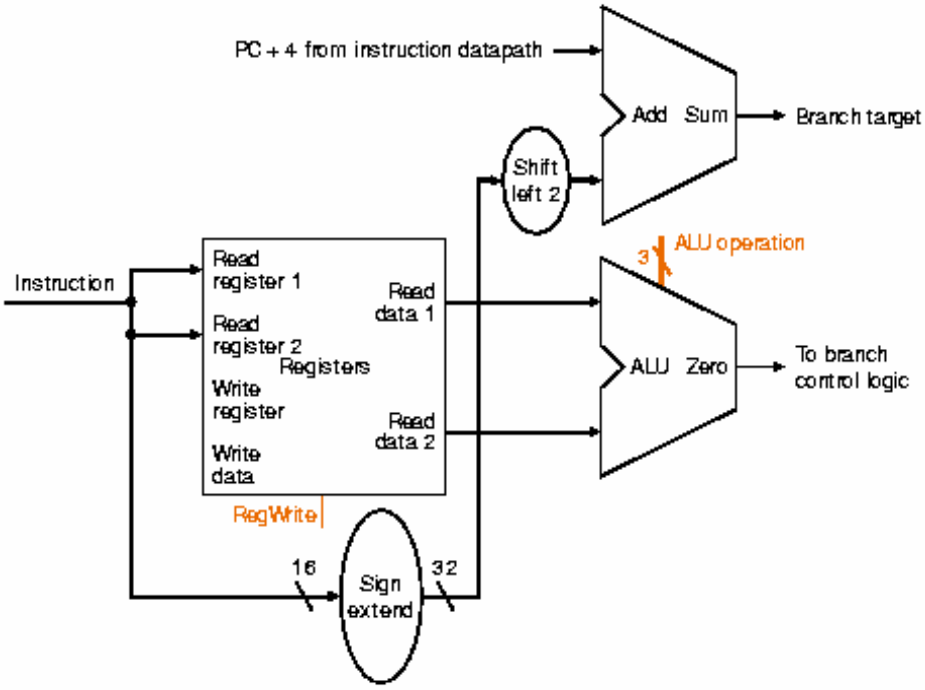
Για την εκτέλεση αυτής της εντολής πρέπει να υπολογίσουμε τη διεύθυνση του στόχου διακλάδωσης, προσθέτοντας το πεδίο offset της εντολής, μετά από την επέκταση προσήμου, στο περιεχόμενο του απαριθμητή προγράμματος. Υπάρχουν δύο λεπτομέρειες στην αρχιτεκτονική συνόλου εντολών που χρειάζονται προσοχή:

- Η αρχιτεκτονική συνόλου εντολών καθορίζει πως η βάση για τον υπολογισμό της διεύθυνσης του στόχου διακλάδωσης είναι η διεύθυνση της εντολής που ακολουθεί τη διακλάδωση. Αφού υπολογίσουμε τη διεύθυνση της επόμενης εντολής (απαριθμητής προγράμματος + 4) κατά την ανάκληση της εντολής στη δίοδο δεδομένων, είναι απλό να χρησιμοποιήσουμε αυτή την τιμή ως βάση για τον υπολογισμό της διεύθυνσης του στόχου διακλάδωσης.

- Η διεύθυνση του στόχου διακλάδωσης πρέπει να απέχει από την προηγούμενη διεύθυνση κατά πολλαπλάσιο του 4 (δεδομένου ότι οι εντολές είναι 32 bits=4 bytes). Αυτό επιτυγχάνεται με τον εξής τρόπο: γράφουμε στο πεδίο offset οποιοδήποτε αριθμό και τον ολισθαίνουμε αριστερά κατά 2 bits.

Για την εκτέλεση της εντολής διακλάδωσης με συνθήκη, πρέπει να προσδιοριστεί εάν η επόμενη εντολή είναι η εντολή που ακολουθεί σειριακά ή είναι η εντολή στη διεύθυνση του στόχου διακλάδωσης. Όταν η συνθήκη είναι αληθής (π.χ. οι τελεστέοι είναι ίσοι), τότε η διεύθυνση του στόχου διακλάδωσης γίνεται η νέα τιμή του απαριθμητή προγράμματος και λέμε ότι η διακλάδωση ακολουθείται. Εάν οι τελεστέοι δεν είναι ίσοι, τότε η ενημερωμένη τιμή του απαριθμητή προγράμματος αντικαθιστά την ήδη υπάρχουσα (όπως γίνεται και στα υπόλοιπα είδη των εντολών). Σ' αυτή την περίπτωση λέμε πως η διακλάδωση δεν ακολουθείται.

Η εντολή διακλάδωσης με συνθήκη επιτελεί δύο λειτουργίες: συγκρίνει τα περιεχόμενα των καταχωρητών και υπολογίζει τη διεύθυνση του στόχου διακλάδωσης. Για τον υπολογισμό της διεύθυνσης του στόχου διακλάδωσης θα χρειαστούμε μια μονάδα για την επέκταση προσήμου, όπως και στο σχήμα 3.2.9, και έναν αθροιστή. Επίσης πρέπει να αλλάξουμε το κομμάτι της διόδου δεδομένων που αφορά στην ανάκληση της εντολής. Για τη σύγκριση, θα χρειαστούμε το αρχείο καταχωρητών του σχήματος 3.2.6 που θα μας δώσει τους δύο τελεστέους. Για τη σύγκριση των δύο τελεστέων μπορούμε να χρησιμοποιήσουμε την ALU. Επίσης αφού η ALU έχει σαν έξοδο ένα σήμα (Zero) το οποίο μας δείχνει αν το αποτέλεσμα της πράξης είναι 0, μπορούμε να στείλουμε τους δύο τελεστέους στην ALU για να γίνει αφαίρεση. Εάν το σήμα Zero στη μονάδα της ALU είναι ενεργό, τότε οι δύο τιμές των τελεστέων είναι ίσες. Παρ' όλο που η έξοδος Zero της ALU μας δείχνει πάντα αν το αποτέλεσμα είναι 0, θα τη χρησιμοποιούμε μόνο για να ελέγχουμε την ισότητα των τελεστέων στις εντολές διακλάδωσης. Αργότερα θα δείξουμε πως συνδέονται τα σήματα ελέγχου στην έξοδο της ALU ώστε να χρησιμοποιηθούν στη δίοδο δεδομένων για την επιλογή της σωστής τιμής του απαριθμητή προγράμματος. Η δίοδος δεδομένων για την εντολή διακλάδωσης φαίνεται στο σχήμα 3.2.11.



Σχήμα 3.2.11 - Η δίοδος δεδομένων για την εντολή διακλάδωσης χρησιμοποιεί την ALU για τον υπολογισμό της συνθήκης διακλάδωσης και έναν ξεχωριστό αθροιστή για τον υπολογισμό της διεύθυνσης του στόχου διακλάδωσης. Για τον υπολογισμό της διεύθυνσης του στόχου διακλάδωσης, προστίθεται η τιμή του απαριθμητή προγράμματος αυξημένη κατά 4, με το πεδίο offset της εντολής (μετά από την επέκταση προσήμου και ολίσθηση κατά 2 bits προς τα αριστερά).

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 7

Πώς γίνεται ο υπολογισμός της διεύθυνσης του στόχου διακλάδωσης; Χρειάζεται κάποια επιπλέον κυκλώματα για να επιτευχθεί αυτό; Αν ναι, ποια είναι αυτά και πως τα συνδέουμε στο τμήμα της διόδου δεδομένων για τις εντολές διακλάδωσης; Να συγκρίνετε την απάντησή σας με την προηγούμενη παράγραφο.

Η εντολή μεταπήδησης (jump) επιδρά στην τιμή του περιεχομένου του απαριθμητή προγράμματος. Λειτουργεί αντικαθιστώντας ένα τμήμα του περιεχομένου του απαριθμητή προγράμματος, με τα 26 λιγότερο σημαντικά bits της εντολής, ολισθημένα αριστερά κατά 2 bits.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 8

Τι πρέπει να αλλάξει στο σχήμα 3.2.11 έτσι ώστε η δίοδος δεδομένων να περιλαμβάνει και τις εντολές μεταπήδησης;

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 8

Η εντολή μεταπήδησης επιδρά στην τιμή του απαριθμητή προγράμματος. Επομένως πρέπει να πάρουμε τα 26 λιγότερο σημαντικά ψηφία της εντολής (με τη μονάδα επέκτασης προσήμου) και κάνουμε ολίσθηση προς τα αριστερά κατά 2. Η τιμή αυτή αντικαθιστά ένα τμήμα του περιεχομένου του απαριθμητή προγράμματος. Άρα, ουσιαστικά δεν αλλάζει τίποτα στο σχήμα 3.2.11, μόνο η τιμή του απαριθμητή προγράμματος.
(Πρέπει να σημειωθεί πως στο σχήμα 3.2.11 υπάρχει το αρχείο καταχωρητών και μία επιπλέον ALU. Αυτά τα κυκλώματα δεν χρειάζονται στην περίπτωση που η δίοδος δεδομένων υλοποιεί μόνο τις εντολές μεταπήδησης, αφού δεν γίνεται ανάγνωση ή εγγραφή σε κάποιον καταχωρητή.)

Αφού έχουμε εξετάσει τις διόδους δεδομένων που χρειάζονται για την εκτέλεση των διαφορετικών τύπων εντολών, μπορούμε να τις συνδυάσουμε έτσι ώστε να έχουμε μία μόνο δίοδο δεδομένων και να προσθέσουμε την μονάδα ελέγχου.

Η δίοδος δεδομένων και η μονάδα ελέγχου ενός κύκλου

Στην υποενότητα αυτή θα αναφερθούμε στην υλοποίηση της διόδου δεδομένων και της μονάδας ελέγχου του υπολογιστή MIPS, όταν η εκτέλεση των εντολών γίνεται σε έναν κύκλο ρολογιού. Αρχικά θα συνδέσουμε όλα τα τμήματα της διόδου δεδομένων, μαζί με τα κατάλληλα σήματα ελέγχου, έτσι ώστε να περιλαμβάνονται όλες οι εντολές. Στη συνέχεια θα κατασκευάσουμε το κύκλωμα που αποτελεί τη μονάδα ελέγχου της ALU, από λογικές πύλες και θα ολοκληρώσουμε την κατασκευή της διόδου δεδομένων του ενός κύκλου, προσθέτοντας τις αρτηρίες, κάποια επιπλέον κυκλώματα, καθώς και τα απαιτούμενα σήματα ελέγχου. Τέλος, θα αναφερθούμε στα μειονεκτήματα αυτού του τρόπου υλοποίησης και τις αιτίες που τα προκαλούν.

Πως υλοποιείται η δίοδος δεδομένων ενός κύκλου του υπολογιστή MIPS

Παρακάτω θα εξετάσουμε μια απλοποιημένη υλοποίηση της διόδου δεδομένων του υπολογιστή MIPS. Θα δημιουργήσουμε αυτή την απλή δίοδο δεδομένων και τη μονάδα ελέγχου, ενώνοντας τα τμήματα της διόδου δεδομένων που έχουμε ήδη κατασκευάσει και προσθέτοντας τις γραμμές ελέγχου όπου χρειάζεται.

Θα κατασκευάσουμε τη δίοδο δεδομένων από τα τμήματα που έχουμε κατασκευάσει στα σχήματα 3.2.5, 3.2.8, 3.2.10 και 3.2.11. Αυτή η απλή δίοδος δεδομένων θα πρέπει να εκτελεί όλες τις εντολές σε έναν κύκλο ρολογιού. Αυτό σημαίνει πως κανένα τμήμα της διόδου δεδομένων δεν θα μπορεί να χρησιμοποιηθεί περισσότερο από μία φορά για κάθε εντολή. Στην περίπτωση που κάποιο κύκλωμα χρειάζεται παραπάνω από μία φορά θα πρέπει να υπάρχει δύο φορές μέσα στη δίοδο δεδομένων. Επιπλέον η μνήμη εντολών πρέπει να είναι διαχωρισμένη από τη μνήμη δεδομένων. Επειδή κάποιες από τις μονάδες πρέπει να υπάρχουν δύο φορές στη δίοδο δεδομένων, όταν συνδέσουμε τα κομμάτια που έχουμε ήδη κατασκευάσει, οι διαφορετικές εντολές θα μπορούν να μοιράζονται από κοινού κάποιο από τα κυκλώματα.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 9

Μπορείτε να εξηγήσετε τον τρόπο με τον οποίο δύο διαφορετικές εντολές μπορούν να μοιραστούν από κοινού κάποιο κύκλωμα;

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 9

Για να γίνει αυτό ανάμεσα σε δύο διαφορετικούς τύπους εντολών, πρέπει να επιτρέπονται πολλαπλές συνδέσεις στην είσοδο του κυκλώματος και να υπάρχει κάποιο σήμα ελέγχου που θα επιλέγει ανάμεσα στις εισόδους. Αυτό εύκολα πραγματοποιείται με έναν πολυπλέκτη, ο οποίος επιλέγει ανάμεσα από πολλές εισόδους, ανάλογα με την τιμή στις γραμμές ελέγχου του.

Παράδειγμα

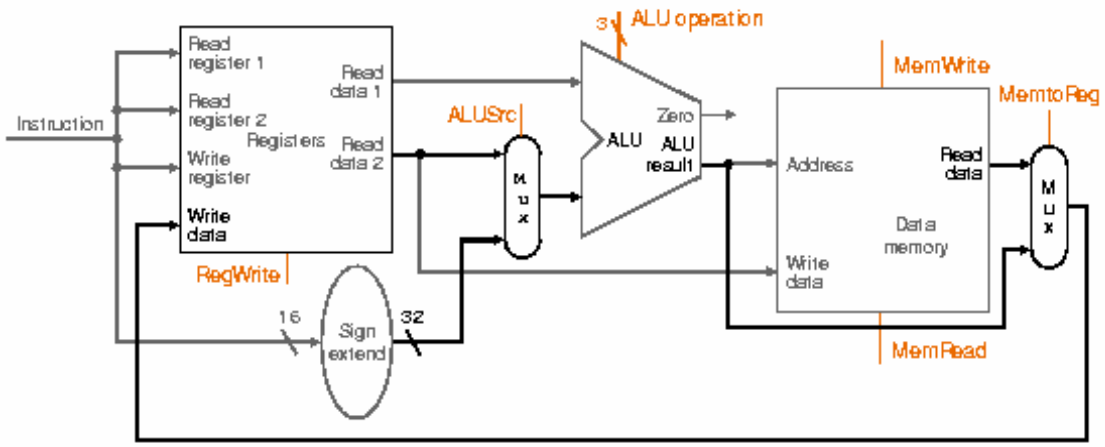
Η δίοδος δεδομένων των εντολών τύπου R του σχήματος 3.2.8 και η δίοδος δεδομένων των εντολών φόρτωσης και αποθήκευσης του σχήματος 3.2.10 είναι πανομοιότυπες. Οι βασικές διαφορές τους είναι οι εξής:

- **Η δεύτερη είσοδος της ALU είναι είτε ένας καταχωρητής (στις εντολές τύπου R), είτε η επέκταση προσήμου του offset το οποίο βρίσκεται στο λιγότερο σημαντικό τμήμα της εντολής (στις εντολές φόρτωσης και αποθήκευσης).**
- **Η τιμή που γράφεται στον καταχωρητή αποτελέσματος (Result register) προέρχεται από την ALU (στις εντολές τύπου R) ή από τη μνήμη (στην περίπτωση μιας εντολής φόρτωσης).**

Να δείξετε πως μπορούν να ενωθούν αυτές οι δύο δίοδοι δεδομένων χρησιμοποιώντας πολυπλέκτες, χωρίς να χρειαστεί να χρησιμοποιήσετε δύο φορές τα κυκλώματα που είναι κοινά στα σχήματα 3.2.8 και 3.2.10. Αγνοήστε την είσοδο ελέγχου στους πολυπλέκτες.

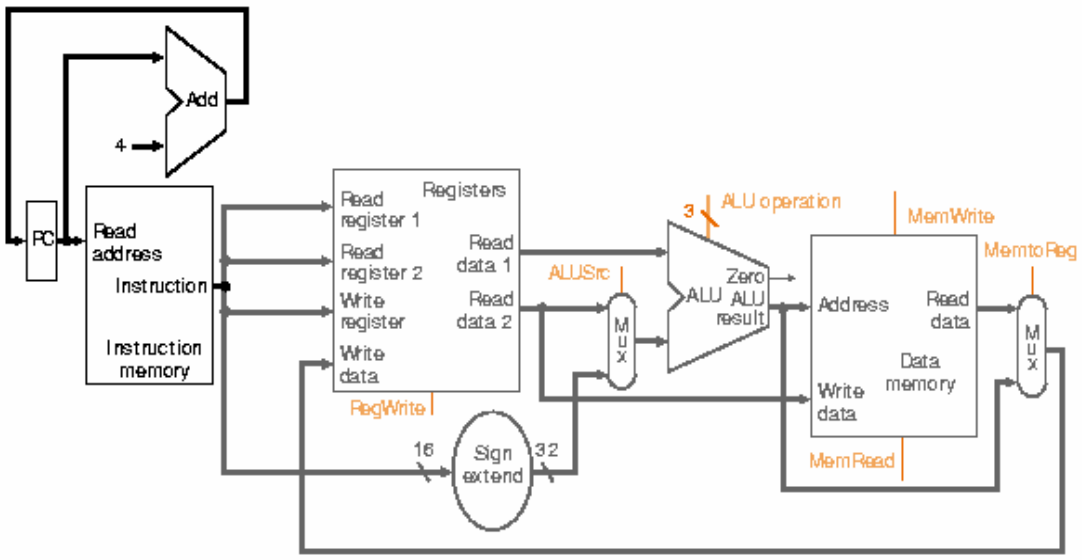
Απάντηση:

Για να συνδυάσουμε τις διόδους δεδομένων και να χρησιμοποιήσουμε μόνο ένα αρχείο καταχωρητών και μόνο μία ALU, πρέπει να υποστηρίζουμε δύο διαφορετικές εισόδους για την δεύτερη είσοδο της ALU καθώς και δύο διαφορετικές εισόδους για το αρχείο καταχωρητών. Έτσι τοποθετούμε έναν πολυπλέκτη στην είσοδο της ALU και άλλον ένα στην είσοδο δεδομένων του αρχείου καταχωρητών. Το σχήμα 3.2.12 απεικονίζει την δίοδο δεδομένων που προκύπτει από τον συνδυασμό των δύο διόδων.



Σχήμα 3.2.12 - Ο συνδυασμός των διόδων δεδομένων των εντολών φόρτωσης και αποθήκευσης και των εντολών τύπου R. Αυτό το παράδειγμα δείχνει τον τρόπο με τον οποίο ενώνουμε σε μία μόνο δίοδο δεδομένων τα τμήματα που είχαμε δημιουργήσει.

Το τμήμα της διόδου δεδομένων που χρησιμοποιείται για την ανάκληση των εντολών και την αύξηση του απαριθμητή προγράμματος του σχήματος 3.2.5, μπορεί εύκολα να προστεθεί στη δίοδο δεδομένων του σχήματος 3.2.12. Το αποτέλεσμα φαίνεται στο σχήμα 3.2.13. Η συνενωμένη δίοδος δεδομένων περιλαμβάνει τη μνήμη εντολών και μία ξεχωριστή μνήμη δεδομένων. Επιπλέον η δίοδος δεδομένων χρειάζεται έναν αθροιστή και μία ALU, αφού ο αθροιστής χρησιμοποιείται στην αύξηση του απαριθμητή προγράμματος, ενώ η ALU χρησιμοποιείται για την εκτέλεση της εντολής στον ίδιο κύκλο ρολογιού.

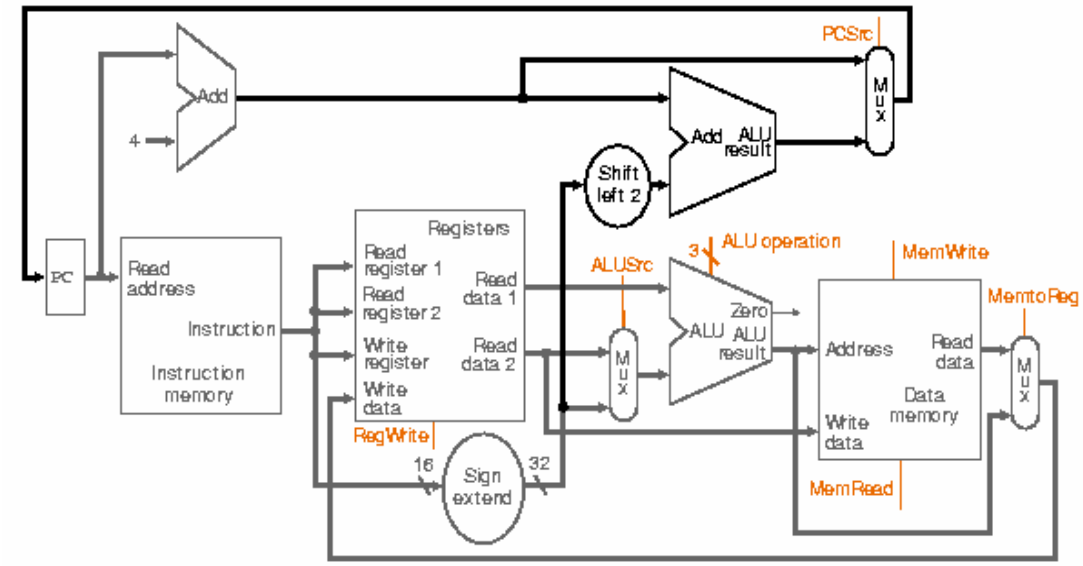


Σχήμα 3.2.13 - Το τμήμα της διόδου δεδομένων που χρησιμοποιείται για την ανάκληση των εντολών και την αύξηση του απαριθμητή προγράμματος του σχήματος 3.2.5 έχει προστεθεί στη δίοδο δεδομένων του σχήματος 3.2.12 (η οποία μπορεί να διαχειριστεί τις εντολές φόρτωσης και αποθήκευσης, καθώς και τις αριθμητικές και λογικές εντολές). Το τμήμα που έχει προστεθεί είναι τονισμένο στο σχήμα με πιο έντονες γραμμές. Το αποτέλεσμα είναι μία δίοδος δεδομένων η οποία υποστηρίζει πολλές λειτουργίες του συνόλου εντολών του MIPS. Τα βασικά τμήματα της διόδου που λείπουν είναι αυτά για τις εντολές διακλάδωσης με συνθήκη και τις εντολές μεταπήδησης.

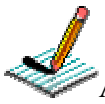
 Τώρα μπορούμε να συνδέσουμε όλα αυτά τα τμήματα για να κατασκευάσουμε μία μόνο δίοδο δεδομένων (για την αρχιτεκτονική του υπολογιστή MIPS), προσθέτοντας την δίοδο δεδομένων για τις εντολές διακλάδωσης του σχήματος 3.2.11. Το σχήμα 3.2.14 δείχνει την δίοδο δεδομένων που παίρνουμε εάν συνδέσουμε τα σχήματα 3.2.11 και 3.2.13. Οι εντολές διακλάδωσης χωρίς συνθήκη χρησιμοποιούν την κεντρική ALU για τη σύγκριση των τελεστέων του καταχωρητή, επομένως πρέπει να κρατήσουμε τον αθροιστή του σχήματος 3.2.11 για τον υπολογισμό της διεύθυνσης του στόχου διακλάδωσης. Επίσης χρειάζεται ένας ακόμα πολυπλέκτης για να επιλέξει τη διεύθυνση της επόμενης εντολής ανάμεσα στην τιμή: απαριθμητής προγράμματος+4 και την επόμενη διεύθυνση του στόχου διακλάδωσης. Επειδή στον απαριθμητή προγράμματος θα γίνει εγγραφή με μία από αυτές τις δύο τιμές σε κάθε κύκλο του ρολογιού, δεν χρειαζόμαστε σήμα ελέγχου εγγραφής.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 10

Να σχεδιάσετε την ολοκληρωμένη δίοδο δεδομένων του υπολογιστή MIPS, έτσι ώστε να μπορεί να εκτελέσει τα βασικά είδη εντολών (φόρτωση ή αποθήκευση δεδομένων, τις αριθμητικές και λογικές εντολές, και τις εντολές διακλάδωσης) σε έναν κύκλο ρολογιού. Να συγκρίνετε το σχήμα που σχεδιάσατε με το σχήμα που ακολουθεί (σχήμα 3.2.14).



Σχήμα 3.2.14 - Η δίοδος δεδομένων για την αρχιτεκτονική του υπολογιστή MIPS συνδυάζει όλα τα κυκλώματα που απαιτούνται για τους διαφορετικούς τύπους εντολών. Η δίοδος δεδομένων μπορεί να εκτελέσει τις βασικές εντολές (φόρτωση ή αποθήκευση δεδομένων, τις αριθμητικές και λογικές εντολές, και τις εντολές διακλάδωσης με συνθήκη), σε ένα μόνο κύκλο ρολογιού. Οι προσθήκες που έγιναν στο σχήμα 3.2.13 για να μπορεί η δίοδος να εκτελεί και εντολές διακλάδωσης, είναι σημειωμένες με πιο έντονες γραμμές.

 Αφού ολοκληρώσαμε την δίοδο δεδομένων, μπορούμε να προσθέσουμε τη μονάδα ελέγχου. Η μονάδα ελέγχου πρέπει να μπορεί να δέχεται εισόδους και να παράγει: ένα σήμα εγγραφής για κάθε στοιχείο μνήμης, τον επιλογέα σε κάθε πολυπλέκτη, και τη μονάδα ελέγχου της ALU.

Η μονάδα ελέγχου της ALU

Η μονάδα ελέγχου της ALU έχει τρεις εισόδους ελέγχου. Μόνο πέντε από τους οχτώ πιθανούς συνδυασμούς χρησιμοποιούνται ως είσοδοι ελέγχου. Οι πέντε αυτοί πιθανοί συνδυασμοί φαίνονται στο παρακάτω σχήμα.

