5.1 Εισαγωγή

Η απόδοση μιας αρχιτεκτονικής καθορίζεται με βάση τρεις παράγοντες: τον αριθμό εκτελούμενων εντολών, το χρόνο κύκλου μηχανής, και τον αριθμό κύκλων ανά εντολή (CPI). Ο μεταγλωττιστής και το σύνολο εντολών καθορίζουν τον πρώτο παράγοντα. Όμως, και οι δύο άλλοι παράγοντες καθορίζονται από την υλοποίηση του επεξεργαστή. Στο κεφάλαιο αυτό, θα ασχοληθούμε με την κατασκευή της μονάδας επεξεργασίας δεδομένων (ΜΕΔ) και της μονάδας ελέγχου (ΜΕ) του επεξεργαστή, για δύο διαφορετικές υλοποιήσεις της αρχιτεκτονικής MIPS.

Πιο συγκεκριμένα, θα υλοποιήσουμε ένα υποσύνολο του συνόλου εντολών της αρχιτεκτονικής MIPS, που θα περιλαμβάνει:

- Τις εντολές προσπέλασης μνήμης lw και sw.
- Τις εντολές αριθμητικών/λογικών πράξεων add, sub, and, or και slt.
- Τις εντολές διακλάδωσης beq και άλματος j.

Παρότι το υποσύνολο αυτό δεν περιλαμβάνει όλες τις εντολές MIPS, οι βασικές αρχές κατασκευής των δύο μονάδων του επεξεργαστή αναδεικνύονται με την υλοποίηση του υποσυνόλου αυτού, ενώ η υλοποίηση των υπόλοιπων εντολών MIPS γίνεται με παρόμοιο τρόπο.

Μελετώντας την κατασκευή της ΜΕΔ και της ΜΕ του επεξεργαστή, θα μπορέσουμε να δούμε πώς το σύνολο εντολών καθορίζει πτυχές της υλοποίησης, καθώς και πώς οι διάφορες επιλογές μας επηρεάζουν το χρόνο κύκλου μηχανής και τον αριθμό CPI της αρχιτεκτονικής. Θα χρησιμοποιήσουμε μερικούς βασικούς κανόνες σχεδίασης, όπως για παράδειγμα τη *Σχεδίαση υψηλότερης ταχύτητας για τις πιο κοινές περιπτώσεις*, καθώς και την *Κανονικότητα σχεδίασης για μεγαλύτερη απλότητα κατασκευής*. Οι πιο πολλές ιδέες που θα ακολουθήσουμε, εφαρμόζονται στην κατασκευή μεγάλου εύρους διαφορετικών επεξεργαστών υψηλής απόδοσης, τόσο γενικού όσο και ειδικού σκοπού.

Μια προεπισκόπηση της υλοποίησης

Έχουμε συναντήσει ένα μεγάλο αριθμό διαφορετικών εντολών MIPS. Ένα μεγάλο μέρος από ό,τι απαιτείται για την υλοποίηση μιας ΜΕΔ και μιας ΜΕ, όπου αυτές οι εντολές θα εκτελούνται, είναι το ίδιο, ανεξάρτητα από την εντολή. Έτσι, ας θυμηθούμε τις δύο πρώτες φάσεις εκτέλεσης μιας εντολής, που είναι κοινές για όλες τις εντολές:

1. Ανάκληση της εντολής από τη μνήμη, με αποστολή στη ΜΔΜ του περιεχομένου του μετρητή προγράμματος, και ανάγνωση από τη μνήμη της αντίστοιχης λέξης εντολής.
2. Αποκωδικοποίηση της εντολής με πιθανή ταυτόχρονη ανάγνωση ενός ή δύο καταχωρητών. Οι αριθμοί των καταχωρητών που διαβάζονται λαμβάνονται απ' ευθείας από τη λέξη εντολής.

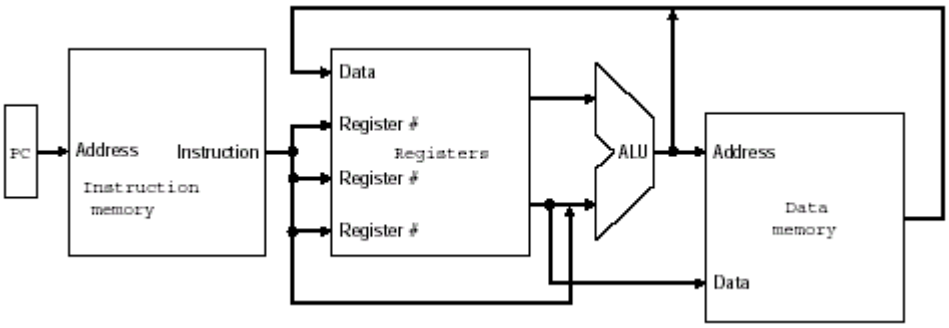
Μετά τις φάσεις αυτές, οι ενέργειες που απαιτούνται για την ολοκλήρωση της εντολής εξαρτώνται από τον τύπο της εντολής. Για κάθε τύπο εντολής που μελετάμε, οι ενέργειες που απαιτούνται είναι λίγο-πολύ οι ίδιες.

Μολαταύτα, ακόμα και για εντολές διαφορετικού τύπου, υπάρχουν πολλές ομοιότητες στις υπολειπόμενες ενέργειες. Για παράδειγμα, όλες οι εντολές που διαβάζουν καταχωρητές χρησιμοποιούν την ΑΛΜ στην επόμενη φάση εκτέλεσής τους: Οι εντολές προσπέλασης μνήμης για τον υπολογισμό της τελικής διεύθυνσης προσπέλασης, οι εντολές αριθμητικών/λογικών πράξεων για την εκτέλεση της αντίστοιχης πράξης, και οι εντολές διακλάδωσης για την αποτίμηση της συνθήκης άλματος.

Πάντως, μετά τη χρήση της ΑΛΜ, οι τελικές ενέργειες που απαιτούνται διαφέρουν μεταξύ των διαφορετικών τύπων εντολών. Μια εντολή προσπέλασης μνήμης θα προσπελάσει τη μνήμη, είτε για να γράψει κάποια δεδομένα αν είναι εντολή αποθήκευσης, είτε για να διαβάσει κάποια δεδομένα αν είναι εντολή φόρτωσης. Μια εντολή αριθμητικής/λογικής πράξης θα αποθηκεύσει το αποτέλεσμα της πράξης από την ΑΛΜ σε κάποιον καταχωρητή. Τέλος, μια

εντολή διακλάδωσης είναι πιθανό να αλλάξει τη διεύθυνση επόμενης εντολής, ανάλογα με το αποτέλεσμα της αποτίμησης της συνθήκης άλματος.

Μια γενική εικόνα της ΜΕΔ που απαιτείται για την εκτέλεση των εντολών που μελετάμε δίνεται στο ακόλουθο διάγραμμα. Στο υπόλοιπο του παρόντος κεφαλαίου θα στηριχτούμε στην εικόνα αυτή, προσθέτοντας τις λεπτομέρειες που λείπουν, και κυρίως, συμπληρώνοντας τη ΜΕ που καθοδηγεί τη ΜΕΔ στην εκτέλεση των εντολών. Πριν προχωρήσουμε όμως σε αυτό, θα αναφερθούμε σε κάποιες απαραίτητες αρχές λογικής σχεδίασης.



Αρχές λογικής σχεδίασης

Σαν πρώτο βήμα κατασκευής ενός συστήματος, πρέπει να πάρουμε κάποιες αποφάσεις πάνω στη λογική και στο χρονισμό των ψηφιακών κυκλωμάτων του συστήματος. Οι βασικές αρχές λογικής σχεδίασης που θα χρησιμοποιηθούν εκτενώς στο υπόλοιπο του κεφαλαίου θεωρούνται γνωστές. Εδώ, θα επιμείνουμε σε κάποιες από αυτές που σχετίζονται περισσότερο με το αντικείμενό μας.

Όταν σχεδιάζουμε ένα ψηφιακό κύκλωμα, δεν είναι απαραίτητο να ταυτίζουμε το λογικά ενεργό σήμα με το ηλεκτρικά ενεργό. Έτσι, σε κάποιες περιπτώσεις, το λογικό "1" (ή "αληθές") αντιστοιχίζεται σε χαμηλή ηλεκτρική τάση, ενώ σε άλλες περιπτώσεις, το ίδιο λογικό σήμα αντιστοιχίζεται σε υψηλή ηλεκτρική τάση. Επειδή η αντιστοίχηση των λογικών σημάτων σε ηλεκτρικά δε σχετίζεται με το θέμα της σχεδίασης της ΜΕΔ και της ΜΕ του επεξεργαστή, στο υπόλοιπο του κεφαλαίου θα αναφερόμαστε μόνο στις λογικές στάθμες των σημάτων, και όχι στις ηλεκτρικές.

Οι υπομονάδες που συναντάμε σε μια αρχιτεκτονική MIPS περιλαμβάνουν δύο ειδών λογικά στοιχεία: στοιχεία που *ενεργούν* σε κάποια δεδομένα, και στοιχεία που *αποθηκεύουν* τα δεδομένα αυτά. Τα πρώτα είναι όλα *συνδυαστικά* στοιχεία, το οποίο σημαίνει ότι οι έξοδοί τους εξαρτώνται αποκλειστικά από τις τρέχουσες εισόδους τους. Για την ίδια είσοδο, ένα συνδυαστικό κύκλωμα παράγει πάντα την ίδια έξοδο. Τυπικό παράδειγμα συνδυαστικού στοιχείου σε μια ΜΕΔ είναι η ΑΛΜ, η οποία για ένα συγκεκριμένο σύνολο εισόδων παράγει την ίδια έξοδο, αφού δεν αποθηκεύει εσωτερικά καμία πληροφορία.

Από την άλλη μεριά, τα υπόλοιπα στοιχεία δεν είναι συνδυαστικά, αλλά στοιχεία *κατάστασης*, που λέγονται έτσι επειδή αποθηκεύουν τα δεδομένα που περιγράφουν την κατάσταση του συστήματος. Αν κάθε φορά που ανοίγουμε το διακόπτη ενός ψηφιακού συστήματος, φορτώνουμε τα στοιχεία κατάστασης με τα ίδια ακριβώς δεδομένα, το σύστημα θα συμπεριφέρεται με τον ίδιο ακριβώς τρόπο. Η κατάσταση δηλαδή του συστήματος καθορίζει πλήρως τη συμπεριφορά του κάθε χρονική στιγμή. Τέτοια στοιχεία κατάστασης σε μια ΜΕΔ είναι οι κρυφές μνήμες εντολών και δεδομένων, όπως και οι καταχωρητές της ΜΕΔ.

Ένα στοιχείο κατάστασης έχει τουλάχιστον δύο εισόδους και μία έξοδο. Οι απαιτούμενες είσοδοι είναι (α) η είσοδος των δεδομένων που θα αποθηκευτούν στο στοιχείο, που θα αποτελέσουν δηλαδή τη νέα κατάσταση του στοιχείου, και (β) η είσοδος ρολογιού που καθορίζει το χρόνο στον οποίο θα γίνει η αποθήκευση, η αλλαγή δηλαδή της κατάστασης του στοιχείου. Η έξοδος ενός στοιχείου κατάστασης παρέχει την τιμή των αποθηκευμένων δεδομένων του στοιχείου, όπως αυτά εγγράφηκαν σε κάποιον προηγούμενο ωρολογιακό παλμό. Παρα-

τηρήστε ότι η εγγραφή ενός στοιχείου κατάστασης γίνεται μόνο τη στιγμή που καθορίζει η είσοδος ρολογιού, ενώ η ανάγνωσή του γίνεται οποιαδήποτε χρονική στιγμή.

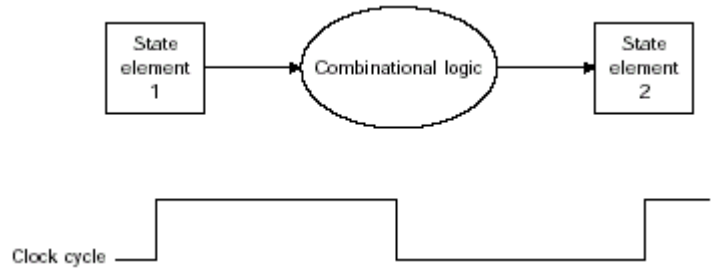
Τα στοιχεία κατάστασης λέγονται και *ακολουθιακά*, επειδή οι έξοδοί τους εξαρτώνται τόσο από τις εισόδους τους, όσο και από την τρέχουσα κατάστασή τους. Για παράδειγμα, η έξοδος της υπομονάδας καταχωρητών εξαρτάται τόσο από το περιεχόμενο των καταχωρητών, όσο και από τους αριθμούς καταχωρητών που παρέχονται σαν είσοδος στην υπομονάδα.

Χρονισμός ψηφιακών κυκλωμάτων

Μια *μέθοδος χρονισμού* καθορίζει τη χρονική στιγμή κατά την οποία ένα ψηφιακό σήμα μπορεί να διαβαστεί από ή να γραφτεί σε ένα στοιχείο κατάστασης. Είναι σημαντικό να καθορίσουμε επακριβώς πότε γίνονται οι αναγνώσεις και πότε γίνονται οι εγγραφές, επειδή αν προσπαθήσουμε να διαβάσουμε ένα σήμα την ίδια στιγμή στην οποία γίνεται η εγγραφή του, η τιμή που θα διαβάσουμε μπορεί να είναι η παλιά τιμή, η νέα τιμή, ή ακόμα κάποιος τυχαίος συνδυασμός των δύο τιμών! Η μέθοδος χρονισμού επιλέγεται, ώστε να αποφεύγεται αυτή ακριβώς η ανεπιθύμητη συμπεριφορά.

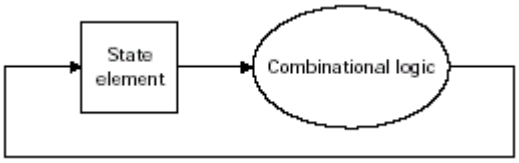
Για απλότητα, θα θεωρήσουμε ότι το σύστημά μας έχει *ακμοπυροδοτούμενο* χρονισμό. Ένας τέτοιος χρονισμός επιτρέπει αποθήκευση τιμών σε στοιχεία κατάστασης μόνο στις ακμές των ωρολογιακών παλμών. Έτσι, κι επειδή μόνο τα στοιχεία κατάστασης αποθηκεύουν δεδομένα στο σύστημα, κάθε συνδυαστικό κύκλωμα του συστήματος λαμβάνει τις εισόδους του από, και στέλνει τις εξόδους του σε στοιχεία κατάστασης, και μάλιστα οι είσοδοι αντιστοιχούν σε κατάσταση που αποθηκεύτηκε σε προηγούμενο ωρολογιακό παλμό, ενώ οι έξοδοι γίνονται διαθέσιμοι στον επόμενο ωρολογιακό παλμό.

Ο χρονισμός που περιγράφηκε πιο πάνω απεικονίζεται στο πιο κάτω σχήμα. Πιο συγκεκριμένα, υποθέτοντας ότι οι αλλαγές κατάστασης συμβαίνουν στην άνοδο των ωρολογιακών παλμών, βλέπουμε ότι όλα τα σήματα εισόδου του συνδυαστικού κυκλώματος προέρχονται από το στοιχείο κατάστασης 1, όπου το τελευταίο σήμα αποθηκεύτηκε στην άνοδο του παρόντος παλμού, περνάνε μέσα από το συνδυαστικό κύκλωμα, και τα σήματα εξόδου αυτού αποθηκεύονται στο στοιχείο κατάστασης 2 με την άνοδο του επόμενου παλμού. Ο ελάχιστος χρόνος που απαιτείται για τη διάδοση των σημάτων μεταξύ δύο στοιχείων κατάστασης ορίζει τη διάρκεια του ωρολογιακού παλμού, δηλαδή τον κύκλο μηχανής του συστήματος.



Αν ένα στοιχείο κατάστασης δε γράφεται σε κάθε παλμό ρολογιού, θα πρέπει στις εισόδους του να συμπεριλάβουμε κάποιο σήμα επίτρεψης εγγραφής. Τότε, η κατάσταση του στοιχείου μεταβάλλεται σε μια ακμή ωρολογιακού παλμού, μόνο αν το σήμα επίτρεψης εγγραφής είναι λογικά ενεργό όταν εμφανίζεται η ακμή αυτή. Στοιχεία κατάστασης που γράφονται σε κάθε παλμό του ρολογιού δεν απαιτούν σήμα επίτρεψης εγγραφής.

Ο ακμοπυροδοτούμενος χρονισμός μας επιτρέπει να διαβάσουμε το περιεχόμενο ενός καταχωρητή, να επεξεργαστούμε τα δεδομένα που διαβάστηκαν με τη βοήθεια κάποιου συνδυαστικού κυκλώματος, και να αποθηκεύσουμε την έξοδο του τελευταίου πίσω στον καταχωρητή, μέσα στον ίδιο ωρολογιακό παλμό. Κάτι τέτοιο απεικονίζεται στο επόμενο σχήμα. Δεν



έχει σημασία, αν οι εγγραφές συμβαίνουν στην άνοδο ή στην πτώση του παλμού, και, αφού οι είσοδοι του συνδυαστικού κυκλώματος δεν αλλάζουν παρά μόνο με μια εγγραφή στον καταχωρητή, οι έξοδοι αυτού παραμένουν σταθερές μέχρι την αποθήκευση των τιμών τους πίσω στον καταχωρητή. Έτσι, δε δημιουργείται κύκλωμα ανάδρασης με επανατροφοδότηση του συνδυαστικού κυκλώματος μέσα στον ίδιο κύκλο ρολογιού, και το απεικονιζόμενο κύκλωμα λειτουργεί σωστά.

Όταν τα δεδομένα που χειρίζεται ο επεξεργαστής είναι μεγέθους 32 bits, οι περισσότερες είσοδοι και έξοδοι των κυκλωμάτων του έχουν επίσης μέγεθος 32 bits. Γραμμές σημάτων μεγέθους μεγαλύτερου από 1 bit λέγονται *αρτηρίες* ή *δίαυλοι*. Στα διαγράμματα που ακολουθούν, οι αρτηρίες απεικονίζονται με παχιές γραμμές. Το μέγεθος (εύρος) μιας αρτηρίας συμβολίζεται με έναν αριθμό πάνω στην αρτηρία, ο οποίος θα παραλείπεται, όταν ισούται με το μέγεθος των δεδομένων του επεξεργαστή ή όταν προκύπτει άμεσα από το διάγραμμα. Σε πολλές περιπτώσεις, ένα βέλος πάνω στην αρτηρία θα δείχνει την κατεύθυνση που ακολουθούν τα δεδομένα σε αυτήν. Τέλος, γραμμές που μεταφέρουν σήματα ελέγχου θα διαχωρίζονται από γραμμές που μεταφέρουν δεδομένα με αυτό το χρώμα.

Η υλοποίηση του υποσυνόλου του συνόλου εντολών MIPS

Θα ξεκινήσουμε την υλοποίηση με ένα σύστημα βασισμένο στο σχήμα που είδαμε νωρίτερα, το οποίο θα ακολουθεί το χρονισμό ενός ωρολογιακού παλμού για κάθε εντολή MIPS. Μ' άλλα λόγια, στο σύστημα αυτό, κάθε εντολή ξεκινάει τον κύκλο της σε μια ακμή ωρολογιακού παλμού, και τον ολοκληρώνει στην αντίστοιχη ακμή του επόμενου παλμού.

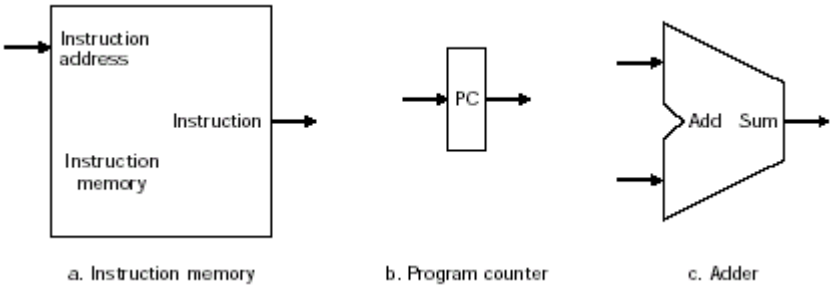
Αν και πιο εύκολα κατανοητή, η υλοποίηση αυτή δεν εφαρμόζεται στην πράξη, επειδή είναι πιο αργή από μια υλοποίηση, στην οποία κάθε διαφορετικός τύπος εντολής απαιτεί διαφορετικό αριθμό πολλαπλών – αλλά πολύ πιο σύντομων – ωρολογιακών παλμών. Αφού κατανοήσουμε τον έλεγχο της απλούστερης πρώτης υλοποίησης, θα προχωρήσουμε στη δεύτερη υλοποίηση των πολλαπλών ωρολογιακών παλμών (ή κύκλων μηχανής) για κάθε εντολή. Η υλοποίηση αυτή είναι πιο ρεαλιστική, αλλά και πιο πολύπλοκη.

Η σχεδίαση της ΜΕ του επεξεργαστή καταλήγει στη μορφή λογικών εξισώσεων ή διαγραμμάτων κατάστασης. Και οι δύο αυτές μορφές μπορούν εύκολα να οδηγήσουν σε σχεδίαση υλικού με τη βοήθεια ειδικού προγράμματος CAD. Το πώς γίνεται αυτό δεν αποτελεί αντικείμενο του παρόντος βιβλίου.

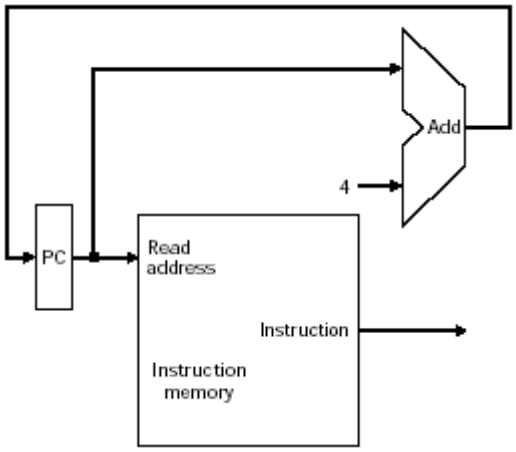
5.2 Σχεδιάζοντας τη ΜΕΔ

Για να ξεκινήσουμε τη σχεδίαση της ΜΕΔ του επεξεργαστή, είναι λογικό να αναλογιστούμε ποιες είναι η κύριες υπομονάδες της ΜΕΔ που απαιτούνται για την εκτέλεση του καθενός διαφορετικού τύπου εντολής που η ΜΕΔ θα υποστηρίζει. Έχοντας υπ' όψη μας ποιες υπομονάδες χρειάζεται κάθε εντολή, μπορούμε να σχεδιάσουμε τη ΜΕΔ, συνδυάζοντας τις υπομονάδες αυτές, εισάγοντας όποια απαραίτητα σήματα ελέγχου διαφοροποιούν την εκτέλεση των εντολών αυτών.

Το πρώτο στοιχείο που χρειαζόμαστε είναι ο χώρος αποθήκευσης των εντολών του προγράμματος. Μια μονάδα μνήμης, η μνήμη εντολών, χρησιμοποιείται για το σκοπό αυτό. Η μνήμη αυτή, όπως και κάθε μονάδα μνήμης αποτελεί στοιχείο κατάστασης για τον επεξεργαστή. Η μνήμη εντολών, εκτός της αποθήκευσης των εντολών του προγράμματος, παρέχει στον επεξεργαστή μια λέξη εντολής, όταν της δίνεται μια διεύθυνση. Η διεύθυνση της εντολής αποθηκεύεται σε ένα άλλο στοιχείο κατάστασης, το *μετρητή προγράμματος* (PC), ο οποίος είναι ένας καταχωρητής ειδικού σκοπού. Μετά την ανάκληση μιας λέξης εντολής, ο PC πρέπει να αυξάνεται κατάλληλα, ώστε να περιέχει τη διεύθυνση της επόμενης εντολής. Έτσι, στα παραπάνω στοιχεία προσθέτουμε έναν αθροιστή, ο οποίος δεν είναι παρά μια απλοποιημένη ΑΛΜ που είναι σχεδιασμένη έτσι, ώστε να εκτελεί μόνο την πράξη της πρόσθεσης. Τα 3 βασικά στοιχεία της ΜΕΔ που είδαμε απεικονίζονται γραφικά στο σχήμα που ακολουθεί.



Η εκτέλεση μιας εντολής ξεκινάει με την ανάκληση αυτής από τη μνήμη εντολών. Για την προετοιμασία εκτέλεσης της επόμενης εντολής, αυξάνουμε το μετρητή προγράμματος, ώστε να δείχνει στην επόμενη εντολή, που για μέγεθος λέξης 32 bits, είναι 4 θέσεις παρακάτω στη μνήμη εντολών[1]. Η βασική ΜΕΔ που υλοποιεί το βήμα αυτό και συνδυάζει τα 3 προηγούμενα στοιχεία φαίνεται στο ακόλουθο διάγραμμα.



Ας θεωρήσουμε στη συνέχεια τον πρώτο τύπο εντολών MIPS που θα μας απασχολήσει, τις *αριθμητικές/λογικές εντολές*, ή αλλιώς *R-εντολές*. Οι εντολές αυτές διαβάζουν δύο καταχωρητές, εκτελούν μια πράξη ΑΛΜ μεταξύ των δεδομένων που διαβάστηκαν, και αποθηκεύουν ένα αποτέλεσμα σε κάποιον καταχωρητή. R-εντολές είναι οι εντολές add, sub, and, or και slt. Ένα παράδειγμα εφαρμογής μιας εντολής add είναι το

```
add $4, $2, $3
```

που διαβάζει τους καταχωρητές $2 και $3, και γράφει τον καταχωρητή $4.

Το σύνολο εντολών MIPS υποστηρίζει 32 καταχωρητές γενικού σκοπού. Η υπομονάδα καταχωρητών υλοποιείται σε μια διάταξη φακέλου, το *φάκελο καταχωρητών* (ΦΚ), έτσι ώστε κάθε προσπέλαση για ανάγνωση ή εγγραφή ενός καταχωρητή να μπορεί να γίνει μέσω ενός αριθμού από 0 μέχρι 31, του αριθμού δηλαδή του καταχωρητή. Ο ΦΚ είναι ένα στοιχείο κατάστασης του επεξεργαστή. Το συνδυαστικό στοιχείο που χρησιμοποιείται στις R-εντολές για την επεξεργασία των δεδομένων που διαβάζονται από το ΦΚ είναι μία ΑΛΜ.

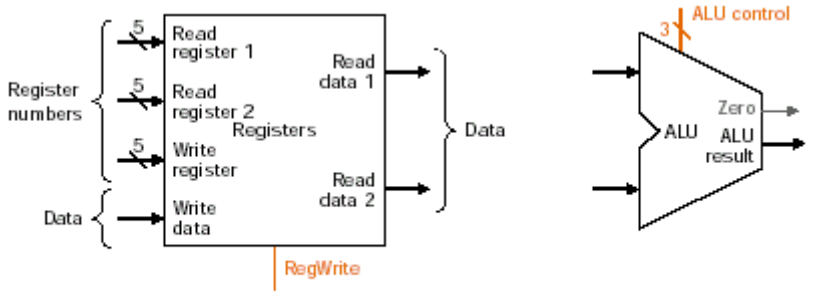
Επειδή οι R-εντολές έχουν τρία τελούμενα, πρέπει να μπορούμε να διαβάζουμε δύο λέξεις δεδομένων από, και να γράφουμε μία λέξη δεδομένων στο ΦΚ σε κάθε εντολή. Επομένως, για κάθε λέξη δεδομένων που διαβάζουμε από το ΦΚ, χρειαζόμαστε μια είσοδο για να παρέχει τον αντίστοιχο αριθμό καταχωρητή, και μια έξοδο για να μεταφέρει τη λέξη εκτός του ΦΚ. Για τη λέξη δεδομένων που γράφουμε στο ΦΚ, χρειαζόμαστε δύο εισόδους, μία για να παρέχει τον αριθμό καταχωρητή, και μία για να παρέχει τη λέξη που γράφεται στο ΦΚ. Ο ΦΚ έχει έγκυρες εξόδους κάθε χρονική στιγμή, ανάλογα με τις τιμές που έχει στις αντίστοιχες δύο εισόδους του. Μ' άλλα λόγια, η ανάγνωση του ΦΚ συμβαίνει σε κάθε κύκλο μηχανής. Αντίθετα, η εγγραφή του ΦΚ γίνεται μόνο σε καθορισμένους κύκλους μηχανής, μια που εκτός από R-εντολές, η ΜΕΔ υποστηρίζει και άλλες εντολές που δε γράφουν αποτέλεσμα στο

[1] Αν ο PC αποθηκεύει διεύθυνση ψηφιολέξης (byte), η αύξηση γίνεται όντως κατά 4. Εναλλακτικά, μπορούμε να υποθέσουμε ότι ο PC αποθηκεύει διεύθυνση λέξης, οπότε η αύξηση θα γίνεται κατά 1.

ΦΚ. Για το λόγο αυτό, η εγγραφή του ΦΚ ελέγχεται από ειδικό σήμα επίτρεψης εγγραφής, το πρώτο σήμα ελέγχου που συναντάμε στη ΜΕΔ που υλοποιούμε. Συνολικά λοιπόν, ο ΦΚ δέχεται 5 εισόδους (3 αριθμούς καταχωρητών, 1 λέξη δεδομένων και 1 σήμα επίτρεψης εγγραφής) και παρέχει 2 εξόδους (2 λέξεις δεδομένων). Οι 3 πρώτες είσοδοι έχουν μέγεθος 5 bits, ώστε να καθορίζουν έναν αριθμό από 0 έως 31, ενώ η είσοδος δεδομένων και οι δύο έξοδοι έχουν μέγεθος 32 bits.