Μονάδα ελέγχου της ALU	Λειτουργία
000	And
001	Or
010	Add
110	Subtract
111	Set-on-less-than

 **Ανάλογα με τον τύπο της εντολής, η ALU χρειάζεται να εκτελέσει μία από αυτές τις πέντε λειτουργίες (functions). Για τις εντολές φόρτωσης και αποθήκευσης, θα χρησιμοποιήσουμε την ALU για να υπολογίσουμε την διεύθυνση της μνήμης, επομένως η λειτουργία που απαιτείται είναι η άθροιση(add). Για τις εντολές τύπου R, η ALU πρέπει να εκτελέσει κάποια από τις λειτουργίες (subtract, add, AND, OR, set-on-less-than), ανάλογα με την τιμή του πεδίου function της εντολής (που είναι τα 6 λιγότερα σημαντικά ψηφία της εντολής). Για τον έλεγχο της ισότητας των τελεστέων που βρίσκονται στους δύο καταχωρητές, στις εντολές διακλάδωσης (beq), η ALU πρέπει να κάνει αφαίρεση**.

Μπορούμε να δημιουργήσουμε την είσοδο ελέγχου της ALU (που αποτελείται από 3 ψηφία), χρησιμοποιώντας μια μικρή μονάδα ελέγχου που θα έχει ως εισόδους το πεδίο function της εντολής, και ένα πεδίο ελέγχου των 2 ψηφίων το οποίο λέγεται ALUOp. Το ALUOp δηλώνει ότι η λειτουργία που θα εκτελεστεί είναι η add (00) για τις εντολές φόρτωσης και αποθήκευσης, η subtract (01) για τις εντολές διακλάδωσης, ή είναι εντολή τύπου R (10), οπότε η λειτουργία της εντολής είναι κωδικοποιημένη στο πεδίο function (10). Η έξοδος της μονάδας ελέγχου της ALU είναι ένα σήμα με 3 ψηφία που ελέγχει κατ΄ ευθείαν την ALU δημιουργώντας έναν από τους πέντε συνδυασμούς των 3 ψηφίων. Στο σχήμα 3.2.15 φαίνεται πως καθορίζονται οι είσοδοι της μονάδας ελέγχου της ALU, του πεδίου ελέγχου ALUOp (2 ψηφία) και του κωδικοποιημένου πεδίου function (6 ψηφία). Επιπλέον στο σχήμα φαίνεται η σχέση μεταξύ των ψηφίων του ALUOp και του κωδικού λειτουργίας (opcode) της εντολής.

Κωδικός λειτουργίας εντολής	ALU0p	Λειτουργία εντολής	Πεδίο Function	Επιθυμητή λειτουργία της ALU	Είσοδος ελέγχου της ALU
LW	00	Load word	*XXXXXX*	add	010
SW	00	Store word	*XXXXXX*	add	010
Branch equal	01	Branch equal	*XXXXXX*	subtract	110
R-type	10	Add	100000	add	010
R-type	10	Subtract	100010	subtract	110
R-type	10	AND	100100	and	000
R-type	10	OR	100101	or	001
R-type	10	Set-on-less-than	101010	set-on-less-than	111

Σχήμα 3.2.15 - Ο παραπάνω πίνακας δείχνει τον τρόπο με τον οποίο δημιουργούνται τα ψηφία ελέγχου της ALU ανάλογα με τα ψηφία ελέγχου του πεδίου ALUOp και την κωδικοποίηση του πεδίου function για τις εντολές τύπου R. Ο κωδικός λειτουργίας εντολής που βρίσκεται στην πρώτη στήλη (opcode), καθορίζει τα ψηφία του ALUOp. Παρατηρούμε πως όταν ο κωδικός του ALUOp είναι 00 ή 01, τα πεδία της εξόδου δεν εξαρτώνται από τον κωδικό του πεδίου function. Σ΄ αυτή την περίπτωση λέμε πως η τιμή του κωδικού του πεδίου function είναι αδιάφορη (don't care), επομένως οι τιμές αυτές συμβολίζονται με XXXXXX. Όταν η τιμή του ALUOp είναι 10, τότε ο κωδικός του function χρησιμοποιείται για να καθορίσει την είσοδο της μονάδας ελέγχου της ALU.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 11

Να απλοποιήσετε περισσότερο τον πίνακα του σχήματος 3.2.16, χρησιμοποιώντας περισσότερους αδιάφορους όρους. Να συγκρίνετε την απάντησή σας με το σχήμα που ακολουθεί (σχήμα 3.2.16).

Υπάρχουν αρκετοί διαφορετικοί τρόποι με τους οποίους μπορούμε να σχεδιάσουμε το κύκλωμα που παρέχει τα 3 ψηφία ελέγχου της ALU από το πεδίο ALUOp (2 ψηφία) και τον κωδικό του πεδίου function (6 ψηφία). Επειδή μόνο μερικές από τις 64 πιθανές τιμές του πεδίου function μας ενδιαφέρουν και το πεδίο function χρησιμοποιείται μόνο όταν τα ψηφία του ALUOp έχουν την τιμή 10, μπορούμε να χρησιμοποιήσουμε λογικά κυκλώματα τα οποία θα αναγνωρίζουν ένα υποσύνολο των πιθανών τιμών και έτσι θα παίρνουμε σωστά τα ψηφία ελέγχου της ALU. Για να κατασκευάσουμε αυτά τα λογικά κυκλώματα θα σχεδιάσουμε ένα πίνακα αληθείας για τους συνδυασμούς του κωδικού του πεδίου function που μας ενδιαφέρουν και τα ψηφία του ALUOp. Ο πίνακας του σχήματος 3.2.16 απεικονίζει τον τρόπο με τον οποίο καθορίζονται τα ψηφία της ALU ανάλογα με τα πεδία στην είσοδο. Αφού ο πίνακας αληθείας είναι πολύ μεγάλος (2^8=256 τιμές) και η μονάδα ελέγχου της ALU δεν χρησιμοποιείται για τους περισσότερους από αυτούς τους συνδυασμούς, στον πίνακα αληθείας φαίνονται μόνο οι τιμές που χρειάζεται η μονάδα ελέγχου της ALU.

ALUOp		Κωδικός του πεδίου Function						Λειτουργία
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	010
X	1	X	X	X	X	X	X	110
1	X	X	X	0	0	0	0	010
1	X	X	X	0	0	1	0	110
1	X	X	X	0	1	0	0	000
1	X	X	X	0	1	0	1	001
1	X	X	X	1	0	1	0	111

Σχήμα 3.2.16 - Ο πίνακας αληθείας για τα 3 ψηφία της μονάδας ελέγχου της ALU συναρτήσει του ALUOp και του κωδικού του πεδίου function. Τα ψηφία του ALUOp ονομάζονται ALUOp1 και ALUOp2. Στο σχήμα απεικονίζονται μόνο οι τιμές που δεν είναι όλες 0 στη μονάδα ελέγχου της ALU, μαζί με κάποιες τιμές που είναι αδιάφορες. Για παράδειγμα, η ALUOp δεν χρησιμοποιεί την κωδικοποίηση 11, έτσι ο πίνακας αληθείας περιέχει τις τιμές 1X και X1, αντί για 10 και 01. Επίσης όταν χρησιμοποιείται ο κωδικός του πεδίου function, τα δύο πρώτα ψηφία (F5 και F4) αυτού του πεδίου είναι πάντα 10 και οι τιμές 00, 01, 11 είναι αδιάφορες (don't care terms), και τις έχουμε αντικαταστήσει με XX στον πίνακα αληθείας.

Επειδή σε μερικές περιπτώσεις δεν ενδιαφερόμαστε για κάποιες από τις τιμές στις εισόδους, και θέλουμε οι πίνακες να είναι μικροί, δεν περιλαμβάνουμε στους πίνακες τις αδιάφορες τιμές. Μια τέτοια τιμή απεικονίζεται στους πίνακες με X και δηλώνει πως η έξοδος είναι αληθής, ανεξάρτητα από την τιμή της αντίστοιχης εισόδου. Για παράδειγμα, όταν τα ψηφία του ALUOp είναι 00, όπως στην πρώτη γραμμή του σχήματος 3.2.16, τα ψηφία της μονάδας ελέγχου της ALU είναι 010, ανεξάρτητα από τον κωδικό function. Σ΄ αυτή την περίπτωση οι τιμές του κωδικού function θα είναι αδιάφορες σ΄ αυτή την γραμμή του πίνακα αληθείας.

Αφού κατασκευάσαμε τον πίνακα αληθείας μπορούμε να τον βελτιώσουμε και να τον υλοποιήσουμε με πύλες.

 Το κύκλωμα που υλοποιεί τα τρία ψηφία της μονάδας ελέγχου της ALU έχει τρεις ξεχωριστές εξόδους, (Operation2, Operation1 και Operation0), η κάθε μία από τις οποίες αντιστοιχεί σε κάποιο από τα 3 ψηφία της μονάδας ελέγχου της ALU. Η λογική συνάρτηση για κάθε έξοδο σχεδιάζεται αν συνδυάσουμε τους πίνακες αληθείας οι οποίοι καθορίζουν την συγκεκριμένη έξοδο. Για παράδειγμα το λιγότερο σημαντικό ψηφίο της μονάδας ελέγχου της ALU (Operation0), καθορίζεται από τα τις δύο τελευταίες τιμές του πίνακα αληθείας του σχήματος 3.2.16. Έτσι ο πίνακας αληθείας για το Operation0 θα έχει αυτές τις δύο τιμές. Επιπλέον αν κοιτάξουμε τους πίνακες αληθείας για κάθε έξοδο ξεχωριστά, μπορούμε να ελαχιστοποιήσουμε τα λογικά κυκλώματα που χρειάζονται εκμεταλλευόμενοι τις ομοιότητες ανάμεσα στους όρους που σχετίζονται με κάθε μία από τις εξόδους. Το σχήμα 3.2.17 δείχνει τους πίνακες αληθείας για κάθε ένα από τα 3 ψηφία της μονάδας ελέγχου της ALU.

ALUOp		Τα 6 ψηφία του πεδίου Function					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
X	1	X	X	X	X	X	X
1	X	X	X	X	X	1	X

a. Ο πίνακας αληθείας για Operation2=1. Αυτός ο πίνακας αντιστοιχεί στο αριστερό ψηφίο του πεδίου Operation του σχήματος 3.2.16.

ALUOp		Τα 6 ψηφία του πεδίου Function					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
0	X	X	X	X	X	X	X
X	X	X	X	X	0	X	X

b. Ο πίνακας αληθείας για Operation1=1.

ALUOp		Τα 6 ψηφία του πεδίου Function					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
1	X	X	X	X	X	X	1
1	X	X	X	1	X	X	X

c. Ο πίνακας αληθείας για Operation0=1.

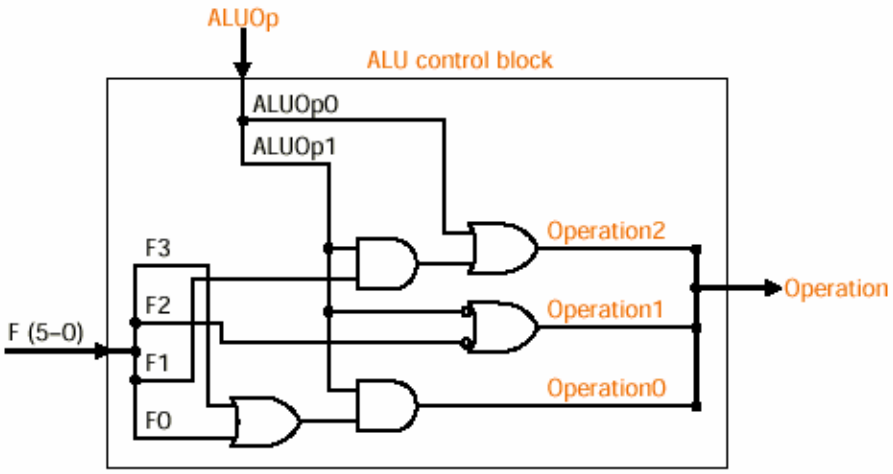
Σχήμα 3.2.17 - Οι πίνακες αληθείας για τα 3 ψηφία της μονάδας ελέγχου της ALU. Μόνο οι τιμές για τις οποίες η έξοδος είναι 1 φαίνονται στα παραπάνω σχήματα. Τα ονόματα των σημάτων που αντιστοιχούν στον κωδικό λειτουργίας των 3 ψηφίων που υποστηρίζονται από την ALU είναι τα: Operation2, Operation1 και Operation0. Έτσι οι πίνακες αληθείας δείχνουν τους συνδυασμούς στις εισόδους για τους οποίους η ALU πρέπει να είναι 010, 001, 110 ή 111 (οι κωδικοί 011, 100 και 101 δεν χρησιμοποιούνται). Όταν το πεδίο ALUOp δεν ισούται με 10 δεν μας ενδιαφέρει η τιμή του κωδικού του πεδίου function και το συμβολίζουμε με X στους πίνακες αληθείας.

 Στο σχήμα 3.2.17 έχουμε εκμεταλλευτεί το γεγονός της κοινής δομής σε κάθε πίνακα αληθείας και έχουμε ενσωματώσει επιπλέον αδιάφορες τιμές. Για παράδειγμα οι πέντε γραμμές στον πίνακα αληθείας του σχήματος 3.2.16 που ορίζουν το Operation1 έχουν μειωθεί σε μόνο δύο γραμμές στο σχήμα 3.2.17. Ένα πρόγραμμα υλοποίησης για ελαχιστοποίηση των λογικών κυκλωμάτων, θα χρησιμοποιήσει τις αδιάφορες τιμές για να μειώσει τον αριθμό των πυλών και τον αριθμό των εισόδων σε κάθε πύλη, σε μία λογική πύλη που θα πραγματοποιεί τους παραπάνω πίνακες αληθείας.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 12

Να σχεδιάσετε το κύκλωμα της μονάδας ελέγχου της ALU χρησιμοποιώντας λογικές πύλες. Να συγκρίνετε το κύκλωμα που σχεδιάσατε, με αυτό του σχήματος που ακολουθεί (σχήμα 3.2.18).
(Υπόδειξη: Να χρησιμοποιήσετε τους πίνακες του σχήματος 3.2.17).

Από τον απλοποιημένο πίνακα αληθείας του σχήματος 3.2.17, μπορούμε να σχεδιάσουμε το λογικό κύκλωμα του σχήματος 3.2.18, το οποίο είναι το κύκλωμα της μονάδας ελέγχου της ALU. Ένα παράδειγμα για τον τρόπο με τον οποίο οι λογικές πύλες κατασκευάζονται από τους πίνακες αληθείας φαίνεται στη λεζάντα του σχήματος 3.2.18.



Σχήμα 3.2.18 - Το κύκλωμα της μονάδας ελέγχου της ALU από το οποίο παίρνουμε τα 3 ψηφία ελέγχου της ALU. Μόνο 4 από τα 6 ψηφία του κωδικού function χρειάζονται στην είσοδο, αφού τα δύο σημαντικότερα ψηφία είναι πάντα αδιάφορα. Ας εξετάσουμε πως φτιάχνουμε το παραπάνω κύκλωμα από τον πίνακα αληθείας του σχήματος 3.2.17. Έστω η έξοδος Operation2 η οποία προκύπτει από τις δύο γραμμές του πίνακα αληθείας για το Operation2. Η δεύτερη γραμμή είναι μια λειτουργία ΚΑΙ που αποτελείται από δύο όρους (F1=1 και ALUOp1=1). Η πρώτη πύλη ΚΑΙ με τις δύο εισόδους ανταποκρίνεται σε αυτόν τον όρο. Ο άλλος όρος για να είναι το Operation2 ενεργό, είναι το ALUOp0. Αυτοί οι δύο όροι συνδέονται με μία πύλη Ή της οποίας η έξοδος είναι το Operation2. Οι έξοδοι Operation0 και Operation1 δημιουργούνται με παρόμοιο τρόπο από τον πίνακα αληθείας.

Το κύκλωμα της μονάδας ελέγχου της ALU μπορεί να κατασκευαστεί εύκολα επειδή έχουμε μόνο τρεις εξόδους, και επειδή μπορούμε να παραλείψουμε πολλούς από τους πιθανούς συνδυασμούς στην είσοδο. Εάν έπρεπε να μετασχηματίσουμε όλους τους κωδικούς function σε σήματα ελέγχου της ALU, τότε αυτή η απλή μέθοδος δεν θα μπορούσε να χρησιμοποιηθεί για την κατασκευή του κυκλώματος της μονάδας ελέγχου της ALU. Τότε θα έπρεπε να χρησιμοποιήσουμε έναν αποκωδικοποιητή, μια μνήμη, ή έναν σύνθετο πίνακα από λογικές πύλες.

Σε γενικές γραμμές, μία λογική εξίσωση και μία απεικόνιση του πίνακα αληθείας μιας λογικής συνάρτησης σχετίζονται μεταξύ τους. Έτσι όταν ένας πίνακας αληθείας καθορίζει μόνο τις εισόδους που το αποτέλεσμα στην έξοδο είναι μη μηδενικό, τότε ο πίνακας δεν περιγράφει ολοκληρωμένα την λογική συνάρτηση. Ένας ολοκληρωμένος πίνακας αληθείας απεικονίζει όλες τις αδιάφορες τιμές. Για παράδειγμα η κωδικοποίηση 11 για το ALUOp πάντα έχει στην έξοδο αδιάφορη τιμή. Έτσι ένας ολοκληρωμένος πίνακας αληθείας έχει XXX στο τμήμα της εξόδου για όλες τις τιμές που το πεδίο ALUOp έχει τιμή 11. Αυτές οι αδιάφορες τιμές μας επιτρέπουν να αντικαθιστούμε τις τιμές 10 και 01 του πεδίου ALUOp με 1X και X1 αντίστοιχα. Η προσθήκη των αδιάφορων τιμών και η ελαχιστοποίηση των κυκλωμάτων είναι πολύπλοκες διαδικασίες και πολύ εύκολα μπορούν να γίνουν λάθη, επομένως τις αφήνουμε σε κάποιο πρόγραμμα να τις εκτελέσει.

Πως σχεδιάζουμε τη βασική μονάδα ελέγχου ενός κύκλου

Αφού εξετάσαμε το σχεδιασμό της ALU η οποία χρησιμοποιεί τον κωδικό function και ένα σήμα των 2 *bits* σαν είσοδο στη μονάδα ελέγχου, μπορούμε να επιστρέψουμε στο σχεδιασμό της υπόλοιπης μονάδας ελέγχου. Για να ξεκινήσουμε αυτή τη διαδικασία ας προσδιορίσουμε όλες τις γραμμές ελέγχου και τα απαιτούμενα κυκλώματα για κάθε εντολή, όπως τα σχεδιάσαμε στη δίοδο δεδομένων του σχήματος 3.2.14. Για να δούμε τον τρόπο με τον οποίο θα προσθέσουμε τις αρτηρίες έτσι ώστε να κατευθύνουμε τις εντολές στη δίοδο δεδομένων, είναι χρήσιμο να επαναλάβουμε την διάταξη των πεδίων για τους τρεις διαφορετικούς τύπους εντολών: τις εντολές τύπου R, διακλάδωσης και φόρτωσης ή αποθήκευσης. Η διάταξη των πεδίων για κάθε τύπο εντολής φαίνεται στο σχήμα 3.2.19.

Για την διάταξη των πεδίων της εντολής βασιζόμαστε στις παρακάτω σημαντικές παρατηρήσεις:

- Το πεδίο op, το οποίο ονομάζεται και κωδικός λειτουργίας (opcode), βρίσκεται πάντα στα bits 31-26. Το πεδίο αυτό θα το αναφέρουμε ως Op[5-0].

- Οι δύο πηγαίοι καταχωρητές που πρόκειται να διαβαστούν τα περιεχόμενά τους, καθορίζονται πάντα από τα πεδία rs και rt στα bits 25-21 και 20-16. Αυτό ισχύει για τις εντολές τύπου R, τις εντολές διακλάδωσης και τις εντολές αποθήκευσης.

- Ο βασικός καταχωρητής για τις εντολές φόρτωσης και αποθήκευσης βρίσκεται πάντα στα bits 25-21(rs).

- Το πεδίο offset (16 bits) για τις εντολές διακλάδωσης, φόρτωσης και αποθήκευσης βρίσκεται πάντα στα bits 15-0.

- Ο καταχωρητής προορισμού για μία εντολή φόρτωσης βρίσκεται στα bits 20-16(rt), ενώ για τις εντολές τύπου R βρίσκεται στα bits 15-11(rd). Έτσι πρέπει να προσθέσουμε ένα πολυπλέκτη για να επιλέγει το πεδίο της εντολής που χρησιμοποιείται, για να μας δείχνει τον αριθμό του καταχωρητή στον οποίο θα γίνει εγγραφή.

Πεδίο	0	rs	rt	rd	shamt	funct
Θέση των bits	31-26	25-21	20-16	15-11	10-6	5-0

a. Εντολές τύπου R

Πεδίο	35 ή 43	rs	rt	address
Θέση των bits	31-26	25-21	20-16	15-0

b. Εντολές φόρτωσης ή αποθήκευσης

Πεδίο	4	rs	rt	address
Θέση των bits	31-26	25-21	20-16	15-0

c. Εντολές διακλάδωσης

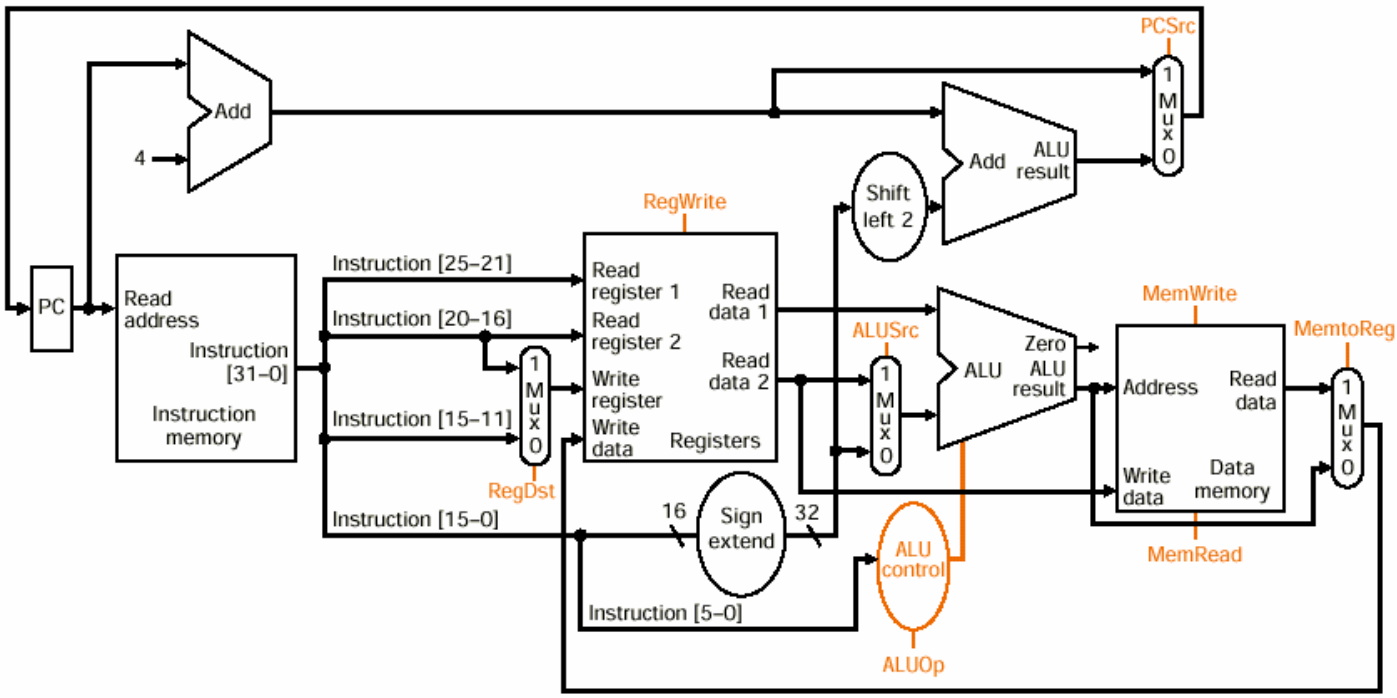
Σχήμα 3.2.19 - Οι τρεις τύποι εντολών χρησιμοποιούν δύο διαφορετικούς τρόπους για την διάταξη των πεδίων τους. Οι εντολές μεταπήδησης (jump) χρησιμοποιούν άλλη διάταξη των πεδίων τους την οποία θα αναφέρουμε περιληπτικά.

a. Η διάταξη των πεδίων της εντολής για τις εντολές τύπου R, οι οποίες έχουν όλες κωδικό λειτουργίας (opcode) 0. Αυτές οι εντολές έχουν τρεις τελεστέους καταχωρητή: rs, rt και rd. Τα πεδία rs και rt είναι πηγαίοι τελεστέοι και το rd είναι ο τελεστέος προορισμού. Η συνάρτηση της ALU βρίσκεται στο πεδίο funct και αποκωδικοποιείται από τη μονάδα ελέγχου της ALU. Οι εντολές που υλοποιούνται με αυτή τη μορφή είναι οι: add, sub, and, or και slt. Το πεδίο shamt χρησιμοποιείται μόνο για ολίσθηση, και το αγνοούμε.
b. Η διάταξη των πεδίων της εντολής για τις εντολές φόρτωσης (κωδικός λειτουργίας=35) και αποθήκευσης (κωδικός λειτουργίας=43). Ο καταχωρητής rs είναι ο βασικός καταχωρητής που προστίθεται στο πεδίο address (16 bits), για να παράγει την διεύθυνση της μνήμης. Στις εντολές φόρτωσης ο rs είναι ο καταχωρητής προορισμού για την τιμή που φορτώθηκε. Στις εντολές αποθήκευσης ο rt είναι ο πηγαίος καταχωρητής του οποίου η τιμή θα αποθηκευτεί στη μνήμη.
c. Η διάταξη των πεδίων της εντολής για τις εντολές διακλάδωσης (κωδικός λειτουργίας=4). Οι καταχωρητές rs και rt είναι οι πηγαίοι καταχωρητές οι οποίοι συγκρίνονται μεταξύ τους για το αν είναι ίσοι. Το πεδίο address (16 bits) ολισθαίνει και προστίθεται στον απαριθμητή προγράμματος για τον υπολογισμό της διεύθυνσης του στόχου διακλάδωσης.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 13

Να περιγράψετε τη διάταξη των πεδίων για τις εντολές τύπου R, τις εντολές φόρτωσης ή αποθήκευσης και τις εντολές διακλάδωσης.