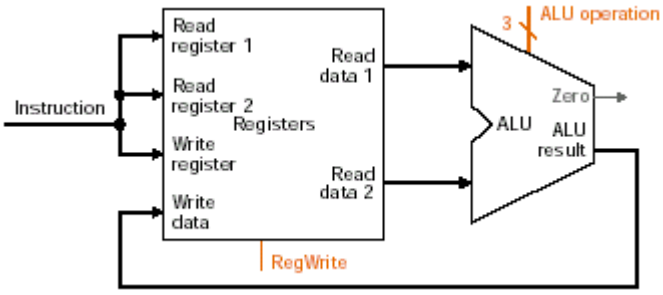
Η ΑΛΜ από την άλλη μεριά, έχει 3 εισόδους (2 λέξεις δεδομένων και 1 σήμα ελέγχου) και 2 εξόδους (1 λέξη δεδομένων και 1 γραμμή μηδενικής τιμής). Το σήμα ελέγχου ΑΛΜ έχει μέγεθος 3 bits, ώστε να καθορίζει 1 από τις 5 πράξεις που υποστηρίζονται. Όπως και στον υπόλοιπο επεξεργαστή, οι λέξεις δεδομένων έχουν μέγεθος 32 bits.

Ο ΦΚ και η ΑΛΜ δείχνονται σχηματικά αμέσως παρακάτω. Το διάγραμμα που ακολουθεί



a. Registers b. ALU

δίνει τη ΜΕΔ που υλοποιεί την εκτέλεση των R-εντολών χρησιμοποιώντας τα δύο παραπάνω στοιχεία, μη συμπεριλαμβανομένων των υπομονάδων που συμμετέχουν στην ανάκληση των εντολών. Καθώς οι αριθμοί των καταχωρητών που αποτελούν τα τελούμενα των R-εντολών λαμβάνονται απ' ευθείας από τη λέξη εντολής, το διάγραμμα δείχνει τους αριθμούς αυτούς να προέρχονται από τις γραμμές που μεταφέρουν την εντολή από τη μνήμη εντολών, σύμφωνα με προηγούμενο διάγραμμα.



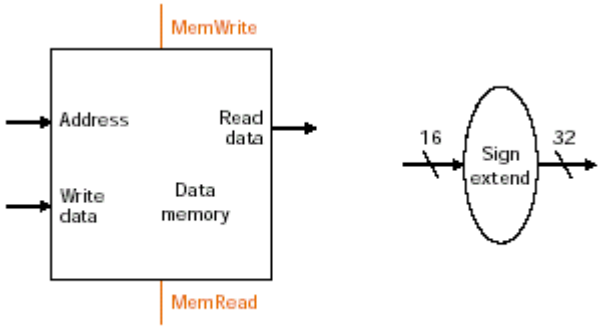
Στη συνέχεια θα μελετήσουμε τις εντολές προσπέλασης μνήμης MIPS lw και sw. Αυτές έχουν τη γενική μορφή

```
lw $t1,offset($t2)
sw $t1,offset($t2)
```

όπου $t1 και $t2 καταχωρητές, ενώ offset είναι μια σταθερά μετατόπισης, η οποία λαμβάνεται από τη λέξη εντολής σαν προσημασμένος αριθμός μεγέθους 16 bits. Οι εντολές αυτές υπολογίζουν την τελική διεύθυνση προσπέλασης προσθέτοντας το περιεχόμενο του καταχωρητή βάσης $t2 με τη σταθερά μετατόπισης. Αν η εντολή είναι εντολή αποθήκευσης, τα δεδομένα αποθήκευσης διαβάζονται από το ΦΚ, και ειδικότερα από τον καταχωρητή $t1, και αποστέλλονται στη μνήμη, όπου και αποθηκεύονται στην παραπάνω διεύθυνση. Αν η εντολή είναι εντολή φόρτωσης, τα δεδομένα φόρτωσης λαμβάνονται από τη μνήμη, και από την παραπάνω διεύθυνση, και αποθηκεύονται στο ΦΚ, και ειδικότερα στον καταχωρητή $t1. Είναι φανερό ότι για την εκτέλεση των εντολών προσπέλασης μνήμης θα χρειαστούμε τόσο το ΦΚ, όσο και την ΑΛΜ, που χρησιμοποιήσαμε και προηγουμένως.

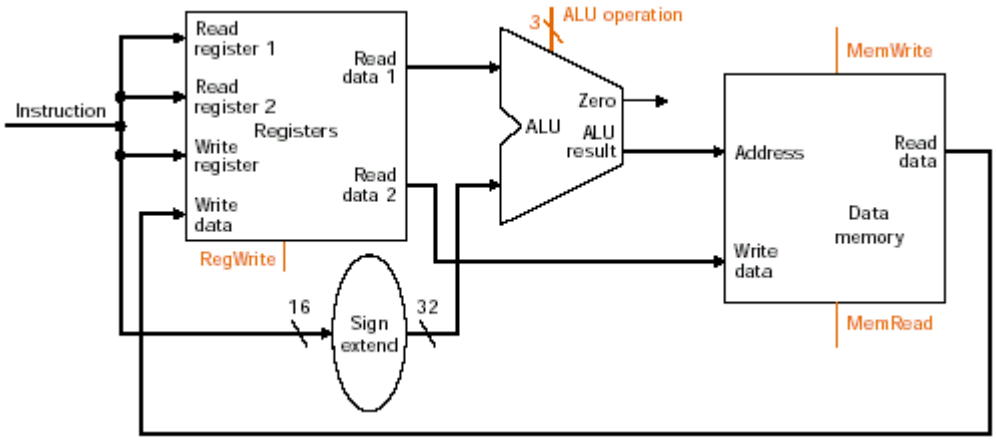
Επιπλέον των παραπάνω, θα χρειαστούμε τώρα μια υπομονάδα προέκτασης προσήμου, η οποία να προεκτείνει τη σταθερά των 16 bits σε 32 bits, καθώς και μια μονάδα μνήμης, τη

μνήμη δεδομένων. Πιο συγκεκριμένα, η μνήμη δεδομένων διαβάζεται σε εντολές φόρτωσης, και γράφεται σε εντολές αποθήκευσης. Έτσι, εκτός της εισόδου διευθύνσεων και της εξόδου δεδομένων, χρειάζεται μια είσοδο δεδομένων για εντολές αποθήκευσης, και εισόδους επίτρεψης ανάγνωσης και εγγραφής. Σχηματικά, τα δύο νέα στοιχεία απεικονίζονται παρακάτω.



a. Data memory unit b. Sign-extension unit

Στο διάγραμμα που ακολουθεί φαίνεται πώς τα δύο νέα στοιχεία συνδέονται με τα προηγούμενα για την εκτέλεση εντολών προσπέλασης μνήμης MIPS, και πάλι χωρίς να συμπεριλάβουμε τις υπομονάδες ανάκλησης των εντολών. Εκτός από τους αριθμούς καταχωρητών, η λέξη εντολής παρέχει και τη σταθερά μετατόπισης, η οποία, αφού περάσει από την υπομονάδα προέκτασης προσήμου, καταλήγει στη δεύτερη είσοδο της ΑΛΜ.



Η εντολή διακλάδωσης beq του συνόλου εντολών MIPS που μας απασχολεί, έχει 3 τελούμενα, δύο καταχωρητές που συγκρίνονται για την αποτίμηση της συνθήκης άλματος, και μια σταθερά μετατόπισης που δίνει τη διεύθυνση προορισμού του άλματος σχετικά με τη διεύθυνση της εντολής διακλάδωσης. Η μορφή της εντολής αυτής είναι η

```
beq $t1,$t2,offset
```

όπου $t1 και $t2 είναι καταχωρητές, και offset μια προσημασμένη σταθερά μετατόπισης μεγέθους 16 bits. Για την εκτέλεση αυτής της εντολής πρέπει να υπολογιστεί η διεύθυνση προορισμού, με πρόσθεση της σταθεράς μετατόπισης – μετά από προέκταση προσήμου – με την τιμή του PC. Ιδιαίτερη προσοχή χρειάζονται τα ακόλουθα χαρακτηριστικά που έχουν εξ ορισμού οι εντολές διακλάδωσης MIPS:

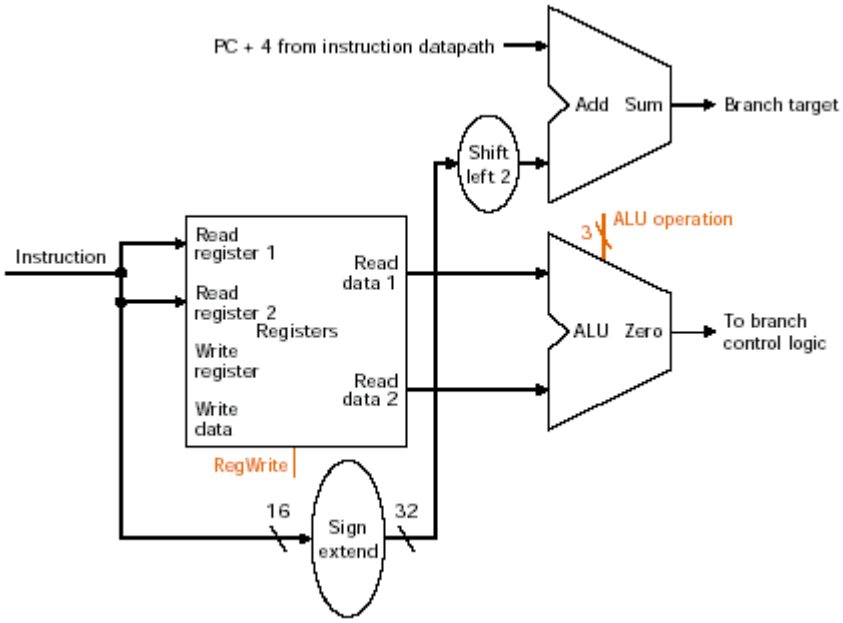
- Η σταθερά μετατόπισης δίνει τη διεύθυνση προορισμού του άλματος σχετικά με τη διεύθυνση της εντολής *που ακολουθεί* την εντολή διακλάδωσης, και όχι της ίδιας της εντολής διακλάδωσης. Από τη στιγμή που ήδη υπολογίζουμε τη διεύθυνση της επόμενης εντολής στις υπομονάδες ανάκλησης, είναι απλό να χρησιμοποιήσουμε την έξοδο του αντίστοιχου αθροιστή σαν είσοδο της υπομονάδας υπολογισμού της διεύθυνσης προορισμού άλματος.
- Η μετατόπιση δίνει μια σχετική διεύθυνση λέξης, και όχι ψηφιολέξης. Επομένως, η μετατόπιση δε λαμβάνεται αυτούσια από τη λέξη εντολής, αλλά ολισθαίνει δύο θέσεις αριστερά, πριν χρησιμοποιηθεί για τον παραπάνω υπολογισμό, ώστε να μας δώσει τε-

λικά μια διεύθυνση ψηφιολέξης[2]. Ας σημειωθεί ότι με τον τρόπο αυτό το εύρος διευθύνσεων προορισμού που καλύπτονται από μια εντολή beq είναι τετραπλάσιο από αυτό που θα ήταν, εάν η μετατόπιση έδινε σχετική διεύθυνση ψηφιολέξης.

Για την αντιμετώπιση του δεύτερου από τα πιο πάνω θέματα, εισάγουμε στη ΜΕΔ μια υπομονάδα αριστερής ολίσθησης κατά δύο θέσεις μεταξύ της εξόδου της υπομονάδας προέκτασης προσήμου και της ΑΛΜ.

Επιπλέον του υπολογισμού της διεύθυνσης προορισμού άλματος, πρέπει να καθορίσουμε εάν η εντολή που θα ακολουθήσει θα είναι όντως η εντολή της διεύθυνσης προορισμού, ή η εντολή που ακολουθεί την εντολή διακλάδωσης. Εάν η συνθήκη άλματος (ισότητα τιμών των δύο καταχωρητών) είναι αληθής, η διεύθυνση προορισμού τοποθετείται στον PC, και λέμε ότι το άλμα *εκτελείται*. Σε διαφορετική περίπτωση, η αυξημένη τιμή του PC είναι αυτή που θα εισαχθεί πίσω στον PC, και λέμε ότι το άλμα *δεν εκτελείται*. Αυτό συμβαίνει και με κάθε εντολή χωρίς άλμα.

Επομένως, η ΜΕΔ που εκτελεί μια διακλάδωση πρέπει να υλοποιεί δύο λειτουργίες: τον υπολογισμό της διεύθυνσης προορισμού και την αποτίμηση της συνθήκης άλματος, όπως φαίνεται στο διάγραμμα που ακολουθεί. Για την πρώτη λειτουργία περιλαμβάνουμε μια υπομονάδα προέκτασης προσήμου, μια υπομονάδα αριστερής ολίσθησης[3] και έναν αθροιστή. Για τη δεύτερη λειτουργία χρησιμοποιούμε το ΦΚ για ανάγνωση, και την ΑΛΜ. Ειδικότερα, με την πράξη της αφαίρεσης στην ΑΛΜ, μπορούμε να χρησιμοποιήσουμε την έξοδο μηδενικής τιμής της ΑΛΜ σα σήμα ισότητας των δύο τελούμενων που διαβάστηκαν από το ΦΚ. Αν το σήμα αυτό είναι λογικά ενεργό, και η εντολή που εκτελείται είναι εντολή beq, τότε το άλμα θα εκτελεστεί. Το πώς ακριβώς υλοποιείται η εκτέλεση του άλματος θα το δούμε αργότερα.



Τέλος, για να ολοκληρώσουμε τη μελέτη των εντολών MIPS που επιθυμούμε να υποστηρίξουμε, θα σημειώσουμε ότι μια εντολή άλματος j εκτελείται απλά με αντικατάσταση των 28 λιγότερο σημαντικών ψηφίων του PC με τα 26 λιγότερο σημαντικά ψηφία της λέξης εντολής, ολισθημένα αριστερά κατά 2 θέσεις.

Στη συνέχεια μπορούμε να συνδυάσουμε τις ΜΕΔ που σχεδιάσαμε για καθένα τύπο εντολής MIPS σε μια κοινή ΜΕΔ, προσθέτοντας σ' αυτήν και την απαιτούμενη ΜΕ. Θα ξεκινήσουμε με τη σχεδίαση μιας ΜΕΔ που εκτελεί κάθε εντολή σε ένα μεγάλο κύκλο μηχανής, και στη συνέχεια θα δούμε μια ΜΕΔ που εκτελεί μάθε εντολή σε πολλαπλούς συντομότερους κύκλους μηχανής.

---

[2] Η ολίσθηση αποφεύγεται αν ο PC περιέχει διεύθυνση λέξης και όχι ψηφιολέξης.

[3] Η αριστερή ολίσθηση μιας λέξης κατά 2 θέσεις δεν απαιτεί στην πραγματικότητα ειδική υπομονάδα, επειδή υλοποιείται απλά με προσθήκη 2 ψηφίων 0 σα λιγότερο σημαντικά ψηφία της λέξης.

5.3 Μια απλή υλοποίηση

Στην ενότητα αυτή θα δούμε μια υλοποίηση που θα μπορούσε να χαρακτηρισθεί ως η απλούστερη δυνατή υλοποίηση των εντολών MIPS που μελετάμε. Κατασκευάζουμε την απλή ΜΕΔ και μια αντίστοιχη ΜΕ, συνδυάζοντας τις ΜΕΔ που σχεδιάσαμε νωρίτερα, και προσθέτοντας κατάλληλα σήματα ελέγχου. Έτσι, με την απλή αυτή υλοποίηση, θα καλύψουμε τις εντολές προσπέλασης μνήμης lw και sw, διακλάδωσης beq, και αριθμητικών/λογικών πράξεων add, sub, and, or και slt, ενώ στο τέλος θα προσθέσουμε υποστήριξη για την εντολή άλματος j.

Κατασκευάζοντας μια κοινή ΜΕΔ για όλες τις εντολές

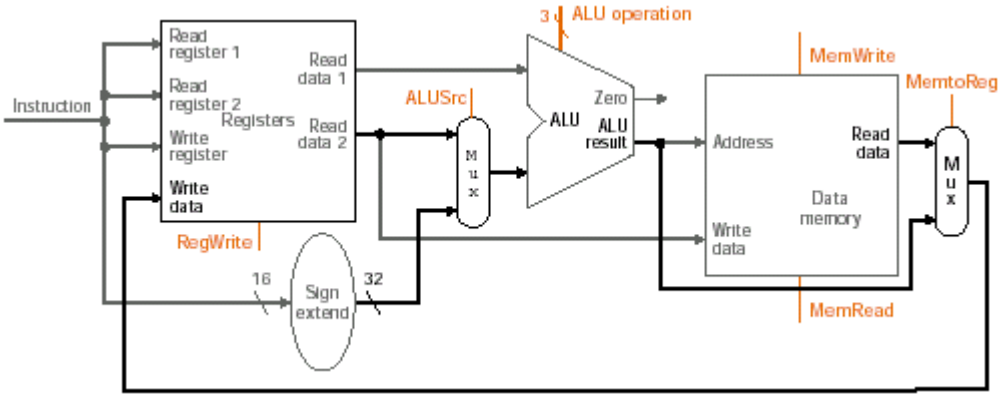
Έστω ότι θέλουμε να σχεδιάσουμε μια ΜΕΔ με βάση τις επιμέρους ΜΕΔ που σχεδιάσαμε πιο πριν. Στην απλούστερη περίπτωση η σύνδεση αυτών οδηγεί σε μια κοινή ΜΕΔ, στην οποία η εκτέλεση κάθε εντολής ολοκληρώνεται σε έναν κύκλο μηχανής. Αυτό σημαίνει ότι δε μπορούμε να χρησιμοποιήσουμε καμία υπομονάδα της ΜΕΔ περισσότερες από μία φορές στην εκτέλεση της ίδιας εντολής, κι έτσι οποτεδήποτε χρειαζόμαστε την ίδια υπομονάδα πολλαπλές φορές σε μια εντολή, την εισάγουμε στη ΜΕΔ σε ισάριθμα αντίγραφα. Για το λόγο αυτό, για παράδειγμα, χρειαζόμαστε διαφορετική μνήμη εντολών από μνήμη δεδομένων. Από την άλλη μεριά, όμως, συνδέοντας τις επιμέρους ΜΕΔ που έχουμε σχεδιάσει, δεν είναι απαραίτητο να εισάγουμε πολλαπλά αντίγραφα για τις υπομονάδες που εμφανίζονται σε περισσότερες της μίας ΜΕΔ, επειδή οι πολλαπλές χρήσεις αυτών γίνονται από εντολές διαφορετικού τύπου.

Για να μπορέσει η ίδια υπομονάδα της ΜΕΔ να χρησιμοποιείται από εντολές διαφορετικού τύπου, όταν οι διαφορετικές χρήσεις αυτής γίνονται με διαφορετικές εισόδους, πρέπει να μπορούμε να επιτρέψουμε συνδέσεις πολλαπλών εισόδων στην υπομονάδα αυτή, με δυνατότητα επιλογής της κατάλληλης εισόδου με τη βοήθεια ειδικών σημάτων ελέγχου. Μια τέτοια επιλογή επιτυγχάνεται συνήθως με ένα κύκλωμα *πολυπλέκτη*. Αυτό επιλέγει μία είσοδο από ένα σύνολο γραμμών εισόδου, με την επιλογή να γίνεται με βάση την τιμή που έχει η είσοδος επιλογής του πολυπλέκτη.

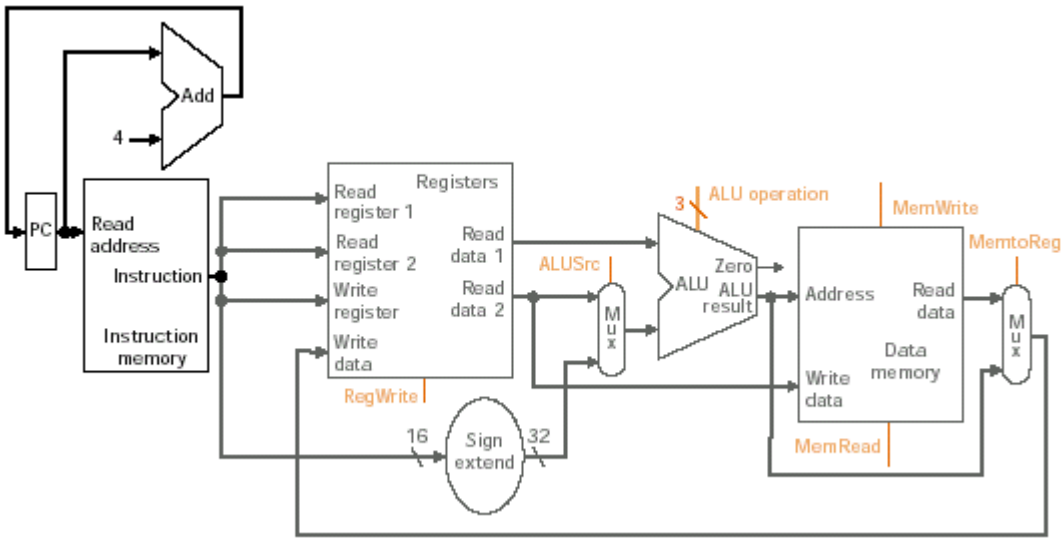
Για παράδειγμα, για να ενοποιήσουμε σε μία κοινή ΜΕΔ τις δύο ΜΕΔ που σχεδιάσαμε για R-εντολές και εντολές προσπέλασης μνήμης, πρέπει να μελετήσουμε τις διαφορές μεταξύ των δύο ΜΕΔ, που είναι οι εξής:

- Η δεύτερη είσοδος της ΑΛΜ προέρχεται από το ΦΚ για τις R-εντολές, και από την υπομονάδα προέκτασης προσήμου για τις εντολές προσπέλασης μνήμης.
- Η τιμή που αποθηκεύεται στο ΦΚ προέρχεται από την ΑΛΜ για τις R-εντολές, και από τη μνήμη για εντολές φόρτωσης.

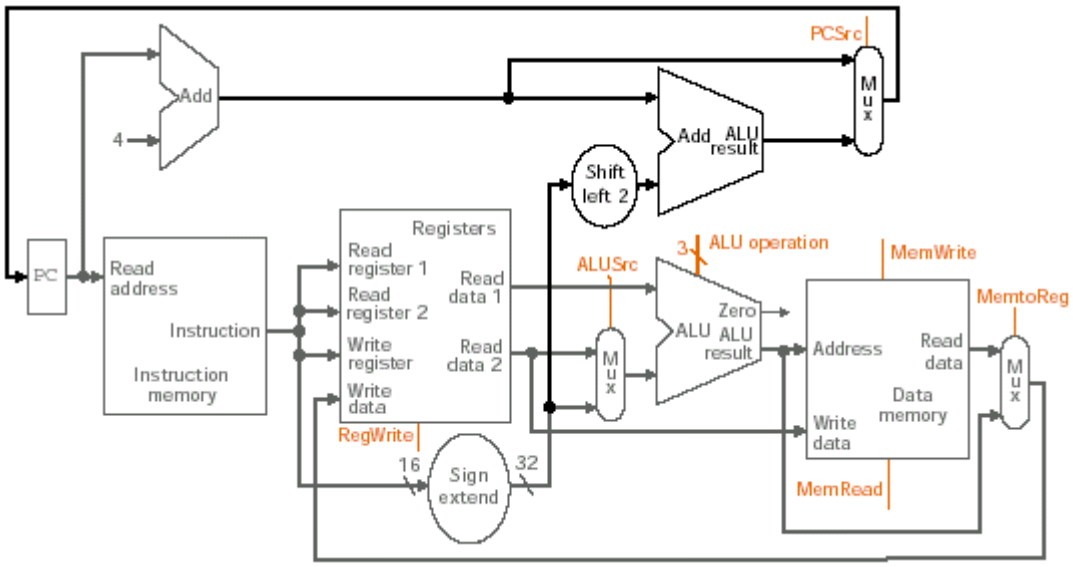
Παρατηρούμε λοιπόν ότι η ενοποίηση των δύο ΜΕΔ παρουσιάζει πρόβλημα στη δεύτερη είσοδο της ΑΛΜ και στην είσοδο δεδομένων του ΦΚ. Η ενοποίηση των δύο ΜΕΔ χωρίς τη χρήση πολλαπλών αντιγράφων των δύο υπομονάδων όπου εμφανίζονται τα δύο παραπάνω προβλήματα, απαιτεί έναν πολυπλέκτη για τη δεύτερη είσοδο της ΑΛΜ κι ένα δεύτερο πολυπλέκτη για την είσοδο δεδομένων του ΦΚ. Η νέα ΜΕΔ δίνεται στο παρακάτω διάγραμμα.



Στη συνέχεια, μπορούμε να προσθέσουμε στη ΜΕΔ τις υπομονάδες ανάκλησης εντολών. Παρατηρούμε ότι οι δύο μονάδες μνήμης, εντολών και δεδομένων, δε μπορούν να ενοποιηθούν σε μία, επειδή η προσπέλαση της μνήμης εντολών γίνεται για όλες τις εντολές, συμπεριλαμβανομένων των εντολών προσπέλασης μνήμης. Ενοποίηση των δύο μνημών θα οδηγούσε σε μια μονάδα που θα χρησιμοποιείτο δύο φορές στην ίδια εντολή, από όλες τις εντολές προσπέλασης μνήμης. Για παρόμοιο λόγο, δε μπορούμε να ενοποιήσουμε τον αθροιστή που αυξάνει την τιμή του PC με την ΑΛΜ. Το διάγραμμα της ΜΕΔ που προκύπτει δίνεται παρακάτω.



Τέλος, ενσωματώνουμε στην παραπάνω ΜΕΔ και τη ΜΕΔ που σχεδιάσαμε για εντολές διακλάδωσης. Η ΜΕΔ εντολών διακλάδωσης χρησιμοποιεί την ΑΛΜ που έχουμε για τη σύγκριση των τελούμενων και την αποτίμηση της συνθήκης άλματος, ενώ ο υπολογισμός της διεύθυνσης προορισμού άλματος πρέπει να γίνεται σε ανεξάρτητη υπομονάδα. Ένας νέος πολυπλέκτης απαιτείται για την επιλογή εισόδου στον PC. Το τελικό διάγραμμα της ΜΕΔ είναι το πιο κάτω.



Τώρα που ολοκληρώσαμε τη ΜΕΔ, θα προσθέσουμε τη ΜΕ. Η ΜΕ, με κατάλληλες εισόδους, παρέχει τα σήματα επίτρεψης ανάγνωσης και εγγραφής για κάθε στοιχείο κατάστασης του επεξεργαστή, τα σήματα επιλογής για κάθε πολυπλέκτη, και το σήμα ελέγχου της ΑΛΜ. Ο έλεγχος της ΑΛΜ είναι διαφορετικός από το συνολικό έλεγχο του επεξεργαστή, και είναι σκόπιμο να ξεκινήσουμε με αυτόν.

Ο έλεγχος της ΑΛΜ

Η ΑΛΜ, όπως αναφέραμε παραπάνω, έχει σαν είσοδο ελέγχου ένα σήμα μεγέθους 3 bits. Αυτό το σήμα καθορίζει ποια από τις 5 επιτρεπτές πράξεις εκτελείται στην ΑΛΜ, σε καθεμιά από τις εντολές που υλοποιούμε. Προφανώς, το σήμα ελέγχου μπορεί να καθορίσει μέχρι 8 πράξεις, κι επομένως 3 συνδυασμοί τιμών του μένουν αχρησιμοποίητες. Θεωρούμε ότι οι συνδυασμοί που χρησιμοποιούνται και οι αντίστοιχες πράξεις είναι οι εξής:

Σήμα Ελέγχου	Πράξη
000	AND
001	OR
010	ADD
110	SUB
111	SLT

Ανάλογα με τον τύπο εντολής, η ΑΛΜ θα εκτελέσει μια από τις 5 παραπάνω πράξεις. Για εντολές προσπέλασης μνήμης η ΑΛΜ χρησιμοποιείται για τον υπολογισμό της τελικής διεύθυνσης προσπέλασης, εκτελώντας την πράξη της πρόσθεσης ADD. Για R-εντολές η ΑΛΜ εκτελεί την πράξη που καθορίζει ο κωδικός τελεστή της εντολής. Για την εντολή διακλάδωσης beq, τέλος, η ΑΛΜ εκτελεί την πράξη της αφαίρεσης SUB.

Μπορούμε να κατασκευάσουμε μια μικρή μονάδα ελέγχου για την ΑΛΜ, η οποία θα δέχεται σαν είσοδο 2 ψηφία που καθορίζουν τον τύπο της εντολής, καθώς και τα 6 ψηφία του κωδικού τελεστή των εντολών MIPS, ενώ θα παράγει σαν έξοδο τα 3 ψηφία ελέγχου που θέλουμε. Έστω ALUOp το σήμα των 2 ψηφίων εισόδου που δίνουν τον τύπο της εντολής. Όταν αυτή είναι εντολή προσπέλασης μνήμης, το σήμα ALUOp θα είναι “00”, και η πράξη ΑΛΜ θα είναι ADD. Όταν η εντολή είναι η εντολή διακλάδωσης beq, το σήμα ALUOp θα είναι “01”, και η πράξη θα είναι SUB. Τέλος, όταν έχουμε R-εντολή, το σήμα ALUOp θα είναι “10”, και η πράξη θα καθορίζεται από τα υπόλοιπα 6 ψηφία εισόδου. Όπως είπαμε νωρίτερα, η έξοδος της μονάδας ελέγχου της ΑΛΜ είναι το σήμα των 3 ψηφίων που καθορίζουν την πράξη που εκτελείται, σύμφωνα με τον παραπάνω πίνακα.