Σύμφωνα με τα παραπάνω μπορούμε να προσθέσουμε στη δίοδο δεδομένων τα επίπεδα της εντολής και έναν επιπλέον πολυπλέκτη, ο οποίος θα μας δίνει τον αριθμό του καταχωρητή (από το αρχείο καταχωρητών), στον οποίο θα γίνει εγγραφή. Το σχήμα 3.2.20 απεικονίζει τα παραπάνω μαζί με τη μονάδα ελέγχου της ALU, τα σήματα εγγραφής για τα στοιχεία μνήμης, το σήμα ανάγνωσης για τη μονάδα μνήμης, και τα σήματα ελέγχου για τους πολυπλέκτες. Αφού όλοι οι πολυπλέκτες έχουν δύο εισόδους, ο καθένας χρειάζεται μία μόνο γραμμή ελέγχου.



Σχήμα 3.2.20 - Η δίοδος δεδομένων του σχήματος 3.2.14 με τους πολυπλέκτες και με όλες τις γραμμές ελέγχου. Οι γραμμές ελέγχου είναι χρωματισμένες στο σχήμα. Επίσης έχει προστεθεί η μονάδα ελέγχου της ALU.

Το σχήμα 3.2.20 απεικονίζει τις επτά γραμμές ελέγχου (οι οποίες αποτελούνται από ένα μόνο bit), μαζί με το σήμα ελέγχου του ALUOp (το οποίο αποτελείται από δύο bits). Έχουμε ήδη εξηγήσει τον τρόπο με τον οποίο λειτουργεί το σήμα ελέγχου του ALUOp. Επομένως είναι απαραίτητο να εξηγήσουμε τις εργασίες που επιτελούν οι άλλες επτά γραμμές ελέγχου, πριν τις χρησιμοποιήσουμε κατά την διάρκεια της εκτέλεσης των εντολών. Το σχήμα 3.2.21 περιγράφει τη λειτουργία αυτών των επτά γραμμών ελέγχου.

Σήμα ελέγχου	Επίδραση όταν δεν είναι ενεργά	Επίδραση όταν είναι ενεργά
MemRead	Τίποτα	Το περιεχόμενο της μνήμης δεδομένων που βρίσκεται στην διεύθυνση που έχει δοθεί για ανάγνωση, τοποθετείται στην έξοδο ανάγνωσης δεδομένων.
MemWrite	Τίποτα	Το περιεχόμενο της μνήμης δεδομένων που βρίσκεται στην διεύθυνση που έχει δοθεί για εγγραφή, αντικαθίσταται από την τιμή που βρίσκεται στην είσοδο εγγραφής των δεδομένων.
ALUSrc	Ο δεύτερος τελεστέος της ALU προέρχεται από τη δεύτερη έξοδο του αρχείου καταχωρητών.	Ο δεύτερος τελεστέος της ALU είναι τα 16 λιγότερα σημαντικά ψηφία της εντολής, μετά από επέκταση προσήμου.
RegDst	Ο αριθμός του καταχωρητή προορισμού στον οποίο θα γίνει η εγγραφή προέρχεται από το πεδίο rt.	Ο αριθμός του καταχωρητή προορισμού στον οποίο θα γίνει εγγραφή προέρχεται από το πεδίο rd.
RegWrite	Τίποτα	Στον καταχωρητή που προσδιορίζεται από την τιμή που έχει δοθεί στην είσοδο εγγραφή καταχωρητή, εγγράφεται η τιμή που έχει δοθεί στην είσοδο εγγραφής δεδομένων.
PCSrc	Ο απαριθμητής προγράμματος (PC) αντικαθίσταται από την έξοδο του αθροιστή που υπολογίζει την τιμή PC + 4.	Ο απαριθμητής προγράμματος (PC) αντικαθίσταται από την έξοδο του αθροιστή, ο οποίος υπολογίζει τον στόχο διακλάδωσης.
MemtoReg	Η τιμή που δίνεται στην είσοδο εγγραφή καταχωρητή προέρχεται από την ALU.	Η τιμή που δίνεται στην είσοδο εγγραφή καταχωρητή προέρχεται από την μνήμη δεδομένων.

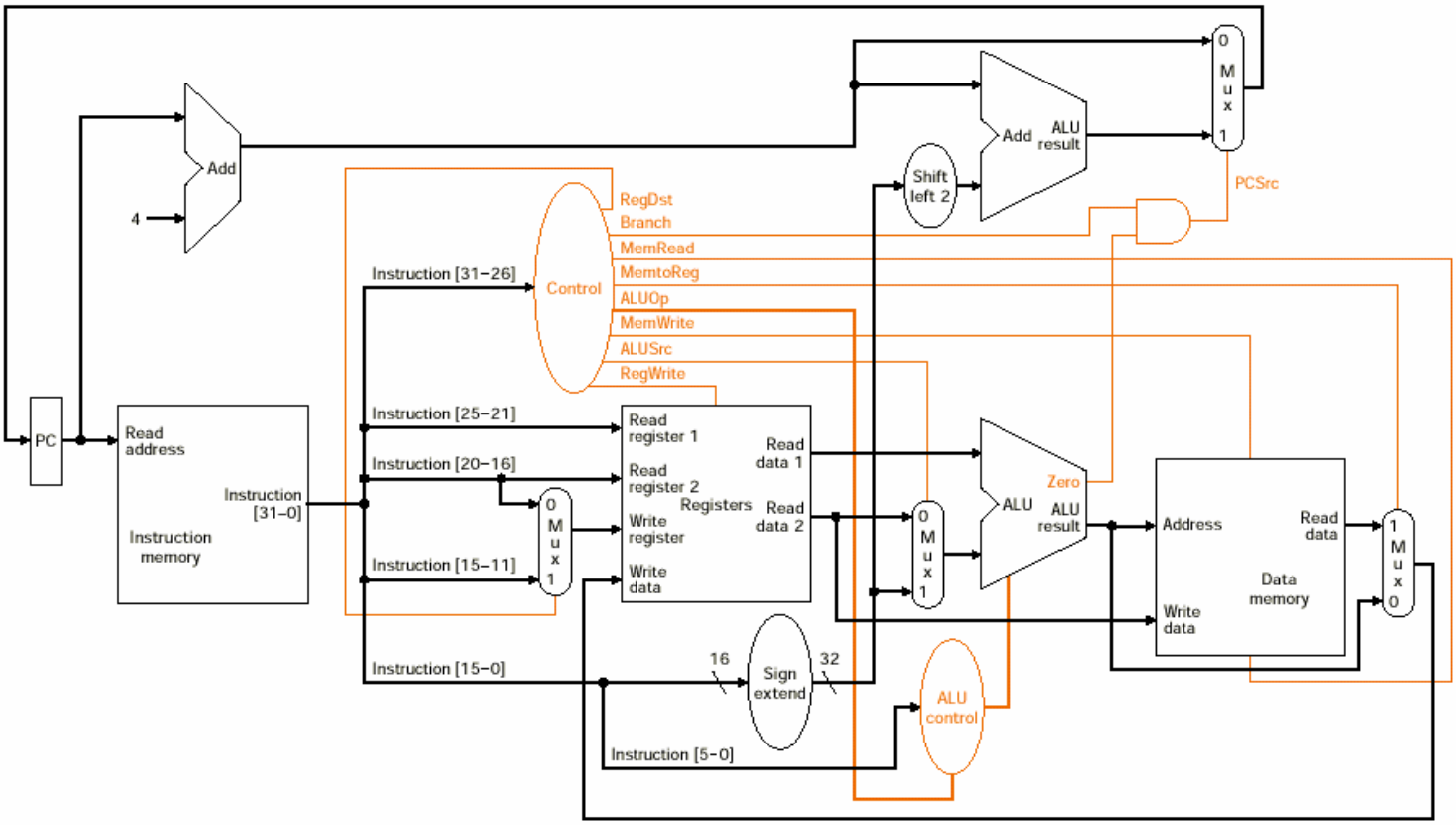
Σχήμα 3.2.21 - Η λειτουργία για κάθε ένα από τα επτά σήματα ελέγχου. Όταν είναι ενεργό το ένα από τα δύο bit σε ένα πολυπλέκτη, η είσοδος επιλογής του πολυπλέκτη έχει την τιμή 1. Διαφορετικά, εάν δεν είναι ενεργό, η είσοδος επιλογής έχει την τιμή 0.

Η μονάδα ελέγχου μπορεί να ενεργοποιήσει όλα αυτά τα σήματα εκτός από το σήμα PCSrc, βασιζόμενη αποκλειστικά στο πεδίο του κωδικού λειτουργίας (opcode field) της εντολής. Η γραμμή ελέγχου ενεργοποιείται μόνο εάν η μονάδα ελέγχου αποφασίσει ότι έχουμε εντολή διακλάδωσης ή ισότητας και η έξοδος 0 (Zero output) της ALU, η οποία χρησιμοποιείται για σύγκριση, είναι αληθής. Για να δημιουργήσουμε το σήμα PCSrc πρέπει να χρησιμοποιήσουμε μια πύλη AND με είσοδο το σήμα από τη μονάδα ελέγχου, το οποίο ονομάζουμε σήμα διακλάδωσης (Branch), και το σήμα της εξόδου 0 της ALU.

Τα εννέα αυτά σήματα ελέγχου ενεργοποιούνται βάσει των έξι σημάτων στην είσοδο της μονάδας ελέγχου, τα οποία είναι τα ψηφία του κωδικού λειτουργίας (opcode bits).

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 14

Να συμπληρώσετε στο σχήμα 3.2.20 τη μονάδα ελέγχου και τις γραμμές ελέγχου, μέσα από τις οποίες μεταφέρονται τα σήματα ελέγχου στα διάφορα κυκλώματα της διόδου δεδομένων. Να συγκρίνετε το κύκλωμα που σχεδιάσατε με το σχήμα που ακολουθεί (σχήμα 3.2.22).



Σχήμα 3.2.22 - Η δίοδος δεδομένων με την μονάδα ελέγχου. Η είσοδος στην μονάδα ελέγχου είναι το πεδίο του κωδικού λειτουργίας (opcode) της εντολής το οποίο αποτελείται από 6 bits. Οι έξοδοι της μονάδας ελέγχου αποτελούνται από τρία σήματα του ενός bit, τα οποία χρησιμοποιούνται στον έλεγχο των πολυπλεκτών (RegDst, ALUSrc, και MemtoReg), από τρία σήματα για τον έλεγχο της εγγραφής και της ανάγνωσης στο αρχείο καταχωρητών και τη μνήμη δεδομένων (RegWrite, MemRead και MemWrite), από ένα σήμα του ενός ψηφίου το οποίο καθορίζει τη διακλάδωση (Branch), και ένα σήμα ελέγχου των 2 ψηφίων για την ALU (ALUOp). Μία πύλη ΚΑΙ χρησιμοποιείται για να συνδέσει το σήμα ελέγχου διακλάδωσης και την έξοδο 0 από την ALU. Η πύλη ΚΑΙ ελέγχει την επιλογή της επόμενη τιμής του απαριθμητή προγράμματος.

Η ενεργοποίηση των γραμμών ελέγχου εξαρτάται μόνο από το πεδίο του κωδικού λειτουργίας (opcode). Επομένως πρέπει να καθορίσουμε εάν το κάθε ένα από τα σήματα ελέγχου είναι 0, 1 ή είναι αδιάφορος όρος (X).

Το σχήμα 3.2.23 καθορίζει τον τρόπο με τον οποίο πρέπει να ενεργοποιηθούν τα σήματα ελέγχου για κάθε κωδικό λειτουργίας, και προέρχεται από τα σχήματα 3.2.15, 3.2.21 και 3.2.22.

Εντολή	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0
Τύπου R	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

Σχήμα 3.2.23 - Η πρώτη γραμμή του πίνακα αντιστοιχεί στις εντολές τύπου R (add, subtract, and, or και slt). Για όλες τις εντολές, τα πεδία του πηγαίου καταχωρητή είναι τα rs και rt και του καταχωρητή προορισμού είναι το rd. Αυτό καθορίζει τον τρόπο με τον οποίο ενεργοποιούνται τα σήματα ALUSrc και RegDst. Επιπλέον μια εντολή τύπου R γράφει σε ένα καταχωρητή (RegWrite = 1), αλλά ούτε διαβάζει ούτε γράφει στη μνήμη δεδομένων. Το πεδίο ALUOp για τις εντολές τύπου R είναι 10 για να δηλώσει πως η μονάδα ελέγχου της ALU πρέπει να παραχθεί από το πεδίο funct. Η δεύτερη και η τρίτη γραμμή του πίνακα είναι για τις εντολές φόρτωσης και αποθήκευσης. Τα ALUSrc και ALUOp καθορίζουν τον υπολογισμό της ενεργής διεύθυνσης. Τα MemRead και MemWrite καθορίζουν την προσπέλαση της μνήμης. Τέλος τα RegDst και RegWrite καθορίζουν ότι στην εντολή φόρτωσης, το αποτέλεσμα αποθηκεύεται στον καταχωρητή rt. Η εντολή διακλάδωσης είναι παρόμοια στη λειτουργία με τις εντολές τύπου R, αφού στέλνει τους καταχωρητές rs και rt στην ALU. Το πεδίο ALUOp για την εντολή διακλάδωσης καθορίζει την αφαίρεση, η οποία χρησιμοποιείται για σύγκριση. Παρατηρούμε πως το πεδίο MemReg δεν έχει σχέση με το αν το σήμα RegWrite είναι 0, (αφού δεν γίνεται εγγραφή στον καταχωρητή, η τιμή των δεδομένων στην είσοδο εγγραφής δεδομένων δεν χρησιμοποιείται). Συνεπώς το MemtoReg στις δύο τελευταίες γραμμές του πίνακα έχει αντικατασταθεί από X, είναι δηλαδή αδιάφορος όρος. Αυτού του τύπου οι αδιάφοροι όροι πρέπει να προστεθούν από τον σχεδιαστή, αφού εξαρτώνται από τον τρόπο λειτουργίας της διόδου δεδομένων. Αδιάφοροι όροι μπορούν επίσης να προστεθούν στο RegDst όταν το RegWrite είναι 0.

Σύμφωνα με τα σχήματα 3.2.21 και 3.2.23, μπορούμε να σχεδιάσουμε τη λογική μονάδα ελέγχου. Πριν τον σχεδιασμό θα εξηγήσουμε τον τρόπο με τον οποίο χρησιμοποιεί κάθε εντολή, τη δίοδο δεδομένων. Σημειώστε ότι ένας πολυπλέκτης ο οποίος βρίσκεται στο 0, αντιδρά με τον ίδιο τρόπο ακόμα και αν η γραμμή ελέγχου δεν είναι ενεργή.

Αντί να εξετάζουμε ολόκληρη τη δίοδο σαν ένα τμήμα συνδυαστικής λογικής, είναι πιο εύκολο να δούμε την εκτέλεση της εντολής σαν μια σειρά από βήματα, εστιάζοντας την προσοχή μας στο κομμάτι της διόδου που αφορά το κάθε βήμα.

Ας ξεκινήσουμε με τις εντολές τύπου R όπως για παράδειγμα η:

add $x, $y, $z.

- **Τα τέσσερα βήματα για την εκτέλεση μιας εντολής τύπου R είναι:**

1. Η εντολή ανακαλείται από τη μνήμη εντολών και ο απαριθμητής προγράμματος αυξάνεται.

2. Οι δύο καταχωρητές, $x και $y, διαβάζονται από το αρχείο καταχωρητών. Επίσης κατά τη διάρκεια αυτού του βήματος, η βασική μονάδα ελέγχου υπολογίζει τον τρόπο ενεργοποίησης των γραμμών ελέγχου.

3. Η ALU επεξεργάζεται τα δεδομένα τα οποία διαβάστηκαν από το αρχείο καταχωρητών, χρησιμοποιώντας τον κωδικό function (τα ψηφία 0-5 της εντολής), που περιγράφει τη λειτουργία της ALU.

4. Το αποτέλεσμα της ALU γράφεται στο αρχείο καταχωρητών, χρησιμοποιώντας τα ψηφία 15-11 της εντολής, για την επιλογή του καταχωρητή προορισμού ($x).

Υπενθυμίζουμε πως η εφαρμογή αυτή χρησιμοποιεί τη συνδυαστική λογική. Δηλαδή οι εντολές τύπου R δεν εκτελούνται σαν μια σειρά από τέσσερα διαφορετικά βήματα. Η δίοδος δεδομένων λειτουργεί σε ένα κύκλο ρολογιού, και τα σήματα στη δίοδο δεδομένων μεταβάλλονται απρόβλεπτα κατά τη διάρκεια αυτού του κύκλου. Τα σήματα σταθεροποιούνται περίπου με τη σειρά των βημάτων της εντολής, επειδή η ροή της πληροφορίας ακολουθεί αυτή την πορεία.

Η εκτέλεση των εντολών φόρτωσης όπως για παράδειγμα η:

lw $s, offset($y)

μπορεί να χωριστεί σε πέντε διαφορετικά βήματα (σε αντιστοιχία με τα τέσσερα βήματα των εντολών τύπου R).

- **Τα πέντε βήματα για την εκτέλεση μιας εντολής φόρτωσης είναι:**

1. Η εντολή ανακαλείται από τη μνήμη εντολών και αυξάνεται ο απαριθμητής προγράμματος.

2. Η τιμή του καταχωρητή ($y) διαβάζεται από το αρχείο καταχωρητών.

3. Η ALU υπολογίζει το άθροισμα της τιμής που διάβασε από το αρχείο καταχωρητών και των 16 λιγότερο σημαντικών ψηφίων της εντολής (offset), μετά από επέκταση προσήμου.

4. Το άθροισμα που υπολογίζεται από την ALU, χρησιμοποιείται στην είσοδο της διεύθυνσης στη μνήμη δεδομένων.

5. Τα δεδομένα από τη μονάδα μνήμης γράφονται στο αρχείο καταχωρητών. Ο καταχωρητής προορισμού δίνεται από τα bits 20-16 της εντολής ($x).

Τέλος μπορούμε να χωρίσουμε σε βήματα την εκτέλεση των εντολών διακλάδωσης με συνθήκη όπως για παράδειγμα η:

beq $x, $y, offset.

Οι εντολές αυτές λειτουργούν περίπου όπως οι εντολές τύπου R, αλλά η έξοδος της ALU χρησιμοποιείται για να καθορίσει την επόμενη τιμή του απαριθμητή προγράμματος που είναι PC +4, ή είναι η διεύθυνση του στόχου διακλάδωσης.

- **Τα τέσσερα βήματα της εκτέλεσης των εντολών διακλάδωσης με συνθήκη είναι:**

1. Η εντολή ανακαλείται από τη μνήμη και αυξάνεται ο απαριθμητής προγράμματος.

2. Οι δύο καταχωρητές, $x και $y, διαβάζονται από το αρχείο καταχωρητών.

3. Η ALU εκτελεί αφαίρεση στις τιμές των δεδομένων που διάβασε από το αρχείο καταχωρητών. Η τιμή του PC + 4 προστίθεται στα 16 λιγότερο σημαντικά ψηφία της εντολής (offset), μετά από επέκταση προσήμου. Το αποτέλεσμα είναι η διεύθυνση του στόχου διακλάδωσης.

4. Η έξοδος 0 της ALU χρησιμοποιείται για την επιλογή της επόμενης τιμής του απαριθμητή προγράμματος.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 15

α)Να περιγράψετε περιληπτικά τα επιμέρους βήματα για την εκτέλεση των εντολών τύπου R, των εντολών φόρτωσης και των εντολών διακλάδωσης με συνθήκη;
β)Για ποιο λόγο χωρίζουμε την εκτέλεση των εντολών σε βήματα;

 Αφού είδαμε τον τρόπο με τον οποίο οι εντολές χωρίζονται σε βήματα, θα συνεχίσουμε με την υλοποίηση του ελέγχου. Η λειτουργία του ελέγχου μπορεί να προσδιοριστεί χρησιμοποιώντας τα περιεχόμενα του σχήματος 3.2.23. Οι έξοδοι είναι οι γραμμές ελέγχου και η είσοδος είναι το πεδίο του κωδικού λειτουργίας (opcode), Op[5-0]. Έτσι μπορούμε να φτιάξουμε ένα πίνακα αληθείας για κάθε έξοδο. Πριν από αυτό ας γράψουμε την κωδικοποίηση για κάθε ένα από τους κωδικούς λειτουργίας που μας ενδιαφέρουν από το σχήμα 3.2.23, και με δεκαδικούς αριθμούς και σαν μια σειρά από bits, τα οποία είναι είσοδος στην μονάδα ελέγχου:

Όνομα εντολής	Κωδικός λειτουργίας σε δεκαδική μορφή	Κωδικός λειτουργίας σε δυαδική μορφή					
		Op5	Op4	Op3	Op2	Op1	Op0
Τύπου R	0	0	0	0	0	0	0
lw	35	1	0	0	0	1	1
sw	43	1	0	1	0	1	1
beq	4	0	0	0	1	0	0

Εάν χρησιμοποιήσουμε αυτόν τον πίνακα, μπορούμε να περιγράψουμε λογικά τη μονάδα ελέγχου σε έναν μεγάλο πίνακα αληθείας που περιέχει όλες τις εξόδους, όπως γίνεται στο σχήμα 3.2.24. Αυτός ο πίνακας καθορίζει ακριβώς την συνάρτηση ελέγχου και μπορούμε να τον υλοποιήσουμε κατ' ευθείαν σε πύλες όπως ακριβώς κάναμε και με την μονάδα ελέγχου της ALU.

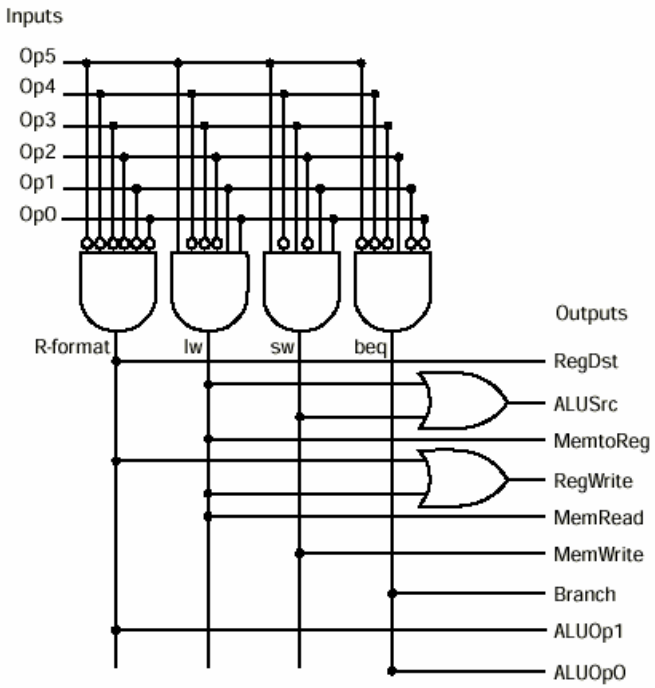
		Τύπου R	lw	sw	beq
Είσοδοι	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Έξοδοι	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

Σχήμα 3.2.24 - Η συνάρτηση ελέγχου όταν οι εντολές εκτελούνται σε έναν κύκλο ρολογιού, καθορίζεται πλήρως από αυτόν τον πίνακα. Στον πίνακα φαίνονται οι συνδυασμοί των σημάτων εισόδου που αντιστοιχούν στους τέσσερις κωδικούς λειτουργίας που καθορίζουν τις εξόδους. (Θυμηθείτε πως το Op[5-0] αντιστοιχεί στα ψηφία 31-26 της εντολής, τα οποία είναι το πεδίο του κωδικού λειτουργίας(opcode).) Μετά τις εισόδους, στον πίνακα δίνονται οι έξοδοι. Έτσι η έξοδος RegWrite είναι ενεργή για δύο διαφορετικούς συνδυασμούς των εισόδων. Εάν θεωρήσουμε μόνο τους τέσσερις κωδικούς λειτουργίας που φαίνονται στον πίνακα, μπορούμε να απλοποιήσουμε τον πίνακα αληθείας χρησιμοποιώντας αδιάφορους όρους. Για παράδειγμα, μπορούμε να βρούμε μια εντολή τύπου R με τη βοήθεια της εξίσωσης Op5 • Op2, αφού αυτό είναι αρκετό για να ξεχωρίσουμε τις εντολές τύπου R από τις εντολές φόρτωσης (lw), αποθήκευσης (sw) και διακλάδωσης(beq). Δεν εκμεταλλευόμαστε αυτή την απλοποίηση αφού οι υπόλοιποι κωδικοί λειτουργίας του υπολογιστή MIPS χρησιμοποιούνται στην τελική υλοποίηση.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 16

Να υλοποιήσετε τη συνάρτηση ελέγχου του σχήματος 3.2.24 χρησιμοποιώντας λογικές πύλες ΚΑΙ και λογικές πύλες Ή. Να συγκρίνετε την απάντησή σας με το σχήμα που ακολουθεί (σχήμα 3.2.25).

Η υλοποίηση της συνάρτησης ελέγχου δεν είναι πολύπλοκη. Το σχήμα 3.2.25 απεικονίζει τη συνάρτηση ελέγχου υλοποιημένη με πύλες ΚΑΙ και πύλες Ή σε παράταξη. Αυτή η δομή ονομάζεται προγραμματιζόμενη λογική παράταξη ή PLA (Programmable Logic Array). Η χρήση κυκλωμάτων PLA είναι μία από τις πιο γνωστές μεθόδους για να υλοποιήσουμε την συνάρτηση ελέγχου.