Συνολικά, η λειτουργία της μονάδας ελέγχου της ΑΛΜ περιγράφεται στον πιο κάτω πίνακα, όπου δίνονται οι τιμές του σήματος ελέγχου της ΑΛΜ με βάση τα 2 bits του σήματος ALUOp και τα 6 bits του κωδικού τελεστή της εντολής. Το πώς ακριβώς λαμβάνουμε το σήμα ALUOp από τη λέξη εντολής θα το δούμε αργότερα.

Τύπος εντολής	ALUOp	Εντολή	Κωδικός τελεστή	Πράξη	Σήμα ελέγχου
Προσπέλαση μνήμης	00	LW	XXXXXX	ADD	010
Προσπέλαση μνήμης	00	SW	XXXXXX	ADD	010
Διακλάδωση	01	BEQ	XXXXXX	SUB	110
R-εντολή	10	ADD	100000	ADD	010
R-εντολή	10	SUB	100010	SUB	110
R-εντολή	10	AND	100100	AND	000
R-εντολή	10	OR	100101	OR	001
R-εντολή	10	SLT	101010	SLT	111

Η υλοποίηση της ΜΕ σε πολλαπλά επίπεδα (σε ανώτερο επίπεδο η κεντρική ΜΕ, και σε κατώτερα επίπεδα οι ΜΕ των υπομονάδων της ΜΕΔ, οι οποίες δέχονται σαν είσοδο σήματα εξόδου της κεντρικής ΜΕ – όπως για παράδειγμα το σήμα ALUOp που είδαμε πιο πάνω) αποτελεί κοινή τεχνική. Τα πολλαπλά επίπεδα ελέγχου μειώνουν το συνολικό μέγεθος της ΜΕ. Επίσης, χρησιμοποιώντας πολλές μικρές ΜΕ για τις υπομονάδες της ΜΕΔ, αυξάνουμε τη συνολική ταχύτητα της ΜΕ. Η βελτιστοποίηση που γίνεται με την κατανομή της ΜΕ σε πολλαπλά επίπεδα είναι σημαντική, επειδή σε πολλές περιπτώσεις ο έλεγχος επιφέρει τη μεγαλύτερη καθυστέρηση στη λειτουργία του επεξεργαστή.

Υπάρχουν αρκετοί διαφορετικοί τρόποι που μετατρέπουν τα 8 ψηφία εισόδου στα 3 ψηφία εξόδου της ΜΕ της ΑΛΜ. Επειδή από τις 64 δυνατές τιμές του κωδικού τελεστή ένα πολύ μικρό μέρος χρησιμοποιείται, και μάλιστα χρησιμοποιείται μόνο όταν το σήμα ALUOp είναι "10", μπορούμε να σχεδιάσουμε ένα απλό κύκλωμα που να αναγνωρίζει τους συνδυασμούς των ψηφίων του σήματος ALUOp και του κωδικού τελεστή που θέλουμε, και να παράγει της σωστές τιμές του ζητούμενου σήματος ελέγχου.

Το βασικό βήμα σχεδίασης του κυκλώματος αυτού είναι η κατασκευή ενός πίνακα αλήθειας που από τα 8 ψηφία εισόδου μας δίνει τα ζητούμενα 3 ψηφία εξόδου της ΜΕ της ΑΛΜ. Επειδή ο πλήρης πίνακας αλήθειας για 8 ψηφία εισόδου έχει 2^8=256 γραμμές, δίνουμε μόνο τις γραμμές του πίνακα που παρέχουν χρήσιμες τιμές εξόδου. Οι υπόλοιπες γραμμές αντιστοιχούν σε αδιάφορες συνθήκες, μια που είτε δεν πρόκειται ποτέ να εμφανιστούν στην πράξη, είτε παράγουν σήματα που δεν επηρεάζουν τελικά την κατάσταση του επεξεργαστή[4]. Ο πίνακας αλήθειας δίνεται στη συνέχεια:

ALUOp		**Κωδικός τελεστή (F)**						**Σήμα ελέγχου**
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

Επειδή σε αρκετές περιπτώσεις κάποιες από τις εισόδους είναι αδιάφορες, ο πίνακας αλήθειας περιλαμβάνει και αδιάφορες τιμές εισόδου. Μια τέτοια τιμή (που συμβολίζεται με 'X' στην αντίστοιχη στήλη του πίνακα) υποδηλώνει ότι η έξοδος δεν εξαρτάται από τη συγκεκριμένη είσοδο σε εκείνη τη γραμμή. Για παράδειγμα, όταν το σήμα ALUOp είναι "00", η έξοδος έχει τιμή "010", ανεξάρτητα από τα ψηφία του κωδικού τελεστή. Τότε, ο κωδικός τελεστή αποτελεί αδιάφορη τιμή για την αντίστοιχη γραμμή του πίνακα αλήθειας. Αργότερα θα δούμε κι άλλες περιπτώσεις αδιάφορων τιμών εισόδου.

Από τη στιγμή που ο πίνακας αλήθειας έχει κατασκευαστεί, η βελτιστοποίησή του και η σχεδίαση του λογικού κυκλώματος που τον υλοποιεί είναι διαδικασίες γνωστές που μπορούν να εφαρμοστούν με τη βοήθεια ειδικών εργαλείων λογικής σχεδίασης.

Η σχεδίαση της κεντρικής ΜΕ

Τώρα που περιγράψαμε πώς να σχεδιάσουμε μια ΑΛΜ που χρησιμοποιεί τον κωδικό τελεστή και το σήμα ελέγχου ALUOp για τον έλεγχό της, θα επιστρέψουμε στη σχεδίαση της κεντρικής ΜΕ. Για να ξεκινήσουμε, θα εξετάσουμε από τη μία τη μορφή της λέξης εντολής MIPS, και από την άλλη τα σήματα ελέγχου που έχουμε εισάγει στη ΜΕΔ που σχεδιάσαμε νωρίτερα. Για να κατανοήσουμε πώς τα πεδία της λέξης εντολής συνδέονται στη ΜΕΔ, ας δούμε τα πεδία αυτά για τις εντολές που μας απασχολούν, δηλαδή τις R-εντολές, τις εντολές προσπέλασης μνήμης και την εντολή διακλάδωσης beq:

Πεδίο	0	rs	rt	rd	sh	F
Ψηφία	31-26	25-21	20-16	15-11	10-6	5-0

α. R-εντολές

[4] Οι δύο αυτές συνθήκες πρέπει να ικανοποιούνται, ώστε να μπορούμε να αγνοήσουμε τις υπόλοιπες γραμμές του πίνακα αλήθειας. Στη γενικότερη περίπτωση, συμπεριλαμβάνουμε τις γραμμές, για τις οποίες αυτές δεν ικανοποιούνται.

Πεδίο	35 ή 43	rs	rt	offset
Ψηφία	31-26	25-21	20-16	15-0

β. Εντολές προσπέλασης μνήμης

Πεδίο	4	rs	rt	offset
Ψηφία	31-26	25-21	20-16	15-0

γ. Εντολή διακλάδωσης beq

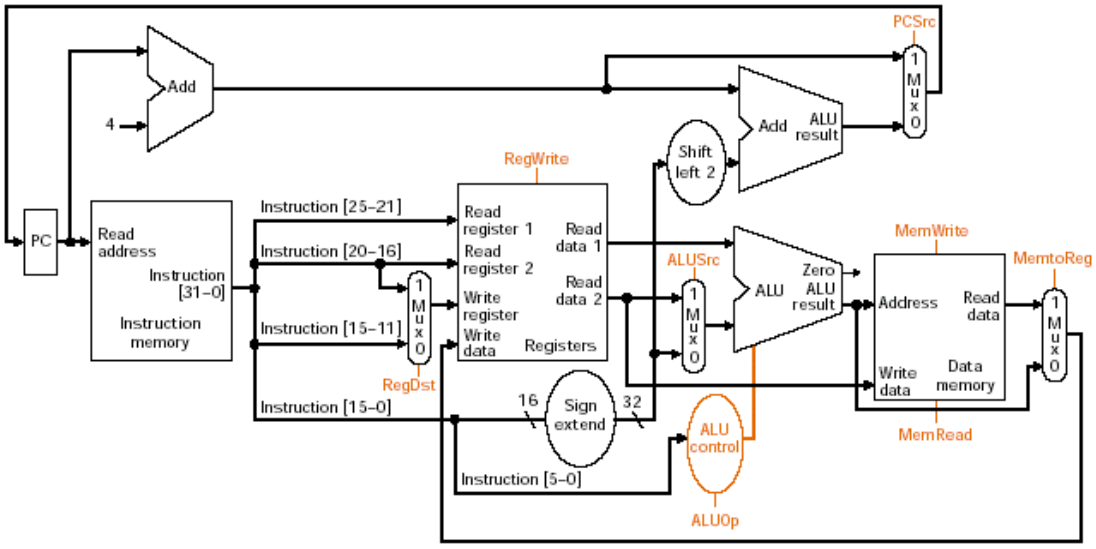
Θα στηρίξουμε τη σχεδίαση της ΜΕ στις ακόλουθες παρατηρήσεις πάνω στη μορφή της λέξης εντολής:

- Ο κωδικός λειτουργίας περιέχεται πάντα στα πιο σημαντικά ψηφία 31-26. Το πεδίο αυτό έχει όνομα Op, ή σα διάνυσμα ψηφίων Op[5-0].
- Οι καταχωρητές για ανάγνωση καθορίζονται πάντα από τα πεδία rs και rt, τα οποία κωδικοποιούνται πάντα στα ψηφία 25-21 και 20-16, αντίστοιχα. Ειδικά για την εντολή φόρτωσης, η τιμή που διαβάζεται από τον καταχωρητή rt δε χρησιμοποιείται.
- Ο καταχωρητής βάσης για τις εντολές προσπέλασης μνήμης είναι ο rs και βρίσκεται πάντα στα ψηφία 25-21.
- Η σταθερά μετατόπισης offset βρίσκεται πάντα στις θέσεις 15-0.
- Ο καταχωρητής αποθήκευσης καθορίζεται άλλοτε από τα ψηφία 15-11 (πεδίο rd), και άλλοτε από τα ψηφία 20-16 (πεδίο rt). Η δεύτερη περίπτωση χρησιμοποιείται στην εντολή φόρτωσης lw, ενώ η πρώτη σε όλες τις R-εντολές. Η επιλογή του αριθμού καταχωρητή αποθήκευσης γίνεται με τη βοήθεια πολυπλέκτη.

Η ακριβής σύνδεση της λέξης εντολής στη ΜΕΔ φαίνεται στο διάγραμμα της επόμενης σελίδας, όπου έχουμε επίσης προσθέσει τη ΜΕ της ΑΛΜ, ενώ έχουμε διατηρήσει και όλα τα άλλα σήματα ελέγχου που είχαμε νωρίτερα. Ας σημειωθεί ότι τα σήματα ελέγχου των πολυπλεκτών έχουν μέγεθος 1 bit, εφ' όσον επιλέγουν μία από δύο εισόδους.

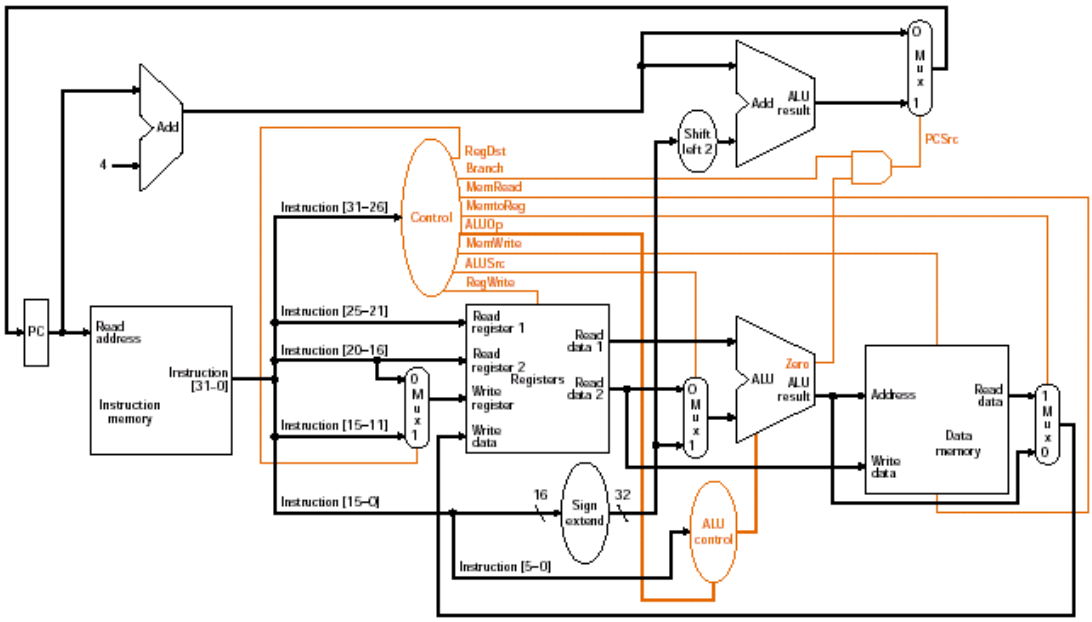
Όπως φαίνεται στο διάγραμμα, έχουμε συνολικά επτά σήματα ελέγχου μεγέθους 1 και ένα σήμα ελέγχου μεγέθους 2 bits. Το σήμα ελέγχου ALUOp – που είναι και το μοναδικό σήμα των 2 bits – έχει περιγραφεί νωρίτερα. Πριν προχωρήσουμε στη σύνθεση της ΜΕ, η οποία θα παράγει τα σήματα αυτά κατά την εκτέλεση των εντολών, ας δούμε περιληπτικά τη λειτουργία των υπόλοιπων επτά σημάτων, μέσα από τον πίνακα που ακολουθεί.

Σήμα	**Λειτουργία απενεργοποίησης**	**Λειτουργία ενεργοποίησης**
RegDst	Ο αριθμός καταχωρητή αποθήκευσης προέρχεται από το πεδίο rt (ψηφία 20-16).	Ο αριθμός καταχωρητή αποθήκευσης προέρχεται από το πεδίο rd (ψηφία 15-11).
RegWrite	Καμία.	Ο καταχωρητής αποθήκευσης γράφεται από την είσοδο δεδομένων του ΦΚ.
ALUSrc	Η δεύτερη είσοδος της ΑΛΜ προέρχεται από το ΦΚ.	Η δεύτερη είσοδος της ΑΛΜ προέρχεται από την υπομονάδα προέκτασης προσήμου.
PCSrc	Η είσοδος του PC προέρχεται από τον αθροιστή που αυξάνει την τιμή του.	Η είσοδος του PC προέρχεται από τον αθροιστή που υπολογίζει τη διεύθυνση προορισμού άλματος.
MemRead	Καμία.	Η μνήμη δεδομένων διαβάζεται από τη διεύθυνση της εισόδου διευθύνσεων.
MemWrite	Καμία.	Η μνήμη δεδομένων γράφεται από την είσοδο δεδομένων στη διεύθυνση της εισόδου διευθύνσεων.
MemtoReg	Η τιμή που αποθηκεύεται στο ΦΚ προέρχεται από την ΑΛΜ.	Η τιμή που αποθηκεύεται στο ΦΚ προέρχεται από τη μνήμη δεδομένων.



Από τη λειτουργία των σημάτων ελέγχου μπορούμε να βρούμε πώς να τα υλοποιήσουμε στη ΜΕ. Μάλιστα, μπορούμε να διαπιστώσουμε ότι η ενεργοποίηση όλων των σημάτων γίνεται με βάση τη λέξη εντολής και μόνο – και συγκεκριμένα τον κωδικό λειτουργίας της λέξης εντολής. Εξαίρεση αποτελεί το σήμα PCSrc, το οποίο ενεργοποιείται λογικά, όταν ο κωδικός λειτουργίας αντιστοιχεί στην εντολή διακλάδωσης beq *και* η έξοδος μηδενικής τιμής της ΑΛΜ (Zero) είναι ενεργοποιημένη. Για την ενεργοποίηση του σήματος αυτού επομένως, θα χρειαστεί να περάσουμε από μια πύλη λογικού ΚΑΙ ένα σήμα Branch από την κεντρική ΜΕ, το οποίο θα ενεργοποιείται όταν η εντολή είναι η beq, και το σήμα Zero.

Σύμφωνα με τα παραπάνω, όλα τα σήματα ελέγχου που χρησιμοποιούμε, εκτός του PCSrc, αλλά συμπεριλαμβανομένου του Branch, προέρχονται από την κεντρική ΜΕ, η οποία υλοποιείται σαν λογικό κύκλωμα με είσοδο τα 6 ψηφία του κωδικού λειτουργίας της λέξης εντολής και έξοδο τα σήματα αυτά. Ένα ολοκληρωμένο διάγραμμα της ΜΕΔ που περιλαμβάνει και τη ΜΕ φαίνεται πιο κάτω.



Πριν προσπαθήσουμε να εξάγουμε τις λογικές εξισώσεις ή τον πίνακα αλήθειας για τα σήματα ελέγχου που παράγει η ΜΕ, είναι σκόπιμο να εξηγήσουμε τη λειτουργία της ΜΕ λιγότερο επίσημα. Επειδή η ενεργοποίηση των σημάτων ελέγχου που παράγει η ΜΕ εξαρτάται αποκλειστικά από τον κωδικό λειτουργίας της λέξης εντολής, σχηματίζουμε τον πιο κάτω

πίνακα που μας δείχνει εάν ένα σήμα ελέγχου λαμβάνει τιμή 0, 1 ή X (αδιάφορη τιμή), ανάλογα με τον κωδικό λειτουργίας της εντολής:

Εντολή	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
R-εντολή	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

Λειτουργία της ΜΕΔ

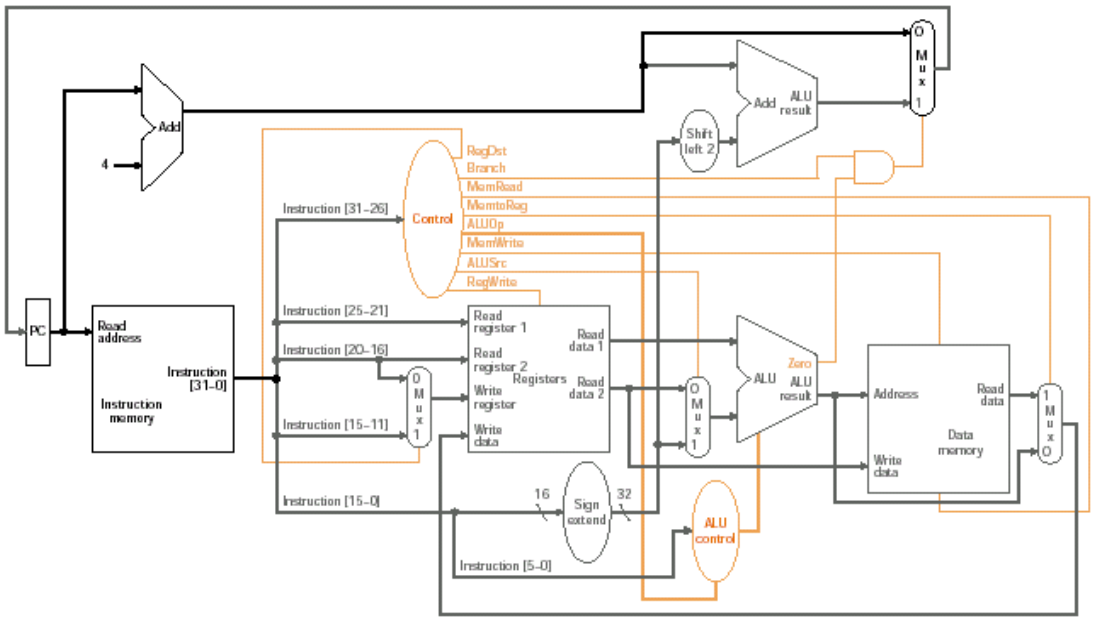
Με την πληροφορία που έχουμε συγκεντρώσει, μπορούμε να σχεδιάσουμε επακριβώς τη ΜΕ της υλοποίησης που έχουμε επιλέξει. Πριν από αυτό, θα δούμε πώς κάθε τύπος εντολής χρησιμοποιεί τη ΜΕΔ μέσω των παραπάνω σημάτων ελέγχου. Με τη βοήθεια των διαγραμμάτων που ακολουθούν, θα δείξουμε τη ροή πληροφορίας μέσα από τη ΜΕΔ για τους 3 τύπους εντολών που υλοποιήσαμε. Κάθε διάγραμμα τονίζει τα ενεργοποιημένα σήματα ελέγχου, καθώς και τις υπομονάδες της ΜΕΔ που χρησιμοποιούνται σε κάθε φάση εκτέλεσης της εντολής.

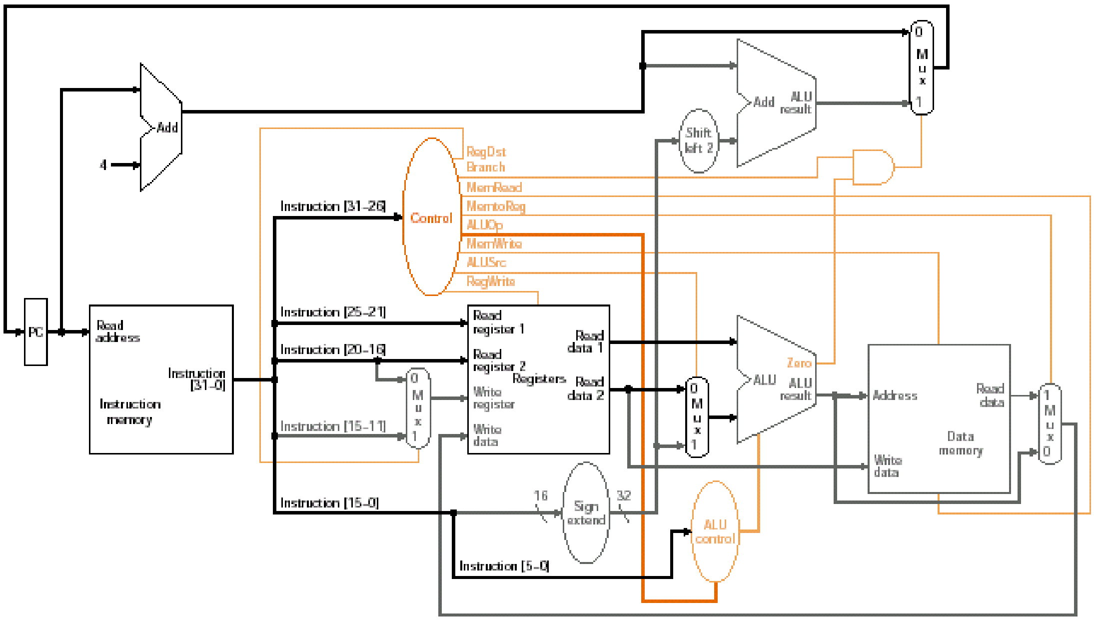
Ας ξεκινήσουμε, όπως συνήθως, με μια R-εντολή, όπως για παράδειγμα την

```
add $t1,$t2,$t3
```

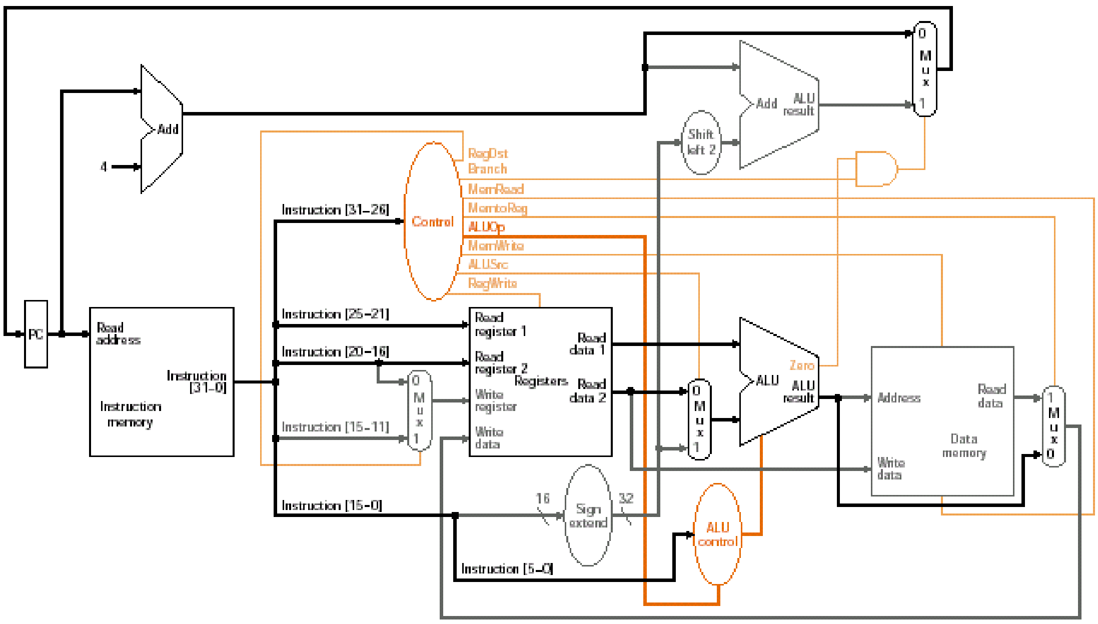
όπου $t1, $t2 και $t3 τρεις καταχωρητές του ΦΚ. Παρ' όλο που στην παρούσα υλοποίηση η ΜΕΔ δεν είναι παρά ένα μεγάλο συνδυαστικό κύκλωμα που εκτελεί κάθε εντολή σε έναν κύκλο μηχανής με είσοδο από, και έξοδο στα στοιχεία κατάστασης της ΜΕΔ, είναι πιο εύκολο να κατανοήσουμε τη λειτουργία της ΜΕΔ, μελετώντας κάθε φάση του κύκλου εντολής ξεχωριστά από τις υπόλοιπες. Έτσι, διακρίνουμε 4 φάσεις στον κύκλο μιας R-εντολής:

1. Η εντολή ανακαλείται από τη μνήμη εντολών, και ο PC αυξάνεται. Τα τμήματα της ΜΕΔ που συμμετέχουν στη φάση αυτή φαίνονται με σκούρο χρώμα στο πρώτο από τα διαγράμματα που ακολουθούν.
2. Δύο καταχωρητές, ο $t2 και ο $t3, διαβάζονται από το ΦΚ. Η ΜΕ παράγει τα σήματα ελέγχου που απαιτούνται για την εκτέλεση της εντολής. Στο δεύτερο διάγραμμα τονίζονται και οι υπομονάδες που συμμετέχουν στη φάση αυτή.
3. Η ΑΛΜ εκτελεί μια πράξη, την οποία καθορίζει η ΜΕ της ΑΛΜ, με βάση τον κωδικό τελεστή της εντολής. Το τρίτο διάγραμμα τονίζει και τις νέες λειτουργίες.
4. Το αποτέλεσμα της πράξης μεταφέρεται από την έξοδο της ΑΛΜ στο ΦΚ, επιλέγοντας σαν καταχωρητή αποθήκευσης αυτόν που καθορίζεται από τα ψηφία 15-11 της λέξης εντολής (στην περίπτωσή μας ο καταχωρητής $t1). Το τελευταίο διάγραμμα τονίζει όλες τις λειτουργίες που αντιστοιχούν στην R-εντολή.

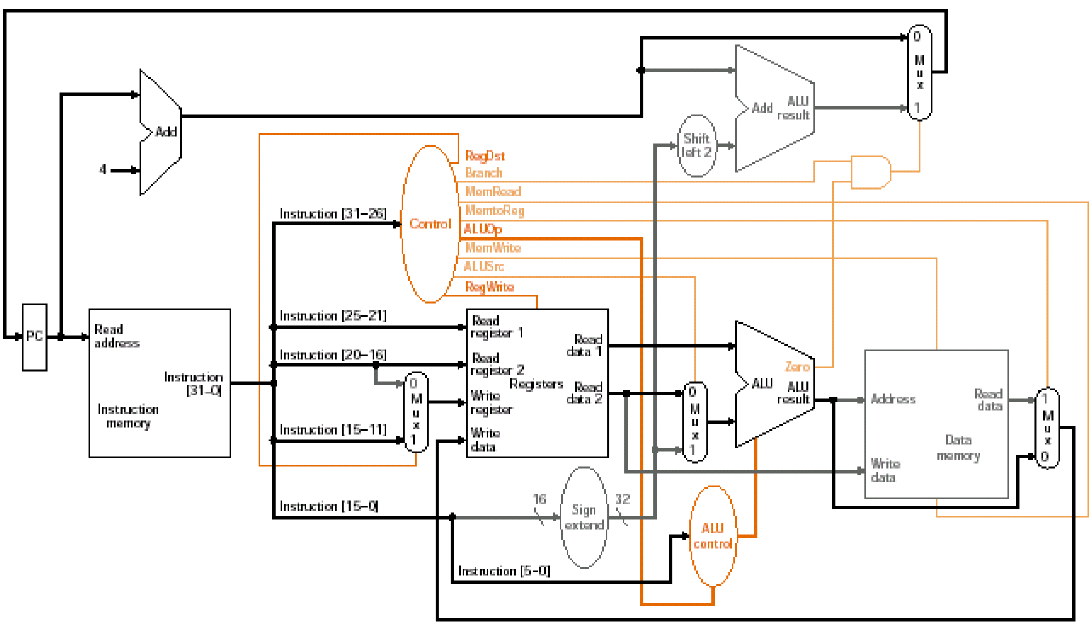




0
M
u
x
1
Add
4
Add
ALU result
Shift left 2
PC
Read address
Instruction [31–0]
Instruction memory
Control
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign extend
32
ALU
Zero
ALU result
ALU control
Address
Read data
Data memory
Write data



0
M
u
x
1
Add
4
Add
ALU result
Shift left 2
PC
Read address
Instruction [31–0]
Instruction memory
Control
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign extend
32
ALU
Zero
ALU result
ALU control
Address
Read data
Data memory
Write data



0
M
u
x
1
Add
4
Add
ALU result
Shift left 2
PC
Read address
Instruction [31–0]
Instruction memory
Control
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign extend
32
ALU
Zero
ALU result
ALU control
Address
Read data
Data memory
Write data

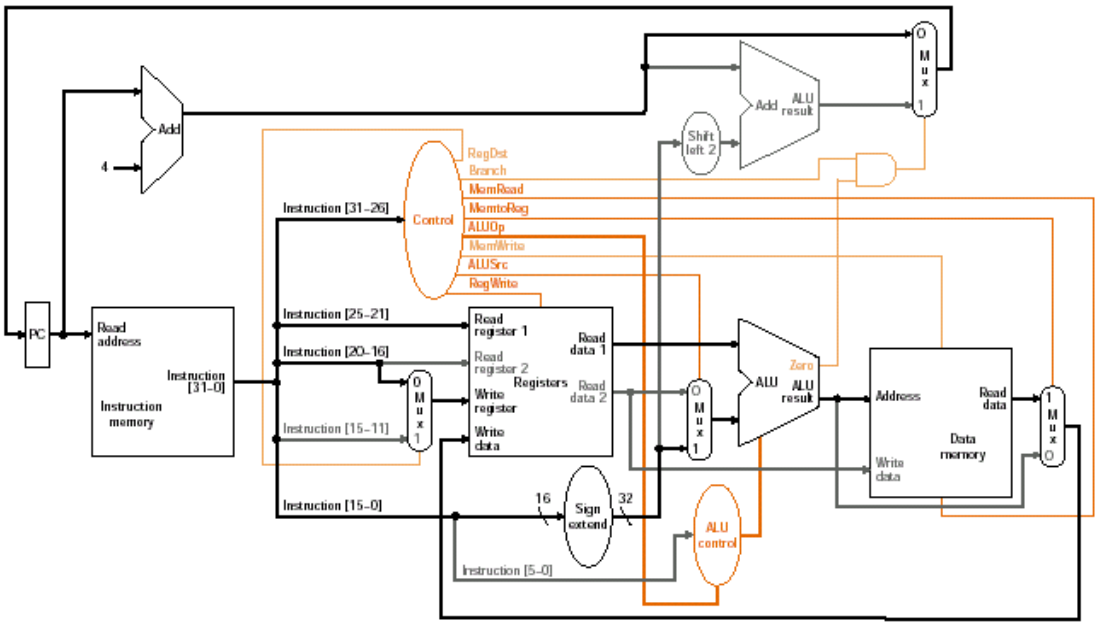
Υπενθυμίζουμε ότι στην παρούσα υλοποίηση δεν διακρίνονται οι παραπάνω φάσεις μεταξύ τους, επειδή η ΜΕΔ εκτελεί κάθε εντολή σε ένα και μόνο κύκλο μηχανής. Τα σήματα που διαδίδονται στη ΜΕΔ μεταβάλλονται κατά τη διάρκεια του ωρολογιακού παλμού, και σταθεροποιούνται σε μια τελική τιμή περίπου σύμφωνα με τη σειρά των τεσσάρων φάσεων που περιγράψαμε, απλά επειδή η ροή της πληροφορίας στη ΜΕΔ ακολουθεί αυτή τη σειρά. Έτσι, το τελευταίο διάγραμμα δείχνει, όχι μόνο τις λειτουργίες της τελευταίας φάσης, αλλά στην πραγματικότητα τις λειτουργίες της συνολικής ΜΕΔ κατά τη διάρκεια ενός κύκλου μηχανής.

Με παρόμοιο τρόπο μπορούμε να δείξουμε τις λειτουργίες της ΜΕΔ για μια εντολή φόρτωσης, όπως για παράδειγμα την

```
lw $t1,offset($t2)
```

όπου $t1 και $t2 δύο καταχωρητές του ΦΚ, και offset η σταθερά μετατόπισης. Το διάγραμμα που ακολουθεί τονίζει τις υπομονάδες της ΜΕΔ που συμμετέχουν στην εκτέλεση της παραπάνω εντολής. Μπορούμε να φανταστούμε την εντολή φόρτωσης να εκτελείται σε 5 φάσεις:

1. Η εντολή ανακαλείται και ο PC αυξάνεται.
2. Ένας καταχωρητής, ο $t2, διαβάζεται από το ΦΚ.
3. Η ΑΛΜ υπολογίζει την τελική διεύθυνση προσπέλασης αθροίζοντας την τιμή που διαβάστηκε από το ΦΚ με την προέκταση προσήμου της σταθεράς μετατόπισης offset.
4. Η έξοδος της ΑΛΜ στέλνεται στην είσοδο διευθύνσεων της μνήμης δεδομένων και μια λέξη διαβάζεται από αυτήν.
5. Η λέξη που φορτώθηκε από τη μνήμη δεδομένων αποθηκεύεται στο ΦΚ, και συγκεκριμένα στον καταχωρητή που καθορίζεται από τα ψηφία 20-16 της λέξης εντολής (εδώ ο $t1).

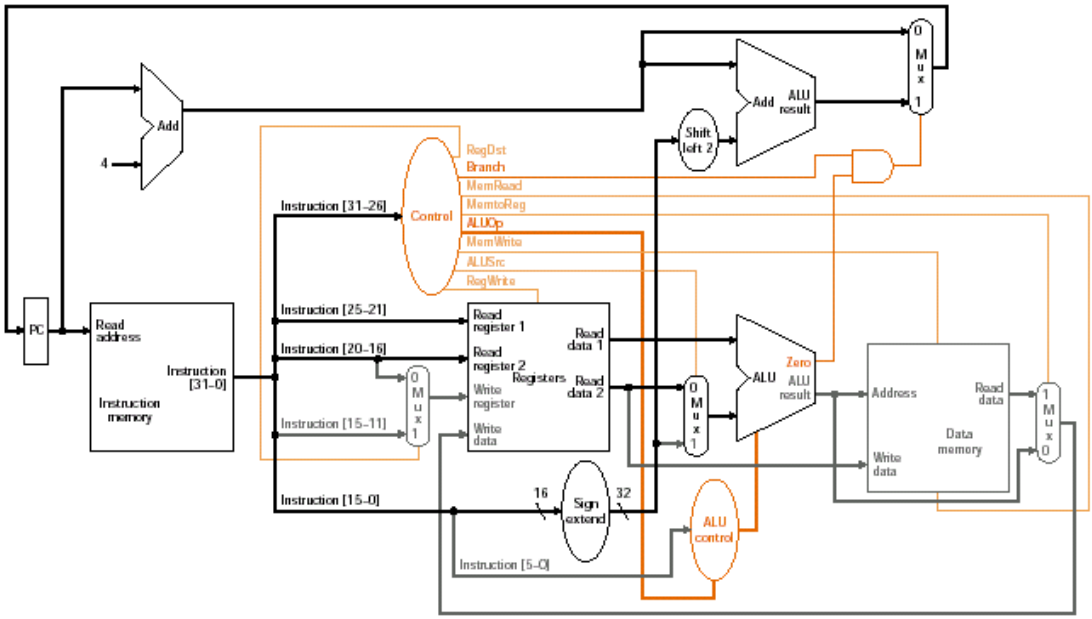


Τέλος, ολοκληρώνουμε την ανάλυση της εκτέλεσης των εντολών στη ΜΕΔ που σχεδιάσαμε, με την εκτέλεση μιας εντολής διακλάδωσης, για παράδειγμα την

```
beq $t1,$t2,offset
```

όπου $t1 και $t2 δύο καταχωρητές του ΦΚ, και offset σταθερά μετατόπισης. Η εντολή διακλάδωσης εκτελείται περίπου σαν R-εντολή, με τη διαφορά ότι η έξοδος της ΑΛΜ δεν αποθηκεύεται, αλλά χρησιμοποιείται για την παραγωγή του σήματος PCSrc. Το αντίστοιχο διάγραμμα ακολουθεί, ενώ οι 4 φάσεις της εντολής είναι οι εξής:

1. Η εντολή ανακαλείται και ο PC αυξάνεται.
2. Δύο καταχωρητές, οι $t1 και $t2, διαβάζονται από το ΦΚ.
3. Η ΑΛΜ εκτελεί αφαίρεση (SUB) μεταξύ των τιμών που διαβάστηκαν από το ΦΚ. Ταυτόχρονα, η αυξημένη τιμή του PC προστίθεται στο αποτέλεσμα της προέκτασης προσήμου της σταθεράς μετατόπισης, ολισθημένου αριστερά κατά δύο θέσεις, με αποτέλεσμα της πρόσθεσης αυτής τη διεύθυνση προορισμού άλματος.



4. Η έξοδος Zero της ΑΛΜ χρησιμοποιείται για την παραγωγή του σήματος PCSrc που θα καθορίσει την είσοδο του PC.

Στην επόμενη υλοποίηση της ΜΕΔ θα δούμε κάθε μία από τις παραπάνω φάσεις εκτέλεσης των εντολών MIPS που υποστηρίζουμε να καταλαμβάνει έναν ξεχωριστό κύκλο μηχανής.

Ολοκλήρωση της σχεδίασης της ΜΕ

Ας συνεχίσουμε τώρα με την υλοποίηση της κεντρικής ΜΕ. Όλη η πληροφορία που χρειαζόμαστε για να ορίσουμε τις λογικές συναρτήσεις των σημάτων ελέγχου βρίσκεται στον πίνακα που δώσαμε νωρίτερα. Οι έξοδοι της ΜΕ είναι τα σήματα ελέγχου, ενώ οι είσοδοι είναι τα 6 ψηφία του κωδικού λειτουργίας της λέξης εντολής. Επομένως, μπορούμε να κατασκευάσουμε άμεσα τον πίνακα αλήθειας για κάθε ένα από τα σήματα ελέγχου της ΜΕ. Ας δούμε όμως πρώτα τους κωδικούς λειτουργίας Op[5-0] για τις εντολές που μας απασχολούν:

Εντολή	Κωδικός	Op5	Op4	Op3	Op2	Op1	Op0
R-εντολή	0_{10}	0	0	0	0	0	0
lw	35_{10}	1	0	0	0	1	1
sw	43_{10}	1	0	1	0	1	1
beq	4_{10}	0	0	0	1	0	0

Με βάση την κωδικοποίηση αυτή, μπορούμε να εκφράσουμε τις λογικές συναρτήσεις της ΜΕ μέσα από ένα μεγάλο πίνακα αλήθειας που περιγράφει όλα τα σήματα ελέγχου για τις αντίστοιχες εισόδους. Ο πίνακας αυτός δίνεται στην επόμενη σελίδα, και αρκεί για την άμεση υλοποίηση της ΜΕ σε ένα λογικό κύκλωμα, όπως μπορεί να γίνει με τη βοήθεια κατάλληλου εργαλείου.

Θα κλείσουμε την παρούσα παράγραφο, προσθέτοντας την εντολή άλματος j στις βασικές ΜΕΔ και ΜΕ που υλοποιήσαμε, ώστε να δείξουμε πώς μπορούμε να επεκτείνουμε την παραπάνω υλοποίηση με πρόσθετες εντολές MIPS.

Η λέξη εντολής άλματος έχει την ακόλουθη μορφή:

Πεδίο	2	address
Ψηφία	31-26	25-0

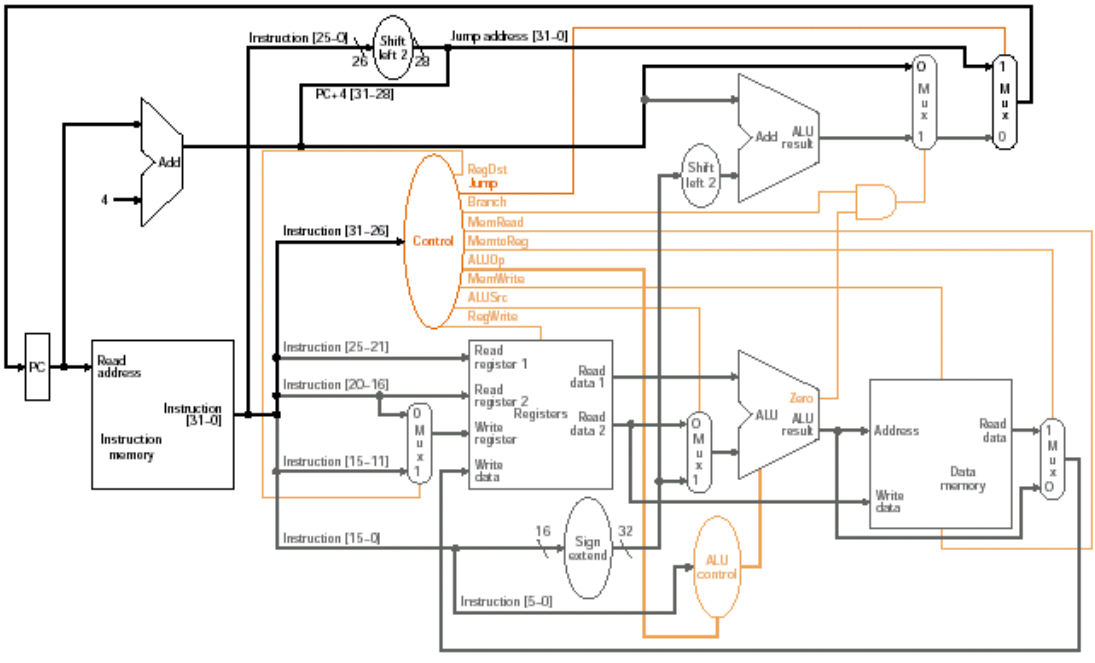
Είσοδος/Εξοδος	Σήμα	R-εντολή	lw	sw	beq
Είσοδοι	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Έξοδοι	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

Η εντολή άλματος j μοιάζει με εντολή διακλάδωσης, αλλά υπολογίζει τη διεύθυνση προορισμού με διαφορετικό τρόπο, ενώ δεν έχει συνθήκη στην εκτέλεση του άλματος. Όπως στην εντολή beq, τα δύο λιγότερο σημαντικά ψηφία της διεύθυνσης προορισμού είναι πάντα τα 00. Τα επόμενα 26 ψηφία της διεύθυνσης προορισμού λαμβάνονται απ' ευθείας από το πεδίο address της λέξης εντολής. Τα υπόλοιπα ψηφία είναι τα τέσσερα πιο σημαντικά ψηφία της διεύθυνσης της εντολής που ακολουθεί την εντολή άλματος.

Η υλοποίηση της εντολής j συνίσταται στην προσθήκη δυνατότητας εισόδου στον PC μιας τιμής που προκύπτει από την παράθεση:

- των 4 πιο σημαντικών ψηφίων της τιμής του αυξημένου PC,
- των 26 ψηφίων του πεδίου address της λέξης εντολής, και
- των δύο ψηφίων 00.

Η προσθήκη του ελέγχου της εντολής j φαίνεται στο παρακάτω διάγραμμα. Ένας πρόσθετος πολυπλέκτης χρησιμοποιείται για την επιλογή της εισόδου του PC, που τώρα πρέπει να είναι μία από: την αυξημένη τιμή του PC, τη διεύθυνση προορισμού εντολής διακλάδωσης, και τη διεύθυνση προορισμού εντολής άλματος. Ο νέος πολυπλέκτης επιλέγει τη σωστή τιμή με τη βοήθεια του νέου σήματος ελέγχου Jump, το οποίο είναι λογικά ενεργό όταν η εντολή είναι η j, δηλαδή όταν ο κωδικός λειτουργίας στη λέξη εντολής έχει τιμή 2_{10}.



Γιατί δε χρησιμοποιείται η υλοποίηση απλού κύκλου μηχανής

Αν και η παραπάνω υλοποίηση δουλεύει σωστά, δε χρησιμοποιείται σε μοντέρνες αρχιτεκτονικές, επειδή έχει χαμηλή απόδοση. Για να το κατανοήσουμε αυτό, ξεκινάμε με την παρατήρηση ότι η διάρκεια εκτέλεσης κάθε εντολής που υλοποιείται είναι ένας κύκλος μηχανής. Εφ' όσον οι παλμοί του ρολογιού που ορίζουν τον κύκλο μηχανής έχουν σταθερή περίοδο, η υλοποίηση απλού κύκλου μηχανής εμφανίζει αριθμό CPI ίσο με 1. Όμως, η διάρκεια του κύκλου μηχανής καθορίζεται από τη μακρύτερη διαδρομή πληροφορίας στον επεξεργαστή. Στην περίπτωσή μας, αυτή η διαδρομή αντιστοιχεί στην εντολή φόρτωσης lw, η οποία χρησιμοποιεί διαδοχικά 5 υπομονάδες της ΜΕΔ: τη μνήμη εντολών, το ΠΚ, την ΑΛΜ, τη μνήμη δεδομένων και πάλι το ΠΚ. Παρ' όλο που ο αριθμός CPI για όλες τις εντολές είναι 1, ο κύκλος μηχανής έχει υπερβολικά μεγάλη διάρκεια, καθώς όλες οι υπόλοιπες εντολές ολοκληρώνουν την εκτέλεσή τους σε χρόνο μικρότερο από το χρόνο ενός κύκλου μηχανής στον οποίο εκτελείται η εντολή φόρτωσης. Έτσι, η συνολική απόδοση της υλοποίησης αυτής δεν αναμένεται να είναι ιδιαίτερα υψηλή.

Παράδειγμα 1

Έστω ότι ο χρόνος λειτουργίας των κυριότερων υπομονάδων της υλοποίησης του απλού κύκλου μηχανής είναι:

- *Προσπέλαση μονάδων μνήμης: 2ns.*
- *Πράξη ΑΛΜ και αθροιστών: 2ns.*
- *Προσπέλαση ΠΚ: 1ns.*

Αν οι πολυπλέκτες, η ΜΕ, οι προσπελάσεις του PC, η προέκταση προσήμου και η μεταφορά των σημάτων επιφέρουν αμελητέα καθυστέρηση στην εκτέλεση των εντολών, συγκρίνετε την απόδοση των δύο υλοποιήσεων απλού κύκλου μηχανής, όπου:

1. *Ο κύκλος μηχανής έχει σταθερή διάρκεια.*
2. *Ο κύκλος μηχανής έχει μεταβλητή διάρκεια με ιδανικό τρόπο, δηλαδή ίση με όση απαιτείται για την εκτέλεση της τρέχουσας εντολής.*

Αν και η δεύτερη υλοποίηση είναι φανταστική, θα μας επιτρέψει να δούμε πόσος χρόνος χάνεται, όταν όλες οι εντολές ολοκληρώνονται στον ίδιο σταθερό χρόνο.

Για τον υπολογισμό της απόδοσης των δύο υλοποιήσεων, υποθέστε ότι έχετε το ακόλουθο μίγμα εντολών: 24% εντολές φόρτωσης, 12% εντολές αποθήκευσης, 44% R-εντολές, 18% εντολές διακλάδωσης και 2% εντολές άλματος.

Ο υπολογισμός του χρόνου εκτέλεσης ενός προγράμματος γίνεται με βάση τη σχέση:

Χρόνος εκτέλεσης = Αριθμός εντολών × CPI × Χρόνος κύκλου μηχανής

Εφ' όσον CPI = 1:

Χρόνος εκτέλεσης = Αριθμός εντολών × Χρόνος κύκλου μηχανής

Για τον ίδιο αριθμό εντολών και CPI, η σύγκριση θα γίνει με βάση το μέσο χρόνο κύκλου μηχανής. Για να υπολογίσουμε τον τελευταίο, σχηματίζουμε κατ' αρχήν τον ακόλουθο πίνακα που μας δείχνει ποιες από τις παραπάνω υπομονάδες χρησιμοποιούνται από κάθε είδος εντολής:

Εντολή	**Υπομονάδες κάθε φάσης του κύκλου εντολής**				
R-εντολή	Μνήμη εντολών	ΠΚ	ΑΛΜ	ΠΚ	
Φόρτωσης	Μνήμη εντολών	ΠΚ	ΑΛΜ	Μνήμη δεδομένων	ΠΚ
Αποθήκευσης	Μνήμη εντολών	ΠΚ	ΑΛΜ	Μνήμη δεδομένων	
Διακλάδωσης	Μνήμη εντολών	ΠΚ	ΑΛΜ		
Άλματος	Μνήμη εντολών				

Λαμβάνοντας υπ' όψη τους χρόνους λειτουργίας κάθε υπομονάδας, ο παραπάνω πίνακας μας δίνει:

Εντολή	Μνήμη εντολών	ΦΚ	ΑΛΜ	Μνήμη δεδομένων	ΦΚ	Σύνολο
R-εντολή	2	1	2	0	1	6ns
Φόρτωσης	2	1	2	2	1	8ns
Αποθήκευσης	2	1	2	2	0	7ns
Διακλάδωσης	2	1	2	0	0	5ns
Άλματος	2	0	0	0	0	2ns

Για την υλοποίηση σταθερού κύκλου μηχανής, η διάρκειά του καθορίζεται από τη μέγιστη διάρκεια εκτέλεσης εντολής, που όπως φαίνεται από τον πίνακα, είναι 8ns.

Για τη δεύτερη υλοποίηση, η διάρκεια του κύκλου μηχανής θα μεταβάλλεται από 2ns μέχρι 8ns. Ο μέσος χρόνος κύκλου μηχανής για το μίγμα εντολών που μας δίνεται θα είναι:

$$\text{Χρόνος κύκλου μηχανής} = 24\% \times 8 + 12\% \times 7 + 44\% \times 6 + 18\% \times 5 + 2\% \times 2$$
$$= 6.3\text{ns}$$

Είναι φανερό ότι η υλοποίηση με το χαμηλότερο χρόνο κύκλου μηχανής έχει υψηλότερη απόδοση. Ο λόγος απόδοσης ΛΑ μεταξύ των δύο υλοποιήσεων είναι:

$$\text{ΛΑ} = \text{Χρόνος εκτέλεσης}^{(\text{σταθερού κύκλου μηχανής})} / \text{Χρόνος εκτέλεσης}^{(\text{μεταβλητού κύκλου μηχανής})}$$
$$= \text{Χρόνος κύκλου μηχανής}^{(\text{σταθερός})} / \text{Χρόνος κύκλου μηχανής}^{(\text{μεταβλητός})}$$
$$= 8 / 6.3$$
$$= 1.27$$

Μ' άλλα λόγια, η δεύτερη υλοποίηση είναι 1.27 φορές πιο γρήγορη από την πρώτη. Δυστυχώς, ο μεταβλητός χρόνος κύκλου μηχανής είναι στην πράξη μη υλοποιήσιμος, γιατί θα απαιτούσε τόση καθυστέρηση στον υπολογισμό του χρόνου κύκλου μηχανής της κάθε εντολής, που θα ξεπερνούσε το κέρδος από τη συντόμευσή του. Γενικά, αντί να υλοποιούμε μεταβλητό χρόνο κύκλου μηχανής, υλοποιούμε μεταβλητό αριθμό κύκλων μηχανής ανά εντολή, με σταθερό βασικό κύκλο μηχανής, κάτι που θα μελετήσουμε πιο κάτω.□

Η επιβάρυνση στην απόδοση ενός επεξεργαστή από την υλοποίηση απλού κύκλου μηχανής για κάθε εντολή είναι μεγάλη, αλλά μπορεί να μας φαίνεται αποδεκτή για το υποσύνολο εντολών MIPS που υποστηρίζουμε. Αν όμως προσπαθήσουμε να ενσωματώσουμε πιο πολύπλοκες υπομονάδες στη ΜΕΔ, όπως για παράδειγμα υπομονάδες πράξεων κινητής υποδιαστολής, η επιβάρυνση θα είναι πραγματικά μεγάλη. Έτσι, εάν επαναλάβουμε το προηγούμενο παράδειγμα, συμπεριλαμβάνοντας μερικές απλές εντολές κινητής υποδιαστολής, θα βρίσκαμε ότι η απόδοση της υλοποίησης με μεταβλητό χρόνο κύκλου μηχανής είναι σχεδόν 3 φορές υψηλότερη από την απόδοση της πρώτης υλοποίησης.

Στην πραγματικότητα, για την υποστήριξη άλλων πολύπλοκων εντολών, ή ακόμα για την υποστήριξη πολύπλοκων διευθυνσιοδοτήσεων, μια εντολή θα μπορούσε να απαιτεί μέχρι εκατοντάδες νανοδευτερόλεπτα για την ολοκλήρωσή της. Επειδή ο χρόνος του κύκλου μηχανής καθορίζεται, όπως είπαμε νωρίτερα, από το μέγιστο χρόνο ολοκλήρωσης μιας εντολής, ο οποίος μπορεί να αντιστοιχεί σε κάποια πολύ σπάνια χρησιμοποιούμενη εντολή, οποιαδήποτε προσπάθεια επίτευξης υψηλής απόδοσης του επεξεργαστή πρέπει να γίνεται με σχεδίαση υψηλής ταχύτητας, ακόμα και για υλικό που χρησιμοποιείται πολύ σπάνια. Κάτι τέτοιο παραβιάζει μια βασική αρχή σχεδίασης που ακολουθούμε, την αρχή της σχεδίασης υψηλότερης ταχύτητας για τις πιο κοινές περιπτώσεις.

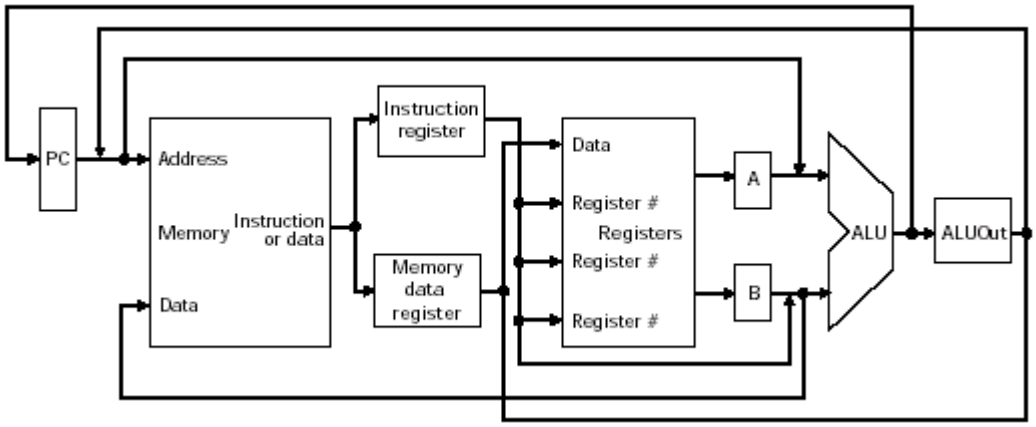
Ακόμα, με την υλοποίηση του απλού κύκλου μηχανής ανά κύκλο εντολής, κάθε υπομονάδα της ΜΕΔ χρησιμοποιείται μέχρι μία φορά σε κάθε κύκλο εντολής. Αυτό σημαίνει ότι κάποιες υπομονάδες πρέπει να υλοποιούνται σε περισσότερα του ενός αντίγραφα στη ΜΕΔ. Κάτι τέτοιο μπορεί να αυξήσει απαγορευτικά το κόστος του επεξεργαστή. Άρα η υλοποίηση αυτή δεν έχει μόνο χαμηλή απόδοση, αλλά και υψηλό κόστος!

Τα προαναφερθέντα προβλήματα αντιμετωπίζονται με μια υλοποίηση πολλαπλών κύκλων μηχανής για κάθε κύκλο εντολής, με χρόνο κύκλου μηχανής που καθορίζεται από το χρόνο λειτουργίας της πιο πολύπλοκης υπομονάδας της ΜΕΔ ή της ΜΕ, και όχι από το συνολικό χρόνο ολοκλήρωσης της πιο πολύπλοκης εντολής. Την υλοποίηση αυτή θα τη μελετήσουμε στη συνέχεια.

5.4 Μια υλοποίηση πολλαπλών κύκλων μηχανής ανά κύκλο εντολής

Νωρίτερα, διασπάσαμε την εκτέλεση κάθε εντολής σε ένα αριθμό φάσεων, που κάθε μία χρησιμοποιούσε μια ή περισσότερες συγκεκριμένες υπομονάδες της ΜΕΔ ή της ΜΕ. Η υλοποίηση πολλαπλών κύκλων μηχανής στηρίζεται σε αυτές τις φάσεις, έτσι ώστε σ' αυτήν την υλοποίηση, κάθε φάση του κύκλου εντολής ολοκληρώνεται σε έναν κύκλο μηχανής. Με τον τρόπο αυτό, κάποια υπομονάδα της ΜΕΔ μπορεί να χρησιμοποιηθεί περισσότερες από μία φορά στον ίδιο κύκλο εντολής, αρκεί αυτό να γίνεται σε διαφορετικούς κύκλους μηχανής.