Σχήμα 3.2.25 - Η υλοποίηση της συνάρτησης ελέγχου σύμφωνα με τον πίνακα αληθείας του σχήματος 3.2.24. Οι είσοδοι στις πύλες ΚΑΙ είναι οι είσοδοι της συνάρτησης και τα συμπληρώματά τους. Οι είσοδοι στις πύλες Ή είναι οι έξοδοι των πυλών ΚΑΙ (ή είναι οι είσοδοι της συνάρτησης και τα συμπληρώματά τους). Η έξοδος των πυλών Ή είναι οι έξοδοι της συνάρτησης.

Παράδειγμα

Το σχήμα 3.2.22 περιλαμβάνει την υλοποίηση των εντολών τύπου R, των εντολών φόρτωσης ή αποθήκευσης και των εντολών διακλάδωσης. Να επεκτείνεται κατάλληλα το σχήμα 3.2.22 έτσι ώστε να περιλαμβάνει και τις εντολές μεταπήδησης (jump). Να περιγράψετε πως θα τοποθετηθούν οι καινούριες γραμμές ελέγχου.

Απάντηση:

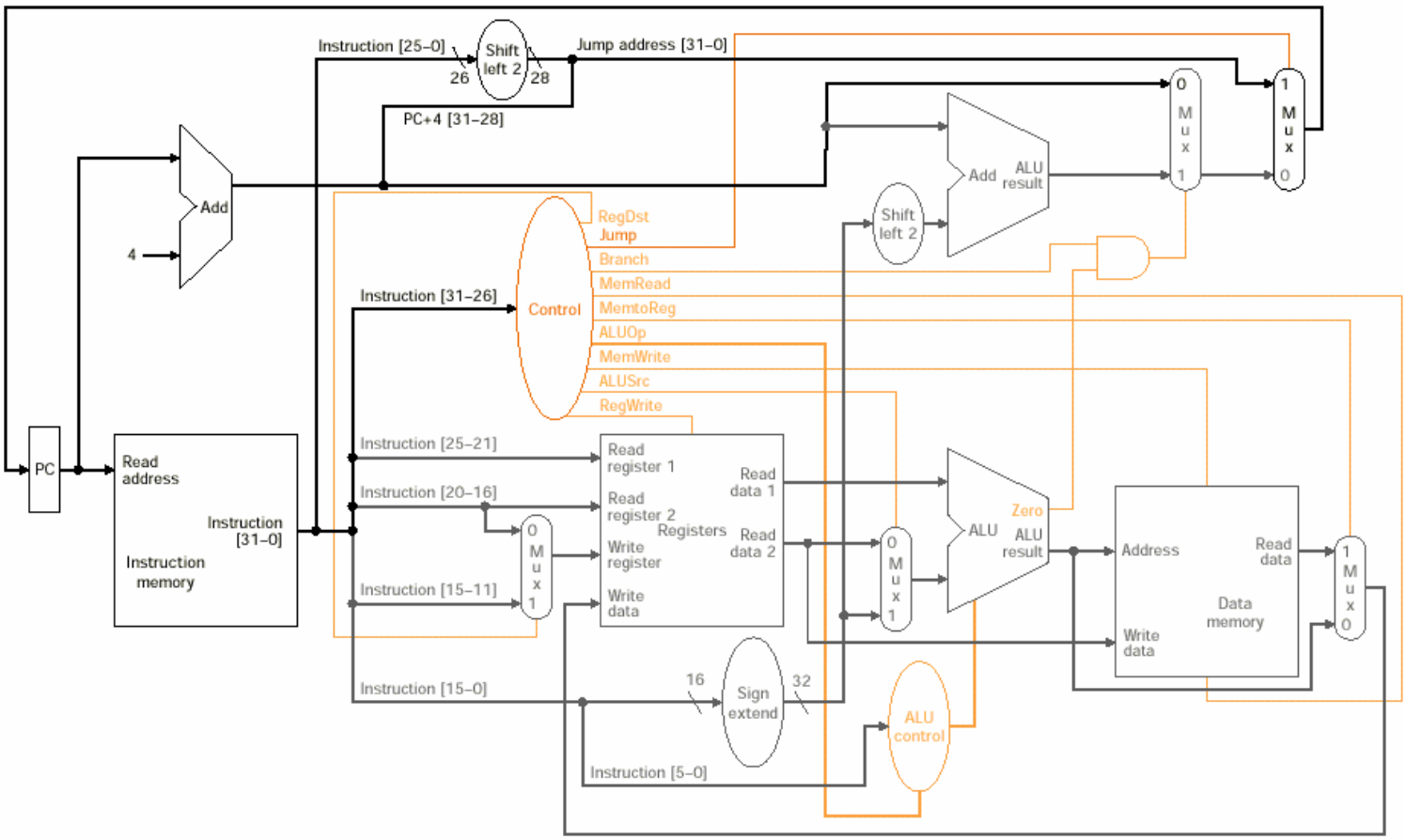
Η εντολή μεταπήδησης μοιάζει με την εντολή διακλάδωσης, με τη διαφορά ότι υπολογίζει την επόμενη τιμή του απαριθμητή προγράμματος διαφορετικά και ότι δεν λειτουργεί υπό συνθήκη. Όπως και στη διακλάδωση, τα 2 λιγότερα σημαντικά ψηφία της διεύθυνσης της εντολής μεταπήδησης είναι πάντα 00_{two}. Τα επόμενα λιγότερο σημαντικά 26 ψηφία από τα 32 ψηφία της διεύθυνσης, προέρχονται από το πεδίο της εντολής όπως φαίνεται στο σχήμα 3.2.26. Τα 4 περισσότερο σημαντικά ψηφία της διεύθυνσης, η οποία θα γίνει η επόμενη τιμή του απαριθμητή προγράμματος, προέρχονται από την τρέχουσα τιμή του απαριθμητή προγράμματος. Έτσι μπορούμε να υλοποιήσουμε μία εντολή μεταπήδησης, αποθηκεύοντας στον απαριθμητή προγράμματος διαδοχικά τα παρακάτω :

- **τα 4 περισσότερο σημαντικά bits του υπάρχοντα απαριθμητή προγράμματος (είναι τα bits 31-28),**
- **το πεδίο της εντολής μεταπήδησης, το οποίο αποτελείται από 26 bits και**
- **τα bits 00_{two}.**

Πεδίο	2	address
Θέση των bits	31-26	25-0

Σχήμα 3.2.26 - Η μορφή της εντολής μεταπήδησης(κωδικός λειτουργίας=2).

Στο σχήμα 3.2.27 φαίνεται η μονάδα ελέγχου, έτσι ώστε να μπορεί να δέχεται και εντολές μεταπήδησης. Χρησιμοποιείται ένας επιπλέον πολυπλέκτης για να επιλέγει τη νέα τιμή του απαριθμητή προγράμματος (PC), η οποία θα είναι η αυξημένη κατά 4 τιμή του απαριθμητής προγράμματος (PC + 4), ή ο στόχος διακλάδωσης, ή ο στόχος μεταπήδησης. Επιπλέον χρειάζεται ένα σήμα ελέγχου για τον πολυπλέκτη που προσθέσαμε. Αυτό το σήμα ελέγχου, το οποίο ονομάζεται σήμα μεταπήδησης (jump), είναι ενεργό μόνο όταν υπάρχει εντολή μεταπήδησης, και αυτό συμβαίνει όταν ο κωδικός λειτουργίας είναι 2 (opcode = 2).



Σχήμα 3.2.27 - Η δίοδος δεδομένων και η μονάδα ελέγχου επεκταμένα κατά τέτοιο τρόπο έτσι ώστε να δέχονται τις εντολές μεταπήδησης. Έχει προστεθεί ένας πολυπλέκτης (πάνω δεξιά στο σχήμα), ο οποίος ελέγχεται από το σήμα ελέγχου μεταπήδησης. Η εντολή μεταπήδησης, σύμφωνα με τον σχεδιασμό των εντολών στον υπολογιστή MIPS, χρησιμοποιεί την τιμή του απαριθμητή προγράμματος [31-28] για την διεύθυνση του στόχου μεταπήδησης, αντί για την τιμή αυτών των bits, αφού έχει αυξηθεί ο απαριθμητής προγράμματος κατά 4.

Τι λειτουργεί λανθασμένα στην υλοποίηση ενός κύκλου

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 17

α)Να προσδιορίσετε τι είναι αυτό που καθορίζεται το μήκος του κύκλου ρολογιού.
β)Για ποιο λόγο η συνολική απόδοση της υλοποίησης της διόδου δεδομένων με χρήση ενός κύκλου ρολογιού για κάθε εντολή, δεν είναι καλή;
Να αιτιολογήσετε την απάντησή σας.

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 17

α)Εξ' ορισμού, ο κύκλος ρολογιού έχει το ίδιο μήκος για κάθε εντολή στην υλοποίηση ενός κύκλου και το CPI (κύκλοι ρολογιού ανά εντολή) θα είναι 1. Σύμφωνα με όσα έχουμε πει, ο κύκλος ρολογιού καθορίζεται από τη μεγαλύτερη πιθανή διαδρομή της μηχανής. Αυτή η διαδρομή είναι σχεδόν πάντα μια εντολή φόρτωσης, η οποία χρησιμοποιεί πέντε λειτουργικές μονάδες σε σειρά: τη μνήμη εντολών, το αρχείο καταχωρητών, την ALU, τη μνήμη δεδομένων και τη μονάδα ελέγχου.
β)Παρ' όλο που το CPI είναι 1 (ένας κύκλος ρολογιού ανά εντολή), η συνολική απόδοση της υλοποίησης που χρησιμοποιεί ένα μόνο κύκλο ρολογιού για κάθε εντολή, δεν είναι πολύ καλή, αφού πολλοί τύποι εντολών μπορούν να εκτελεστούν και σε μικρότερο κύκλο ρολογιού.

Παράδειγμα

Να υποθέσετε ότι ο χρόνος λειτουργίας για τις βασικές μονάδες στην υλοποίηση είναι:

Μονάδες μνήμης : 10ns
ALU και αθροιστές : 10 ns
Αρχείο καταχωρητών (για εγγραφή ή ανάγνωση) : 5ns

Ας υποθέσουμε ότι οι πολυπλέκτες, η μονάδα ελέγχου, η προσπέλαση του απαριθμητή προγράμματος, η μονάδα επέκτασης προσήμου και τα καλώδια δεν προκαλούν καμία καθυστέρηση. Σ' αυτή την περίπτωση, ποια από τις παρακάτω υλοποιήσεις είναι πιο γρήγορη και κατά πόσο;

1. Μία υλοποίηση στην οποία κάθε εντολή λειτουργεί σε ένα κύκλο ρολογιού συγκεκριμένου μήκους.

2. Μία υλοποίηση όπου κάθε εντολή εκτελείται σε ένα κύκλο ρολογιού, χρησιμοποιώντας ένα "μεταβλητό" ρολόι, έτσι ώστε να χρησιμοποιεί κάθε φορά όσο χρόνο χρειάζεται. (Μια τέτοια προσέγγιση δεν είναι πρακτικά εύκολη, αλλά μας επιτρέπει να δούμε τι χάνουμε όταν όλες οι εντολές πρέπει να εκτελεστούν σε ένα κύκλο ρολογιού του ίδιου μήκους.)

Απάντηση:

Ας ξεκινήσουμε συγκρίνοντας τον χρόνο εκτέλεσης της ΚΜΕ για διαφορετικού τύπου εντολές. Γνωρίζουμε ότι :

χρόνος εκτέλεσης ΚΜΕ = πλήθος εντολών × CPI × χρόνος κύκλου ρολογιού

Αφού το CPI είναι 1, μπορούμε να απλοποιήσουμε την παραπάνω εξίσωση ως εξής :

χρόνος εκτέλεσης ΚΜΕ = πλήθος εντολών × χρόνος κύκλου ρολογιού

Το μόνο που χρειαζόμαστε είναι να βρούμε το χρόνο του κύκλου ρολογιού για τις δύο υλοποιήσεις. Η "κρίσιμη" διαδρομή για τους διαφορετικούς τύπους εντολών φαίνεται στο παρακάτω σχήμα :

Τύπος εντολής	**Λειτουργικές μονάδες που χρησιμοποιούνται από τις εντολές**				
Τύπου R	Ανάκληση εντολής	Προσπέλαση καταχωρητή	ALU	Προσπέλαση καταχωρητή	
Φόρτωση	Ανάκληση εντολής	Προσπέλαση καταχωρητή	ALU	Προσπέλαση μνήμης	Προσπέλαση καταχωρητή
Αποθήκευση	Ανάκληση εντολής	Προσπέλαση καταχωρητή	ALU	Προσπέλαση μνήμης	
Διακλάδωση	Ανάκληση εντολής	Προσπέλαση καταχωρητή	ALU		
Μεταπήδηση	Ανάκληση εντολής				

Χρησιμοποιώντας τις "κρίσιμες" διαδρομές, μπορούμε να υπολογίσουμε το απαιτούμενο μήκος του κύκλου ρολογιού, για κάθε τύπο εντολής :

Τύπος εντολής	**Μνήμη εντολών**	**Ανάγνωση του καταχωρητή**	**Λειτουργία ALU**	**Μνήμη δεδομένων**	**Εγγραφή του καταχωρητή**	**Συνολικά**
Τύπου R	10	5	10	0	5	30 ns
Φόρτωση	10	5	10	10	5	40 ns
Αποθήκευση	10	5	10	10		35 ns
Διακλάδωση	10	5	10	0		25 ns
Μεταπήδηση	10					10 ns

Επομένως ο κύκλος ρολογιού για έναν υπολογιστή με σταθερό κύκλο είναι 40ns για όλες τις εντολές, ενώ για έναν υπολογιστή με μεταβλητό κύκλο ρολογιού είναι ανάμεσα στα 10ns και 40ns.

Μπορούμε να υπολογίσουμε το μέσο μήκος του κύκλου ρολογιού για έναν υπολογιστή με μεταβλητό μήκος στον κύκλο ρολογιού, χρησιμοποιώντας τους παραπάνω πίνακες και τη συχνότητα εμφάνισης μιας εντολής. Θεωρώντας γνωστές τις συχνότητες εμφάνισης των εντολών, έχουμε: 22% για τις εντολές φόρτωσης, 11% για τις εντολές αποθήκευσης, 49% για τις εντολές τύπου R, 16% για τις εντολές διακλάδωσης και 2% για τις εντολές μεταπήδησης.

Έτσι ο μέσος χρόνος για κάθε εντολή με μεταβλητό κύκλο ρολογιού είναι:

Κύκλος ρολογιού ΚΜΕ = 40×22% + 35×11% + 30×49% + 25×16% +10×2% = 31.6 ns

Αφού η υλοποίηση με μεταβλητό κύκλο ρολογιού έχει μικρότερο κύκλο, είναι σαφώς ταχύτερη. Η απόδοση υπολογίζεται ως εξής:

$$\frac{\textbf{Απόδοση ΚΜΕ με μεταβλητό κύκλο ρολογιού}}{\textbf{Απόδοση ΚΜΕ με ένα μόνο κύκλο ρολογιού}} =$$

$$\frac{\textbf{Χρόνος εκτέλεσης ΚΜΕ με ένα μόνο κύκλο ρολογιού}}{\textbf{Χρόνος εκτέλεσης ΚΜΕ με μεταβλητό κύκλο ρολογιού}} =$$

$$\frac{\textbf{IC × κύκλος ρολογιού της ΚΜΕ με ένα μόνο κύκλο ρολογιού}}{\textbf{IC × κύκλος ρολογιού της ΚΜΕ με μεταβλητό κύκλο ρολογιού}} =$$

$$\frac{\textbf{κύκλος ρολογιού της ΚΜΕ με ένα μόνο κύκλο ρολογιού}}{\textbf{κύκλος ρολογιού της ΚΜΕ με μεταβλητό κύκλο ρολογιού}} =$$

$$\frac{\textbf{40}}{\textbf{31.6}} = \textbf{1.27}$$

(Όπου IC = πλήθος εντολών)

Επομένως η υλοποίηση με μεταβλητό κύκλο ρολογιού είναι ταχύτερη κατά 1.27 φορές.

Η υλοποίηση ενός κύκλου ρολογιού με διαφορετική ταχύτητα για κάθε εντολή, είναι εξαιρετικά δύσκολη διαδικασία και το κόστος υλοποίησης μιας τέτοιας εφαρμογής είναι μεγαλύτερο από τα πλεονεκτήματα που κερδίζουμε. Ένας εναλλακτικός τρόπος υλοποίησης είναι να χρησιμοποιήσουμε μικρότερο κύκλο ρολογιού και να μεταβάλλουμε τον αριθμό των κύκλων ρολογιού για τους διαφορετικούς τύπους εντολής.

Το κόστος για την εφαρμογή ενός κύκλου με σταθερό κύκλο ρολογιού είναι μεγάλο, αλλά σε πρώτη προσέγγιση μπορούμε να το θεωρήσουμε αποδεκτό. Ωστόσο εάν προσπαθήσουμε να υλοποιήσουμε μια μονάδα κινητής υποδιαστολής ή να υλοποιήσουμε πιο πολύπλοκες εντολές, ή να χρησιμοποιήσουμε πιο σύνθετες

τεχνικές υλοποίησης, δεν θα έχουμε σωστά αποτελέσματα με τη σχεδίαση ενός κύκλου.

Ακολουθεί ένα παράδειγμα για την υλοποίηση μιας μονάδας με κινητή υποδιαστολή.

Παράδειγμα

Υποθέτουμε ότι έχουμε μια μονάδα κινητής υποδιαστολής η οποία χρειάζεται 20 ns για την πρόσθεση των αριθμών κινητής υποδιαστολής και 60 ns για τον πολλαπλασιασμό κλασματικών αριθμών. Όλοι οι άλλοι χρόνοι στις λειτουργικές μονάδες είναι ίδιοι με το προηγούμενο παράδειγμα και η εντολή κινητής υποδιαστολής είναι παρόμοια με μια αριθμητική και λογική εντολή, με τη διαφορά ότι χρησιμοποιεί την ALU κινητής υποδιαστολής αντί για την ALU που είχαμε μέχρι τώρα. Χρησιμοποιώντας τις συχνότητες εμφάνισης των εντολών, να βρείτε το λόγο των αποδόσεων ανάμεσα στην εφαρμογή με διαφορετικό κύκλο ρολογιού για κάθε τύπο εντολής και στην εφαρμογή στην οποία όλες οι εντολές έχουν τον ίδιο κύκλο ρολογιού. Να υποθέσετε ότι :

- **Οι εντολές φόρτωσης και αποθήκευσης διπλής ακρίβειας χρειάζονται τον ίδιο χρόνο όπως και οι εντολές φόρτωσης και αποθήκευσης των 32 ψηφίων.**
- **Οι εντολές διακλάδωσης κινητής υποδιαστολής καταναλώνουν τον ίδιο χρόνο με τις εντολές διακλάδωσης ακεραίου.**
- **Η αφαίρεση και η σύγκριση κλασματικών αριθμών χρειάζονται τον ίδιο χρόνο με την πρόσθεση.**
- **Η διαίρεση κλασματικών αριθμών χρειάζεται τον ίδιο χρόνο με τον πολλαπλασιασμό.**

Απάντηση:

Από το προηγούμενο παράδειγμα γνωρίζουμε ότι:

$$\frac{\textbf{Απόδοση ΚΜΕ με μεταβλητό κύκλο ρολογιού}}{\textbf{Απόδοση ΚΜΕ με ένα μόνο κύκλο ρολογιού}} = \frac{\textbf{κύκλος ρολογιού της ΚΜΕ με ένα μόνο κύκλο ρολογιού}}{\textbf{κύκλος ρολογιού της ΚΜΕ με μεταβλητό κύκλο ρολογιού}}$$

Ο κύκλος ρολογιού σε εφαρμογή ενός κύκλου είναι ίσος με τον μεγαλύτερο χρόνο που χρειάζεται μια εντολή, ο οποίος είναι ο χρόνος εκτέλεσης της εντολής του πολλαπλασιασμού αριθμών κινητής υποδιαστολής. **Ο χρόνος που απαιτείται για τον πολλαπλασιασμό αριθμών κινητής υποδιαστολής, και ορίζει επομένως τον κύκλο ρολογιού, είναι : 10 + 5 + 60 + 5 = 80 ns.**

Έστω ένα υπολογιστής στον οποίο οι εντολές έχουν διαφορετικό κύκλο ρολογιού. Ο χρόνος που απαιτείται για μία εντολή πρόσθεσης κινητής υποδιαστολής είναι 10 + 5 + 20 + 5 = 40 ns. Εάν πάρουμε τη συχνότητα εμφάνισης των εντολών, έχουμε 26% εντολές φόρτωσης, 14% εντολές αποθήκευσης, 31% εντολές τύπου R, 8% εντολές διακλάδωσης, 2% εντολές μεταπήδησης και 19% εντολές κινητής υποδιαστολής (από τις οποίες το 9% είναι πολλαπλασιασμοί και διαιρέσεις και το 10% είναι προσθέσεις, αφαιρέσεις και συγκρίσεις). **Έτσι το μέσο μήκος του κύκλου ρολογιού είναι :**

Κύκλος ρολογιού ΚΜΕ = 40×26% + 35×14% + 30×31% + 25×8%
+ 10×2% + 80×9% + 40×10% = 38.0 ns

Η βελτίωση στην απόδοση είναι :

$$\frac{\textbf{Απόδοση ΚΜΕ με μεταβλητό κύκλο ρολογιού}}{\textbf{Απόδοση ΚΜΕ με ένα μόνο κύκλο ρολογιού}} =$$

$$\frac{\textbf{κύκλος ρολογιού της ΚΜΕ με ένα μόνο κύκλο ρολογιού}}{\textbf{κύκλος ρολογιού της ΚΜΕ με μεταβλητό κύκλο ρολογιού}} = \frac{80}{38} = 2.11$$

Επομένως ένα μεταβλητό ρολόι βελτιώνει την απόδοση περισσότερο από το διπλάσιο.

 Υποθέσαμε πως ο κύκλος ρολογιού ισούται με την χειρότερη περίπτωση καθυστέρησης της εκτέλεσης μιας εντολής. Επομένως πρέπει να χρησιμοποιήσουμε τεχνικές υλοποίησης που να μειώνουν τον χειρότερο κύκλο ρολογιού. Δεν αρκεί δηλαδή να βελτιώσουμε οποιαδήποτε εντολή, αλλά μόνο αυτή που προκαλεί την μεγαλύτερη καθυστέρηση. Έτσι η εφαρμογή ενός κύκλου δεν βελτιώνει την ταχύτητα και κάθε λειτουργική μονάδα μπορεί να χρησιμοποιηθεί μόνο μία φορά σε κάθε κύκλο ρολογιού. Αυτό έχει σαν αποτέλεσμα να πρέπει να χρησιμοποιήσουμε κάποιες λειτουργικές μονάδες περισσότερο από μία φορά, το οποίο αυξάνει το κόστος της εφαρμογής.

Μπορούμε να αποφύγουμε αυτές τις δυσκολίες με τη χρήση μικρότερου κύκλου ρολογιού και αυτό απαιτεί πολλούς κύκλους ρολογιού για κάθε εντολή.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 18

Ποια είναι τα μειονεκτήματα όταν χρησιμοποιούμε υλοποίηση ενός κύκλου ρολογιού και τι συμπεραίνετε σχετικά με την απόδοση. Να αιτιολογήσετε την απάντηση σας και να υποδείξετε εναλλακτικούς τρόπους υλοποίησης.

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 18

Όπως διαπιστώνεται και από τα προηγούμενα παραδείγματα όλες οι εντολές απαιτούν το ίδιο ποσό χρόνου το οποίο είναι ίσο με τον χρόνο της πιο αργής εντολής (συνήθως μια εντολή φόρτωσης). Ο ίδιος χρόνος κύκλου ρολογιού καταναλώνεται και για τύπους εντολών με μικρότερο κύκλο ρολογιού και αυτό έχει σαν αποτέλεσμα την μείωση της απόδοσης του επεξεργαστή.

Ένα άλλο μειονέκτημα στην υλοποίηση ενός κύκλου είναι ότι κάποιες λειτουργικές μονάδες και συγκεκριμένα η μνήμη και η ALU είναι ανάγκη να υπάρχουν περισσότερο από μία φορές στο κύκλωμα του επεξεργαστή πράγμα που σημαίνει αυξημένο κόστος υλοποίησης του.

Ως εναλλακτικοί τρόποι για τη βελτίωση τα απόδοσης του επεξεργαστή μπορούν να αναφερθούν : α) Η χρήση μεταβλητού κύκλου ρολογιού. Η υλοποίηση ενός τέτοιου ρολογιού με διαφορετική ταχύτητα για κάθε εντολή είναι εξαιρετικά δύσκολη διαδικασία και το κόστος μιας τέτοιας εφαρμογής είναι μεγαλύτερο από τα πλεονεκτήματα που αυτή μας προσφέρει. β) Η χρήση πολλών κύκλων για την εκτέλεση μιας εντολής. Μπορούμε να χρησιμοποιήσουμε μικρότερο κύκλο ρολογιού και μια εντολή να εκτελείται σε περισσότερους από έναν κύκλους ρολογιού. Ανάλογα με την εντολή έχουμε και κάποιον αριθμό κύκλων ρολογιού που αυτές απαιτούν για την εκτέλεση τους. Αυτή η τεχνική είναι υλοποιήσιμη και έχει επικρατήσει σήμερα.

Η δίοδος δεδομένων και η μονάδα ελέγχου της εφαρμογής πολλών κύκλων

Στην υποενότητα αυτή θα αναφερθούμε στην υλοποίηση της διόδου δεδομένων και της μονάδας ελέγχου του υπολογιστή MIPS, όταν η εκτέλεση των εντολών χρειάζεται πολλούς κύκλους ρολογιού για να ολοκληρωθεί. Αρχικά θα τροποποιήσουμε κατάλληλα τη δίοδο δεδομένων και τα σήματα ελέγχου, έτσι ώστε να ανταποκρίνονται στην εφαρμογή πολλών κύκλων. Στη συνέχεια θα εξηγήσουμε τον τρόπο με τον οποίο χωρίζονται οι εντολές σε πολλούς κύκλους ρολογιού. Τέλος θα κατασκευάσουμε τη μονάδα ελέγχου χρησιμοποιώντας τις μηχανές πεπερασμένων καταστάσεων.