Είναι προφανές ότι μπορούμε έτσι να πετύχουμε οικονομία στο υλικό του επεξεργαστή. Η δυνατότητα εντολών διαφορετικού τύπου να ολοκληρώνονται σε διαφορετικό αριθμό κύκλων μηχανής, όπως και η δυνατότητα επαναχρησιμοποίησης υπομονάδων της ΜΕΔ από την ίδια εντολή, είναι τα δύο βασικά πλεονεκτήματα αυτής της υλοποίησης σε σχέση με αυτή του απλού κύκλου μηχανής ανά κύκλο εντολής. Μια πρώτη αφηρημένη εικόνα της νέας ΜΕΔ που σχεδιάζουμε δίνεται στο διάγραμμα που ακολουθεί, από το οποίο, μετά από σύγκριση με το



ολοκληρωμένο διάγραμμα της υλοποίησης απλού κύκλου μηχανής που είδαμε νωρίτερα, παρατηρούμε τα εξής:

- Χρησιμοποιούμε μια μονάδα μνήμης, κοινή για εντολές και δεδομένα, αντί για δύο ξεχωριστές μονάδες.
- Χρησιμοποιούμε μια μοναδική ΑΛΜ για όλες τις πράξεις, καταργώντας τους δύο αθροιστές που υπολογίζουν τις δύο πιθανές νέες τιμές του PC.
- Ένας ή περισσότεροι καταχωρητές ειδικού σκοπού χρησιμοποιούνται στη νέα ΜΕΔ, και πιο συγκεκριμένα στις εξόδους των κυριότερων υπομονάδων της, για αποθήκευση της πληροφορίας που μεταφέρεται μεταξύ των φάσεων του κύκλου εντολής, που τώρα ολοκληρώνονται σε διαφορετικό κύκλο μηχανής.

Στο τέλος κάθε κύκλου μηχανής, κάθε πληροφορία που παράγεται από τη ΜΕΔ ή τη ΜΕ για να χρησιμοποιηθεί σε κάποιον επόμενο κύκλο, αποτελεί μέρος της κατάστασης του επεξεργαστή, κι επομένως πρέπει να αποθηκευτεί σε κάποιο στοιχείο κατάστασης. Οποιαδήποτε πληροφορία μεταφέρεται σε *επόμενη* εντολή αποθηκεύεται σε στοιχεία που προσπελαύνονται μέσα από το σύνολο εντολών του επεξεργαστή. Τα στοιχεία αυτά είναι τα στοιχεία κατάστασης που ήδη έχουμε, δηλαδή οι καταχωρητές γενικού σκοπού του ΦΚ, η μονάδα μνήμης ή ο PC. Αντίθετα, κάποια πληροφορία που χρησιμοποιείται από την *ίδια* εντολή σε κάποιον κύκλο μηχανής που ακολουθεί, χωρίς να είναι ορατή σε επόμενη εντολή, δε μπορεί να αποθηκευτεί σε κανένα από τα υπάρχοντα στοιχεία κατάστασης, κι επομένως πρέπει να αποθηκευτεί σε κάποιο νέο καταχωρητή.

Με βάση τα παραπάνω, συμπεραίνουμε ότι η θέση των νέων καταχωρητών που εισάγουμε στη ΜΕΔ καθορίζεται από δύο παράγοντες: (α) ποια στοιχεία ολοκληρώνουν τη λειτουργία τους στο τέλος ενός κύκλου, και (β) ποια πληροφορία είναι απαραίτητη για τα στοιχεία που εκτελούν τη λειτουργία τους στην αρχή κάποιου επόμενου κύκλου μηχανής της ίδιας εντολής. Όσο αφορά τον πρώτο παράγοντα, στην υλοποίησή μας υποθέτουμε ότι ένας κύκλος

μηχανής μπορεί να "χωρέσει" το πολύ μία από τις ακόλουθες λειτουργίες: μια προσπέλαση μνήμης, μια προσπέλαση του ΦΚ (για δύο αναγνώσεις και μία εγγραφή), ή μια πράξη ΑΛΜ. Επομένως, η τιμή κάθε εξόδου των τριών αντίστοιχων υπομονάδων της ΜΕΔ είναι απαραίτητο να αποθηκευτεί προσωρινά, ώστε να μπορέσει να χρησιμοποιηθεί σε κάποιον επόμενο κύκλο μηχανής.

Λόγω των παραπάνω, στη νέα ΜΕΔ έχουν εισαχθεί οι ακόλουθοι καταχωρητές ειδικού σκοπού:

- Ο καταχωρητής εντολών (IR) και ο καταχωρητής δεδομένων μνήμης (MDR) έχουν προστεθεί για την αποθήκευση της λέξης εντολής και των δεδομένων φόρτωσης από τη μνήμη, αντίστοιχα. Παρατηρήστε ότι, αντί να χρησιμοποιήσουμε έναν καταχωρητή για την αποθήκευση της τιμής εξόδου της μονάδας μνήμης, χρησιμοποιούμε δύο διαφορετικούς καταχωρητές, επειδή, όπως θα δούμε καλύτερα σε λίγο, οι δύο παραπάνω τιμές χρειάζονται ταυτόχρονα στον ίδιο κύκλο μηχανής.
- Οι καταχωρητές Α και Β έχουν προστεθεί για την αποθήκευση των δύο τιμών που διαβάζονται από το ΦΚ.
- Ο καταχωρητής C (ή ALUOut) έχει προστεθεί για την αποθήκευση της τιμής εξόδου της ΑΛΜ.

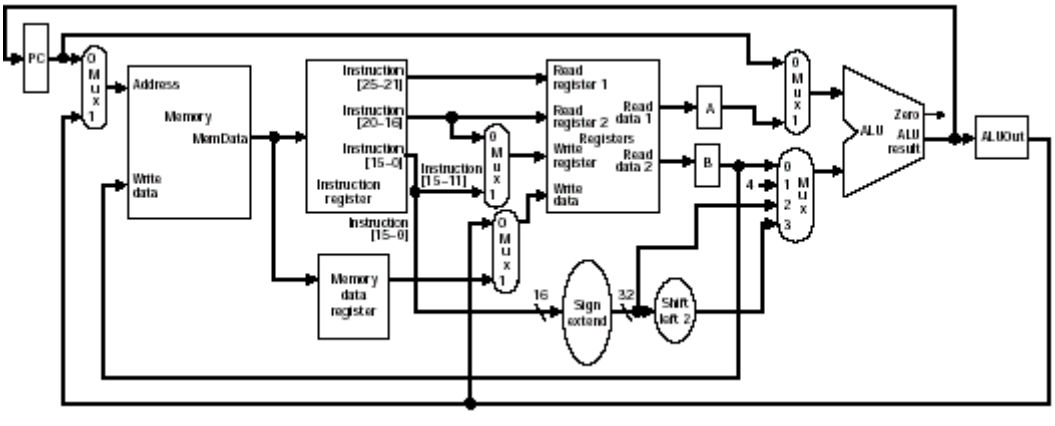
Όλοι οι παραπάνω καταχωρητές πλην του IR αποθηκεύουν κάποια τιμή που χρησιμοποιείται αποκλειστικά στον αμέσως επόμενο κύκλο μηχανής. Για το λόγο αυτό δε χρειάζονται σήμα επίτρεψης για την εγγραφή τους. Αντίθετα, ο IR διατηρεί την τιμή του – τη λέξη εντολής – για όλη τη διάρκεια της εντολής, κι επομένως απαιτεί σήμα επίτρεψης για την εγγραφή του, ώστε να αποτρέπεται η επανεγγραφή του πριν την ολοκλήρωση της εντολής. Η διαφορά αυτή στην εγγραφή των καταχωρητών που προσθέσαμε, θα γίνει περισσότερο κατανοητή μόλις δείξουμε τις λειτουργίες της ΜΕΔ ανά κύκλο μηχανής, για κάθε έναν τύπο εντολής που υλοποιούμε.

Επειδή αρκετές από τις υπομονάδες της ΜΕΔ χρησιμοποιούνται πολλαπλές φορές στην ίδια εντολή, είναι απαραίτητο να προσθέσουμε πολυπλέκτες – ή να επεκτείνουμε ήδη υπάρχοντες πολυπλέκτες – στις εισόδους τους, ώστε να επιλέγονται οι είσοδοι που χρειάζονται σε κάθε χρήση αυτών. Για παράδειγμα, εφ' όσον χρησιμοποιούμε μια μνήμη τόσο για εντολές όσο και για δεδομένα, χρειαζόμαστε έναν πολυπλέκτη για να επιλέγουμε την είσοδο διεύθυνσης είτε από τον PC (για ανάγνωση εντολής) είτε από τον C (για προσπέλαση δεδομένων).

Η αντικατάσταση των τριών υπομονάδων πράξεων (ΑΛΜ και δύο αθροιστές) με μία ΑΛΜ σημαίνει ότι οι είσοδοι που κατευθύνονταν σ' εκείνες τις υπομονάδες πρέπει τώρα να πηγαίνουν στην ΑΛΜ. Αυτό οδηγεί στις εξής διαφοροποιήσεις τις νέας ΜΕΔ από την προηγούμενη:

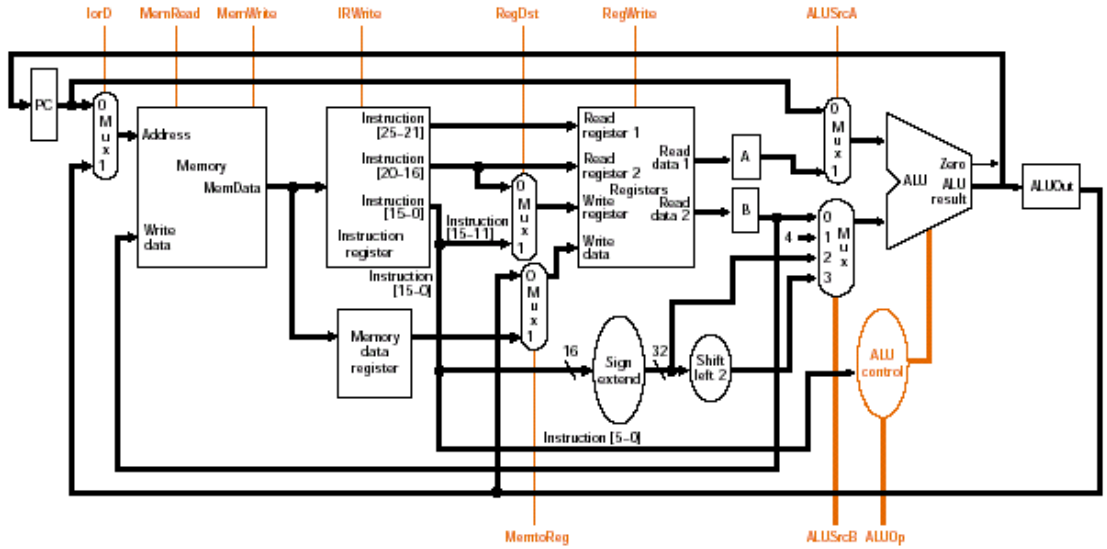
1. Ένας πρόσθετος πολυπλέκτης τοποθετείται στην πρώτη είσοδο της ΑΛΜ. Ο πολυπλέκτης αυτός χρησιμοποιείται για την επιλογή εισόδου είτε από τον καταχωρητή Α είτε από τον PC.
2. Ο πολυπλέκτης της δεύτερης εισόδου της ΑΛΜ επεκτείνεται από 2x1 σε 4x1. Οι δύο πρόσθετες επιλογές εισάγουν τη σταθερά 4 (για την αύξηση του PC) και την προέκταση προσήμου της ολισθημένης μετατόπισης (για τον υπολογισμό της διεύθυνσης προορισμού άλματος των εντολών διακλάδωσης) στην ΑΛΜ.

Οι παραπάνω τροποποιήσεις φαίνονται στο διάγραμμα που ακολουθεί. Με την προσθήκη



των νέων καταχωρητών και πολυπλεκτών μπορέσαμε να αφαιρέσουμε μια μονάδα μνήμης και δύο αθροιστές. Επειδή τα πρώτα στοιχεία είναι μικρά σε σχέση με τα δεύτερα, συμπεραίνουμε ότι η νέα ΜΕΔ είναι σημαντικά πιο οικονομική από την προηγούμενη σε κόστος υλικού.

Εφ' όσον η ΜΕΔ του παραπάνω διαγράμματος εκτελεί κάθε εντολή σε πολλαπλούς κύκλους μηχανής, απαιτεί διαφορετικά σήματα ελέγχου από τη ΜΕΔ του απλού κύκλου μηχανής. Τα στοιχεία κατάστασης απαιτούν πιθανά σήματα επίτρεψης εγγραφής, ενώ η μονάδα μνήμης απαιτεί και σήμα ανάγνωσης. Η ΑΛΜ από την άλλη μεριά, επειδή ολοκληρώνει τη λειτουργία της σε έναν κύκλο μηχανής, μπορεί να χρησιμοποιήσει τα σήματα ελέγχου που σχεδιάσαμε για την ΑΛΜ της ΜΕΔ του απλού κύκλου μηχανής. Τέλος, όλοι οι πολυπλέκτες απαιτούν αντίστοιχα σήματα επιλογής, που για πολυπλέκτες 4x1 είναι μεγέθους 2 bits. Προσθέτοντας τα σήματα ελέγχου στο παραπάνω, λαμβάνουμε το ακόλουθο διάγραμμα:



Η ΜΕΔ πολλαπλών κύκλων μηχανής χρειάζεται ακόμα την υποστήριξη διακλαδώσεων και αλμάτων. Θα δούμε τι είναι αναγκαίο γι' αυτό, και στη συνέχεια θα δούμε πώς εκτελούνται στη νέα ΜΕΔ οι εντολές του υποσυνόλου MIPS που υλοποιούμε, και πώς παράγονται τα σήματα ελέγχου.

Έτσι, για τις δύο εντολές j και beq υπάρχουν 3 δυνατές επιλογές εισόδου νέας τιμής στον PC:

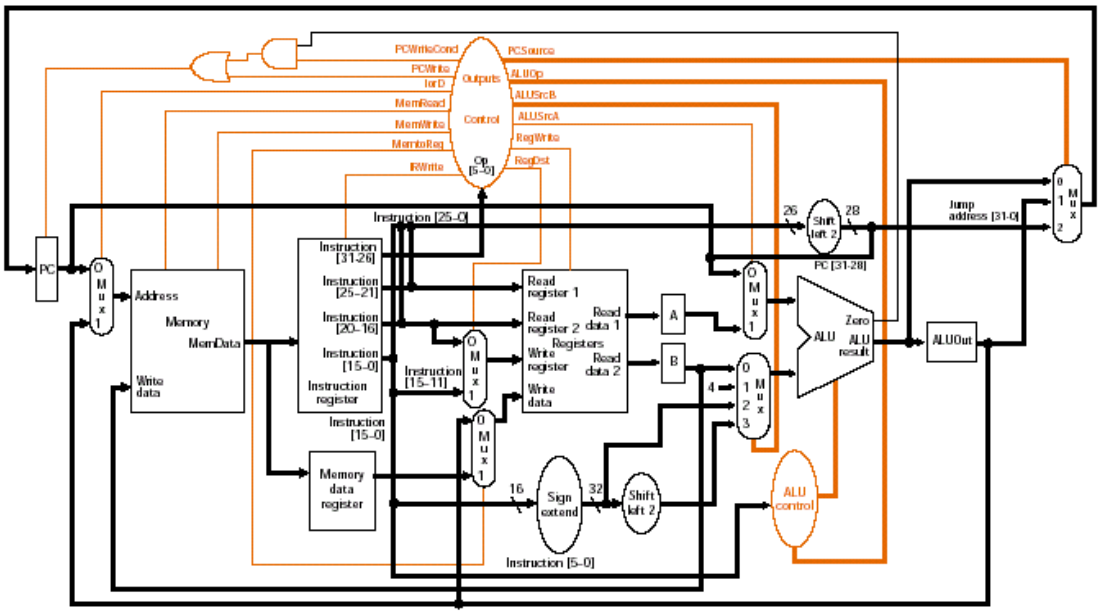
1. Η έξοδος της ΑΛΜ μετά την ανάκληση της εντολής, η οποία παρέχει την τιμή PC+4. Η τιμή αυτή αποθηκεύεται άμεσα στον PC.
2. Ο καταχωρητής C (ALUOut), ο οποίος περιέχει τη διεύθυνση προορισμού άλματος μιας εντολής διακλάδωσης, μόλις γίνει ο υπολογισμός της στην ΑΛΜ.
3. Τα 26 λιγότερο σημαντικά ψηφία του IR, ολισθημένα αριστερά κατά 2 θέσεις, τα οποία, μαζί με τα 4 περισσότερο σημαντικά ψηφία του αυξημένου PC, αποτελούν τη διεύθυνση προορισμού σε μια εντολή άλματος.

Όπως διαπιστώσαμε και στη ΜΕΔ του απλού κύκλου μηχανής, ο PC γράφεται άλλοτε με, και άλλοτε χωρίς συνθήκη. Όταν δεν εκτελείται κάποιο άλμα, ή όταν έχουμε εντολή άλματος, ο PC γράφεται χωρίς συνθήκη. Όταν όμως έχουμε εντολή διακλάδωσης, η τιμή του αυξημένου PC αντικαθίσταται από τη διεύθυνση προορισμού που περιέχει ο C, μόνο εάν η συνθήκη διακλάδωσης ικανοποιείται. Επομένως, ο έλεγχος εγγραφής του PC γίνεται με δύο διαφορετικά σήματα, με ονόματα PCWrite και PCWriteCond, για τις δύο παραπάνω περιπτώσεις, αντίστοιχα.

Για να συνδέσουμε τα δύο σήματα ελέγχου εγγραφής του PC σε αυτόν, χρησιμοποιούμε λογικές πύλες που τα συνδυάζουν, μαζί με το σήμα μηδενικής τιμής εξόδου της ΑΛΜ (Zero), σε ένα κοινό σήμα επίτρεψης εγγραφής. Κάτι παρόμοιο είχαμε κάνει και στη ΜΕΔ του απλού κύκλου μηχανής. Έτσι, για τον έλεγχο εγγραφής του PC με συνθήκη – τη συνθήκη διακλάδωσης της εντολής beq, το σήμα Zero της ΑΛΜ συνδυάζεται με το σήμα PCWriteCond μέσα

από μια πύλη λογικού ΚΑΙ. Η έξοδος της πύλης αυτής συνδυάζεται με το σήμα PCWrite, που ελέγχει την εγγραφή του PC χωρίς συνθήκη, μέσα από μια πύλη λογικού Η. Η έξοδος της δεύτερης αυτής πύλης υλοποιεί το τελικό σήμα επίτρεψης εγγραφής του PC.

Το διάγραμμα που δίνεται στη συνέχεια περιλαμβάνει την πλήρη νέα ΜΕΔ με την αντίστοιχη ΜΕ. Όπως κάναμε και στην ανάλυση του ελέγχου της ΜΕΔ απλού κύκλου μηχανής, και πριν προχωρήσουμε στην μελέτη της εκτέλεσης των εντολών στη νέα ΜΕΔ, θα δώσουμε στους δύο πίνακες που ακολουθούν τη συνοπτική λειτουργία των σημάτων ελέγχου, όταν αυτά είναι ενεργοποιημένα και όταν αυτά είναι απενεργοποιημένα.



Σήμα 1 bit	Λειτουργία απενεργοποίησης	Λειτουργία ενεργοποίησης
RegDst	Ο αριθμός καταχωρητή αποθήκευσης προέρχεται από το πεδίο rt (ψηφία 20-16 του IR).	Ο αριθμός καταχωρητή αποθήκευσης προέρχεται από το πεδίο rd (ψηφία 15-11 του IR).
RegWrite	Καμία.	Ο καταχωρητής αποθήκευσης γράφεται από την είσοδο δεδομένων του ΦΚ.
ALUSrcA	Η πρώτη είσοδος της ΑΛΜ προέρχεται από τον PC.	Η πρώτη είσοδος της ΑΛΜ προέρχεται από τον καταχωρητή Α.
MemRead	Καμία.	Η μνήμη διαβάζεται από τη διεύθυνση της εισόδου διευθύνσεων.
MemWrite	Καμία.	Η μνήμη γράφεται από την είσοδο δεδομένων στη διεύθυνση της εισόδου διευθύνσεων.
MemtoReg	Η τιμή που αποθηκεύεται στο ΦΚ προέρχεται από τον καταχωρητή C.	Η τιμή που αποθηκεύεται στο ΦΚ προέρχεται από τον καταχωρητή MDR.
IorD	Η είσοδος διευθύνσεων της μνήμης προέρχεται από τον PC.	Η είσοδος διευθύνσεων της μνήμης προέρχεται από τον καταχωρητή C.
IRWrite	Καμία.	Η έξοδος της μνήμης γράφεται στον καταχωρητή IR.
PCWrite	Καμία.	Η τιμή που καθορίζεται από το σήμα PCSource γράφεται στον PC.
PCWriteCond	Καμία.	Η τιμή που καθορίζεται από το σήμα PCSource γράφεται στον PC, εάν το σήμα Zero είναι ενεργοποιημένο.

Σήμα 2 bits	Τιμή	Λειτουργία
ALUSrcB	00	Η δεύτερη είσοδος της ΑΛΜ προέρχεται από τον καταχωρητή B.
	01	Η δεύτερη είσοδος της ΑΛΜ είναι η σταθερά 4.
	10	Η δεύτερη είσοδος της ΑΛΜ προέρχεται από την προέκταση προσήμου των 16 λιγότερο σημαντικών ψηφίων του καταχωρητή IR.
	11	Η δεύτερη είσοδος της ΑΛΜ προέρχεται από την προέκταση προσήμου των 16 λιγότερο σημαντικών ψηφίων του καταχωρητή IR, ολισθημένων 2 θέσεις αριστερά.
PCSource	00	Η είσοδος του PC προέρχεται από την έξοδο της ΑΛΜ.
	01	Η είσοδος του PC προέρχεται από τον καταχωρητή C.
	10	Η είσοδος του PC προέρχεται από την παράθεση των 4 περισσότερο σημαντικών ψηφίων του PC και των 26 λιγότερο σημαντικών ψηφίων του καταχωρητή IR, ολισθημένων αριστερά κατά 2 θέσεις.
ALUOp	00	Η ΑΛΜ εκτελεί πράξη πρόσθεσης (ADD).
	01	Η ΑΛΜ εκτελεί πράξη αφαίρεσης (SUB).
	10	Η ΑΛΜ εκτελεί την πράξη που ορίζει ο κωδικός τελεστή της εντολής.

Η εκτέλεση των εντολών στην υλοποίηση πολλαπλών κύκλων μηχανής

Έχοντας σχεδιάσει τη ΜΕΔ πολλαπλών κύκλων μηχανής ανά κύκλο εντολής, πρέπει στη συνέχεια να αποφασίσουμε ποιο μέρος του κύκλου εντολής θα εκτελείται σε κάθε κύκλο μηχανής, ώστε να διαπιστώσουμε αν τα παραπάνω σήματα αρκούν, πριν προχωρήσουμε στην υλοποίηση της ΜΕ. Για να διαιρέσουμε χρονικά τον κύκλο εντολής σε κύκλους μηχανής στη ΜΕΔ αυτή, μοιράζουμε σε διαφορετικούς κύκλους μηχανής τις λειτουργίες που εκτελούνται σε κάθε εντολή, έχοντας σα στόχο ο συνολικός χρόνος εκτέλεσης ανά κύκλο μηχανής να είναι λίγο-πολύ σταθερός, έτσι ώστε να πετύχουμε τον ελάχιστο χρόνο κύκλου μηχανής. Κατ' αρχήν, προσπαθούμε να διαιρέσουμε τον κύκλο κάθε εντολής σε βήματα, έτσι ώστε καθένα από αυτά να εκτελείται σε έναν κύκλο μηχανής. Για να ελαχιστοποιήσουμε το χρόνο κύκλου μηχανής, θα περιορίσουμε κάθε βήμα, ώστε να περιλαμβάνει το πολύ μία από τις βασικές λειτουργίες που αναφέραμε νωρίτερα, δηλαδή μία πράξη ΑΛΜ, μία προσπέλαση του ΦΚ, ή μία προσπέλαση μνήμης. Με τον περιορισμό αυτό, ο κύκλος μηχανής θα είναι τόσο σύντομος, όσος είναι ο χρόνος εκτέλεσης της πιο μεγάλης από τις λειτουργίες αυτές.

Πριν προχωρήσουμε, ας θυμηθούμε ότι στο τέλος κάθε κύκλου μηχανής, οποιαδήποτε δεδομένα παράγονται σε αυτόν και θα χρησιμοποιηθούν σε επόμενο κύκλο μηχανής πρέπει να αποθηκεύονται σε κάποιο στοιχείο κατάστασης. Τέτοιο στοιχείο θα είναι είτε κάποιο στοιχείο κατάστασης προσπελάσιμο από το σύνολο εντολών που υλοποιούνται (όπως ο PC, ο ΦΚ ή η μνήμη) είτε κάποιος προσωρινός καταχωρητής (όπως ο A, ο B, ο C, ο IR ή ο MDR). Ακόμα, όταν ο χρονισμός είναι ακμοπυροδοτούμενος, η τρέχουσα τιμή ενός στοιχείου κατάστασης μπορεί να διαβαστεί καθ' όλη τη διάρκεια του κύκλου μηχανής. Οποιαδήποτε νέα τιμή αποθηκεύεται σε αυτόν στην επόμενη ακμή του ωρολογιακού παλμού (δηλαδή στο τέλος του κύκλου μηχανής).

Στην υλοποίηση του απλού κύκλου μηχανής, η εκτέλεση κάθε εντολής επιτυγχάνεται με μια διαδοχή λειτουργιών στοιχείων της ΜΕΔ. Άλλα από τα στοιχεία λειτουργούν στη σειρά, δηλαδή η έξοδος ενός στοιχείου λαμβάνεται στην είσοδο ενός ή περισσότερων άλλων στοιχείων, και άλλα από τα στοιχεία λειτουργούν παράλληλα. Σαν παράδειγμα παράλληλης λειτουργίας, η ανάγνωση της μνήμης εντολών γίνεται την ίδια στιγμή με την αύξηση της τιμής του PC. Παρόμοια εκτέλεση έχουμε και στην υλοποίηση πολλαπλών κύκλων μηχανής. Όλες οι βασικές λειτουργίες του ίδιου βήματος εκτελούνται παράλληλα, και όλες οι βασικές λειτουργίες διαδοχικών βημάτων εκτελούνται στη σειρά. Ο περιορισμός της μίας πράξης ΑΛΜ, μίας προσπέλασης του ΦΚ ή μίας προσπέλασης μνήμης καθορίζει απλά τι μπορεί να "χωρέσει" σε έναν κύκλο μηχανής.

Παρατηρήστε ότι διαχωρίζουμε τις προσπελάσεις των καταχωρητών ειδικού σκοπού από τις προσπελάσεις των καταχωρητών γενικού σκοπού του ΦΚ. Η προσπέλαση των πρώτων

μπορεί να γίνει σα μέρος των λειτουργιών ενός βήματος της εκτέλεσης μιας εντολής. Αντίθετα, η προσπέλαση του ΦΚ είναι μια βασική λειτουργία της ΜΕΔ και απαιτεί έναν κύκλο μηχανής για να ολοκληρωθεί. Αυτό συμβαίνει επειδή ο ΦΚ είναι πολύπλοκη μονάδα με εσωτερικό έλεγχο που απαιτεί σημαντικά περισσότερο χρόνο για την προσπέλασή του από τους καταχωρητές ειδικού σκοπού, που είναι μεμονωμένοι, δεν έχουν δικό τους έλεγχο, και επομένως έχουν ελάχιστο χρόνο προσπέλασης. Η δέσμευση ενός ολόκληρου κύκλου μηχανής για την προσπέλαση του ΦΚ επιβάλλεται τελικά από την ανάγκη ελαχιστοποίησης του χρόνου κύκλου μηχανής.

Τα βήματα στα οποία διαιρούμε τον κύκλο εντολής δίνονται στη συνέχεια. Κάθε εντολή, ανάλογα με τον τύπο της, απαιτεί από τρία έως πέντε βήματα για να εκτελεστεί:

1. Ανάκληση εντολής

Το 1ο βήμα περιλαμβάνει την προσκόμιση της εντολής από τη μνήμη και την αύξηση της τιμής του PC[5]:

```
IR = Memory[PC];
PC = PC + 4;
```

Λειτουργία: Στείλε στην είσοδο διευθύνσεων της μονάδας μνήμης το περιεχόμενο του PC, εκτέλεσε μια ανάγνωση μνήμης, και επέστρεψε την τιμή που διαβάστηκε στον καταχωρητή IR, όπου και αποθήκευσέ την. Επίσης, αύξησε την τιμή του PC κατά 4.