Η υλοποίηση πολλών κύκλων

Στην προηγούμενη υποενότητα χωρίσαμε κάθε εντολή σε μια σειρά από βήματα σε αντιστοιχία με τις λειτουργίες των μονάδων που είχαμε. Μπορούμε να χρησιμοποιήσουμε αυτά τα βήματα για να δημιουργήσουμε την υλοποίηση πολλών κύκλων.

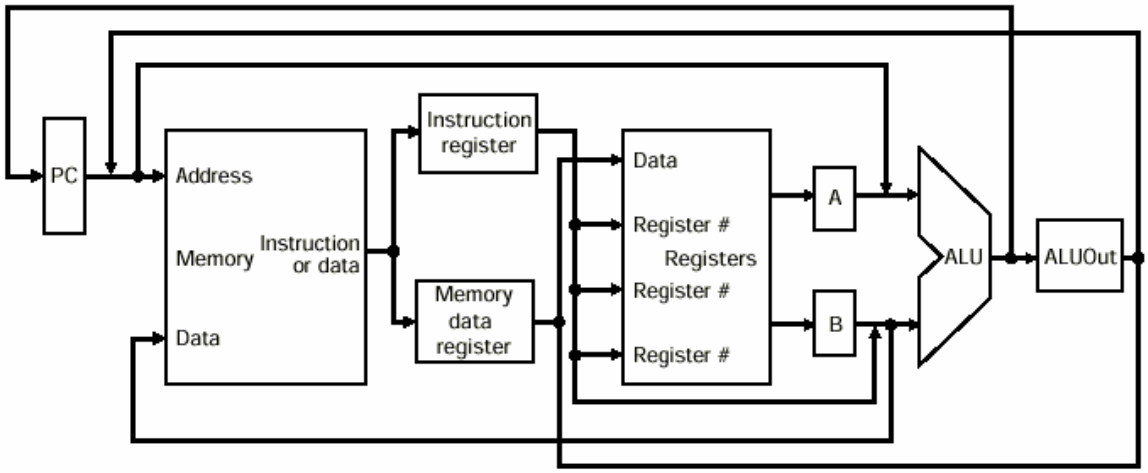
ΔΡΑΣΤΗΡΙΟΤΗΤΑ 19

Μπορείτε να αναφέρετε τους λόγους για τους οποίους η εφαρμογή πολλών κύκλων υπερέχει της εφαρμογής του ενός κύκλου;

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 19

Στην υλοποίηση πολλών κύκλων, για την εκτέλεση κάθε βήματος, απαιτείται ένας κύκλος ρολογιού. Αυτή η υλοποίηση επιτρέπει σε μια λειτουργική μονάδα να χρησιμοποιείται περισσότερο από μία φορά ανά εντολή, για όσο χρόνο χρειάζεται στους διαφορετικούς κύκλους ρολογιού. Αυτό βοηθάει στη μείωση των κυκλωμάτων που απαιτούνται. Η δυνατότητα να χρησιμοποιούν οι εντολές διαφορετικό αριθμό κύκλων ρολογιού και να μοιράζονται τις λειτουργικές μονάδες κατά τη διάρκεια της εκτέλεσης μιας εντολής, είναι τα βασικά πλεονεκτήματα της σχεδίασης με πολλούς κύκλους ρολογιού.

Το σχήμα 3.2.28 απεικονίζει ένα τμήμα της διόδου δεδομένων της υλοποίησης πολλών κύκλων.



Σχήμα 3.2.28 - Ένα τμήμα της διόδου δεδομένων της εφαρμογής πολλών κύκλων. Στο σχήμα φαίνονται: η μονάδα μνήμης, και η ALU της οποίας διαμοιράζονται οι εντολές και οι δίοδοι δεδομένων οι οποίες συνδέουν τις διαμοιραζόμενες αυτές μονάδες.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 20

Να συγκρίνετε το σχήμα 3.2.28 με το σχήμα 3.2.14. Ποιες διαφορές παρατηρείτε σε αυτά τα δύο σχήματα;

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 20

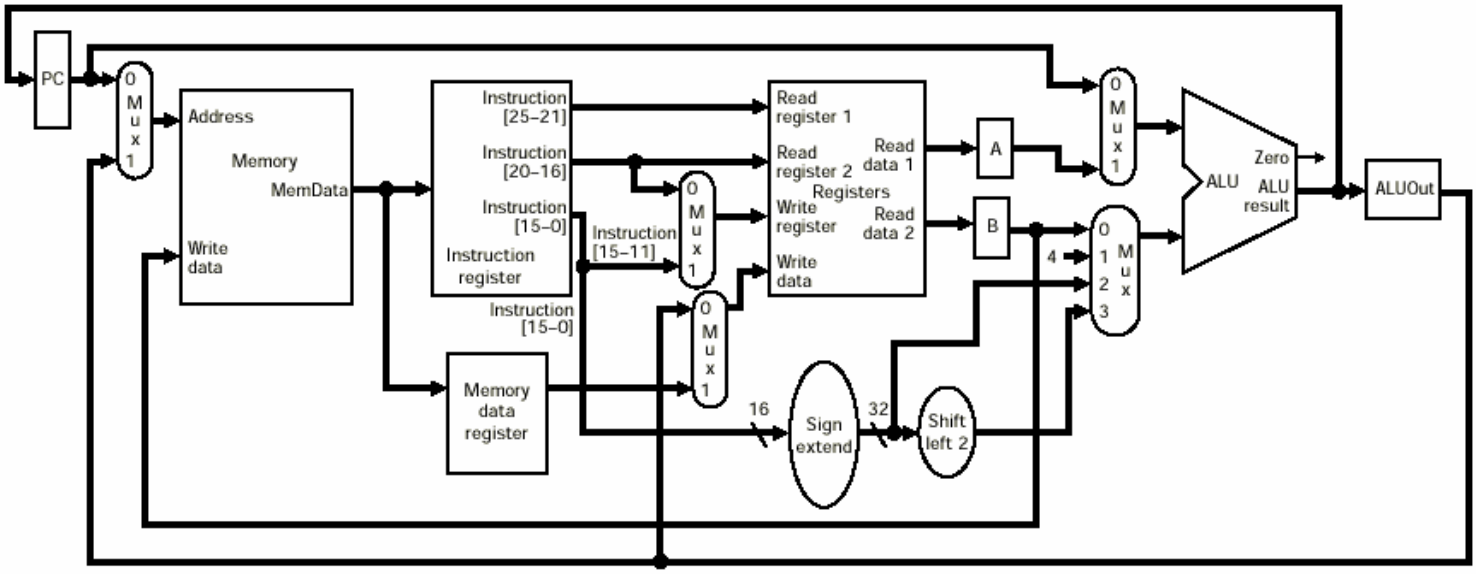
Εάν συγκρίνουμε το σχήμα 3.2.28 με το σχήμα 3.2.14 παρατηρούμε τις εξής διαφορές:

- **Μία μόνο μονάδα μνήμης χρησιμοποιείται από τις εντολές και τα δεδομένα.**
- **Ένας καταχωρητής χρησιμοποιείται για φύλαξη της εντολής μετά την ανάγνωσή της. Αυτός ο καταχωρητής εντολής (Instruction Register - IR), χρειάζεται επειδή η μνήμη μπορεί χρησιμοποιηθεί ξανά αργότερα για την προσπέλαση δεδομένων κατά τη διάρκεια εκτέλεσης μιας εντολής.**
- **Υπάρχει μόνο μία ALU, αντί για μία ALU και δύο αθροιστές.**

Επειδή οι διάφορες λειτουργικές μονάδες διαμοιράζονται για διαφορετικούς λόγους, χρειάζεται να προσθέτουμε πολυπλέκτες και να επεκτείνουμε τους πολυπλέκτες που ήδη υπάρχουν. Αφού η μνήμη χρησιμοποιείται για τις εντολές και τα δεδομένα, για την διεύθυνση της μνήμης χρειαζόμαστε ένα πολυπλέκτη, για να επιλέγει ανάμεσα στον απαριθμητή προγράμματος (για την προσπέλαση των εντολών) και στο αποτέλεσμα της ALU (για την προσπέλαση των δεδομένων).

Για τον διαμοιρασμό της ALU απαιτείται η εισαγωγή ενός πολυπλέκτη στην πρώτη είσοδο της ALU, η οποία μπορεί να είναι είτε ένας καταχωρητής, είτε ο απαριθμητής προγράμματος. Επίσης χρειάζεται μια τροποποίηση στον πολυπλέκτη στην δεύτερη είσοδο της ALU, έτσι ώστε από δύο να έχει τέσσερις εισόδους. Για την μετατροπή του πολυπλέκτη χρειάζονται δύο επιπλέον είσοδοι: η σταθερά 4 (που χρησιμοποιείται στην αύξηση του απαριθμητή προγράμματος) και το πεδίο offset μετά από επέκταση προσήμου και ολίσθηση (που χρησιμοποιείται στον υπολογισμό του στόχου διακλάδωσης).

Το σχήμα 3.2.29 απεικονίζει τη δίοδο δεδομένων με τους επιπλέον πολυπλέκτες. Με την είσοδο ενός καταχωρητή και τριών πολυπλεκτών, μπορούμε να μειώσουμε τις μονάδες μνήμης από δύο σε μία και να αφαιρέσουμε δύο αθροιστές. Αφού οι καταχωρητές και οι πολυπλέκτες είναι μικρά κυκλώματα, αυτό μπορεί να αποφέρει σημαντική μείωση στο κόστος κατασκευής.

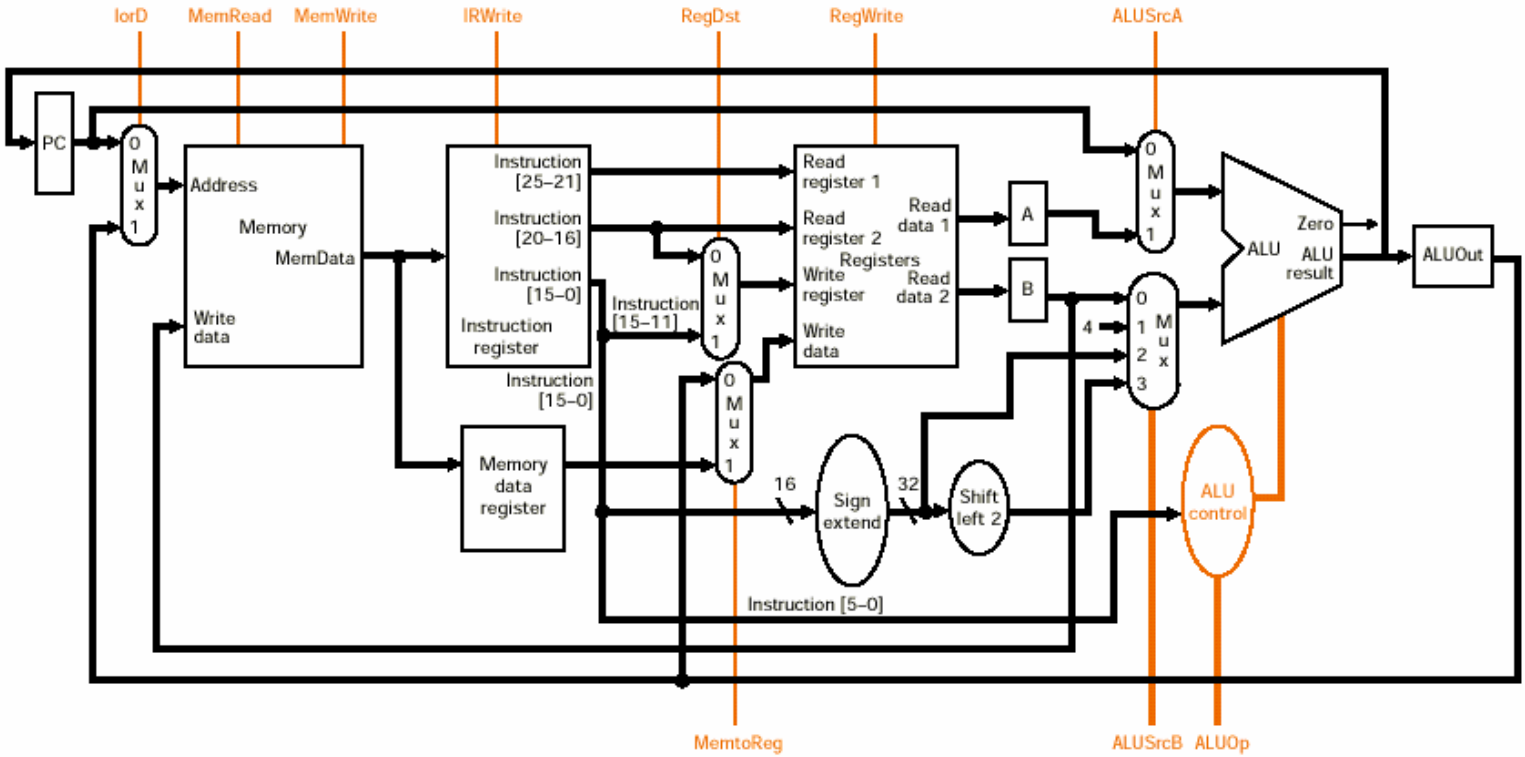


Σχήμα 3.2.29 - Η δίοδος δεδομένων πολλών καταστάσεων του υπολογιστή MIPS για τον χειρισμό όλων των βασικών εντολών. Οι προσθήκες που έχουν γίνει σε σχέση με την δίοδο δεδομένων ενός κύκλου, περιλαμβάνουν έναν πολυπλέκτη για την διεύθυνση της μνήμης, έναν πολυπλέκτη για την πρώτη είσοδο της ALU, και μία επέκταση του πολυπλέκτη στην δεύτερη είσοδο της ALU. Αυτές οι προσθήκες μας επιτρέπουν να αφαιρέσουμε δύο αθροιστές και μία μονάδα μνήμης.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 21

Να συμπληρώσετε τις γραμμές ελέγχου στο σχήμα 3.2.29 της διόδου δεδομένων του υπολογιστή MIPS. Να συγκρίνετε το σχήμα που σχεδιάσατε με αυτό που ακολουθεί (σχήμα 3.2.30).
(Υπόδειξη: Τα σήματα ελέγχου είναι: MemRead, MemWrite, ALUSrcA, RegDst, RegWrite, MemtoReg, IorD, IRWrite, ALUSrcB και ALUOp.)

Επειδή η δίοδος δεδομένων του σχήματος 3.2.29 χρειάζεται πολλούς κύκλους ρολογιού ανά εντολή, πρέπει τα σήματα ελέγχου να ενεργοποιούνται με διαφορετικό τρόπο. Θα χρειαστούμε ένα σήμα εγγραφής για κάθε στοιχείο μνήμης: τη μνήμη, τον απαριθμητή προγράμματος, τους καταχωρητές γενικής χρήσης και τον καταχωρητή εντολών, καθώς και ένα σήμα ανάγνωσης (για τη μνήμη). Μπορούμε να χρησιμοποιήσουμε τη μονάδα ελέγχου της ALU από προηγούμενα παραδείγματα (στα σχήματα 3.2.17 και 3.2.18), για τον έλεγχο της ALU. Τέλος καθένας από τους πολυπλέκτες με τις δύο εισόδους χρειάζεται μία γραμμή ελέγχου και οι πολυπλέκτες με τις τέσσερις γραμμές ελέγχου, χρειάζονται δύο γραμμές ελέγχου. Το σχήμα 3.2.30 απεικονίζει τη δίοδο δεδομένων του σχήματος 3.2.29 μαζί με τις γραμμές ελέγχου.



Σχήμα 3.2.30 - Η δίοδος δεδομένων της εφαρμογής πολλών κύκλων με τις γραμμές ελέγχου. Τα σήματα ALUOp και ALUSrcB είναι σήματα ελέγχου των δύο bits, ενώ όλες οι

άλλες γραμμές ελέγχου είναι σήματα του ενός bit. Το σήμα MemRead έχει μεταφερθεί στην κορυφή της μονάδας μνήμης για απλοποίηση του σχήματος.


Πριν εξετάσουμε τα βήματα για την εκτέλεση κάθε εντολής, είναι χρήσιμο να δούμε την επίδραση των σημάτων ελέγχου που έχουν προστεθεί, όταν αυτά είναι ή δεν είναι ενεργά (όπως ακριβώς κάναμε και στην εφαρμογή ενός κύκλου στο σχήμα 3.2.21). Αυτό φαίνεται στο σχήμα 3.2.31. Τα σήματα ελέγχου του ενός bit φαίνονται στον πίνακα a του σχήματος, ενώ τα σήματα ελέγχου ALUSrcB και ALUOp των δύο bits φαίνονται στον πίνακα b του σχήματος.

Σήμα ελέγχου	**Επίδραση όταν δεν είναι ενεργά**	**Επίδραση όταν είναι ενεργά**
MemRead	Τίποτα	Το περιεχόμενο της μνήμης που βρίσκεται στην διεύθυνση που έχει δοθεί για ανάγνωση, τοποθετείται στην έξοδο ανάγνωσης δεδομένων.
MemWrite	Τίποτα	Το περιεχόμενο της μνήμης που βρίσκεται στην διεύθυνση που έχει δοθεί για εγγραφή, αντικαθίσταται από την τιμή που βρίσκεται στην είσοδο εγγραφής των δεδομένων.
ALUSrcA	Ο πρώτος τελεστέος της ALU είναι ο απαριθμητής προγράμματος.	Ο πρώτος τελεστέος της ALU προέρχεται από τον καταχωρητή που δίνεται από το πεδίο rs.
RegDst	Ο αριθμός του καταχωρητή προορισμού στον οποίο θα γίνει η εγγραφή προέρχεται από το πεδίο rt.	Ο αριθμός του καταχωρητή προορισμού στον οποίο θα γίνει εγγραφή προέρχεται από το πεδίο rd.
RegWrite	Τίποτα	Στον καταχωρητή που προσδιορίζεται από την τιμή που έχει δοθεί στην είσοδο εγγραφή καταχωρητή, εγγράφεται η τιμή που έχει δοθεί στην είσοδο εγγραφής δεδομένων.
MemtoReg	Η τιμή που δίνεται στην είσοδο εγγραφή καταχωρητή προέρχεται από την ALU.	Η τιμή που δίνεται στην είσοδο εγγραφή καταχωρητή προέρχεται από την μνήμη δεδομένων.
IorD	Ο απαριθμητής προγράμματος χρησιμοποιείται για να δώσει την διεύθυνση στην μονάδα μνήμης.	Η έξοδος της ALU χρησιμοποιείται για να δώσει την διεύθυνση στην μονάδα μνήμης.
IRWrite	Τίποτα	Η τιμή από την μονάδα μνήμης γράφεται στον καταχωρητή εντολών (Instruction Register-IR).

a. Τα αποτελέσματα των σημάτων ελέγχου του ενός bit.

Σήμα ελέγχου	Τιμή σήματος	Αποτέλεσμα
ALUSrcB	00	Η δεύτερη είσοδος της ALU προέρχεται από τον καταχωρητή που δίνεται από το πεδίο rt.
	01	Η δεύτερη είσοδος της ALU είναι η σταθερά 4.
	10	Η δεύτερη είσοδος της ALU είναι τα 16 λιγότερο σημαντικά ψηφία του καταχωρητή εντολών, μετά από επέκταση προσήμου.
	11	Η δεύτερη είσοδος της ALU είναι τα 16 λιγότερο σημαντικά ψηφία του καταχωρητή εντολών, μετά από επέκταση προσήμου και ολίσθηση.
ALUOp	00	Η ALU εκτελεί μία πρόσθεση.
	01	Η ALU εκτελεί μία αφαίρεση.
	10	Το πεδίο function της εντολής καθορίζει την λειτουργία της ALU.

b. Τα αποτελέσματα των σημάτων ελέγχου των δύο bits.

Σχήμα 3.2.31. Τα αποτελέσματα από την ενεργοποίηση του κάθε σήματος ελέγχου. Μόνο οι γραμμές ελέγχου που επηρεάζουν τους πολυπλέκτες επιτελούν κάποια λειτουργία όταν είναι 0. Το παραπάνω σχήμα είναι παρόμοιο με το σχήμα 3.2.21, με τη διαφορά ότι έχουν προστεθεί οι καινούριες γραμμές ελέγχου (ALUSrcA, IorD, IRWrite, και ALUSrcB) και έχουν αφαιρεθεί οι γραμμές ελέγχου που δεν χρησιμοποιούνται πια, ή έχουν αντικατασταθεί (για τις εντολές μεταπήδησης, διακλάδωσης και το ALUSrc).

Για την μείωση του αριθμού των γραμμών ελέγχου που συνδέονται στις λειτουργικές μονάδες, οι σχεδιαστές χρησιμοποιούν διαμοιραζόμενες αρτηρίες. Μια διαμοιραζόμενη αρτηρία είναι μια συλλογή γραμμών οι οποίες συνδέουν πολλές μονάδες μεταξύ τους. Στις περισσότερες περιπτώσεις, περιλαμβάνουν πολλές πηγές οι οποίες τοποθετούν τα δεδομένα στην αρτηρία και μπορούν να διαβαστούν από πολλές λειτουργικές μονάδες. Έτσι, όπως μειώσαμε τις λειτουργικές μονάδες της διόδου δεδομένων, μπορούμε να μειώσουμε των αριθμό των αρτηριών που συνδέονται στις μονάδες, χρησιμοποιώντας διαμοιρασμό των αρτηριών. Για παράδειγμα, εάν υπάρχουν πέντε πηγές προς την είσοδο της ALU, μόνο δύο από αυτές χρειάζονται ανά πάσα στιγμή. Επομένως ένα ζευγάρι αρτηριών μπορεί να χρησιμοποιηθεί για να κρατάει τις τιμές που στέλνονται στην ALU. Αντί να τοποθετήσει έναν μεγάλο πολυπλέκτη στην είσοδο της ALU, ο σχεδιαστής μπορεί να χρησιμοποιήσει μια διαμοιραζόμενη αρτηρία και μετά να εξασφαλίσει ότι μόνο μία από τις πηγές χρησιμοποιεί την αρτηρία.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 22

Να εξηγήσετε τι είναι οι διαμοιραζόμενες αρτηρίες και το λόγο για τον οποίο τις χρησιμοποιούν οι σχεδιαστές.

Πως χωρίζεται η εκτέλεση των εντολών σε κύκλους ρολογιού

Δεδομένης της διόδου δεδομένων του σχήματος 3.2.30, πρέπει να εξετάσουμε τι συμβαίνει σε κάθε κύκλο ρολογιού στην εκτέλεση των εντολών με πολλούς κύκλους, αφού αυτό καθορίζει τα επιπλέον κυκλώματα της διόδου δεδομένων (για παράδειγμα τους προσωρινούς καταχωρητές) και τα επιπλέον σήματα ελέγχου που πιθανόν να χρειαστούν.

Αρχικά πρέπει να προσθέσουμε έναν καταχωρητή για να κρατάει την τιμή του σήματος για όσο ισχύουν οι δύο παρακάτω συνθήκες:

1. Το σήμα υπολογίζεται σε ένα κύκλο ρολογιού και χρησιμοποιείται σε έναν άλλο κύκλο ρολογιού.
2. Το κύκλωμα που έχει σαν έξοδο το σήμα, μπορεί να αλλάξει πριν την εγγραφή του σήματος σε ένα στοιχείο μνήμης.

Παραδείγματος χάριν, χρειάζεται να αποθηκεύσουμε την εντολή στον καταχωρητή εντολών, επειδή η μνήμη η οποία παράγει την τιμή, αλλάζει την έξοδο πριν ολοκληρωθεί η χρήση όλων των πεδίων της εντολής. Απ' την άλλη πλευρά, όταν χρησιμοποιείται η ALU στις εντολές τύπου R, δεν χρειάζεται να αποθηκεύσουμε την έξοδο, ακόμα και αν δεν χρησιμοποιήσουμε την έξοδο μέχρι τον επόμενο κύκλο ρολογιού. Αυτό συμβαίνει επειδή η έξοδος της ALU δεν αλλάζει (είναι σταθερή) κατά την διάρκεια του κύκλου ρολογιού, όταν γράφεται στο αρχείο καταχωρητών. Η έξοδος της ALU είναι σταθερή επειδή οι είσοδοί της που προέρχονται από το αρχείο καταχωρητών και η έξοδος του αρχείου καταχωρητών, καθορίζονται από τα πεδία rs και rt του καταχωρητή εντολών, ο οποίος είναι σταθερός επειδή είναι ένα στοιχείο μνήμης που εγγράφεται μόνο μία φορά κατά την διάρκεια εκτέλεσης μιας εντολής. Έτσι οι λειτουργικές μονάδες, από το αρχείο καταχωρητών μέχρι και την ALU, αποτελούν ένα τμήμα συνδυαστικής λογικής, του οποίου οι είσοδοι προέρχονται από τον καταχωρητή εντολών (ο οποίος είναι στοιχείο μνήμης), και οι έξοδοι γράφονται στο αρχείο καταχωρητών (που είναι επίσης στοιχείο μνήμης). Η δομή αυτή μοιάζει με τη δομή του σχήματος 3.2.2. Παρ' όλο που η υλοποίηση ενός κύκλου χρησιμοποιούσε πάντα στοιχεία μνήμης στα οποία γινόταν εγγραφή σε κάθε κύκλο ρολογιού (όπως στο σχήμα 3.2.3), η υλοποίηση πολλών κύκλων θα κάνει εγγραφή στα στοιχεία μνήμης επιλεκτικά, όπως στο σχήμα 3.2.2.

Ο στόχος μας για να χωρίσουμε την εκτέλεση των εντολών σε κύκλους ρολογιού, είναι να ισορροπηθεί ο αριθμός των λειτουργιών που εκτελούνται σε ένα κύκλο ρολογιού, έτσι ώστε να μειωθεί ο χρόνος κύκλου ρολογιού. Μπορούμε να χωρίσουμε την εκτέλεση των εντολών σε πέντε βήματα, κάθε ένα από τα οποία να χρειάζεται ένα κύκλο ρολογιού, περίπου του ίδιου μήκους. Για παράδειγμα, θα περιορίσουμε κάθε βήμα έτσι ώστε να περιέχει το πολύ μία λειτουργία της ALU, ή μία προσπέλαση στο αρχείο καταχωρητών, ή μία

προσπέλαση στη μνήμη. Με αυτόν τον περιορισμό ο κύκλος ρολογιού μπορεί να είναι τόσο μικρός όσο η μεγαλύτερη από αυτές τις λειτουργίες.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 23

Σύμφωνα με τα παραπάνω θα περιορίσουμε κάθε βήμα εκτέλεσης μιας εντολής, έτσι ώστε να περιέχει το πολύ μία λειτουργία της ALU, ή μία προσπέλαση στο αρχείο καταχωρητών, ή μία προσπέλαση στη μνήμη. Μπορείτε να απαριθμήσετε τα βήματα στα οποία θα χωρίσουμε την εκτέλεση των εντολών;

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 23

Τα πέντε βήματα εκτέλεσης των εντολών είναι:
1. Ανάκληση της εντολής.
2. Αποκωδικοποίηση της εντολής και ανάκληση των καταχωρητών.
3. Εκτέλεση της εντολής, υπολογισμός της διεύθυνσης μνήμης ή ολοκλήρωση της εντολής διακλάδωσης με συνθήκη.
4. Πρόσβαση στη μνήμη ή ολοκλήρωση των εντολών τύπου R.
5. Επανεγγραφή (ή εγγραφή δεδομένων).

 Σε μία δίοδο δεδομένων ενός κύκλου η κάθε εντολή πρέπει να χρησιμοποιεί ένα σύνολο κυκλωμάτων για την εκτέλεσή της. Πολλά από τα κυκλώματα της διόδου δεδομένων λειτουργούν σε σειρά, χρησιμοποιώντας σαν είσοδο, την έξοδο κάποιου άλλου κυκλώματος. Κάποια άλλα κυκλώματα της διόδου δεδομένων λειτουργούν παράλληλα. Για παράδειγμα, η αύξηση του απαριθμητή προγράμματος και η ανάγνωση της εντολής γίνονται ταυτόχρονα. Κάτι παρόμοιο γίνεται και στην δίοδο δεδομένων πολλών κύκλων. Όλες οι λειτουργίες που γίνονται σε ένα βήμα, γίνονται παράλληλα σε ένα κύκλο ρολογιού, ενώ τα διαδοχικά βήματα λειτουργούν σε σειρά και σε διαφορετικούς κύκλους ρολογιού. Ο περιορισμός σε μία λειτουργία της ALU, ή σε μία προσπέλαση του αρχείου καταχωρητών, ή σε μία προσπέλαση της μνήμης, καθορίζει τη λειτουργία που μπορεί να γίνει σε ένα βήμα.

Παρακάτω δίνονται τα πέντε βήματα και η δράση τους.

1. **<u>Βήμα ανάκλησης της εντολής :</u>**

Ανάκληση της εντολής από τη μνήμη και αύξηση του απαριθμητή προγράμματος (PC).

IR = Memory[PC];

PC = PC + 4;

- **Λειτουργία:** Η διεύθυνση (που είναι το περιεχόμενο του απαριθμητή προγράμματος) στέλνεται στη μνήμη, εκτελείται ανάγνωση και ανάκληση της εντολής στον καταχωρητή εντολών (Instruction Register - IR), οπού θα γίνει η αποθήκευση. Για την υλοποίηση αυτού του βήματος θα χρειαστεί να ενεργοποιήσουμε τα σήματα ελέγχου MemRead και IRWrite και να θέσουμε το IorD στο 0 για να επιλέξει τον απαριθμητή προγράμματος σαν πηγή για την διεύθυνση. Επίσης σε αυτό το στάδιο αυξάνουμε τον απαριθμητή προγράμματος κατά 4, το οποίο απαιτεί να είναι το σήμα ALUSrcB 01, το σήμα ALUSrcA 0 και το ALUOp 00 (για να γίνει η πρόσθεση στην ALU). Τέλος θέλουμε να αποθηκεύσουμε την επαυξημένη διεύθυνση μνήμης στον απαριθμητή προγράμματος. Θα προσθέσουμε αυτό το τμήμα της διόδου δεδομένων και τα σήματα ελέγχου, αφού έχουμε καθορίσει τη μονάδα ελέγχου για τον απαριθμητή προγράμματος, μαζί με τις εντολές διακλάδωσης. Η αύξηση του απαριθμητή προγράμματος και η προσπέλαση της μνήμης εντολών, μπορούν να γίνονται παράλληλα.

2. Βήμα αποκωδικοποίησης της εντολής και ανάκλησης των καταχωρητών:

Στο προηγούμενο βήμα, καθώς και σε αυτό, δεν γνωρίζουμε το είδος της εντολής και παρουσιάζουμε ενέργειες που εφαρμόζονται σε όλες τις εντολές (όπως η ανάκληση της εντολής στο βήμα 1), ή δεν επηρεάζουν τη διαδικασία, στην περίπτωση που η εντολή δεν είναι αυτή στην οποία αναφερόμαστε. Έτσι σε αυτό το βήμα μπορούμε να διαβάσουμε τους δύο καταχωρητές που καθορίζουν τα πεδία rs και rt, αφού η ανάγνωση των καταχωρητών δεν είναι επιζήμια, ακόμα και αν δεν χρειάζεται. Τα περιεχόμενα των καταχωρητών μπορεί να χρειαστούν σε επόμενα στάδια, έτσι τα ονομάζουμε A και B στην παρακάτω περιγραφή. Οι έξοδοι των καταχωρητών δεν χρειάζεται να σωθούν σε προσωρινό καταχωρητή, αφού οι είσοδοι στο αρχείο καταχωρητών (επομένως και τα δεδομένα στις εξόδους), δεν αλλάζουν κατά την διάρκεια εκτέλεσης της εντολής.

Θα υπολογίσουμε επίσης την διεύθυνση του στόχου διακλάδωσης στην ALU, το οποίο επίσης δεν επηρεάζει την διαδικασία, επειδή μπορούμε να αγνοήσουμε την τιμή εάν δεν έχουμε εντολή διακλάδωσης. Επειδή δε γνωρίζουμε εάν έχουμε εντολή διακλάδωσης με συνθήκη και επειδή χρειάζεται να χρησιμοποιήσουμε την ALU στα επόμενα βήματα, πρέπει να σώσουμε την διεύθυνση του στόχου διακλάδωσης που υπολογίσαμε, σε έναν καινούριο καταχωρητή, τον οποίο ονομάζουμε Στόχο (Target).

Εκτελώντας αυτές τις λειτουργίες νωρίτερα, έχουμε το πλεονέκτημα της μείωσης του αριθμού κύκλων ρολογιού που απαιτούνται για την εκτέλεση μιας εντολής. Μπορούμε να κάνουμε αυτές τις ενέργειες εξαιτίας της δομής των εντολών. Για παράδειγμα, εάν η εντολή έχει δύο καταχωρητές εισόδου, είναι πάντα τα πεδία rs και rt και αν έχουμε εντολή διακλάδωσης με συνθήκη, το offset είναι πάντα τα 16 λιγότερο σημαντικά ψηφία:

A = Register[IR[25 – 21]];

B = Register[IR[20 - 16]];

Target = PC + (sign-extend (IR[15 - 0]) << 2);

- **Λειτουργία :** Προσπέλαση του αρχείου καταχωρητών για την ανάγνωση των καταχωρητών με τη χρήση των πεδίων rs και rt. Δεν απαιτείται ενεργοποίηση των γραμμών ελέγχου. Υπολογισμός της διεύθυνσης του στόχου διακλάδωσης και αποθήκευση αυτής στον καταχωρητή Στόχο (Target). Αυτό απαιτεί το σήμα ALUSrcB να είναι 11 (έτσι ώστε να γίνει επέκταση προσήμου και ολίσθηση στο πεδίο offset), το σήμα ALUSrcA να είναι 0 (έτσι ώστε να πάρει την τιμή του απαριθμητή προγράμματος) και το σήμα ALUOp να είναι 00 (έτσι ώστε να γίνει πρόσθεση στην ALU). Επιπλέον πρέπει να προσθέσουμε μία γραμμή ελέγχου εγγραφής για τον καταχωρητή Στόχο, η οποία πρέπει να είναι ενεργοποιημένη κατά τη διάρκεια αυτού του βήματος. Οι προσπελάσεις των καταχωρητών και ο υπολογισμός της διεύθυνσης του στόχου διακλάδωσης γίνονται παράλληλα. Μετά από αυτόν τον κύκλο ρολογιού η λειτουργία που θα γίνει εξαρτάται από τον κωδικό της κάθε εντολής.

3. Βήμα εκτέλεσης εντολής, υπολογισμού της διεύθυνσης μνήμης ή ολοκλήρωσης της εντολής διακλάδωσης:

Αυτός είναι ο πρώτος κύκλος ρολογιού στον οποίο η λειτουργία της διόδου δεδομένων καθορίζεται από τον τύπο της εντολής. Σε όλες τις περιπτώσεις η ALU χρησιμοποιεί τους τελεστέους του προηγούμενου βήματος, εκτελώντας μία από τις τρεις λειτουργίες της, ανάλογα με τον τύπο της εντολής. Το αποτέλεσμα της ALU το ονομάζουμε ALUoutput για να το χρησιμοποιήσουμε σε επόμενα στάδια. Αφού οι είσοδοι της ALU είναι σταθερές, η τιμή αυτή δεν χρειάζεται να σωθεί σε κάποιον καταχωρητή. Ωστόσο τα σήματα που ενεργοποιούνται σε αυτό τον κύκλο ρολογιού και επηρεάζουν το αποτέλεσμα της ALU, πρέπει να διατηρηθούν σταθερά μέχρι τα αποτελέσματα της ALU να εγγραφούν σε έναν καταχωρητή, ή μέχρι να μην χρειάζονται πια.

Παρακάτω καθορίζονται οι λειτουργίες που γίνονται ανάλογα με τον τύπο της κάθε εντολής :

- **Αναφορά στη μνήμη:**

ALUoutpu = A + sign-extend (IR[15-0]);

- **Λειτουργία :** Η ALU προσθέτει τους τελεστέους για να σχηματιστεί η διεύθυνση μνήμης. Αυτό απαιτεί το σήμα ALUSrcA να είναι 1, οπότε χρησιμοποιείται η πρώτη έξοδος του αρχείου καταχωρητών σαν πρώτη είσοδος της ALU. Επίσης πρέπει το σήμα ALUSrcB να είναι 10, οπότε χρησιμοποιείται η μονάδα επέκτασης προσήμου σαν τη δεύτερη είσοδο της ALU. Το σήμα ALUOp πρέπει να είναι 00, έτσι ώστε η ALU να εκτελέσει πρόσθεση.

- **Αριθμητικές και λογικές εντολές (τύπου R):**

ALUoutput = A op B ;

- **Λειτουργία :** Η ALU εκτελεί τη λειτουργία που προσδιορίστηκε από τον κωδικό λειτουργίας, στους δύο καταχωρητές που έγινε ανάγνωση στον προηγούμενο κύκλο ρολογιού. Αυτό απαιτεί το σήμα ALUSrcA να είναι 1, και το σήμα ALUSrcB να είναι 00. Με αυτή την ενεργοποίηση των σημάτων, οι έξοδοι του αρχείου καταχωρητών χρησιμοποιούνται σαν είσοδοι της ALU. Το σήμα ALUOp πρέπει να είναι 10, έτσι ώστε ο κωδικός λειτουργίας να καθορίζει την ενεργοποίηση του κατάλληλου σήματος ελέγχου της ALU.

- **Εντολές διακλάδωσης με συνθήκη:**

if (A = = B) PC = Target ;

- **Λειτουργία :** Η ALU συγκρίνει τα περιεχόμενα των δύο καταχωρητών (εάν είναι ίσα), στους οποίους έγινε ανάγνωση στο προηγούμενο βήμα. Η έξοδος του σήματος 0 της ALU καθορίζει πότε θα εκτελεστεί ή όχι η εντολή διακλάδωσης με συνθήκη. Αυτό απαιτεί το σήμα ALUSrcA να είναι 1 και το σήμα ALUSrcB να είναι 00, όπως ακριβώς στις εντολές τύπου R. Τα σήματα ALUOp πρέπει να είναι 01, έτσι ώστε να εκτελείται αφαίρεση, η οποία χρησιμοποιείται για τον έλεγχο της ισότητας. Επίσης χρειάζεται να ενεργοποιηθεί ένα σήμα ελέγχου εγγραφής για την ενημέρωση του απαριθμητή προγράμματος, αν η έξοδος 0 της ALU είναι ενεργοποιημένη. Αυτό θα καθοριστεί αργότερα, όταν προσθέσουμε την μονάδα ελέγχου του απαριθμητή προγράμματος.

4. Βήμα πρόσβασης στη μνήμη ή ολοκλήρωσης των εντολών τύπου R:

Κατά την διάρκεια αυτού του βήματος, εγγράφονται τα αποτελέσματα της προσπέλασης της μνήμης για τις εντολές φόρτωσης και αποθήκευσης, καθώς και τα αποτελέσματα των αριθμητικών και λογικών λειτουργιών. Ονομάζουμε την έξοδο της μνήμης memory-data (δεδομένα μνήμης), αν και δεν σχετίζεται με κάποιον καταχωρητή, αφού η έξοδος είναι σταθερή κατά την διάρκεια του επόμενου κύκλου ρολογιού, όταν γίνεται εγγραφή της εξόδου αυτής σε έναν καταχωρητή. Η έξοδος αυτή της μνήμης τοποθετείται στην αρτηρία δεδομένων που συνδέει την ΚΜΕ με τη μνήμη.

- **Αναφορά στη μνήμη:**

memory-data = Memory [ALUoutput] ;

or

Memory [ALUoutput] = B ;

- **Λειτουργία :** Για τις εντολές φόρτωσης κατά τη διάρκεια αυτού του βήματος, στέλνεται από την ΚΜΕ προς τη μνήμη η διεύθυνση (μέσω της αρτηρίας διευθύνσεων που συνδέει την ΚΜΕ με τη μνήμη), και η μνήμη τοποθετεί τα δεδομένα που περιέχονται σ' αυτή τη διεύθυνση (memory-data) στην αρτηρία δεδομένων. Επιπλέον για τις εντολές αποθήκευσης, η τιμή που θα αποθηκευτεί μέσω της αρτηρία δεδομένων στη μνήμη, περιέχεται στον καταχωρητή Β. (Ο καταχωρητής Β έχει πάρει τιμή στον δεύτερο κύκλο ρολογιού.) Στις δύο αυτές περιπτώσεις, η διεύθυνση που χρησιμοποιείται είναι αυτή που υπολογίστηκε στο προηγούμενο βήμα και βρίσκεται στην έξοδο της ALU (ALUoutput). Η ενεργοποίηση των σημάτων ελέγχου της ALU πρέπει να παραμείνει σταθερή κατά την διάρκεια αυτού του κύκλου ρολογιού. Το σήμα MemRead (για μία εντολή φόρτωσης), ή το σήμα MemWrite (για μία εντολή αποθήκευσης), πρέπει να είναι ενεργό. Επιπλέον το σήμα IorD πρέπει να είναι 1, έτσι ώστε η διεύθυνση της μνήμης να προέρχεται από την ALU και όχι από τον απαριθμητή προγράμματος.

- **Αριθμητικές και λογικές εντολές (τύπου R):**

Reg[IR[15-11]] = ALUoutput ;

- **Λειτουργία :** Το αποτέλεσμα της λειτουργίας της ALU τοποθετείται στον καταχωρητή προορισμού. Το σήμα RegDst πρέπει να είναι 1 έτσι ώστε το πεδίο rd (που είναι τα bits 15-11), να επιλέξει τον καταχωρητή στον οποίο θα γίνει εγγραφή. Το σήμα RegWrite πρέπει να είναι ενεργό και το σήμα MemtoReg πρέπει να είναι 0 ώστε να εγγραφεί η έξοδος της ALU. Τα σήματα ALUSrcA, ALUSrcB και ALUOp παραμένουν όπως στον προηγούμενο κύκλο ρολογιού.

5. **Βήμα επανεγγραφής :**

Reg[IR[20-16]] = memory-data ;

- **Λειτουργία :** Γίνεται εγγραφή των δεδομένων της φόρτωσης από την μνήμη στο αρχείο καταχωρητών. Σε αυτό το βήμα το σήμα MemtoReg πρέπει να είναι 1, για να γίνει εγγραφή του αποτελέσματος από τη μνήμη, και το σήμα RegWrite, για να προκαλέσει μία εγγραφή. Επίσης το σήμα RegDst πρέπει να είναι 0, για την

επιλογή του πεδίου rt (που είναι τα bits 20-16), ώστε το πεδίο να είναι ο αριθμός του καταχωρητή. Τα σήματα ALUSrcA, ALUSrcB και ALUOp πρέπει να μείνουν σταθερά μέχρι το τέλος αυτού του κύκλου ρολογιού.

Η σειρά των πέντε αυτών βημάτων συνοψίζεται στο σχήμα 3.2.32.

Βήμα	**Εντολές τύπου R**	**Εντολές αναφοράς στη μνήμη**	**Εντολές διακλάδωσης με συνθήκη**
Ανάκληση εντολής	IR = Memory[PC] PC = PC + 4 ;		
Αποκωδικοποίηση εντολής / ανάκληση καταχωρητών	A = Registers[IR[25-21]] B = Registers[IR[20-16]] Target = PC + (sign-extend(IR[15-0]) << 2)		
Εκτέλεση εντολών, υπολογισμός διεύθυνσης, ή ολοκλήρωση των εντολών διακλάδωσης	ALUoutput = A op B	ALUoutput = A + sign-extend(IR[15-0])	if (A = = B) then PC = Target
Προσπέλαση της μνήμης ή ολοκλήρωση των εντολών τύπου R	Reg[IR[15-11]] = ALUoutput	memory-data = Memory[ALUouput] or Memory[ALUouput] = B	
Επανεγγραφή		Reg[IR[20-16]] = memory-data	

Σχήμα 3.2.32 - Τα βήματα εκτέλεσης των διαφορετικών τύπων εντολών. Οι εντολές χρειάζονται από τρία έως πέντε βήματα για να εκτελεστούν. Τα δύο πρώτα βήματα είναι ανεξάρτητα από τον τύπο της εντολής.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 24

Ποια είναι τα πέντε βήματα εκτέλεσης των εντολών; Να αναφέρετε συνοπτικά τη λειτουργία των εντολών σε κάθε ένα από αυτά.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 25

Ποιο είναι το αποτέλεσμα της εκτέλεσης των παρακάτω εντολών του MIPS;
α) add $s1, $s2, $s3
β) lw $s1, 100($s2)
γ) beq $s4, $s5, L

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 25

α) add $s1, $s2, $s3 : Κατά την εκτέλεση της εντολής αυτής, γίνεται πρόσθεση των περιεχομένων των καταχωρητών s2 και s3 και εγγραφή του αποτελέσματος στον καταχωρητή s1.
β) lw $s1, 100($s2) : Κατά την εκτέλεση της εντολής αυτής, φορτώνεται στον καταχωρητή s1, το περιεχόμενο της διεύθυνσης μνήμης, η οποία υπολογίζεται προσθέτοντας την τιμή 100 στο περιεχόμενο του καταχωρητή s2. Επομένως γίνεται η πρόσθεση της τιμής 50 και του περιεχομένου του καταχωρητή s2 και το αποτέλεσμα πηγαίνει στον καταχωρητή s1.
γ) beq $s4, $s5, L : Κατά την εκτέλεση της εντολής αυτής συγκρίνονται τα περιεχόμενα των καταχωρητών s4 και s5. Στην περίπτωση που είναι ίσα τότε ο καταχωρητής Στόχος περιέχει τη διεύθυνση του στόχου διακλάδωσης.

Από αυτή την ακολουθία βημάτων μπορούμε να προσδιορίσουμε το τι πρέπει να κάνει η μονάδα ελέγχου σε κάθε κύκλο ρολογιού. Πριν όμως σχεδιάσουμε την μονάδα ελέγχου, πρέπει να προσθέσουμε το τμήμα της μονάδας ελέγχου εγγραφής του απαριθμητή προγράμματος και τους απαραίτητους πολυπλέκτες για την επιλογή της σωστής τιμής που θα εγγραφεί στον απαριθμητή προγράμματος, καθώς επίσης και το τμήμα που αφορά τον καταχωρητή Στόχο. Αφού η υλοποίηση των εντολών μεταπήδησης έχει σχέση με αυτές τις δυνατότητες, θα ενσωματώσουμε την μονάδα ελέγχου για τις εντολές μεταπήδησης μαζί με τις άλλες μονάδες ελέγχου. Εάν συμπεριλάβουμε και τις εντολές μεταπήδησης, υπάρχουν τρεις πιθανές πηγές της τιμής η οποία θα εγγραφεί στον απαριθμητή προγράμματος. Αυτές είναι:

- Το ALUoutput, το οποίο είναι η πηγή όταν ο απαριθμητής προγράμματος έχει αυξηθεί κατά την σειριακή ανάκληση των εντολών.

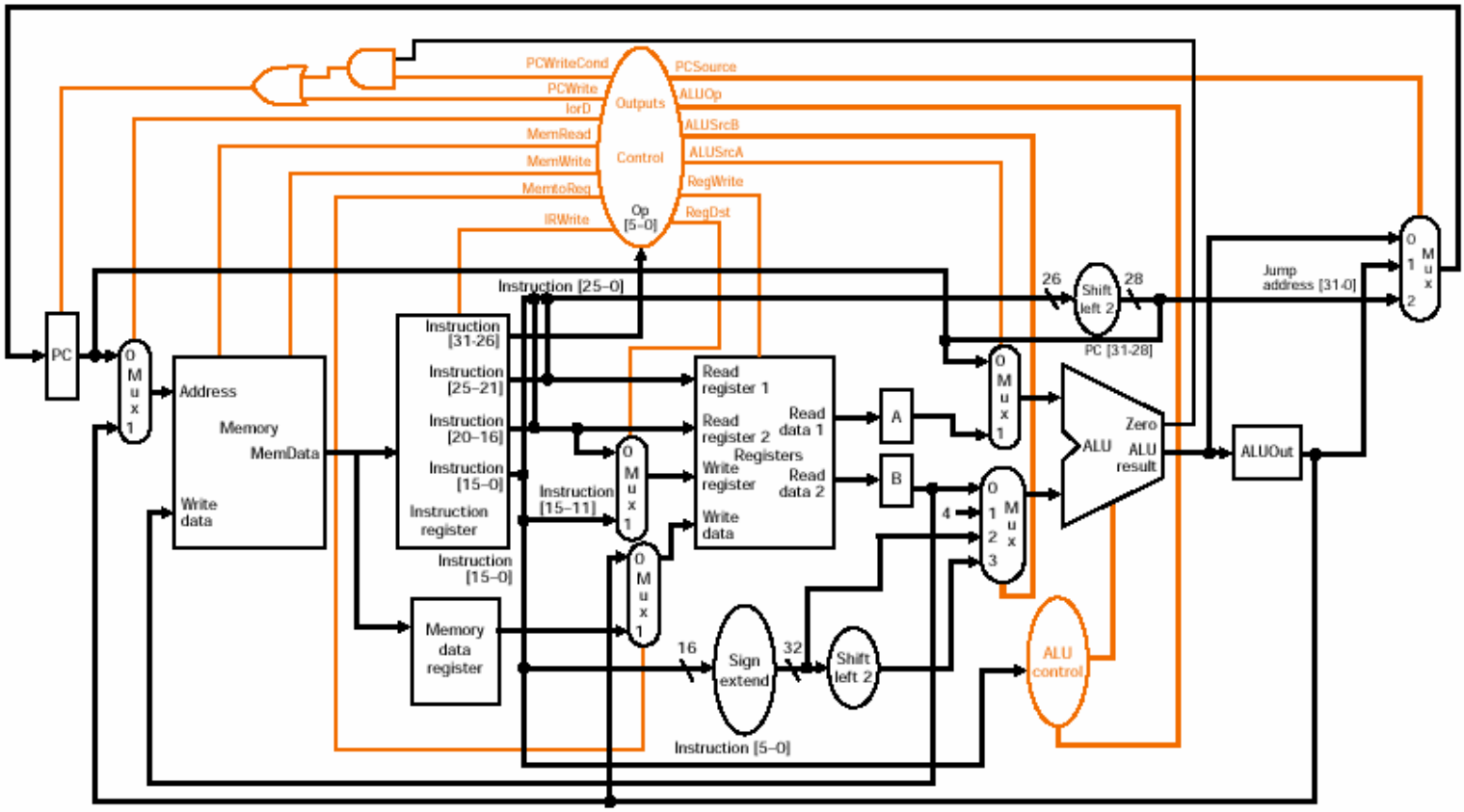
- Ο καταχωρητής Στόχος, όταν έχουμε εντολή διακλάδωσης υπό συνθήκη. Επίσης χρειαζόμαστε ένα σήμα για εγγραφή στον καταχωρητή, που ονομάζεται TargetWrite.

- Τα 26 λιγότερο σημαντικά ψηφία του καταχωρητή εντολών (IR), ολισθημένα αριστερά κατά δύο και συνδεδεμένα με τα τέσσερα περισσότερο σημαντικά ψηφία του απαριθμητή προγράμματος, τα οποία είναι η πηγή της τιμής όταν έχουμε εντολή μεταπήδησης.

Κωδικοποιούμε αυτές τις τρεις πιθανές πηγές χρησιμοποιώντας ένα σήμα ελέγχου των δύο ψηφίων, το PCSource. Οι τρεις πιθανές πηγές κωδικοποιούνται ως εξής: ALUoutput(00), Traget(01) και IR(10) αντίστοιχα. Το σήμα PCSource τότε ελέγχει έναν πολυπλέκτη των τριών εισόδων.