Για την υλοποίηση αυτού του βήματος, πρέπει κατ' αρχήν να ενεργοποιήσουμε τα σήματα ελέγχου MemRead και IRWrite, ώστε να επιλεγεί λειτουργία ανάγνωσης στη μνήμη και να επιτραπεί η εγγραφή του καταχωρητή IR, αντίστοιχα, καθώς και να απενεργοποιήσουμε το σήμα IorD, ώστε να επιλεγεί ο PC στην είσοδο διευθύνσεων της μνήμης. Για την αύξηση της τιμής του PC, πρέπει το σήμα ALUSrcA να είναι απενεργοποιημένο, ώστε να επιλεγεί ο PC στην πρώτη είσοδο της ΑΛΜ, το σήμα ALUSrcB να έχει τιμή "01", ώστε να επιλεγεί η σταθερά 4 στη δεύτερη είσοδο της ΑΛΜ, και το σήμα ALUOp να έχει τιμή "00", ώστε να επιλεγεί η πράξη ADD στην ΑΛΜ. Επειδή τέλος θέλουμε να αποθηκεύσουμε την αυξημένη τιμή του PC πίσω σε αυτόν, πρέπει να ενεργοποιήσουμε το σήμα PCWrite. Η αύξηση της τιμής του PC και η ανάγνωση της μνήμης είναι δύο λειτουργίες που εκτελούνται παράλληλα. Η νέα τιμή του PC δεν γίνεται ορατή πριν το τέλος του κύκλου μηχανής. (Ας σημειωθεί ότι η έξοδος της ΑΛΜ, δηλαδή η αυξημένη τιμή του PC, αποθηκεύεται και στον καταχωρητή C, κάτι που δε μας χρειάζεται, αλλά ούτε μας ενοχλεί.)

2. Αποκωδικοποίηση εντολής και ανάγνωση τελούμενων από το ΦΚ

Τόσο στο προηγούμενο, όσο και σε αυτό το βήμα, δεν έχει γίνει γνωστό ποια είναι η εντολή που εκτελείται, κι επομένως μπορούμε να εκτελούμε μόνο λειτουργίες, που είτε αφορούν όλες τις εντολές (όπως για παράδειγμα η ανάκληση της εντολής), είτε δε μας ενοχλεί αν εκτελεστούν χωρίς να είναι απαραίτητες. Έτσι, στο βήμα αυτό μπορούμε να διαβάσουμε από το ΦΚ τους δύο καταχωρητές που υποδεικνύουν τα πεδία rs και rt της λέξης εντολής – που είναι αποθηκευμένη στον καταχωρητή IR, επειδή δε μας ενοχλεί η ανάγνωσή τους, ακόμα κι αν η εντολή δε χρειαστεί τελικά τις τιμές τους. Οι τιμές που διαβάζονται αποθηκεύονται στους δύο καταχωρητές A και B, ώστε να είναι διαθέσιμες στον επόμενο κύκλο μηχανής.

Στο βήμα αυτό, θα υπολογίσουμε στην ΑΛΜ τη διεύθυνση προορισμού άλματος διακλάδωσης, υποθέτοντας ότι η εντολή είναι εντολή διακλάδωσης. Αυτή είναι μια λειτουργία που επίσης δεν ενοχλεί, αν διαπιστωθεί ότι η υπόθεσή μας είναι λανθασμένη, και η τιμή που θα έχουμε υπολογίσει απλά δε θα χρησιμοποιηθεί. Η τιμή που υπολογίζεται αποθηκεύεται στο τέλος του κύκλου μηχανής στον καταχωρητή C.

Η "αισιόδοξη" εκτέλεση των πιο πάνω λειτουργιών, πριν διαπιστώσουμε αν είναι ή δεν είναι απαραίτητες για την εντολή, οδηγεί σε μικρότερο αριθμό βημάτων – δηλαδή κύκλων μηχανής – για την ολοκλήρωση μιας εντολής. Η εκτέλεση αυτή είναι επιτρεπτή, επειδή η λε-

[5] Οι βασικές λειτουργίες κάθε βήματος θα δίνονται σε μορφή κώδικα κάποιας γλώσσας υψηλού επιπέδου, πριν αναλυθούν στη συνέχεια.

ξη εντολής MIPS έχει κανονικότητα στη μορφή της: Για κάθε εντολή που έχει δύο τελούμενα εισόδου από το ΦΚ, οι αριθμοί καταχωρητών που διαβάζονται λαμβάνονται πάντα από τα πεδία rs και rt της λέξης εντολής. Για κάθε εντολή διακλάδωσης, η σταθερά μετατόπισης λαμβάνεται πάντα από τα 16 λιγότερο σημαντικά ψηφία της λέξης εντολής.

Στο δεύτερο βήμα του κύκλου εντολής έχουμε λοιπόν:

```
A = Reg[IR[25-21]];
B = Reg[IR[20-16]];
C = PC + (sign-extend(IR[15-0]) << 2);
```

Λειτουργία: Προσπέλασε το ΦΚ για την ανάγνωση των καταχωρητών rs και rt και αποθήκευσε τις τιμές αυτών στους καταχωρητές A και B, αντίστοιχα. Επίσης, πρόσθεσε την τιμή του PC με την προέκταση προσήμου των 16 λιγότερο σημαντικών ψηφίων του IR, ολισθημένων αριστερά κατά 2 θέσεις, και αποθήκευσε το αποτέλεσμα στον C.

Η λειτουργία ανάγνωσης του ΦΚ δε χρησιμοποιεί κανένα σήμα ελέγχου. Οι καταχωρητές A και B δε διαθέτουν σήμα επίτρεψης εγγραφής και παίρνουν νέες τιμές σε κάθε κύκλο μηχανής. Παρόμοια, ο ΦΚ δε διαθέτει σήμα ανάγνωσης και διαβάζεται σε κάθε κύκλο μηχανής. Οι τιμές που διαβάζονται από το ΦΚ και αποθηκεύονται στους δύο καταχωρητές A και B είναι οι ίδιες σε διαδοχικούς κύκλους μηχανής, όσο δεν αλλάζει τιμή ο IR και δε μεσολαβεί εγγραφή στο ΦΚ. Όσο αφορά την υλοποίηση της δεύτερης από τις παραπάνω λειτουργίες, το σήμα ALUSrcA πρέπει να είναι απενεργοποιημένο, ώστε η πρώτη είσοδος της ΑΛΜ να προέλθει από τον PC, ενώ τα σήματα ALUSrcB και ALUOp πρέπει να έχουν τιμές “11” και “00”, αντίστοιχα, ώστε η δεύτερη είσοδος της ΑΛΜ να προέλθει από τη μονάδα αριστερής ολίσθησης, και η πράξη ΑΛΜ να είναι η ADD. Η έξοδος της ΑΛΜ αποθηκεύεται στον καταχωρητή C σε κάθε κύκλο μηχανής, επειδή αυτός δε διαθέτει σήμα επίτρεψης εγγραφής. Έτσι, και εάν η εντολή που αποκωδικοποιείται είναι εντολή διακλάδωσης, η τιμή που αποθηκεύεται τώρα στον C πρέπει να χρησιμοποιηθεί στον επόμενο κύκλο μηχανής, πριν ο C αποκτήσει νέα τιμή, πιθανά διαφορετική. Παρατηρήστε ότι οι δύο λειτουργίες αυτού του βήματος είναι παράλληλες.

Μετά το δεύτερο βήμα, οι λειτουργίες των υπόλοιπων βημάτων είναι εν μέρει διαφορετικές για κάθε τύπο εντολής.

3. Εκτέλεση πράξης ΑΛΜ, υπολογισμός τελικής διεύθυνσης προσπέλασης μνήμης ή εκτέλεση άλματος

Το 3ο βήμα είναι το πρώτο βήμα, οι λειτουργίες του οποίου εξαρτώνται από τον τύπο εντολής. Σε κάθε περίπτωση, η ΑΛΜ εκτελεί κάποια πράξη πάνω σε δεδομένα που παρέχονται από καταχωρητές, όπου αποθηκεύτηκαν στο προηγούμενο βήμα. Η πράξη ΑΛΜ καθορίζεται από τον τύπο της εντολής. Για κάθε τύπο εντολής έχουμε:

Εντολή προσπέλαση μνήμης

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

Λειτουργία: Πρόσθεσε στην τιμή του A την προέκταση προσήμου των 16 λιγότερο σημαντικών ψηφίων του IR.

Το αποτέλεσμα της παραπάνω πράξης αποθηκεύεται στον C, ώστε να χρησιμοποιηθεί από τη μονάδα μνήμης στον επόμενο κύκλο μηχανής σαν την τελική διεύθυνση προσπέλασης. Για τη λειτουργία αυτή απαιτείται ενεργοποίηση του σήματος ALUSrcA, ώστε να επιλεγεί ο καταχωρητής A στην πρώτη είσοδο της ΑΛΜ, ενώ τα σήματα ALUSrcB και ALUOp πρέπει να έχουν τιμές “10” και “00”, αντίστοιχα, ώστε να επιλεγεί η έξοδος της μονάδας προέκτασης προσήμου στη δεύτερη είσοδο της ΑΛΜ, και η πράξη ADD στην ΑΛΜ.

R-εντολή

```
C = A op B
```

Λειτουργία: Εκτέλεσε την πράξη που καθορίζεται από τον κωδικό τελεστή της εντολής πάνω στα δεδομένα που παρέχονται από τους καταχωρητές A και B.

Το αποτέλεσμα της πράξης ΑΛΜ αποθηκεύεται στον C, ώστε να μεταφερθεί στο ΦΚ τον επόμενο κύκλο μηχανής. Για τη λειτουργία αυτή πρέπει να ενεργοποιηθεί το σήμα ALUSrcA,

ώστε να επιλεγεί ο καταχωρητής A στην πρώτη είσοδο της ΑΛΜ, ενώ τα σήματα ALUSrcB και ALUOp πρέπει να έχουν τιμές “00” και “10”, αντίστοιχα, ώστε να επιλεγεί ο καταχωρητής B στη δεύτερη είσοδο της ΑΛΜ, και η πράξη ΑΛΜ που καθορίζει ο κωδικός τελεστή της εντολής.

Εντολή διακλάδωσης

```
if (A == B) PC = C;
```

Λειτουργία: Σύγκρινε τα περιεχόμενα των καταχωρητών A και B, και αν είναι ίσα, μετέφερε την τιμή του C στον PC.

Η σύγκριση των περιεχομένων των A και B για ισότητα γίνεται στην ΑΛΜ με την πράξη SUB. Το λογικό αποτέλεσμα της σύγκρισης αυτής, το οποίο και θα καθορίσει την έκβαση της διακλάδωσης, λαμβάνεται από την έξοδο Zero της ΑΛΜ. Έτσι, η υλοποίηση της εντολής beq απαιτεί ενεργοποίηση του σήματος ALUSrcA, ώστε να επιλεγεί ο A στην πρώτη είσοδο της ΑΛΜ, τιμή “00” για το σήμα ALUSrcB, ώστε να επιλεγεί ο B στη δεύτερη είσοδο της ΑΛΜ, και τιμή “01” για το σήμα ALUOp, ώστε να επιλεγεί η πράξη SUB στην ΑΛΜ. Το σήμα PCCondWrite πρέπει να είναι ενεργοποιημένο, ώστε να επιτραπεί εγγραφή στον PC, αν το σήμα Zero ενεργοποιηθεί. Τέλος, για να επιλεγεί ο C στην είσοδο του PC, πρέπει το σήμα PCSource να έχει τιμή “01”, αφού η διεύθυνση προορισμού άλματος είναι αποθηκευμένη στον C από λειτουργία του προηγούμενου κύκλου μηχανής. Παρατηρήστε ότι για άλματα διακλαδώσεων που εκτελούνται, ο PC γράφεται δύο φορές στον κύκλο εντολής, μια φορά στο 1ο βήμα, οπότε λαμβάνει τη νέα τιμή του από την έξοδο της ΑΛΜ, και μια φορά στο 3ο βήμα, οπότε λαμβάνει τη νέα τιμή του από τον καταχωρητή C. Η διεύθυνση που θα χρησιμοποιηθεί για την ανάκληση της επόμενης εντολής είναι η τιμή που γράφτηκε τελευταία.

Εντολή άλματος

```
PC = PC[31-28] || (IR[25-0] << 2);
```

Λειτουργία: Αντικατάστησε την τιμή του PC με τον αριθμό που προκύπτει από την παράθεση των 4 πιο σημαντικών ψηφίων της παλιάς τιμής του PC με τα 26 λιγότερα σημαντικά ψηφία του IR, ολισθημένα αριστερά κατά 2 θέσεις.

Στην εντολή j ο PC αλλάζει τιμή χωρίς συνθήκη, κι επομένως το σήμα που πρέπει να είναι ενεργοποιημένο για να επιτρέψει την εγγραφή του PC είναι τώρα το PCWrite. Η προέλευση της νέας τιμής του PC καθορίζεται από την τιμή του σήματος PCSource, που στην εντολή αυτή πρέπει να είναι “10”.

4. Προσπέλαση μνήμης ή ολοκλήρωση R-εντολής

Στο βήμα αυτό, μια εντολή lw ή sw προσπελαύνει τη μνήμη, και μια αριθμητική/λογική εντολή γράφει το αποτέλεσμά της στο ΦΚ. Μια λέξη δεδομένων που προέρχεται από τη μνήμη γράφεται προσωρινά στον καταχωρητή MDR, απ’ όπου θα πρέπει να διαβαστεί στον επόμενο κύκλο μηχανής. Για κάθε περίπτωση έχουμε:

Προσπέλαση μνήμης

```
MDR = Memory[C];
```

ή

```
Memory[C] = B;
```

Λειτουργία: Εάν η εντολή είναι εντολή φόρτωσης, διάβασε μια λέξη δεδομένων από τη μνήμη και αποθήκευσέ την στον MDR. Εάν η εντολή είναι εντολή αποθήκευσης, γράψε το περιεχόμενο του B στη μνήμη. Σα διεύθυνση προσπέλασης χρησιμοποίησε και για τις δύο περιπτώσεις το περιεχόμενο του C.

Για την υλοποίηση της προσπέλασης μνήμης, πρέπει να ενεργοποιηθεί το σήμα MemRead για ανάγνωση (σε εντολή φόρτωσης) ή το σήμα MemWrite για εγγραφή (σε εντολή αποθήκευσης). Επίσης, πρέπει να ενεργοποιηθεί το σήμα IorD, ώστε να επιλεγεί ο C στην είσοδο διευθύνσεων της μνήμης. Ο MDR δε διαθέτει σήμα επίτρεψης και γράφεται σε κάθε κύκλο μηχανής, γι’ αυτό και μετά από κάθε εντολή φόρτωσης η τιμή που αποθηκεύεται στον MDR είναι απαραίτητο να χρησιμοποιηθεί στον αμέσως επόμενο κύκλο μηχανής. Αξίζει, τέλος, να

σημειωθεί, ότι σε εντολή αποθήκευσης, η τιμή που στέλνεται στη μνήμη από τον καταχωρητή B είναι η τιμή που διαβάζεται από το ΦΚ στο 3ο βήμα του κύκλου εντολής, κι όχι αυτή που διαβάζεται στο 2ο, άσχετα αν οι δύο τιμές συμπίπτουν.

Αποθήκευση αποτελέσματος R-εντολής

```
Reg[IR[15-11]] = C;
```

Λειτουργία: Αποθήκευσε το περιεχόμενο του C στο ΦΚ.

Η λειτουργία αυτή μεταφέρει την τιμή που γράφτηκε στον καταχωρητή C στον προηγούμενο κύκλο μηχανής στον καταχωρητή αποτελέσματος της εντολής, που καθορίζεται από το πεδίο rd της λέξης εντολής. Τα σήματα RegWrite και RegDst πρέπει να είναι ενεργοποιημένα, ώστε να επιτραπεί η εγγραφή του ΦΚ και να επιλεγεί το σωστό πεδίο από τον IR, αντίστοιχα. Το σήμα MemtoReg πρέπει να είναι απενεργοποιημένο, ώστε η είσοδος δεδομένων του ΦΚ να προέλθει από τον C.

5. Ολοκλήρωση εντολής φόρτωσης

Στο βήμα αυτό ολοκληρώνεται μια εντολή φόρτωσης με την αποθήκευση στο ΦΚ της τιμής που διάβασε από τη μνήμη:

```
Reg[IR[20-16]] = MDR;
```

Λειτουργία: Αποθήκευσε το περιεχόμενο του MDR στο ΦΚ.

Η λειτουργία αυτή μεταφέρει την τιμή που γράφτηκε στον καταχωρητή MDR στον προηγούμενο κύκλο μηχανής στον καταχωρητή αποτελέσματος της εντολής, που στην περίπτωση αυτή καθορίζεται από το πεδίο rt της λέξης εντολής. Το σήμα RegWrite πρέπει να είναι ενεργοποιημένο, ώστε να επιτραπεί η εγγραφή του ΦΚ, και το σήμα RegDst πρέπει να είναι απενεργοποιημένο, ώστε να επιλεγεί το σωστό πεδίο από τον IR. Το σήμα MemtoReg τέλος, πρέπει να είναι ενεργοποιημένο, ώστε η είσοδος δεδομένων του ΦΚ να προέλθει από τον MDR.

Οι λειτουργίες των πέντε παραπάνω βημάτων συνοψίζονται στον παρακάτω πίνακα. Η σειρά διαδοχής τους που φαίνεται σε αυτόν θα μας καθοδηγήσει στη σχεδίαση της ΜΕ της ΜΕΔ των πολλαπλών κύκλων μηχανής για κάθε κύκλο εντολής.

Βήμα	R-εντολές	Εντολές προσπέλασης μνήμης	Διακλαδώσεις	Άλματα
1ο	IR = Memory[PC] PC = PC + 4			
2ο	A = Reg[IR[25-21]] B = Reg[IR[20-16]] C = PC + (sign-extend(IR[15-0]) << 2)			
3ο	C = A op B	C = A + sign-extend (IR[15-0])	if (A == B) PC = C	PC = PC[31-28] || (IR[25-0] << 2)
4ο	Reg[IR[15-11]] = C	*lw*: MDR = Memory[C] *sw*: Memory[C] = B		
5ο		*lw*: Reg[IR[20-16]] = MDR		

Υλοποιώντας τη ΜΕ

Τώρα που καθορίσαμε τις λειτουργίες, κι επομένως τις τιμές που πρέπει να έχουν τα σήματα ελέγχου σε κάθε κύκλο μηχανής, μπορούμε να προχωρήσουμε στη σχεδίαση της ΜΕ της νέας ΜΕΔ. Στην περίπτωση του απλού κύκλου μηχανής ανά κύκλο εντολής, η ΜΕ υλοποιήθηκε με τη βοήθεια πινάκων αλήθειας για τα σήματα ελέγχου, που εξαρτιόνταν μόνο από τη λέξη εντολής. Στην περίπτωση των πολλαπλών κύκλων μηχανής, όμως, ο έλεγχος των λειτουργιών είναι πιο πολύπλοκος, επειδή κάθε εντολή εκτελείται σε διαδοχικά βήματα. Έτσι, η ΜΕ πρέ-

πει και να ορίζει τις τιμές των σημάτων ελέγχου για κάθε κύκλο μηχανής, αλλά και να βρίσκει το σωστό επόμενο βήμα.

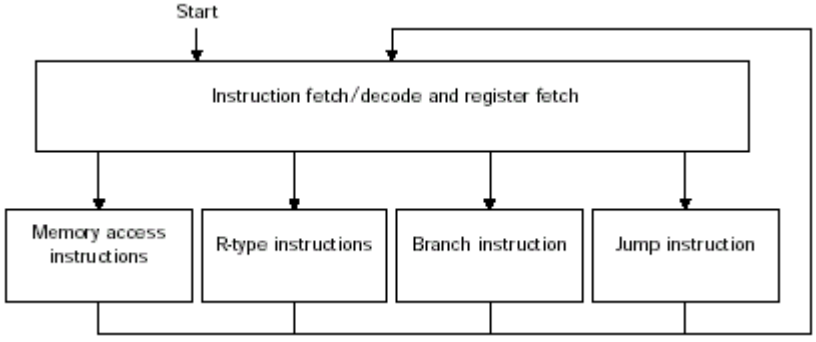
Στην παρούσα παράγραφο και στην παράγραφο 5.5 θα δούμε δύο διαφορετικές μεθόδους υλοποίησης της ΜΕ μιας ΜΕΔ πολλαπλών κύκλων μηχανής ανά κύκλο εντολής. Η πρώτη μέθοδος στηρίζεται σε μηχανές (πεπερασμένων) καταστάσεων, που συνήθως απεικονίζονται γραφικά με διαγράμματα μετάβασης, και ορίζουν ακολουθιακά κυκλώματα. Η δεύτερη μέθοδος ορίζει τη ΜΕ μέσα από ένα προγραμματιστικό μοντέλο, που λέγεται *μικροπρογραμματισμός*. Και τα δύο αυτά μοντέλα αποτελούν τρόπους αναπαράστασης της ΜΕ που οδηγούν στην υλοποίησή της με εργαλεία CAD. Εμείς δε θα ασχοληθούμε με την τελική υλοποίηση του υλικού από μια τέτοια αναπαράσταση, κάτι που ξεφεύγει από τους σκοπούς του παρόντος κεφαλαίου, παρά μόνο με την παραγωγή της αναπαράστασης για τη ΜΕ μιας συγκεκριμένης ΜΕΔ.

Η πρώτη μέθοδος που χρησιμοποιούμε για να ορίσουμε τη ΜΕ μιας ΜΕΔ πολλαπλών κύκλων μηχανής είναι η *μηχανή καταστάσεων*. Μια μηχανή καταστάσεων αποτελείται από ένα σύνολο καταστάσεων και μεταβάσεις μεταξύ αυτών. Οι μεταβάσεις ορίζονται με τη βοήθεια της *συνάρτησης επόμενης κατάστασης*, που απεικονίζει μια παρούσα κατάσταση και ένα σύνολο τιμών εισόδου σε μια νέα – την επόμενη – κατάσταση. Με τη χρήση της μηχανής καταστάσεων για την υλοποίηση μιας ΜΕ, κάθε κατάσταση ορίζει ένα σύνολο τιμών εξόδου που αντιστοιχούν σε ενεργοποιημένα σήματα ελέγχου της ΜΕΔ. Σήματα που δεν είναι ενεργοποιημένα σε μια κατάσταση θεωρούνται απενεργοποιημένα, δηλαδή τους αποδίδεται μηδενική τιμή. Η σωστή λειτουργία της ΜΕΔ στηρίζεται σε αυτή την παραδοχή, και όχι στην παραδοχή ότι τα μη ενεργοποιημένα σήματα είναι αδιάφορες συνθήκες. Για παράδειγμα, το σήμα RegWrite πρέπει να είναι ενεργοποιημένο μόνο όταν έχουμε εγγραφή στο ΦΚ. Έτσι, όταν το σήμα αυτό δε δηλώνεται στις εξόδους κάποιας κατάστασης, τότε αυτό θα σημαίνει ότι δεν έχουμε εγγραφή του ΦΚ όταν η ΜΕΔ βρίσκεται σε αυτή την κατάσταση, κι επομένως το σήμα θα πρέπει να είναι απενεργοποιημένο.

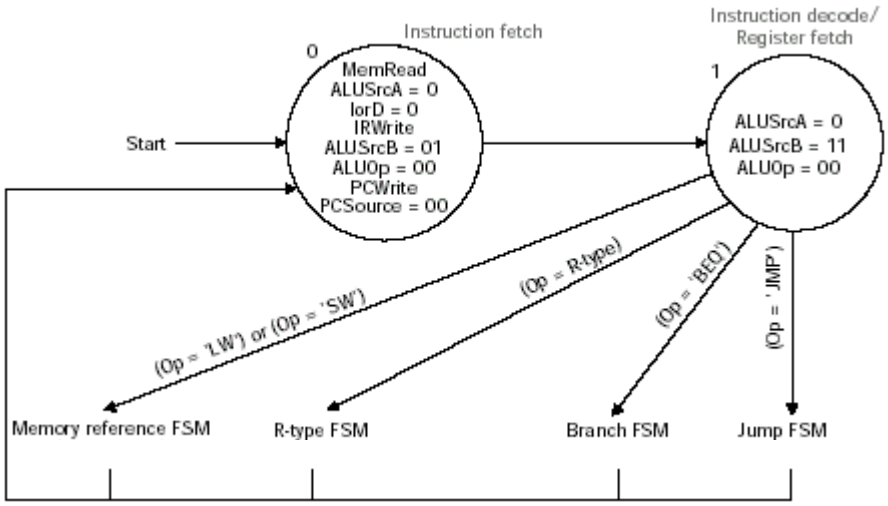
Εξαίρεση στον παραπάνω κανόνα αποτελούν τα σήματα επιλογής των πολυπλεκτών. Τέτοια σήματα εκτελούν μια επιλογή, είτε είναι ενεργοποιημένα είτε όχι. Γι' αυτό, σε μια μηχανή καταστάσεων, πάντα δίνουμε τις τιμές των σημάτων επιλογής των πολυπλεκτών που αφορούν την κατάσταση, ενώ τα σήματα επιλογής των πολυπλεκτών που δε μας ενδιαφέρουν γίνονται αδιάφορες συνθήκες.

Στη συγκεκριμένη ΜΕ που σχεδιάζουμε, η μηχανή καταστάσεων ουσιαστικά περιγράφει τη διαδοχή των βημάτων που είδαμε στον παραπάνω πίνακα. Κάθε βήμα για κάθε τύπο εντολής αντιστοιχίζεται σε μία κατάσταση που επομένως διαρκεί έναν κύκλο μηχανής. Η μηχανή καταστάσεων αποτελείται από περισσότερα του ενός τμήματα. Εφ' όσον τα πρώτα δύο βήματα του κύκλου εντολής είναι κοινά για όλους τους τύπους εντολών, οι δύο πρώτες καταστάσεις της μηχανής καταστάσεων θα είναι οι ίδιες για όλες τις εντολές. Τα υπόλοιπα βήματα όμως διαφέρουν από εντολή σε εντολή, ανάλογα με τον τύπο των εντολών. Εφ' όσον μετά το τελευταίο βήμα κάθε εντολής ακολουθεί το πρώτο, ανεξάρτητα από τον τύπο εντολής, έτσι και στη μηχανή καταστάσεων, την τελευταία κατάσταση κάθε τύπου εντολής ακολουθεί η πρώτη.

Μια αφηρημένη εικόνα της μηχανής που σχεδιάζουμε σύμφωνα με τα παραπάνω φαίνεται στο πιο κάτω σχήμα. Για να συγκεκριμενοποιήσουμε κάθε τμήμα της, θα ξεκινήσουμε με το τμήμα των δύο πρώτων καταστάσεων, και μετά θα προχωρήσουμε στα υπόλοιπα.

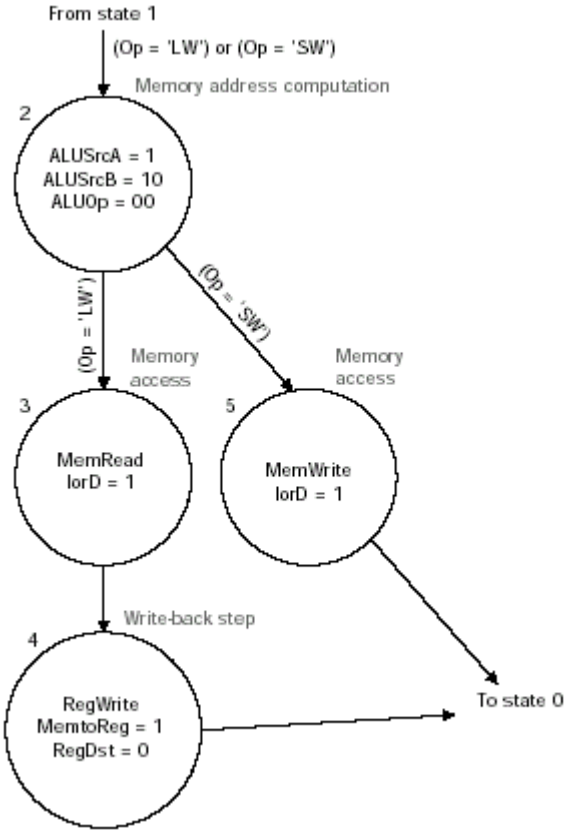


Έτσι λοιπόν, οι δύο πρώτες καταστάσεις της μηχανής καταστάσεων φαίνονται στο παρακάτω διάγραμμα. Η αρίθμηση των καταστάσεων είναι αυθαίρετη, κι εμείς για λόγους απλότητας ξεκινάμε με τον αριθμό 0 για την κατάσταση που αντιστοιχεί στο 1ο βήμα του κύκλου εντολής.



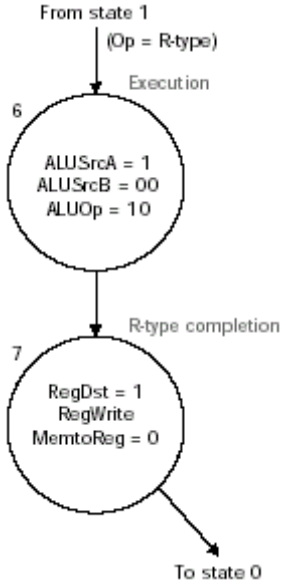
Τα σήματα ελέγχου που είναι ενεργοποιημένα – ή οι συγκεκριμένες τιμές σημάτων επιλογής πολυπλεκτών – σε κάθε κατάσταση παρουσιάζονται στο εσωτερικό του κύκλου που απεικονίζει την κατάσταση. Οι μεταβάσεις μεταξύ των καταστάσεων δείχνονται με κατευθυνόμενα τόξα, οι προαιρετικές ετικέτες των οποίων υποδηλώνουν συνθήκες μετάβασης. Μετά την κατάσταση 1, οι ενεργοποιήσεις των σημάτων ελέγχου εξαρτώνται από τον τύπο της εντολής. Έτσι, η κατάσταση 1 έχει τέσσερις επόμενες καταστάσεις, ανάλογα με τους τύπους εντολών που υλοποιούμε: προσπέλασης μνήμης, R-εντολές, διακλάδωσης και άλματος. Η διαδικασία της *αποκωδικοποίησης* είναι στην ουσία η διαδικασία επιλογής μιας από πολλές επόμενες καταστάσεις.