 Όπως έχουμε παρατηρήσει, όταν υλοποιούμε την μονάδα ελέγχου του ενός κύκλου, στον απαριθμητή προγράμματος γίνεται εγγραφή με δύο διαφορετικούς τρόπους. Εάν η εντολή δεν είναι διακλάδωση υπό συνθήκη (beq), η τιμή του απαριθμητή προγράμματος δεν εξαρτάται από την συνθήκη. Εάν έχουμε εντολή διακλάδωσης υπό συνθήκη, ο αυξημένος απαριθμητής προγράμματος αντικαθίσταται με την τιμή του καταχωρητή Στόχου, μόνο εάν ισχύει η συνθήκη στην οποία η έξοδος 0 της ALU είναι επίσης ενεργή. Έτσι χρειαζόμαστε δύο σήματα εγγραφής του απαριθμητή προγράμματος, τα οποία ονομάζουμε PCWrite και PCWriteCond. Το σήμα PCWriteCond και το σήμα 0 της ALU συνδέονται με μία πύλη ΚΑΙ, η οποία συνδέεται μετά με το σήμα PCWrite για να δημιουργηθεί ένα σήμα εγγραφής για τον απαριθμητή προγράμματος.

Το σχήμα 3.2.33 απεικονίζει την ολοκληρωμένη δίοδο δεδομένων πολλών κύκλων και την μονάδα ελέγχου, μαζί με τα επιπλέον σήματα ελέγχου, τον καταχωρητή Στόχο και τον πολυπλέκτη για την ενημέρωση του απαριθμητή προγράμματος. Το σχήμα 3.2.34 απεικονίζει τα αποτελέσματα των επιπλέον σημάτων ελέγχου. Μαζί με το σχήμα 3.2.31 οι πίνακες αυτοί καθορίζουν τα αποτελέσματα όλων των σημάτων ελέγχου στην δίοδο δεδομένων των πολλών κύκλων του σχήματος 3.2.33.



Σχήμα 3.2.33 - Η ολοκληρωμένη δίοδος δεδομένων στη υλοποίηση πολλών κύκλων μαζί με τις απαραίτητες γραμμές ελέγχου. Οι γραμμές ελέγχου του σχήματος 3.2.30 είναι προσαρτημένες στην μονάδα ελέγχου. Επίσης στο σχήμα περιλαμβάνονται τα κυκλώματα ελέγχου και της διόδου δεδομένων που καθορίζουν τις αλλαγές του απαριθμητή προγράμματος. Οι βασικές αλλαγές από το σχήμα 3.2.30 περιλαμβάνουν: τον καταχωρητή Στόχο (στην πάνω δεξιά γωνία), τον πολυπλέκτη τριών εισόδων για την επιλογή της πηγής της ενημερωμένης τιμής του απαριθμητή προγράμματος (πάνω δεξιά), δύο πύλες για την σύνδεση των σημάτων εγγραφής του απαριθμητή προγράμματος, και τα σήματα ελέγχου PCsourse, PCWrite, PCWriteCond και TargetWrite.

Σήμα ελέγχου	Επίδραση όταν δεν είναι ενεργό	Επίδραση όταν είναι ενεργό
PCWrite	Τίποτα	Γίνεται εγγραφή στον απαριθμητή προγράμματος. Η πηγή ελέγχεται από το PCSource.
PCWriteCond	Τίποτα	Γίνεται εγγραφή στον απαριθμητή προγράμματος εάν η έξοδος 0 της ALU είναι επίσης ενεργή.
TargetWrite	Τίποτα	Η έξοδος της ALU γράφεται στον καταχωρητή Στόχο.

a. Οι επιδράσεις των επιπλέον σημάτων ελέγχου του ενός ψηφίου.

Σήμα ελέγχου	Τιμή σήματος	Επίδραση
PCSourse	00	Η έξοδος της ALU αποστέλλεται στον απαριθμητή προγράμματος για εγγραφή.
	01	Το περιεχόμενο του καταχωρητή Στόχου αποστέλλεται στον απαριθμητή προγράμματος για εγγραφή.
	10	Η διεύθυνση του στόχου μεταπήδησης (PC + 4[29-26] συνδεδεμένη με IR[25-0] και ολισθημένη προς τα αριστερά κατά δύο ψηφία), αποστέλλεται στον απαριθμητή προγράμματος για εγγραφή.

b. Οι επιδράσεις του επιπλέον σήματος ελέγχου των δύο ψηφίων, του PCSource.

Σχήμα 3.2.34 - Οι επιδράσεις των σημάτων ελέγχου, τα οποία καθορίζουν τον τρόπο με τον οποίο γίνονται οι εγγραφές στον απαριθμητή προγράμματος. Το σχήμα αυτό μαζί με το σχήμα 3.2.31 προσδιορίζει τις λειτουργίες όλων των σημάτων ελέγχου στην υλοποίηση πολλών κύκλων της διόδου δεδομένων.

Πως σχεδιάζεται η μονάδα ελέγχου

Αφού καθορίσαμε τα σήματα ελέγχου και το πότε αυτά πρέπει να είναι ενεργά, μπορούμε να υλοποιήσουμε τη μονάδα ελέγχου. Στον σχεδιασμό της μονάδας ελέγχου της διόδου δεδομένων ενός κύκλου, χρησιμοποιήσαμε πίνακες αληθείας που καθόριζαν την ενεργοποίηση των σημάτων ελέγχου, ανάλογα με τον τύπο της εντολής. Έπειτα υλοποιήσαμε τον πίνακα αληθείας με λογικές πύλες όπως φαίνεται στο σχήμα 3.2.25. Στην δίοδο δεδομένων πολλών κύκλων, η μονάδα ελέγχου είναι πιο πολύπλοκη επειδή οι εντολές εκτελούνται σε μια σειρά από βήματα. Η μονάδα

ελέγχου αυτή πρέπει να καθορίζει τα σήματα που ενεργοποιούνται σε κάθε βήμα, καθώς και το επόμενο βήμα.

Σε αυτήν την ενότητα θα δούμε μία τεχνική για την υλοποίηση της μονάδας ελέγχου, η οποία βασίζεται στις μηχανές πεπερασμένων καταστάσεων (finite state machines), οι οποίες συνήθως παριστάνονται γραφικά. Η τεχνική αυτή παρουσιάζει την μονάδα ελέγχου σε μία μορφή που επιτρέπει την λεπτομερή υλοποίηση (με πύλες, ROMs, ή PLAs) χρησιμοποιώντας ένα σύστημα CAD.

Η μέθοδος που θα χρησιμοποιήσουμε για την υλοποίηση της μονάδας ελέγχου είναι μία μηχανή πεπερασμένων καταστάσεων.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 26

Να εξηγήσετε τι είναι η μηχανή πεπερασμένων καταστάσεων και από τι αποτελείται.

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 26

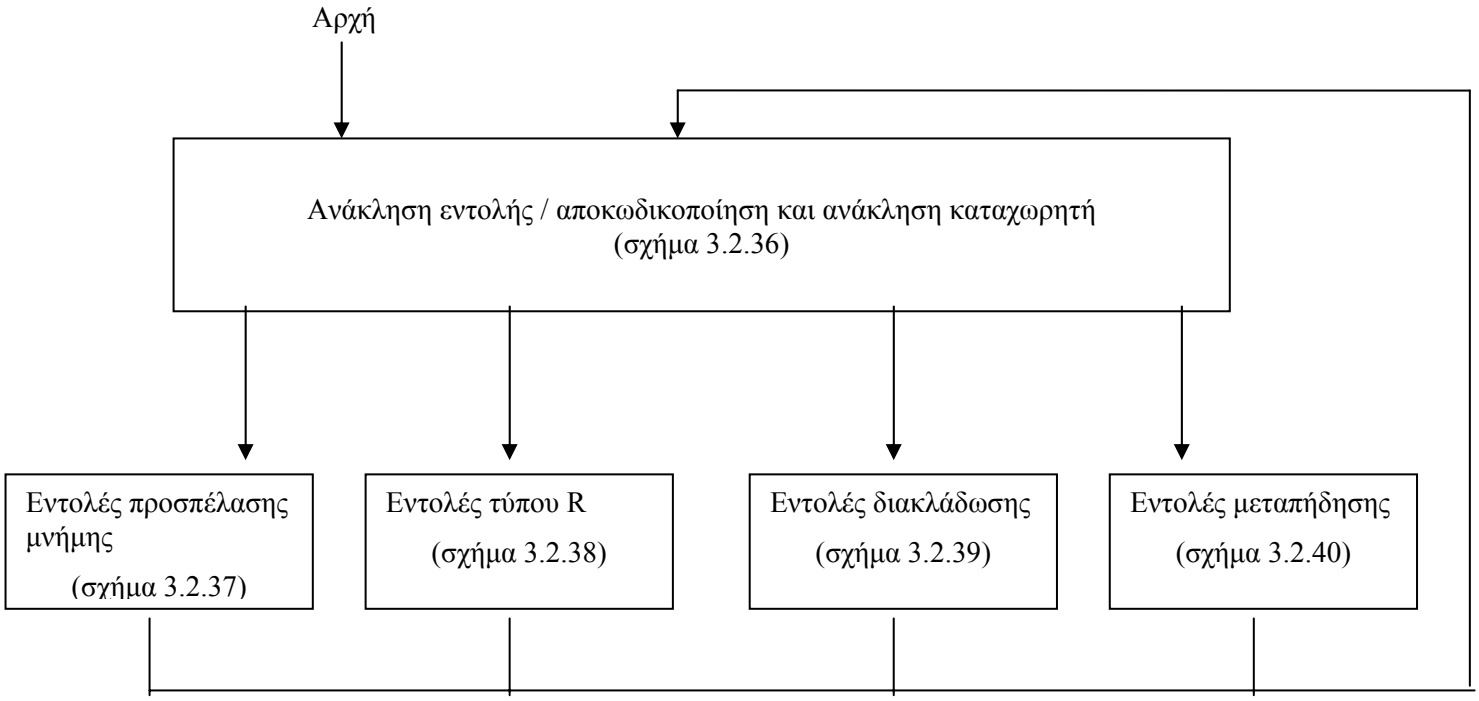
Η μηχανή πεπερασμένων καταστάσεων είναι μία συλλογή από καταστάσεις με μία αρχική κατάσταση και μια συνάρτηση μετάβασης η οποία καθορίζει τη μετάβαση από τη μία κατάσταση στην άλλη. Οι μηχανές πεπερασμένων καταστάσεων χρησιμοποιούνται για να εκφράσουν διαδικασίες απόφασης και χρησιμοποιούνται στην υλοποίηση της μονάδας ελέγχου για να απλοποιήσουν την υλοποίηση.
Η μηχανή πεπερασμένων καταστάσεων είναι χρήσιμη στην σχεδίαση της διόδου δεδομένων επειδή έχουν μικρό αριθμό καταστάσεων. Εάν ο αριθμός των καταστάσεων είναι μεγάλος, τότε η γραφική αναπαράσταση της μηχανής πεπερασμένων καταστάσεων δεν είναι η κατάλληλη μέθοδος για την υλοποίηση της διόδου δεδομένων. Σε τέτοιες περιπτώσεις χρησιμοποιούνται άλλες τεχνικές.
Η μηχανή πεπερασμένων καταστάσεων αποτελείται από καταστάσεις και βέλη που δείχνουν την κατεύθυνση στη μετάβαση των καταστάσεων. Οι κατευθύνσεις καθορίζονται από τη συνάρτηση της επόμενης κατάστασης (next-state function), η οποία εξαρτάται από την παρούσα κατάσταση και τις εισόδους της καινούριας κατάστασης. Κάθε κατάσταση καθορίζει επίσης ένα σύνολο εξόδων που είναι ενεργές όταν η μηχανή βρίσκεται σε αυτή την κατάσταση.

Στην υλοποίηση μιας μηχανής πεπερασμένων καταστάσεων συνήθως υποθέτουμε ότι όλες οι έξοδοι που δεν είναι σαφώς ενεργοποιημένες, είναι απενεργοποιημένες και η σωστή λειτουργία της διόδου δεδομένων συχνά εξαρτάται στο γεγονός ότι ένα σήμα είναι απενεργοποιημένο. Για παράδειγμα, το σήμα

RegWrite πρέπει να είναι ενεργό μόνο όταν είναι να γίνει εγγραφή σε έναν καταχωρητή και όταν δεν είναι σαφώς ενεργοποιημένο, πρέπει να είναι απενεργοποιημένο.

Οι είσοδοι ελέγχου στους πολυπλέκτες είναι κάπως διαφορετικές, αφού επιλέγουν μία από τις εισόδους, ανάλογα με το αν είναι 0 ή 1. Έτσι στη μηχανή πεπερασμένων καταστάσεων πάντα καθορίζουμε τις μονάδες ελέγχου των πολυπλεκτών που μας ενδιαφέρουν. Όταν υλοποιούμε μια μηχανή πεπερασμένων καταστάσεων με λογικά κυκλώματα, το να είναι η μονάδα ελέγχου στο 0 μπορεί να είναι προκαθορισμένο και να μην χρειάζονται πύλες.

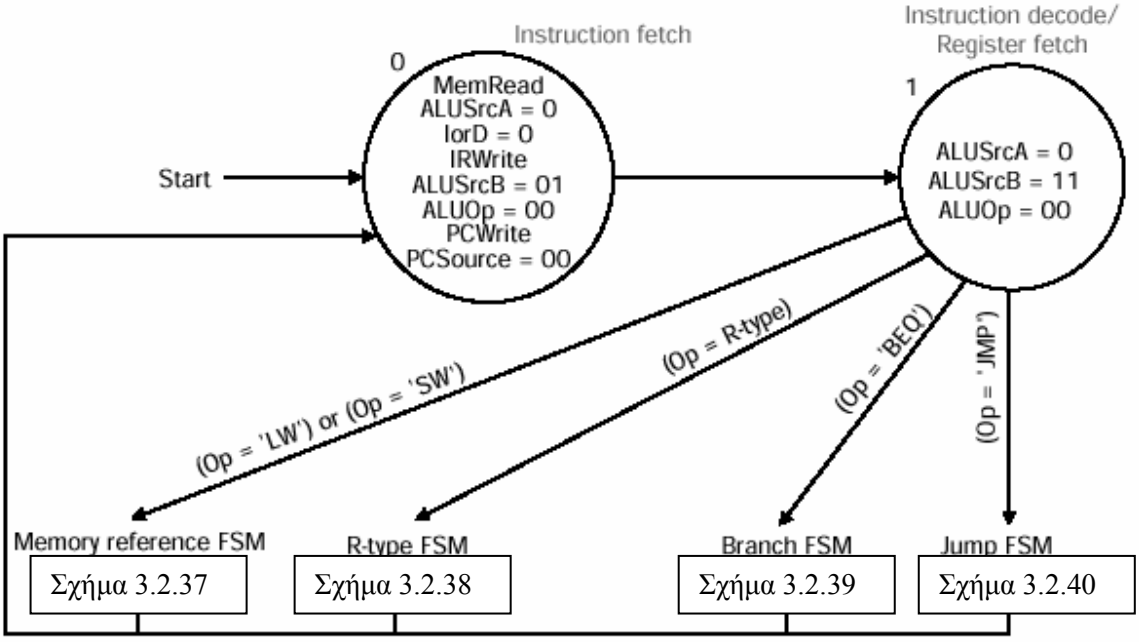
Η μονάδα ελέγχου μιας μηχανής πεπερασμένων καταστάσεων ανταποκρίνεται στα πέντε βήματα της εκτέλεσης των εντολών. Κάθε κατάσταση της μηχανής θα καταλαμβάνει έναν κύκλο ρολογιού. Η μηχανή πεπερασμένων καταστάσεων αποτελείται από διάφορα μέρη. Αφού τα δύο πρώτα βήματα είναι ίδια για όλες τις εντολές, οι δύο αρχικές καταστάσεις της μηχανής θα είναι κοινές για όλες τις εντολές. Τα βήματα 3 έως 5 διαφέρουν ανάλογα με τον κωδικό λειτουργίας. Μετά την εκτέλεση του τελευταίου βήματος, για ένα συγκεκριμένο τύπο εντολής, η μηχανή πεπερασμένων καταστάσεων επιστρέφει στην αρχική κατάσταση για να ξεκινήσει την ανάκληση της επόμενης εντολής. Το σχήμα 3.2.35 απεικονίζει μια αναπαράσταση της μηχανής πεπερασμένων καταστάσεων. Στη συνέχεια θα αναπτύξουμε το τμήμα ανάκλησης της εντολής και το τμήμα της αποκωδικοποίησης και μετά θα δούμε τις καταστάσεις (και τις λειτουργίες τους) για τους διαφορετικούς τύπους εντολών.



Σχήμα 3.2.35 - Μια εικόνα του υψηλού επιπέδου της μονάδας ελέγχου της μηχανής πεπερασμένων καταστάσεων. Κάθε κουτί σε αυτό το σχήμα μπορεί να είναι από μία έως πολλές καταστάσεις. Το βέλος 'Αρχή' δείχνει την αρχική κατάσταση, στην οποία γίνεται ανάκληση της πρώτης εντολής.

 Το σχήμα 3.2.36 απεικονίζει τις δύο πρώτες καταστάσεις μιας μηχανής πεπερασμένων καταστάσεων. Οι καταστάσεις είναι αριθμημένες για απλοποίηση της επεξήγησης του σχήματος. Η κατάσταση 0 αναφέρεται στο βήμα 1 και είναι η αρχική κατάσταση της μηχανής.

Τα σήματα που είναι ενεργά σε κάθε κατάσταση φαίνονται στο σχήμα μέσα στην κατάσταση. Τα βέλη μεταξύ των καταστάσεων καθορίζουν την επόμενη κατάσταση και έχουν πάνω τους τις συνθήκες οι οποίες προσδιορίζουν την επόμενη κατάσταση, όταν περισσότερες από μία καταστάσεις είναι πιθανές. Μετά την κατάσταση 1, η ενεργοποίηση των σημάτων εξαρτάται από τον τύπο της εντολής. Έτσι η μηχανή του σχήματος 3.2.36 έχει τέσσερα βέλη να εξέρχονται από την κατάσταση 1, που ανταποκρίνονται στους τέσσερις διαφορετικούς τύπους των εντολών: εντολές αναφοράς στη μνήμη, τύπου R, διακλάδωσης και μεταπήδησης. Η κατάσταση αυτή ονομάζεται αποκωδικοποίηση και επομένως οι ενέργειες που ακολουθούν εξαρτώνται από τον τύπο της εντολής.



Σχήμα 3.2.36 - Η ανάκληση της εντολής και το τμήμα της αποκωδικοποίησης είναι ίδια για όλες τις εντολές. Οι καταστάσεις αυτές ανταποκρίνονται στο κουτί που βρίσκεται στην κορυφή του σχήματος 3.2.35. Στην πρώτη κατάσταση ενεργοποιούμε κάποια σήματα για να διαβαστεί μια εντολή από τη μνήμη και να εγγραφεί στον καταχωρητή εντολών (MemRead και MemWrite), και θέτουμε το IorD=0 για την επιλογή του απαριθμητή προγράμματος ως την πηγή της διεύθυνσης. Τα σήματα PCWrite, PCSourse, ALUSrcA, ALUOp και ALUSrcB είναι ενεργοποιημένα για τον υπολογισμό του PC+4 και την αποθήκευσή αυτής της τιμής στον απαριθμητή προγράμματος. Στην επόμενη κατάσταση, υπολογίζουμε την διεύθυνση του στόχου διακλάδωσης θέτοντας το ALUSrcB=11 (για να προκαλέσουμε την αποστολή των 16 λιγότερο σημαντικών ψηφίων του IR μετά από επέκταση προσήμου και ολίσθηση, στην ALU). Επίσης θέτουμε τα σήματα ALUSrcA=0 και ALUOp=00 και αποθηκεύουμε το αποτέλεσμα στον καταχωρητή Στόχο (χρησιμοποιώντας το TargetWrite). Μετά υπάρχουν τέσσερις διαφορετικές καταστάσεις που εξαρτώνται από τον τύπο της εντολής, ο οποίος είναι γνωστός κατά την διάρκεια αυτού του βήματος. Εάν έχουμε εντολή φόρτωσης ή

αποθήκευσης πηγαίνουμε σε μία κατάσταση, ενώ τα άλλα βέλη ασχολούνται με τους κωδικούς λειτουργίας μίας εντολής. Η είσοδος της μονάδας ελέγχου, που ονομάζεται op, χρησιμοποιείται για να καθορίσει ποιο από αυτά τα βέλη πρέπει να ακολουθήσουμε, δηλαδή ποια πρέπει να είναι η επόμενη κατάσταση.

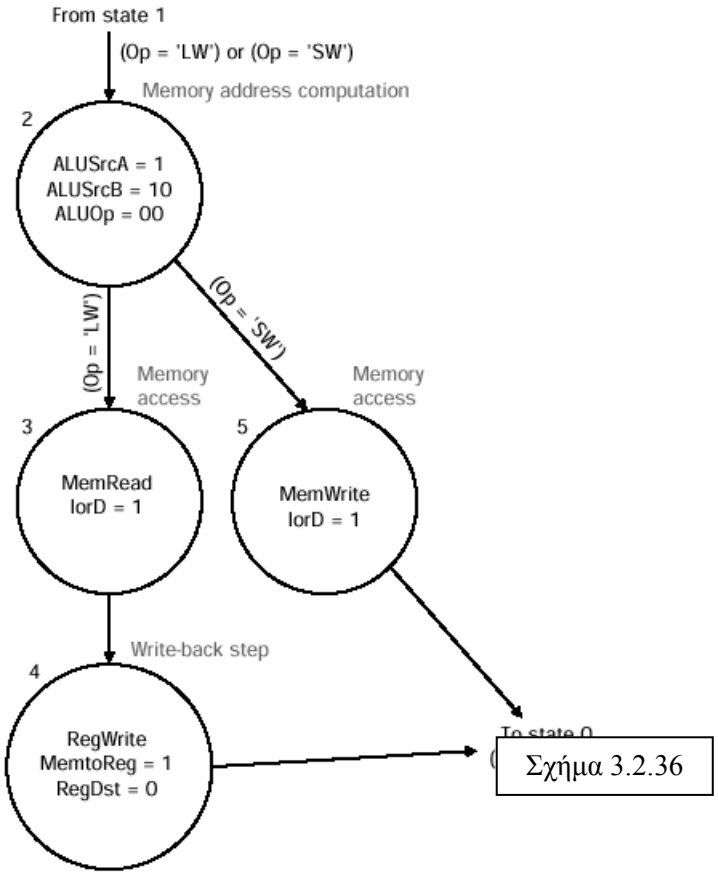
 Το τμήμα της μηχανής πεπερασμένων καταστάσεων που απαιτείται για την υλοποίηση των εντολών αναφοράς στη μνήμη φαίνεται στο σχήμα 3.2.37.

- Γι' αυτές τις εντολές, η πρώτη κατάσταση μετά την ανάκληση της εντολής και την ανάκληση των καταχωρητών, φαίνεται στην κατάσταση 2. Για τον υπολογισμό της διεύθυνσης μνήμης, οι πολυπλέκτες στην είσοδο της ALU πρέπει να είναι ενεργοποιημένοι με τέτοιο τρόπο, ώστε η πρώτη είσοδος να είναι ο καταχωρητής να αντιστοιχεί στο πεδίο rs, ενώ η δεύτερη είσοδος να είναι το πεδίο μετά από επέκταση προσήμου.

- Μετά τον υπολογισμό της διεύθυνσης μνήμης, στην μνήμη πρέπει να γίνει ανάγνωση ή εγγραφή και αυτό απαιτεί δύο διαφορετικές καταστάσεις. Εάν έχουμε εντολή φόρτωσης και ο κωδικός λειτουργίας (op) είναι lw, τότε η κατάσταση 3 (που είναι αντίστοιχη με το βήμα προσπέλασης της μνήμης), εκτελεί την ανάγνωση της μνήμης (το σήμα MemRead είναι ενεργό). Εάν έχουμε εντολή αποθήκευσης και ο κωδικός λειτουργίας (op) είναι sw, η κατάσταση 5 εκτελεί ανάγνωση της μνήμης (το σήμα MemWrite είναι ενεργό). Στις καταστάσεις 3 και 5, το σήμα IorD είναι 1 έτσι ώστε η διεύθυνση της μνήμης να προέρχεται από την ALU.

- Μετά την εγγραφή έχει τελειώσει η εκτέλεση της εντολής αποθήκευσης και η επόμενη κατάσταση είναι η κατάσταση 0. Ωστόσο, στις εντολές φόρτωσης, μία άλλη κατάσταση (η κατάσταση 4) χρειάζεται για να εγγραφεί το αποτέλεσμα από την μνήμη στο αρχείο καταχωρητών.

 Η μνήμη βρίσκεται σε κατάσταση ανάγνωσης στην ίδια διεύθυνση (με την ενεργοποίηση των σημάτων Memread και IorD). Αυτά τα σήματα πρέπει να διατηρηθούν ενεργά στην διάρκεια των καταστάσεων, επειδή η έξοδος της ALU και η μνήμη δεν είναι αποθηκευμένες σε κάποιον καταχωρητή. Εάν οι τιμές της μονάδας ελέγχου άλλαζαν, τότε η έξοδος της ALU και η μνήμη θα άλλαζαν και η τιμή του αποτελέσματος σε μια εντολής φόρτωσης δεν θα ήταν σωστή. Με αυτές τις τιμές σταθερές, τα σήματα ελέγχου του πολυπλέκτη (MemtoReg=1 και RegDst=0), θα στείλουν της έξοδο της ALU στο αρχείο καταχωρητών για εγγραφή, χρησιμοποιώντας το rs σαν αριθμό του καταχωρητή. Μετά από αυτή την κατάσταση, που αντιστοιχεί στο βήμα επανεγγραφής, η επόμενη κατάσταση είναι η κατάσταση 0.