Το μέρος της μηχανής καταστάσεων – μετά την κατάσταση 1, που υλοποιεί τον έλεγχο των εντολών προσπέλασης μνήμης, παρουσιάζεται στο επόμενο διάγραμμα. Για τις εντολές



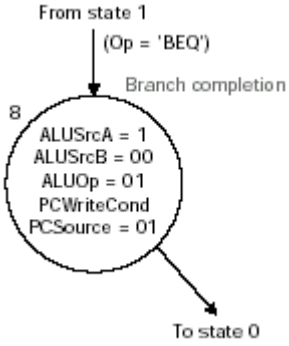
αυτές, το 3ο βήμα υπολογίζει την τελική διεύθυνση προσπέλασης μνήμης και αντιστοιχίζεται στην κατάσταση 2. Στον υπολογισμό της διεύθυνσης προσπέλασης πρέπει οι πολυπλέκτες της πρώτης και της δεύτερης εισόδου της ΑΛΜ να επιλέγουν τιμές από τον καταχωρητή A και την υπομονάδα προέκτασης προσήμου της σταθεράς μετατόπισης της εντολής, αντίστοιχα. Η έξοδος της ΑΛΜ γράφεται στον καταχωρητή C. Αφού υπολογιστεί η διεύθυνση στη μνήμη, γίνεται η προσπέλασή της, για ανάγνωση ή για εγγραφή, κάτι που απαιτεί δύο διαφορετικές καταστάσεις. Έτσι, αν ο κωδικός λειτουργίας υποδεικνύει την εντολή lw, τότε έχουμε μετάβαση στην κατάσταση 3, που υλοποιεί το 4ο βήμα του κύκλου εντολής για εντολές φόρτωσης, ενεργοποιώντας το σήμα ανάγνωσης μνήμης MemRead. Αν από την άλλη μεριά ο κωδικός λειτουργίας υποδεικνύει την εντολή sw, τότε έχουμε μετάβαση στην κατάσταση 5, που υλοποιεί το ίδιο βήμα για εντολές αποθήκευσης, ενεργοποιώντας το σήμα MemWrite. Και στις δύο καταστάσεις 3 και 5, ενεργοποιείται το σήμα IorD, ώστε ο πολυπλέκτης της εισόδου διευθύνσεων της μνήμης να επιλέγει τιμή από τον C. Η έξοδος της μονάδας μνήμης σε κάθε ανάγνωσή της γράφεται στον καταχωρητή MDR. Μετά από μια εγγραφή στη μνήμη, η εντολή sw έχει ολοκληρωθεί, κι έτσι επόμενη κατάσταση της 5 είναι η κατάσταση 0. Αντίθετα, μετά από μια ανάγνωση, μια νέα κατάσταση είναι αναγκαία για τη μεταφορά των δεδομένων από τον MDR στο ΦΚ. Η κατάσταση 4 υλοποιεί το 5ο βήμα – την εγγραφή του ΦΚ – μιας εντολής φόρτωσης lw, ενεργοποιώντας και απενεργοποιώντας τα σήματα επιλογής πολυπλεκτών MemtoReg και RegDst, αντίστοιχα, ώστε η είσοδος δεδομένων του ΦΚ να επιλεγεί από τον καταχωρητή MDR, ενώ η είσοδος διευθύνσεων του ΦΚ να επιλεγεί από το πεδίο rt της λέξης εντολής που περιέχεται στον IR. Μετά την κατάσταση 4, ακολουθεί η κατάσταση 0.

Για την υλοποίηση του ελέγχου των R-εντολών απαιτούνται δύο νέες καταστάσεις, στις οποίες να αντιστοιχίζονται τα βήματα 3 και 4 των εντολών αυτών. Το τμήμα της μηχανής καταστάσεων που περιλαμβάνει τις καταστάσεις αυτές δίνεται στο διάγραμμα που ακολουθεί.



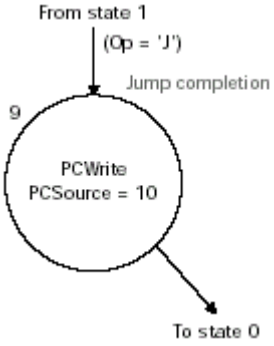
Η κατάσταση 6 θέτει τιμές ‘1’ και “00” στα σήματα επιλογής πολυπλεκτών ALUSrcA και ALUSrcB, αντίστοιχα. Έτσι, οι είσοδοι της ΑΛΜ επιλέγονται από τους καταχωρητές A και B. Επίσης, τίθεται τιμή “10” στο σήμα ALUOp , ώστε ο έλεγχος της πράξης ΑΛΜ να προέλθει από τον κωδικό τελεστή της λέξης εντολής. Η κατάσταση 7 υλοποιεί την αποθήκευση στο ΦΚ. Το σήμα RegWrite ενεργοποιείται, για να επιτρέψει την εγγραφή του ΦΚ, ενώ το σήμα επιλογής RegDst γίνεται ‘1’, ώστε η είσοδος διευθύνσεων του ΦΚ να προέλθει από το πεδίο rd της λέξης εντολής. Στην κατάσταση 7 έχουμε ακόμα απενεργοποίηση του σήματος MemtoReg, γεγονός που επιλέγει το περιεχόμενο του καταχωρητή C για την είσοδο δεδομένων του ΦΚ.

Οι εντολές διακλάδωσης απαιτούν μία μόνο επιπλέον κατάσταση, αφού ολοκληρώνουν την εκτέλεσή τους στο 3ο βήμα του κύκλου εντολής που μελετήσαμε νωρίτερα. Στην κατάσταση αυτή – την κατάσταση 8 του διαγράμματος της επόμενης σελίδας, πρέπει να πάρουν



τιμές τα σήματα που υλοποιούν τη σύγκριση μεταξύ των περιεχομένων των A και B, και την εγγραφή υπό συνθήκη του PC από το περιεχόμενο του C. Για την πρώτη λειτουργία, η κατάσταση 8 θέτει τιμές ‘1’ και “00” στα σήματα ALUSrcA και ALUSrcB, αντίστοιχα. Το σήμα ALUOp παίρνει τιμή “01”, ώστε η ΑΛΜ να εκτελέσει πράξη αφαίρεσης (SUB). Η κατάσταση 8 χρησιμοποιεί την έξοδο Zero της ΑΛΜ, και όχι το αποτέλεσμα της αφαίρεσης. Για την εγγραφή του PC απαιτείται ενεργοποίηση του σήματος PCWriteCond. Το σήμα PCSource πρέπει να πάρει τιμή “01”, ώστε η είσοδος του PC να επιλεγεί από τον καταχωρητή C, ο οποίος περιέχει τη διεύθυνση προορισμού που υπολογίστηκε στο 1ο βήμα του κύκλου εντολής. Η εγγραφή στον PC γίνεται μόνο εάν το ψηφίο Zero έχει τιμή ‘1’.

Οι εντολές άλματος απαιτούν, όπως και οι εντολές διακλάδωσης, μόνο μία κατάσταση μετά την 1. Στην κατάσταση αυτή, που δίνεται στο παρακάτω διάγραμμα, ενεργοποιείται το



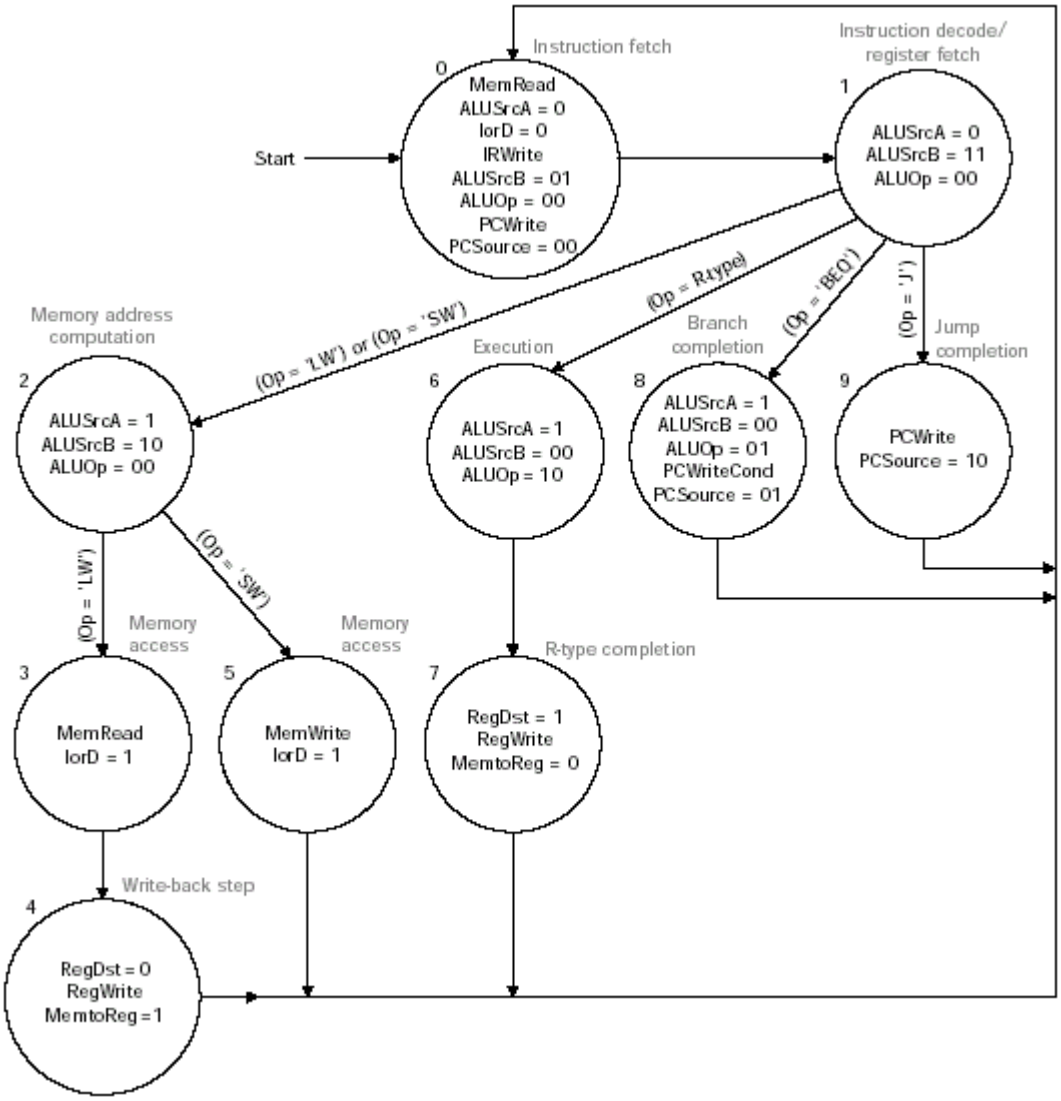
σήμα PCWrite για εγγραφή του PC χωρίς συνθήκη. Τέλος, το σήμα PCSource πρέπει να πάρει τιμή “10”, ώστε η τιμή που θα εισαχθεί στον PC να είναι τα 4 πιο σημαντικά ψηφία του PC, σε παράθεση με τα 26 λιγότερο σημαντικά ψηφία του IR, ολισθημένα αριστερά κατά 2 θέσεις.

Συνδέοντας τα επιμέρους τμήματα της μηχανής καταστάσεων, για να πάρουμε τη μηχανή που ορίζει τη ΜΕ της ΜΕΔ πολλαπλών κύκλων μηχανής ανά κύκλο εντολής, λαμβάνουμε το διάγραμμα της επόμενης σελίδας. Τα σήματα ελέγχου κάθε κατάστασης καθορίζονται όπως φαίνονται στο διάγραμμα. Η επόμενη κάθε κατάστασης εξαρτάται από τον κωδικό λειτουργίας της λέξης εντολής, γι’ αυτό τοποθετούμε ετικέτες στα τόξα μετάβασης που να υποδεικνύουν τον κατάλληλο κωδικό.

Έχοντας ολοκληρώσει τη σχεδίαση της ΜΕ με τη βοήθεια της μηχανής καταστάσεων που την περιγράφει, ας δούμε ποιος είναι ο αριθμός CPI της ΜΕΔ πολλαπλών κύκλων μηχανής για ένα τυπικό μίγμα εντολών, όταν κάθε κατάσταση της ΜΕ διαρκεί έναν κύκλο μηχανής.

Παράδειγμα 2

Θεωρήστε την παραπάνω υλοποίηση μιας ΜΕΔ MIPS πολλαπλών κύκλων μηχανής ανά κύκλο εντολής, και το μετροπρόγραμμα gcc με το ακόλουθο μίγμα εντολών: 22% εντολές φόρτωσης, 11% εντολές αποθήκευσης, 49% R-εντολές, 16% διακλαδώσεις και 2% εντολές άλματος.
Ποιος είναι ο αριθμός CPI της εκτέλεσης του προγράμματος αυτού, όταν κάθε κατάσταση της ΜΕ διαρκεί 1 κύκλο μηχανής;



Όπως προκύπτει από το διάγραμμα μετάβασης της μηχανής καταστάσεων, ο αριθμός κύκλων μηχανής για κάθε τύπο εντολής είναι:

- Εντολές φόρτωσης: 5
- Εντολές αποθήκευσης: 4
- R-εντολές: 4
- Διακλαδώσεις: 3
- Άλματα: 3

Ο αριθμός CPI βρίσκεται ως εξής:

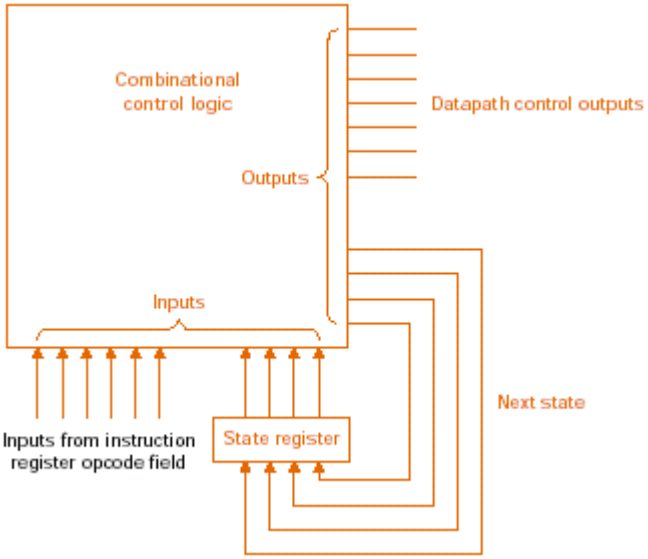
$$\begin{aligned} \text{CPI} &= \text{Αριθμός κύκλων μηχανής} / \text{Αριθμός εντολών} \\ &= \sum \{\text{Αριθμός εντολών}_i \times \text{CPI}_i\} / \text{Αριθμός εντολών} \\ &= \sum \{ (\text{Αριθμός εντολών}_i / \text{Αριθμός εντολών}) \times \text{CPI}_i \} \end{aligned}$$

Το πηλίκο σε παρένθεση είναι η συχνότητα εμφάνισης των εντολών τύπου i, που μας δίνεται στην εκφώνηση για κάθε διαφορετικό i. Επομένως, αντικαθιστώντας τις συχνότητες και τους επιμέρους αριθμούς CPI, έχουμε:

$$\begin{aligned} \text{CPI} &= 22\% \times 5 + 11\% \times 4 + 49\% \times 4 + 16\% \times 3 + 2\% \times 3 \\ &= 1.1 + 0.44 + 1.96 + 0.48 + 0.06 \\ &= 4.04 \end{aligned}$$

Ο παραπάνω αριθμός CPI είναι καλύτερος από τον αριθμό CPI που θα είχαμε, αν – στη χειρότερη περίπτωση – όλες οι εντολές ολοκληρώνονταν στο μέγιστο χρόνο των 5 κύκλων μηχανής.□

Μια μηχανή καταστάσεων μπορεί να υλοποιηθεί σε υλικό με τη βοήθεια ενός καταχωρητή που αποθηκεύει την παρούσα κατάσταση, και ενός συνδυαστικού κυκλώματος που καθορίζει τόσο τις τιμές των σημάτων ελέγχου της ΜΕΔ, όσο και την επόμενη κατάσταση. Το ακόλουθο σχήμα δίνει τη γενική μορφή μιας τέτοιας υλοποίησης.



Η μηχανή που περιγράψαμε αντιστοιχεί στην κατηγορία των μηχανών καταστάσεων που ονομάζονται μηχανές Moore. Το χαρακτηριστικό αυτών των μηχανών είναι ότι οι έξοδοί τους εξαρτώνται αποκλειστικά από την παρούσα κατάσταση, και όχι από τις εισόδους τους. Έτσι, το συνδυαστικό μέρος αυτών των μηχανών μπορεί να χωριστεί σε δύο τμήματα, από τα οποία το πρώτο παράγει τις εξόδους σα συνάρτηση της παρούσας κατάστασης, και το δεύτερο την επόμενη κατάστασή τους. Σε σύγκριση με άλλες μηχανές, οι μηχανές Moore οδηγούν σε υλοποιήσεις ΜΕ που είναι πιο γρήγορες και καταλαμβάνουν λιγότερο χώρο στη ΜΕΔ.

Στην επόμενη παράγραφο θα δούμε μια εναλλακτική υλοποίηση της ΜΕ της ΜΕΔ πολλαπλών κύκλων μηχανής. Οι δύο υλοποιήσεις αποτελούν απλά διαφορετική αναπαράσταση της ίδιας πληροφορίας που ορίζει τη ΜΕ.

5.5 Μικροπρογραμματισμός: Απλοποιώντας τη σχεδίαση της ΜΕ

Για την υλοποίηση της ΜΕ της απλουστευμένης ΜΕΔ MIPS που έχουμε, μια γραφική αναπαράσταση της μηχανής καταστάσεων, όπως αυτή που είδαμε, είναι αρκετή. Μπορούμε να σχεδιάσουμε ένα διάγραμμα μετάβασης σχετικά εύκολα σε μια σελίδα Α4, καθώς και να βρούμε χωρίς πολλά λάθη και τις αντίστοιχες λογικές εξισώσεις, αν το επιθυμούμε. Θεωρήστε τώρα μια ΜΕΔ για το πλήρες σύνολο εντολών MIPS, που περιλαμβάνει πάνω από 100 εντολές. Σε κάποια υλοποίηση αυτής της ΜΕΔ, οι εντολές είναι δυνατό να ολοκληρώνονται σε χρόνο από 1 έως πάνω από 20 κύκλους μηχανής. Είναι φανερό ότι η αντίστοιχη ΜΕ θα είναι αρκετά πιο πολύπλοκη από αυτή που είδαμε πιο πάνω. Θεωρήστε ακόμα ένα σύνολο εντολών CISC, όπως για παράδειγμα το σύνολο εντολών 80x86, που περιλαμβάνει εντολές πολλών διαφορετικών τύπων. Η ΜΕ της ΜΕΔ που το υλοποιεί μπορεί πολύ εύκολα να απαιτεί χιλιάδες καταστάσεις, με εκατοντάδες διαφορετικές ακολουθίες μεταβάσεων, μια που – ειδικά για το σύνολο αυτό – υποστηρίζει πολύ περισσότερους συνδυασμούς διευθυνσιοδοτήσεων, καθώς και κωδικούς λειτουργίας εντολών.

Σε τέτοιες περιπτώσεις, η σχεδίαση τη ΜΕ με κάποια γραφική αναπαράσταση είναι εξαιρετικά επίπονη, αφού η μηχανή καταστάσεων θα έχει τόσο πολλές καταστάσεις – και ακόμα περισσότερες μεταβάσεις! Η γραφική αναπαράσταση, χρήσιμη για μια μικρή μηχανή καταστάσεων, δε θα χωράει σε μια σελίδα, και θα είναι εξαιρετικά δυσνόητη και επομένως δύσχρηστη, όσο γίνεται μεγαλύτερη. Κάτι παρόμοιο αντιμετωπίζει κι ένας προγραμματιστής:

Όσο ένα πρόγραμμα γίνεται μεγαλύτερο, ειδικές δομές – για παράδειγμα υποπρογράμματα – είναι απαραίτητες, για να διατηρούν το πρόγραμμα κατανοητό. Επίσης, η εύρεση λογικών εξισώσεων για σήματα ελέγχου από μια τέτοια αναπαράσταση, χωρίς αυτές να περιέχουν λάθη, είναι ουσιαστικά αδύνατη.

Μπορούμε να δανειστούμε ιδέες προγραμματισμού, ώστε να βρούμε μια μέθοδο υλοποίησης της ΜΕ μιας πολύπλοκης ΜΕΔ που να επιτρέπει εύκολη κατανόηση και σχεδίαση αυτής; Ας προσπαθήσουμε να δούμε την ενεργοποίηση των σημάτων ελέγχου μιας κατάστασης σαν την εκτέλεση μιας εντολής χαμηλού επιπέδου. Για να μη μπερδέψουμε μια τέτοια εντολή – που εκτελείται σε μια και μόνη κατάσταση – με μια εντολή του συνόλου εντολών του επεξεργαστή – που εκτελείται σε πολλαπλές διαδοχικές καταστάσεις, την ονομάζουμε *μικροεντολή*. Μια μικροεντολή λοιπόν ορίζει το σύνολο των σημάτων ελέγχου που ενεργοποιούνται σε κάποια συγκεκριμένη κατάσταση, και εκτελείται, ενεργοποιώντας τα σήματα αυτά.

Για να προχωρήσουμε όμως σε μια υλοποίηση της ΜΕ βασισμένοι στην εκτέλεση μικροεντολών, πρέπει, εκτός από τον καθορισμό των σημάτων ελέγχου που ενεργοποιούνται σε κάθε μικροεντολή, να υλοποιήσουμε ένα μηχανισμό διαδοχής μικροεντολών, ένα μηχανισμό δηλαδή που να μας δίνει ποια είναι η επόμενη κάθε μικροεντολής. Παρατηρώντας το διάγραμμα μετάβασης της μηχανής καταστάσεων που υλοποιήσαμε νωρίτερα, βλέπουμε ότι η επόμενη κατάσταση καθορίζεται με έναν από δύο τρόπους. Κάποιες φορές η παρούσα κατάσταση ακολουθείται από μία μοναδική κατάσταση. Για παράδειγμα, η κατάσταση 1 είναι η μόνη κατάσταση που ακολουθεί την κατάσταση 0, και μάλιστα δεν υπάρχει άλλος τρόπος μετάβασης στην 1, παρά μόνο από την 0. Άλλες φορές η παρούσα κατάσταση ακολουθείται από περισσότερες της μίας καταστάσεις, και η επιλογή μετάβασης γίνεται με βάση την είσοδο. Αυτό συμβαίνει στην κατάσταση 1, η οποία ακολουθείται από 4 διαφορετικές καταστάσεις.

Όταν γράφουμε κάποιο πρόγραμμα, συναντάμε το ανάλογο φαινόμενο. Κάποιες φορές οι εντολές του προγράμματος εκτελούνται σε μια μοναδική σειρά, και άλλες φορές μεταξύ αυτών παρεμβάλλονται διακλαδώσεις. Μάλιστα, η εκτέλεση σε σειρά είναι υπονοούμενη, ενώ η διακλάδωση πρέπει να δηλώνεται με κάποια ειδική εντολή. Στην υλοποίηση της ΜΕ σαν πρόγραμμα μικροεντολών – το οποίο λέγεται *μικροπρόγραμμα*, επίσης αφήνουμε την εκτέλεση σε σειρά ως υπονοούμενη, και απαιτούμε τη χρήση ειδικών κωδικών για υποστήριξη διακλαδώσεων. Ο μηχανισμός διαδοχής μικροεντολών μπορεί να υλοποιηθεί μέσω ενός ακολουθιακού κυκλώματος, ή ακόμα καλύτερα με τη χρήση ειδικού μετρητή μικροεντολών, όπως θα δούμε αργότερα.

Η σχεδίαση της ΜΕ σαν πρόγραμμα που υλοποιεί τις εντολές μηχανής του επεξεργαστή με τη βοήθεια των απλούστερων μικροεντολών ονομάζεται *μικροπρογραμματισμός*. Η βασική ιδέα είναι η απεικόνιση των τιμών των σημάτων ελέγχου της ΜΕΔ στις μικροεντολές να γίνεται με ένα συμβολικό τρόπο, ώστε το μικροπρόγραμμα να γράφεται, όπως γράφεται η συμβολική γλώσσα σαν απεικόνιση της γλώσσας μηχανής. Συντάσσοντας μια συμβολική γλώσσα, αναπαριστάμε τις εντολές μηχανής σα σύνολα από διαφορετικά πεδία (πχ. κωδικός λειτουργίας, αριθμοί καταχωρητών, άμεσο τελούμενο), και αυτό θέλουμε να κάνουμε και τώρα.

Ορίζοντας τη μορφή μιας μικροεντολής

Το μικροπρόγραμμα είναι η συμβολική αναπαράσταση του ελέγχου των εντολών, που με τη βοήθεια κάποιου προγράμματος μεταφράζεται σε μια ακολουθία κωδικών τιμών ελέγχου. Έτσι, μπορούμε να επιλέξουμε πόσα πεδία χρειάζεται μια μικροεντολή, και ποια σήματα ελέγχου καθορίζει κάθε πεδίο. Η μορφή της μικροεντολής πρέπει να απλοποιεί την αναπαράσταση, ώστε να είναι εύκολη τόσο η συγγραφή όσο και η κατανόηση του μικροπρογράμματος. Για παράδειγμα, είναι χρήσιμο να υπάρχει ένα πεδίο ελέγχου της ΑΛΜ, και τρία πεδία που υποδεικνύουν τις πηγές των δύο εισόδων και τον προορισμό της εξόδου της ΑΛΜ. Εκτός της απλοποίησης, άλλος σκοπός της συμβολικής αναπαράστασης είναι να κάνει αδύνατη τη συγγραφή ασύμβατων μικροεντολών. Μια μικροεντολή είναι ασύμβατη, εάν αποδίδει δύο διαφορετικές τιμές στο ίδιο σήμα ελέγχου.

Για να αποτρέψουμε τη συγγραφή ασύμβατων μικροεντολών, πρέπει τα πεδία της μικροεντολής να καθορίζουν μη επικαλυπτόμενα σύνολα σημάτων ελέγχου. Για να διαμερίσουμε τα σήματα ελέγχου σε διαφορετικά πεδία, πρέπει να έχουμε υπ' όψη:

- τη συνολική εικόνα της ΜΕΔ με όλα τα σήματα ελέγχου, και
- τους πίνακες που δίνουν τις λειτουργίες των σημάτων ελέγχου.

Σήματα που δεν ενεργοποιούνται ποτέ ταυτόχρονα μπορούν να παίρνουν τιμές στο ίδιο πεδίο της μικροεντολής. Ο παρακάτω πίνακας δείχνει μια δυνατή μορφή μικροεντολής με 7 πεδία για την απλοποιημένη ΜΕΔ MIPS των πολλαπλών κύκλων μηχανής, την οποία και θα χρησιμοποιήσουμε στη συνέχεια. Τα 6 από τα 7 πεδία αποδίδουν τιμές στα σήματα ελέγχου της ΜΕΔ, ενώ το 7ο πεδίο οδηγεί το μηχανισμό εύρεσης της επόμενης μικροεντολής.

Όνομα πεδίου	Περιγραφή πεδίου
ALU control	Καθορίζει την πράξη ΑΛΜ για τον παρόντα κύκλο μηχανής. Το αποτέλεσμα της πράξης γράφεται στον καταχωρητή C.
SRC1	Επιλέγει την πηγή της πρώτης εισόδου της ΑΛΜ.
SRC2	Επιλέγει την πηγή της δεύτερης εισόδου της ΑΛΜ.
Register control	Καθορίζει τη λειτουργία προσπέλασης του ΦΚ και επιλέγει την πηγή εισόδου δεδομένων για προσπέλαση εγγραφής.
Memory	Καθορίζει τη λειτουργία προσπέλασης της μονάδας μνήμης και επιλέγει την πηγή της εισόδου διευθύνσεων. Για ανάγνωση, επιλέγει τον καταχωρητή αποθήκευσης – IR ή MDR.
PCWrite control	Καθορίζει την εγγραφή του PC.
Sequencing	Καθορίζει τον τρόπο επιλογής της επόμενης μικροεντολής.

Οι μεταφρασμένες μικροεντολές τοποθετούνται συνήθως σε μια μονάδα ROM ή PLA, όπου αποκτούν αριθμητικές διευθύνσεις. Οι διευθύνσεις αποδίδονται συνήθως στη σειρά, με τον ίδιο τρόπο που εμείς αποδώσαμε αριθμούς στις καταστάσεις της προηγούμενης υλοποίησης της ΜΕ. Η επιλογή της επόμενης προς εκτέλεση μικροεντολής γίνεται με έναν από τους ακόλουθους τρεις τρόπους[6]:

1. Αυξάνοντας την τιμή της διεύθυνσης της παρούσας μικροεντολής κατά 1, ώστε η επόμενη μικροεντολή να προσκομιστεί από τη θέση που ακολουθεί σειριακά τη θέση της παρούσας. Η επιλογή αυτή αναπαριστάται στο πεδίο Sequencing της μικροεντολής με το συμβολισμό "Seq". Σε πολλά συστήματα μικροπρογραμματισμού, η επιλογή αυτή είναι προεπιλεγμένη και υπονοείται.
2. Εκτελώντας άλμα στη μικροεντολή με την οποία ξεκινάει ο κύκλος εντολής. Η μικροεντολή αυτή αντιστοιχεί στην κατάσταση 0 της μηχανής καταστάσεων που είδαμε νωρίτερα, και το άλμα σε αυτή συμβολίζεται στο πεδίο Sequencing ως "Fetch".
3. Επιλέγοντας τη νέα διεύθυνση από έναν πίνακα διευθύνσεων. Αυτός ο πίνακας διευθυνσιοδοτείται με βάση την είσοδο της ΜΕ και μπορεί να υλοποιηθεί σε μια άλλη μονάδα ROM ή PLA. Ανάλογα με το πόσες φορές χρησιμοποιούμε αυτό τον τρόπο επιλογής στο μικροκώδικα, υλοποιούμε ανάλογο αριθμό τέτοιων πινάκων. Στην περίπτωσή μας χρειαζόμαστε δύο πίνακες, για επιλογή επόμενης μικροεντολής από αυτές που αντιστοιχούν στις καταστάσεις 1 (πίνακας 1) και 2 (πίνακας 2) της μηχανής καταστάσεων. Ο τρόπος αυτός επιλογής αναπαριστάται με το συμβολισμό "Dispatch i", όπου i είναι ο αριθμός του πίνακα.