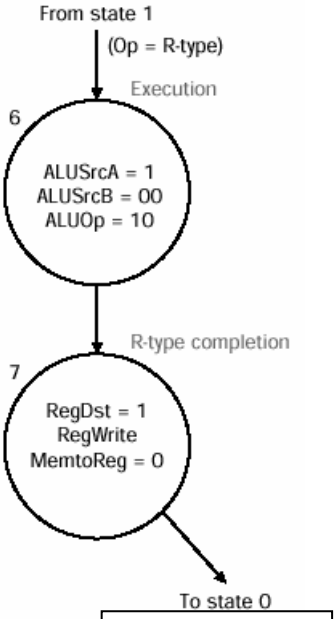


Σχήμα 3.2.37 - Η μηχανή πεπερασμένων καταστάσεων για τον έλεγχο των εντολών που αναφέρονται στη μνήμη έχει τέσσερις καταστάσεις. Αυτές οι καταστάσεις αντιστοιχούν στο κουτί "Εντολές προσπέλασης μνήμης" του σχήματος 3.2.35. Μετά τον υπολογισμό της διεύθυνσης μνήμης, γίνεται οι καταστάσεις διαφοροποιούνται για τις εντολές φόρτωσης και τις εντολές αποθήκευσης. Η ενεργοποίηση των σημάτων ALUSrcA, ALUSrcB και ALUOp χρησιμοποιείται για τον υπολογισμό της διεύθυνσης μνήμης. Αυτά τα σήματα πρέπει να διατηρηθούν σταθερά μέχρι να γίνει εγγραφή της τιμής σε έναν καταχωρητή (αν πρόκειται για εντολή φόρτωσης), ή στην μνήμη (αν πρόκειται για εντολή αποθήκευσης).

 Για την υλοποίηση των εντολών τύπου R χρειάζεται μια μηχανή δύο καταστάσεων, η οποία αντιστοιχεί στα βήματα εκτέλεσης και ολοκλήρωσης μιας εντολής τύπου R. Το σχήμα 3.2.38 απεικονίζει το τμήμα της μηχανής πεπερασμένων καταστάσεων με τις δύο καταστάσεις.

- Η κατάσταση 6 ενεργοποιεί το σήμα ALUSrcA και αφήνει τα σήματα ALUSrcB απενεργοποιημένα. Έτσι οι δύο καταχωρητές που διαβάστηκαν από το αρχείο καταχωρητών, χρησιμοποιούνται σαν είσοδοι της ALU. Θέτοντας το ALUOp=10, η μονάδα ελέγχου της ALU χρησιμοποιεί των κωδικό function για να ενεργοποιήσει τα σήματα ελέγχου της ALU.

- Στην κατάσταση 7, το σήμα RegWrite είναι ενεργό για να προκαλέσει την εγγραφή του καταχωρητή και το RegDst είναι επίσης ενεργό έτσι ώστε το πεδίο rd να χρησιμοποιηθεί σαν τον αριθμό του καταχωρητή προορισμού.



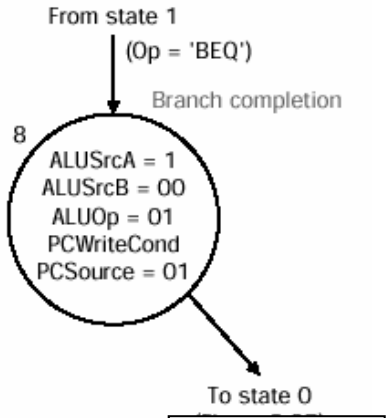
Σχήμα 3.2.36

Σχήμα 3.2.38 - Η υλοποίηση των εντολών τύπου R με μια μηχανή πεπερασμένων καταστάσεων. Οι καταστάσεις αυτές αντιστοιχούν στο κουτί με τίτλο "Εντολές τύπου R" του σχήματος 3.2.35. Στην πρώτη κατάσταση η ALU εκτελεί τις λειτουργίες της, ενώ στη δεύτερη το αποτέλεσμα της ALU γράφεται στο αρχείο καταχωρητών. Τα σήματα που σχετίζονται με την ALU είναι σταθερά κατά την διάρκεια των κύκλων ρολογιού. Τα τρία επιπλέον σήματα ενεργοποιούνται στην κατάσταση 7. Με την ενεργοποίηση αυτή, η έξοδος της ALU εγγράφεται στον καταχωρητή που καθορίζει το πεδίο rd του καταχωρητή εντολών.

 Για τις εντολές διακλάδωσης, χρειάζεται μόνο μία επιπλέον κατάσταση, για την ολοκλήρωση της εκτέλεσης της εντολής κατά τη διάρκεια του τρίτου βήματος.

- Στην κατάσταση αυτή ενεργοποιούνται τα σήματα ελέγχου που υπαγορεύουν στην ALU να εκτελέσει σύγκριση των τιμών των δύο καταχωρητών. Επίσης πρέπει να ενεργοποιηθούν τα σήματα για την εγγραφή υπό συνθήκη του απαριθμητή προγράμματος, της διεύθυνσης από τον καταχωρητή Στόχο (όταν η συνθήκη δεν αληθεύει η τιμή παραμένει PC + 4). Για την σύγκριση χρειάζεται να ενεργοποιηθεί το σήμα ALUSrcA και η τιμή του ALUOp να είναι 01 (για να γίνει αφαίρεση). Για τον έλεγχο της εγγραφής του απαριθμητή προγράμματος, ενεργοποιείται το σήμα PCWriteCond και τίθεται το PCSourse με 01, έτσι ώστε να γίνει εγγραφή της τιμής του καταχωρητή Στόχου στον απαριθμητή προγράμματος, εάν είναι ενεργοποιημένο το bit της εξόδου 0 της ALU.

Το σχήμα 3.2.39 απεικονίζει ένα τμήμα της μηχανής καταστάσεων.

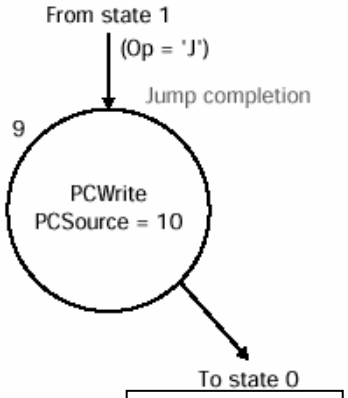


Σχήμα 3.2.36

Σχήμα 3.2.39 - Οι εντολές διακλάδωσης χρειάζονται μόνο μία κατάσταση. Οι πρώτες τρεις έξοδοι που είναι ενεργοποιημένες υπαγορεύουν στην ALU να κάνει σύγκριση ανάμεσα στους καταχωρητές (οι έξοδοι είναι τα σήματα ALUSrcA, ALUSrcB και ALUOp). Τα σήματα PCSourse και PCWriteCond εκτελούν την εγγραφή υπό συνθήκη εάν η συνθήκη διακλάδωσης είναι αληθής.

 Ο τελευταίος τύπος εντολών είναι οι εντολές μεταπήδησης. Όπως και στις εντολές διακλάδωσης χρειάζεται μία μόνο κατάσταση (όπως φαίνεται στο σχήμα 3.2.40) για την ολοκλήρωση της εκτέλεσης των εντολών μεταπήδησης.

- Στην κατάσταση αυτή το σήμα PCWrite είναι ενεργοποιημένο έτσι ώστε να γίνει εγγραφή στον απαριθμητή προγράμματος. Το σήμα PCSource τίθεται 01 και η τιμή που πρόκειται να εγγραφεί θα είναι τα 26 λιγότερο σημαντικά ψηφία του καταχωρητή εντολών, ολισθημένα αριστερά κατά δύο και συνδεδεμένα με τα 4 σημαντικότερα ψηφία του απαριθμητή προγράμματος.



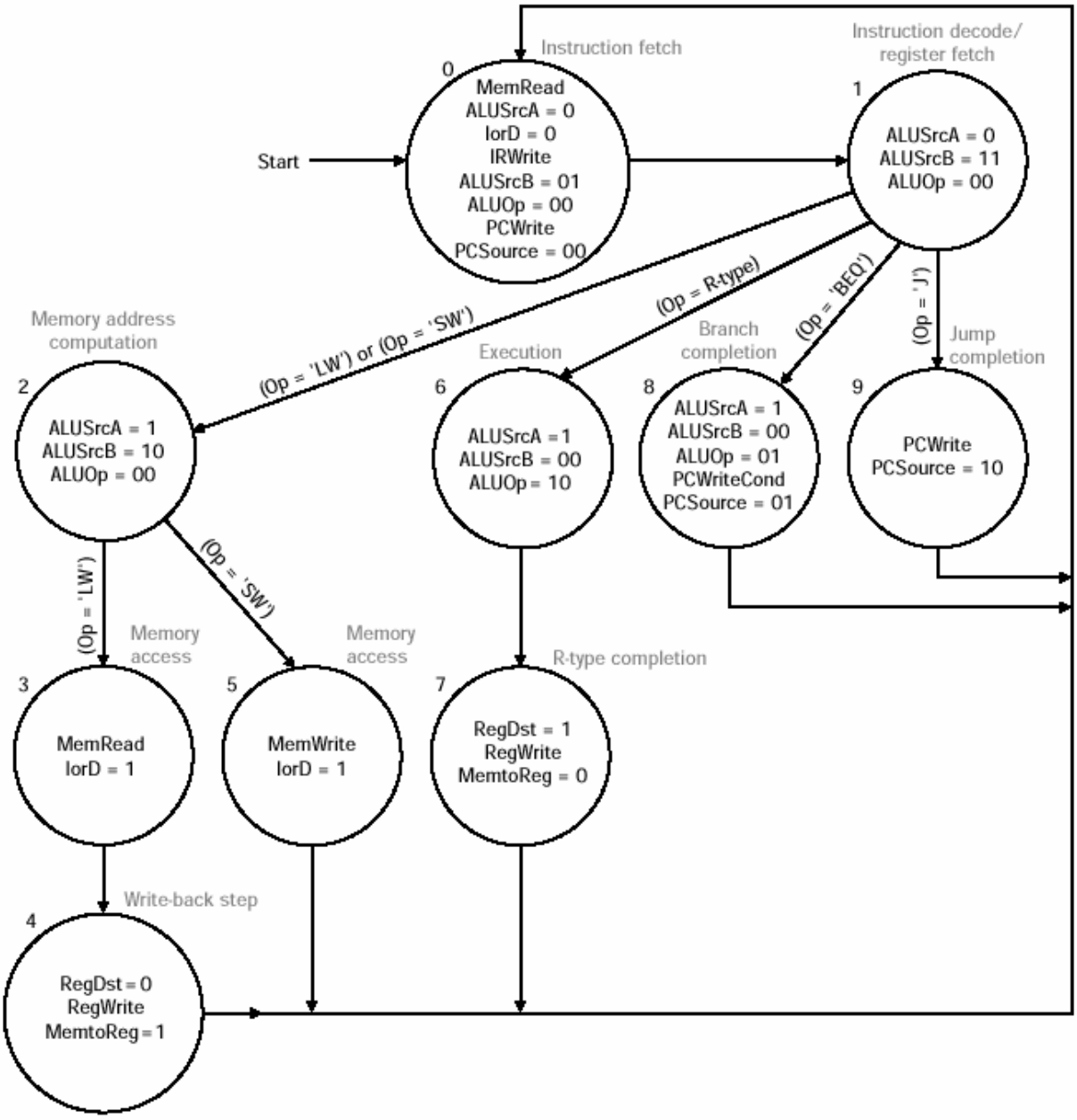
Σχήμα 3.2.36

Σχήμα 3.2.40 - Οι εντολές μεταπήδησης χρειάζονται μία μόνο κατάσταση που να ενεργοποιεί μόνο δύο σήματα ελέγχου, για την εγγραφή του απαριθμητή προγράμματος με τα 26 λιγότερο σημαντικά ψηφία του καταχωρητή εντολών, ολισθημένα αριστερά κατά δύο.

 Τώρα μπορούμε να ενώσουμε τα τμήματα της διόδου δεδομένων σε ένα, για την ολοκλήρωση της μονάδας ελέγχου, όπως φαίνεται στο σχήμα 3.2.41. Σε κάθε κατάσταση φαίνονται τα σχήματα που είναι ενεργοποιημένα. Η λειτουργία της επόμενης κατάστασης εξαρτάται από τα ψηφία του κωδικού λειτουργίας (opcode bits) της εντολής. Επομένως σημειώνουμε πάνω στα βέλη που αντιστοιχούν στην λειτουργία της επόμενης κατάστασης, τον κωδικό λειτουργίας της εντολής που χρησιμοποιείται σαν είσοδος στη μονάδα ελέγχου. Έχοντας την υλοποίηση της μονάδας ελέγχου και γνωρίζοντας ότι κάθε κατάσταση χρειάζεται ένα κύκλο ρολογιού, μπορούμε να βρούμε τους κύκλους ρολογιού για κάθε τύπο εντολής.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 27

Να ενώσετε κατάλληλα τα σχήματα 3.2.36, 3.2.37, 3.2.38, 3.2.39, 3.2.40, έτσι ώστε να σχεδιάσετε την ολοκληρωμένη μηχανή πεπερασμένων καταστάσεων της μονάδας ελέγχου του υπολογιστή MIPS. Να συγκρίνετε το σχήμα που σχεδιάσατε με το σχήμα που ακολουθεί (σχήμα 3.2.41).



Σχήμα 3.2.41 - Η ολοκληρωμένη μηχανή πεπερασμένων καταστάσεων της μονάδας ελέγχου για την δίοδο δεδομένων του σχήματος 3.2.33. Στα βέλη υπάρχουν συνθήκες οι οποίες ελέγχονται για να καθορίσουν την επόμενη κατάσταση. Όταν για την επόμενη κατάσταση δεν υπάρχει συνθήκη, δεν υπάρχει ετικέτα στο βέλος. Οι ετικέτες μέσα στους κόμβους καθορίζουν τα σήματα εξόδου που είναι ενεργοποιημένα κατά την διάρκεια μιας κατάστασης. Η ενεργοποίηση του σήματος ελέγχου ενός πολυπλέκτη καθορίζεται πάντα εάν χρειάζεται σε κάποια από τις λειτουργίες. Έτσι σε μερικές καταστάσεις η είσοδος ελέγχου του πολυπλέκτη είναι ενεργοποιημένη στο 0.

Παράδειγμα

Χρησιμοποιώντας την μονάδα ελέγχου του σχήματος 3.2.41 και τις συχνότητες εμφάνισης των εντολών, να καθορίσετε το μέσο CPI (κύκλοι ρολογιού ανά εντολή), υποθέτοντας ότι κάθε κατάσταση χρειάζεται ένα κύκλο ρολογιού.

Απάντηση:

Η συχνότητα εμφάνισης των εντολών είναι: 22% για τις εντολές φόρτωσης, 11% για τις εντολές αποθήκευσης, 49% για τις εντολές τύπου R, 16% για τις εντολές διακλάδωσης και 2% για τις εντολές μεταπήδησης.

Ο αριθμός των κύκλων ρολογιού για κάθε τύπο εντολής είναι:

- Εντολές φόρτωσης: 5
- Εντολές αποθήκευσης: 4
- Εντολές τύπου R: 4
- Εντολές διακλάδωσης: 3
- Εντολές μεταπήδησης: 3

Το CPI δίνεται από τον παρακάτω τύπο:

$$\text{CPI} = \frac{\text{κύκλοι ρολογιού ΚΜΕ}}{\text{πλήθος εντολών}} = \frac{\sum \text{πλήθος εντολών}_i \times \text{CPI}_i}{\text{πλήθος εντολών}}$$

$$= \sum \frac{\text{πλήθος εντολών}_i}{\text{πλήθος εντολών}} \times \text{CPI}_i$$

Ο λόγος :

$$\frac{\text{πλήθος εντολών}_i}{\text{πλήθος εντολών}}$$

είναι η συχνότητα της εντολής για την κατηγορία εντολής i.

Έτσι αν αντικαταστήσουμε έχουμε:

$$\text{CPI} = 0.22\times5 + 0.1\times4 + 0.49\times4 + 0.16\times3 + 0.02\times3$$
$$= 1.1 + 0.44 + 1.96 + 0.48 + 0.06 = 4.04$$

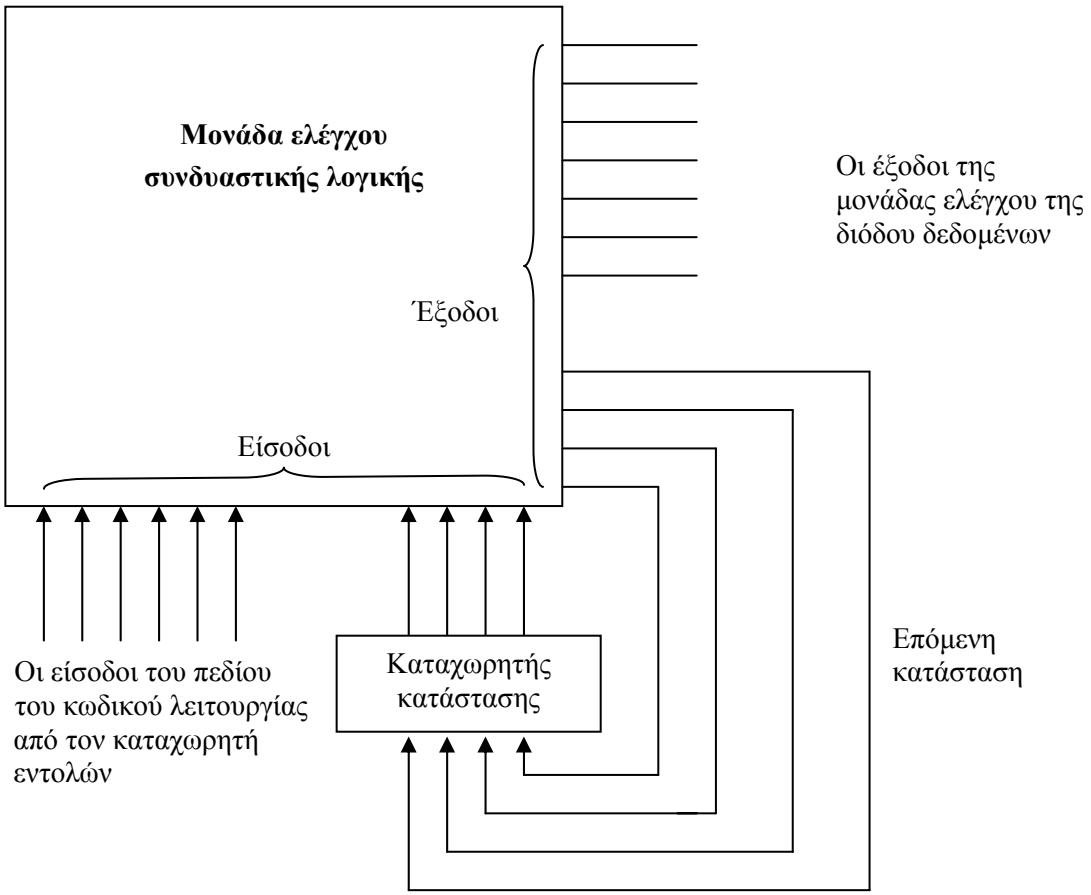
Αυτό το CPI είναι πολύ καλύτερο από το CPI της χειρότερης περίπτωσης, δηλαδή αυτό που θα είχαμε εάν όλες οι εντολές χρειάζονταν τον ίδιο αριθμό κύκλων ρολογιού για να εκτελεστούν (5).

Μία μηχανή πεπερασμένων καταστάσεων μπορεί να υλοποιηθεί με έναν καταχωρητή, ο οποίος διατηρεί την παρούσα κατάσταση και ένα κύκλωμα συνδυαστικής λογικής που καθορίζει τα σήματα της διόδου δεδομένων που είναι ενεργά, όπως επίσης καθορίζει και την επόμενη κατάσταση.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 28

Μπορείτε να απεικονίσετε τον τρόπο με τον οποίο μια μηχανή πεπερασμένων καταστάσεων μπορεί να υλοποιηθεί; Να συγκρίνετε το σχήμα που σχεδιάσατε με το σχήμα που ακολουθεί (σχήμα 3.2.42).

Το σχήμα 3.2.42 απεικονίζει μια τέτοια υλοποίηση.



Σχήμα 3.2.42 - Ο ελεγκτής της μηχανής πεπερασμένων καταστάσεων είναι υλοποιημένος με ένα κύκλωμα συνδυαστικής λογικής και έναν καταχωρητή. Οι έξοδοι της συνδυαστικής λογικής είναι ο αριθμός της επόμενης κατάστασης και τα σήματα ελέγχου που θα ενεργοποιηθούν στην παρούσα κατάσταση. Οι είσοδοι της συνδυαστικής λογικής είναι η παρούσα κατάσταση και οποιαδήποτε είσοδος χρειάζεται για το καθορισμό της επόμενης κατάστασης. Σε αυτή την περίπτωση, οι είσοδοι είναι τα ψηφία του κωδικού λειτουργίας του καταχωρητή.

 **Η μηχανή πεπερασμένων καταστάσεων του σχήματος 3.2.41 ονομάζεται μηχανή Moore (από τον Edward Moore). Το χαρακτηριστικό της είναι ότι η έξοδος εξαρτάται μόνο από την παρούσα κατάσταση. Ένας διαφορετικός τύπος είναι οι μηχανή Mealy (από τον George Mealy). Στη μηχανή Mealy η έξοδος καθορίζεται και από την είσοδο και από την παρούσα κατάσταση.**

Ανακεφαλαιώνοντας λοιπόν …

Η λειτουργία του ρολογιού καθορίζει το πότε θα γίνει ανάγνωση ή εγγραφή ενός σήματος. Στην ακμοπυροδοτούμενη λειτουργία γίνεται ανάγνωση ή εγγραφή ενός σήματος, από ένα στοιχείο μνήμης σε ένα άλλο, μέσω ενός συνδυαστικού κυκλώματος, σε έναν ή δύο κύκλους ρολογιού.

Το τμήμα της διόδου δεδομένων που χρησιμοποιείται για την ανάκληση των εντολών και την αύξηση του απαριθμητή προγράμματος, αποτελείται από τη μνήμη εντολών, τον απαριθμητή προγράμματος και έναν αθροιστή.

Το τμήμα της διόδου δεδομένων για τις εντολές τύπου R αποτελείται από το αρχείο καταχωρητών και την ALU. Οι δύο μονάδες που χρειάζονται για την εκτέλεση των εντολών φόρτωσης και αποθήκευσης, είναι η μονάδα μνήμης δεδομένων και η μονάδα επέκτασης προσήμου. Η δίοδος δεδομένων για τις εντολές διακλάδωσης με συνθήκη χρησιμοποιεί την ALU για τον υπολογισμό της συνθήκης διακλάδωσης και έναν αθροιστή για τον υπολογισμό της διεύθυνσης του στόχου διακλάδωσης.

Για την κατασκευή της διόδου δεδομένων ενός κύκλου, ενώνουμε τα τμήματα της διόδου δεδομένων για κάθε τύπο εντολής και προσθέτουμε πολυπλέκτες και τα απαραίτητα σήματα ελέγχου.

Για την κατασκευή της μονάδας ελέγχου της ALU κωδικοποιούμε τις τρεις εισόδους ελέγχου που έχει, ανάλογα με τη λειτουργία που εκτελεί κάθε φορά. Για την κατασκευή της βασικής μονάδας ελέγχου, πρέπει να προσθέσουμε τις αρτηρίες με τέτοιο τρόπο ώστε να κατευθύνουμε τις εντολές στη δίοδο δεδομένων. Επιπλέον πρέπει να προσθέσουμε τις γραμμές ελέγχου.

Στην υλοποίηση της μονάδας ελέγχου με λογικά κυκλώματα, χωρίζουμε τις εντολές σε μια σειρά από βήματα και κατασκευάζουμε τους αντίστοιχους πίνακες αληθείας. Η δομή που χρησιμοποιείται για την υλοποίηση αυτή ονομάζεται προγραμματιζόμενη λογική ελέγχου ή PLA.

Η συνολική απόδοση της υλοποίησης με μεταβλητό κύκλο ρολογιού είναι ταχύτερη από την απόδοση της υλοποίησης που χρησιμοποιεί ένα μόνο κύκλο ρολογιού για κάθε εντολή. Για να έχουμε μεγαλύτερη απόδοση είναι καλύτερη η χρήση μικρότερου κύκλου ρολογιού, το οποίο απαιτεί πολλούς κύκλους ρολογιού ανά εντολή.

Τα βασικά πλεονεκτήματα της υλοποίησης πολλών κύκλων στη δίοδο δεδομένων, είναι: οι εντολές χρησιμοποιούν διαφορετικό κύκλο ρολογιού και μοιράζονται τις λειτουργικές μονάδες κατά την διάρκεια εκτέλεσής τους.

Η εκτέλεση των εντολών χωρίζεται σε κύκλους ρολογιού έτσι ώστε να ισορροπηθεί ο αριθμός λειτουργιών που εκτελούνται σε ένα κύκλο ρολογιού και να μειωθεί ο χρόνος κύκλου ρολογιού. Οι εντολές χρειάζονται από 3 έως 5 βήματα για να εκτελεστούν, κάθε ένα από τα οποία χρειάζεται ένα κύκλο ρολογιού, περίπου του ίδιου μήκους. Τα δύο πρώτα βήματα είναι ανεξάρτητα από τον τύπο της εντολής.

Μια μέθοδος για την υλοποίηση της μονάδας ελέγχου είναι οι μηχανές πεπερασμένων καταστάσεων. Οι μηχανές πεπερασμένων καταστάσεων αποτελούνται από καταστάσεις και βέλη, τα οποία δείχνουν την κατεύθυνση στην αλλαγή των καταστάσεων.

Κατά την εκτέλεση των εντολών, τα δύο πρώτα βήματα είναι ίδια για όλες τις εντολές. Επομένως, το τμήμα της μηχανής πεπερασμένων καταστάσεων για τα δύο πρώτα βήματα των εντολών είναι κοινό. Τα υπόλοιπα τμήματα αποτελούνται από καταστάσεις ανάλογα με την λειτουργία της κάθε εντολής.

Μια μηχανή πεπερασμένων καταστάσεων μπορεί να υλοποιηθεί με έναν καταχωρητή και ένα κύκλωμα συνδυαστικής λογικής.