Στον πίνακα της επόμενης σελίδας δίνονται οι δυνατές τιμές όλων των πεδίων της μικροεντολής για τη ΜΕ που σχεδιάζουμε, και η λειτουργία – ή οι λειτουργίες – για κάθε τιμή. Υπενθυμίζουμε ότι το μικροπρόγραμμα είναι μια συμβολική αναπαράσταση και ότι η μορφή της μικροεντολής που χρησιμοποιούμε είναι μία από πολλές δυνατές μορφές.

[6] Εκτός από αυτούς τους τρεις που χρησιμοποιούμε για τη συγκεκριμένη ΜΕΔ MIPS, άλλοι τρόποι περιλαμβάνουν: (α) άμεσο άλμα σε τυχαία θέση του μικροπρογράμματος, και (β) διακλάδωση με τη βοήθεια κωδικού συνθήκης, η οποία αποτιμάται κατά την εκτέλεση της μικροεντολής.

Όνομα πεδίου	Τιμή πεδίου	Λειτουργία
Label	*οποιαδήποτε συμβολοσειρά*	Επιπλέον των 7 πεδίων που περιγράψαμε νωρίτερα, το πεδίο Label μας επιτρέπει να ονομάσουμε κάποια διεύθυνση στο μικροπρόγραμμα με κάποια ετικέτα. Το πεδίο αυτό δε μεταφράζεται, απλά καθιστά το μικροπρόγραμμα πιο κατανοητό, και βοηθάει στην κατασκευή των πινάκων διευθύνσεων. Αν μια ετικέτα τελειώνει σε '1' ή '2', η διεύθυνσή της τοποθετείται σε κατάλληλη θέση του αντίστοιχου πίνακα.
ALU control	Add	Πράξη πρόσθεσης (ADD) στην ΑΛΜ.
	Subt	Πράξη αφαίρεσης (SUB) στην ΑΛΜ.
	Func code	Πράξη ΑΛΜ που καθορίζεται από τον κωδικό τελεστή της λέξης εντολής.
SRC1	PC	Επιλογή PC στην πρώτη είσοδο της ΑΛΜ.
	A	Επιλογή καταχωρητή A στην πρώτη είσοδο της ΑΛΜ.
SRC2	B	Επιλογή καταχωρητή B στη δεύτερη είσοδο της ΑΛΜ.
	4	Επιλογή σταθεράς 4 στη δεύτερη είσοδο της ΑΛΜ.
	Extend	Επιλογή της εξόδου της υπομονάδας προέκτασης προσήμου στη δεύτερη είσοδο της ΑΛΜ.
	Extshft	Επιλογή της εξόδου της υπομονάδας αριστερής ολίσθησης στη δεύτερη είσοδο της ΑΛΜ.
Register control	Read	Ανάγνωση του ΦΚ, με βάση τα πεδία rs και rt της λέξης εντολής. Οι έξοδοι αποθηκεύονται στους καταχωρητές A και B, αντίστοιχα.
	Write ALU	Εγγραφή του περιεχομένου του C στο ΦΚ, στον καταχωρητή του πεδίου rd της λέξης εντολής.
	Write MDR	Εγγραφή του περιεχομένου του MDR στο ΦΚ, στον καταχωρητή του πεδίου rt της λέξης εντολής.
Memory	Read PC	Ανάγνωση μνήμης, από τη διεύθυνση που περιέχει ο PC. Η έξοδος αποθηκεύεται στον IR (και στον MDR).
	Read ALU	Ανάγνωση μνήμης, από τη διεύθυνση που περιέχει ο C. Η έξοδος αποθηκεύεται μόνο στον MDR.
	Write ALU	Εγγραφή του περιεχομένου του B στη μνήμη, στη διεύθυνση που περιέχει ο C.
PCWrite control	ALU	Αποθήκευση της εξόδου της ΑΛΜ απ' ευθείας στον PC.
	C-cond	Αντιγραφή του περιεχομένου του C στον PC, μόνο εάν το σήμα Zero είναι ενεργοποιημένο.
	Jump address	Εγγραφή της διεύθυνσης προορισμού άμεσου άλματος στον PC.
Sequencing	Seq	Επόμενη μικροεντολή θα είναι η μικροεντολή που ακολουθεί την παρούσα.
	Fetch	Επόμενη μικροεντολή θα είναι η πρώτη μικροεντολή του μικροπρογράμματος.
	Dispatch i	Η διεύθυνση της επόμενης μικροεντολής λαμβάνεται από τον πίνακα i.

Γράφοντας το μικροπρόγραμμα

Ας γράψουμε τώρα το μικροπρόγραμμα της ΜΕ που θέλουμε. Σε κάθε μικροεντολή θα αποδώσουμε τιμές στα πεδία της, χρησιμοποιώντας τα συμβολικά ονόματα του πίνακα. Ακόμα, θα αποδώσουμε ετικέτες σε κατάλληλες μικροεντολές, οι οποίες να μπορούν να χρησιμοποιηθούν για την κατασκευή των πινάκων διευθύνσεων 1 και 2. Υπάρχουν δύο περιπτώσεις, στις

οποίες μπορούμε να αφήσουμε κάποια πεδία χωρίς τιμές. Για πεδία που ορίζουν σήματα ελέγχου υπομονάδων ή σήματα επίτρεψης εγγραφής, η μη απόδοση τιμής ισοδυναμεί με απενεργοποίηση των αντίστοιχων σημάτων. Για πεδία που ορίζουν σήματα επιλογής πολυπλεκτών, όμως, η μη απόδοση τιμής σημαίνει ότι τα αντίστοιχα σήματα δε μας ενδιαφέρουν για τις συγκεκριμένες μικροεντολές, κι επομένως μπορεί να έχουν οποιαδήποτε τιμή.

Ο ευκολότερος τρόπος να κατανοήσουμε το μικροπρόγραμμα είναι να το χωρίσουμε σε τμήματα, που το καθένα αφορά συγκεκριμένο μέρος του κύκλου εντολής, για κάθε τύπο εντολής, όπως ακριβώς κάναμε και στη σχεδίαση της μηχανής καταστάσεων λίγο νωρίτερα.

Το πρώτο μέρος του κύκλου κάθε εντολής προσκομίζει την εντολή, την αποκωδικοποιεί, και υπολογίζει την επόμενη διεύθυνση ανάκλησης και τη διεύθυνση προορισμού διακλάδωσης. Το μέρος αυτό καλύπτει τα δύο πρώτα βήματα που είδαμε στην προηγούμενη υλοποίηση. Οι δύο μικροεντολές που χρειαζόμαστε θα είναι:

Label	ALU control	SRC1	SRC2	Register control	Memory	PCWrite control	Sequencing
Fetch	Add	PC	4		Read PC	ALU	Seq
	Add	PC	Extshft	Read			Dispatch 1

Για να καταλάβουμε καλύτερα τι κάνει καθεμία από τις δύο αυτές μικροεντολές, ας δούμε τι λειτουργίες εκτελούνται από τις τιμές που αποδώσαμε στα πεδία της. Πιο συγκεκριμένα, για την πρώτη μικροεντολή έχουμε:

Πεδία	Λειτουργίες
ALU control, SRC1, SRC2	Υπολογισμός του αθροίσματος PC + 4. (Το αποτέλεσμα αποθηκεύεται στον C, αν και δεν το διαβάζουμε από εκεί.)
Memory	Ανάγνωση από τη διεύθυνση που περιέχει ο PC και αποθήκευση της εξόδου στον IR.
PCWrite control	Αποθήκευση της εξόδου της ΑΛΜ στον PC.
Sequencing	Επόμενη μικροεντολή είναι αυτή που ακολουθεί.

Το πεδίο Label, που έχει την τιμή “Fetch”, υποδεικνύει τη διεύθυνση της μικροεντολής, στην οποία γίνεται μετάβαση με την αντίστοιχη επιλογή του πεδίου Sequencing στο τέλος του κύκλου εντολής.

Για τη δεύτερη από τις παραπάνω μικροεντολές έχουμε τις παρακάτω λειτουργίες:

Πεδία	Λειτουργίες
ALU control, SRC1, SRC2	Υπολογισμός του αθροίσματος PC + (sign-extend(IR[15-0]) << 2). Το αποτέλεσμα αποθηκεύεται στον C.
Register control	Ανάγνωση του ΜΚ και μεταφορά του περιεχομένου των καταχωρητών που υποδεικνύουν τα πεδία rs και rt της λέξης εντολής στους καταχωρητές A και B, αντίστοιχα.
Sequencing	Η διεύθυνση της επόμενης μικροεντολής λαμβάνεται από τον πίνακα διευθύνσεων 1.

Η λειτουργία που υποδηλώνεται με τιμή “Dispatch 1” μπορεί να παραλληλιστεί με τις εντολές case της Pascal ή switch της C, με τον κωδικό λειτουργίας της λέξης εντολής να χρησιμοποιείται για την επιλογή της κατάλληλης διεύθυνσης από τον πίνακα 1. Ο πίνακας 1 θα πρέπει επομένως να περιέχει τις διευθύνσεις που συμβολίζουν οι ετικέτες:

- “Mem1” για εντολές προσπέλασης μνήμης,
- “Rformat1” για R-εντολές,
- “BEQ1” για την εντολή διακλάδωσης beq, και
- “JUMP1” για την εντολή άμεσου άλματος j.

Οι παραπάνω ετικέτες έχουν επιλεγεί αυθαίρετα, και θα δούμε στη συνέχεια τις μικροεντολές στις οποίες αυτές αντιστοιχούν.

Ο μικροκώδικας για εντολές προσπέλασης μνήμης περιλαμβάνει τέσσερις μικροεντολές, που παρουσιάζονται στη συνέχεια. Η πρώτη από αυτές υπολογίζει την τελική διεύθυνση προσπέλασης. Δύο μικροεντολές απαιτούνται για την ολοκλήρωση μιας εντολής φόρτωσης (ανάγνωση μνήμης που ακολουθείται από εγγραφή του ΦΚ), ενώ μία μικροεντολή απαιτείται για την ολοκλήρωση μιας εντολής αποθήκευσης, μετά τον υπολογισμό της τελικής διεύθυνσης προσπέλασης:

Label	ALU control	SRC1	SRC2	Register control	Memory	PCWrite control	Sequencing
Mem1	Add	A	Extend				Dispatch 2
LW2					Read ALU		Seq
				Write MDR			Fetch
SW2					Write ALU		Fetch

Για τα πεδία της πρώτης από τις παραπάνω μικροεντολές έχουμε:

Πεδία	Λειτουργίες
ALU control, SRC1, SRC2	Υπολογισμός του αθροίσματος Reg[rs] + sign-extend(IR[15-0]). Το αποτέλεσμα αποθηκεύεται στον C.
Sequencing	Η διεύθυνση της επόμενης μικροεντολής λαμβάνεται από τον πίνακα διευθύνσεων 2.

Ο πίνακας διευθύνσεων 2 θα περιέχει τις διευθύνσεις που συμβολίζουν οι ετικέτες:

- “LW2” για εντολές φόρτωσης, και
- “SW2” για εντολές αποθήκευσης.

Η επόμενη μικροεντολή είναι αυτή στην οποία γίνεται μετάβαση μέσω του πίνακα διευθύνσεων 2 για τις εντολές φόρτωσης, και στην οποία έχει αποδοθεί η ετικέτα “LW2”. Το αποτέλεσμα εκτέλεσης της μικροεντολής αυτής για κάθε πεδίο της είναι:

Πεδία	Λειτουργίες
Memory	Ανάγνωση από τη διεύθυνση που περιέχει ο καταχωρητής C, και αποθήκευση της εξόδου στον MDR.
Sequencing	Επόμενη μικροεντολή είναι αυτή που ακολουθεί.

Η τρίτη μικροεντολή ολοκληρώνει την εκτέλεση μιας εντολής φόρτωσης ως εξής:

Πεδία	Λειτουργίες
Register control	Εγγραφή του περιεχομένου του MDR στο ΦΚ, στον καταχωρητή του πεδίου rt της λέξης εντολής.
Sequencing	Επόμενη μικροεντολή είναι αυτή με ετικέτα “Fetch”.

Η τέταρτη μικροεντολή ολοκληρώνει την εκτέλεση μιας εντολής αποθήκευσης. Έχει ετικέτα “SW2” και μετάβαση σε αυτήν γίνεται μέσω του πίνακα διευθύνσεων 2. Οι λειτουργίες της περιγράφονται ως εξής:

Πεδία	Λειτουργίες
Memory	Εγγραφή του περιεχομένου του καταχωρητή B στη διεύθυνση μνήμης που περιέχει ο καταχωρητής C.
Sequencing	Επόμενη μικροεντολή είναι αυτή με ετικέτα “Fetch”.

Ο μικροκώδικας για R-εντολές αποτελείται από δύο μικροεντολές. Η πρώτη, που έχει ετικέτα “Rformat1”, εκτελεί την πράξη ΑΛΜ, ενώ η δεύτερη αποθηκεύει το αποτέλεσμα στο ΦΚ:

Label	ALU control	SRC1	SRC2	Register control	Memory	PCWrite control	Sequencing
Rformat1	Func code	A	B				Seq
				Write ALU			Fetch

Παρατηρούμε ότι οι παραπάνω δύο μικροεντολές δεν εμφανίζουν επικάλυψη στα πεδία τους, μ' άλλα λόγια αποδίδουν τιμές ελέγχου σε διαφορετικά πεδία. Αυτό όμως δε σημαίνει ότι μπορούν να συνδυαστούν σε μία και μόνο μικροεντολή. Με σκοπό τη βέλτιστη συγγραφή μικροκώδικα, μπορούμε να προσπαθούμε να συνδυάζουμε διαδοχικές μικροεντολές σε μία. Στην προκειμένη περίπτωση όμως, το αποτέλεσμα της πράξης ΑΛΜ αποθηκεύεται στον καταχωρητή C, από τον οποίο δε μπορεί να διαβαστεί πριν τον επόμενο κύκλο μηχανής. Γι' αυτό και δε μπορούμε να συνδυάσουμε τις δύο μικροεντολές, διαφορετικά θα διαβάζαμε λάθος τιμή από τον C για αποθήκευση στο ΦΚ! Αν πάλι δοκιμάζαμε να αφαιρέσουμε τον καταχωρητή C, ώστε η έξοδος της ΑΛΜ να αποστέλλεται κατ' ευθείαν στο ΦΚ, θα μπορούσαμε να επιτύχουμε τον παραπάνω συνδυασμό, αλλά θα έπρεπε να επιμηκύνουμε τη διάρκεια του κύκλου μηχανής, ώστε η πράξη ΑΛΜ και η προσπέλαση του ΦΚ να εκτελούνται στον ίδιο κύκλο. (Θυμηθείτε πώς και γιατί μεταβήκαμε από τη ΜΕΔ απλού κύκλου μηχανής στη ΜΕΔ των πολλαπλών κύκλων μηχανής ανά κύκλο εντολής!!)

Η πρώτη από τις δύο μικροεντολές που αφορούν την εκτέλεση των R-εντολών έχει το εξής αποτέλεσμα:

Πεδία	Λειτουργίες
ALU control, SRC1, SRC2	Εκτέλεση της πράξης ΑΛΜ που υποδεικνύεται από τον κωδικό τελεστή της λέξης εντολής, πάνω στις τιμές που περιέχουν οι A και B. Το αποτέλεσμα αποθηκεύεται στον C.
Sequencing	Επόμενη μικροεντολή είναι αυτή που ακολουθεί.

Η δεύτερη μικροεντολή αποθηκεύει το αποτέλεσμα στο ΦΚ:

Πεδία	Λειτουργίες
Register control	Εγγραφή του περιεχομένου του C στο ΦΚ, στον καταχωρητή του πεδίου rd της λέξης εντολής.
Sequencing	Επόμενη μικροεντολή είναι αυτή με ετικέτα "Fetch".

Όσο αφορά την εντολή διακλάδωσης beq, επειδή η διεύθυνση προορισμού άλματος έχει ήδη υπολογιστεί και είναι αποθηκευμένη στον C, η εκτέλεσή της ολοκληρώνεται με μία μόνο μικροεντολή, η οποία έχει ετικέτα "BEQ1":

Label	ALU control	SRC1	SRC2	Register control	Memory	PCWrite control	Sequencing
BEQ1	Subt	A	B			C-cond	Fetch

Η εκτέλεση της μικροεντολής αυτής περιγράφεται ως εξής:

Πεδία	Λειτουργίες
ALU control, SRC1, SRC2	Εκτέλεση αφαίρεσης της τιμής που περιέχει ο B από αυτήν που περιέχει ο A. (Το αποτέλεσμα αποθηκεύεται στον C, αν και δεν το διαβάζουμε από εκεί.)
PCWrite control	Αντιγραφή του περιεχομένου του C στον PC, αν η πράξη ΑΛΜ ενεργοποιεί το σήμα Zero.
Sequencing	Επόμενη μικροεντολή είναι αυτή με ετικέτα "Fetch".

Ο μικροκώδικας της εντολής άμεσου άλματος αποτελείται επίσης από μία μικροεντολή:

Label	ALU control	SRC1	SRC2	Register control	Memory	PCWrite control	Sequencing
JUMP1						Jump address	Fetch

Η εκτέλεση του άμεσου άλματος απαιτεί απόδοση τιμής σε δύο μόνο πεδία της μικροεντολής, με το ακόλουθο αποτέλεσμα:

Πεδία	Λειτουργίες
PCWrite control	Ο PC παίρνει τιμή τη διεύθυνση προορισμού άμεσου άλματος.
Sequencing	Επόμενη μικροεντολή είναι αυτή με ετικέτα “Fetch”.

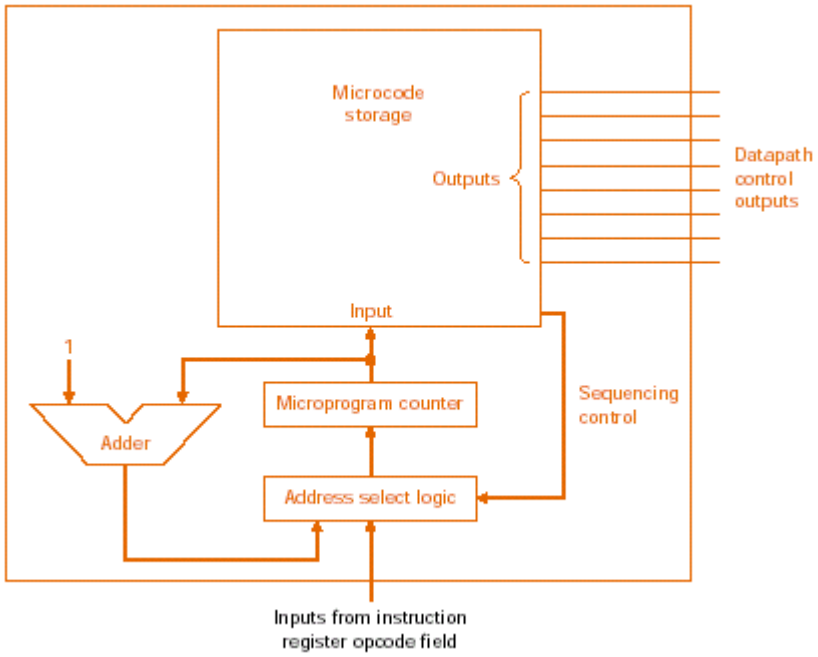
Το συνολικό μικροπρόγραμμα που υλοποιεί τη ζητούμενη ΜΕΔ δίνεται στον πίνακα που ακολουθεί. Αποτελείται από τις 10 μικροεντολές που μελετήσαμε πιο πάνω. Παρατηρούμε την αντιστοίχηση των μικροεντολών αυτών με τις 10 καταστάσεις της μηχανής καταστάσεων της προηγούμενης υλοποίησης της ΜΕ. Η αντιστοίχηση αυτή είναι αναμενόμενη, αφού οι δύο υλοποιήσεις έχουν σα βάση τα 5 βήματα του κύκλου εντολής που περιγράψαμε, όταν μεταβήκαμε από την υλοποίηση του απλού κύκλου μηχανής στην υλοποίηση των πολλαπλών κύκλων μηχανής ανά κύκλο εντολής. Σε πιο πολύπλοκες αρχιτεκτονικές, ο μικροκώδικας μπορεί να αποτελείται από εκατοντάδες ή και χιλιάδες μικροεντολές, και θα αποτελεί την πιο εύκολη μέθοδο σχεδίασης της ΜΕ. Οι ΜΕΔ τέτοιων αρχιτεκτονικών συνήθως χρειάζονται πολλούς βοηθητικούς καταχωρητές ειδικού σκοπού για αποθήκευση ενδιάμεσων αποτελεσμάτων, όπως έχουμε εμείς για παράδειγμα τους A και B. Επομένως θα διαθέτουν και πιο πολύπλοκες διασυνδέσεις μεταξύ αυτών και των υπομονάδων της ΜΕΔ, γεγονός που ενισχύει ακόμα περισσότερο το επιχείρημα υπέρ της χρήσης του μικροπρογραμματισμού σα μεθόδου υλοποίησης της ΜΕ, αντί της καλωδιωμένης λογικής του ακολουθιακού κυκλώματος.

Label	ALU control	SRC1	SRC2	Register control	Memory	PCWrite control	Sequencing
Fetch	Add	PC	4		Read PC	ALU	Seq
	Add	PC	Extshft	Read			Dispatch 1
Mem1	Add	A	Extend				Dispatch 2
LW2					Read ALU		Seq
				Write MDR			Fetch
SW2					Write ALU		Fetch
Rformat1	Func code	A	B				Seq
				Write ALU			Fetch
BEQ1	Subt	A	B			C-cond	Fetch
JUMP1						Jump address	Fetch

Υλοποιώντας το μικροπρόγραμμα σαν ΜΕ

Η υλοποίηση μιας ΜΕ με μικροπρογραμματισμό συνίσταται στην αντιμετώπιση των εξής δύο προβλημάτων: την υλοποίηση του μηχανισμού εύρεσης της επόμενης μικροεντολής, και την αποθήκευση του μικροπρογράμματος. Ένα μικροπρόγραμμα μπορεί να θεωρηθεί σαν την αναπαράσταση σε μορφή κειμένου μιας μηχανής καταστάσεων, κι επομένως μπορεί να υλοποιηθεί ακριβώς όπως αυτή, με έναν καταχωρητή, όπου θα αποθηκεύεται η διεύθυνση της παρούσας, και ένα συνδυαστικό κύκλωμα, που θα υπολογίζει τη διεύθυνση της επόμενης μικροεντολής και θα παράγει τις τιμές των πεδίων ελέγχου. Συχνά όμως, ιδιαίτερα για μεγάλα μικροπρογράμματα, τα δύο παραπάνω προβλήματα αντιμετωπίζονται ξεχωριστά.

Η εναλλακτική υλοποίηση λοιπόν αποθηκεύει το μικροπρόγραμμα σε μια μνήμη μόνο για ανάγνωση (ROM), και υλοποιεί ξεχωριστά το μηχανισμό εύρεσης της διεύθυνσης της επόμενης μικροεντολής. Το διάγραμμα της επόμενης σελίδας δείχνει μια τέτοια υλοποίηση. Ένας αύξων μετρητής χρησιμοποιείται για την αποθήκευση της διεύθυνσης μικροεντολής και την



προεπιλεγμένη μετάβαση στην εντολή που ακολουθεί. Ο αποθηκευμένος μικροκώδικας παρέχει τις τιμές των σημάτων ελέγχου, και πληροφορίες που οδηγούν το μηχανισμό εύρεσης της διεύθυνσης της επόμενης μικροεντολής. Ο τελευταίος περιέχει τους πίνακες διευθύνσεων, που μπορούν να είναι επίσης υλοποιημένοι σε μία ή περισσότερες μονάδες ROM, και οι οποίοι παρέχουν τη ζητούμενη διεύθυνση, εάν οι πληροφορίες του μικροκώδικα υποδεικνύουν τη χρήση αυτών. Το πλεονέκτημα μιας τέτοιας υλοποίησης του παραπάνω μηχανισμού είναι ότι δε χρειάζεται να υλοποιεί τη συνήθη μετάβαση στη μικροεντολή που ακολουθεί, κάτι που γίνεται με τη βοήθεια του αύξοντα μετρητή. Έτσι, ο μηχανισμός απαιτεί λιγότερο υλικό στη ΜΕ.

Με την αποθήκευση του μικροπρογράμματος σε μονάδα ROM και τη χρήση ενός μετρητή για τη διευθυνσιοδότηση αυτής, η εκτέλεση του μικροκώδικα γίνεται λίγο-πολύ όπως η εκτέλεση του κώδικα μηχανής ενός κανονικού προγράμματος, που είναι αποθηκευμένο στη μνήμη εντολών που διευθυνσιοδοτείται με τη βοήθεια του μετρητή προγράμματος PC. Αυτή η ομοιότητα μάλιστα αποτέλεσε και το λόγο της ονομασίας μικροπρογραμματισμός.

Η επιλογή της μεθόδου υλοποίησης της ΜΕ μιας ΜΕΔ – καλωδιωμένη λογική ή μικροπρογραμματισμός – γίνεται με βάση τόσο την πολυπλοκότητα των σημάτων ελέγχου, όσο και τη διαθέσιμη τεχνολογία για την υλοποίηση της ΜΕ. Πριν προχωρήσουμε σε συμπεράσματα, όμως, θα ρίξουμε μια σύντομη ματιά σε ένα από τα δυσκολότερα θέματα του ελέγχου: τις ειδικές περιπτώσεις.

5.6 Ειδικές περιπτώσεις

Ο έλεγχος αποτελεί τη μεγαλύτερη πρόκληση στη σχεδίαση ενός επεξεργαστή: είναι το δυσκολότερο κομμάτι του, τόσο ως προς την ορθότητα, όσο και ως προς την ταχύτητα. Ένα από τα πιο πολύπλοκα σημεία του ελέγχου είναι η υλοποίηση των *ειδικών περιπτώσεων* και των *διακοπών* – γεγονότων άλλων από διακλαδώσεις και άλματα που τροποποιούν την κανονική ροή στην εκτέλεση των εντολών. Οι ειδικές περιπτώσεις είναι απροσδόκητα γεγονότα μέσα στον επεξεργαστή: όπως, για παράδειγμα, οι αριθμητικές υπερχειλίσεις. Οι διακοπές είναι επίσης απροσδόκητα γεγονότα, με τη διαφορά ότι προέρχονται από έξω από τον επεξεργαστή. Οι διακοπές γενικά χρησιμοποιούνται από περιφερειακές συσκευές για την επικοινωνία τους με τον επεξεργαστή.

Σε πολλές αρχιτεκτονικές, όπως και σε πολλά συγγράμματα, δε γίνεται διάκριση μεταξύ διακοπών και ειδικών περιπτώσεων, και συχνά χρησιμοποιείται ο παλαιότερος όρος *διακοπή*

και για τους δύο αυτούς τύπους γεγονότων. Θα ακολουθήσουμε τη σύμβαση της MIPS, χρησιμοποιώντας τον όρο *ειδική περίπτωση*, για να αναφερθούμε σε οποιοδήποτε απροσδόκητο γεγονός που τροποποιεί την κανονική ροή εντολών, ανεξάρτητα αν αυτό προέρχεται από μέσα ή από έξω από τον επεξεργαστή. Ειδικότερα, θα χρησιμοποιούμε τον όρο *διακοπή* μόνο για γεγονότα που προέρχονται από έξω από τον επεξεργαστή. Οι αρχιτεκτονικές 80x86 της Intel χρησιμοποιούν τον όρο *διακοπή* για όλα τα παραπάνω γεγονότα, ενώ οι αρχιτεκτονικές PowerPC χρησιμοποιούν τον όρο *ειδική περίπτωση* για να δηλώσουν την εμφάνιση ενός απροσδόκητου γεγονότος, και τον όρο *διακοπή* για να δηλώσουν την αλλαγή στον έλεγχο ροής του κώδικα.

Οι διακοπές δημιουργήθηκαν αρχικά για το χειρισμό των αιτήσεων εξυπηρέτησης περιφερειακών συσκευών (Ε/Ε). Ο ίδιος βασικός μηχανισμός επεκτάθηκε για το χειρισμό και των γεγονότων που συμβαίνουν εσωτερικά στον επεξεργαστή. Ο παρακάτω πίνακας δίνει παραδείγματα γεγονότων που οδηγούν σε ειδική περίπτωση ή σε διακοπή, αναφέροντας και από πού προέρχεται το κάθε γεγονός:

Γεγονός	Απο πού προέρχεται	Ορολογία MIPS
Αίτηση εξυπηρέτησης Ε/Ε	Εξωτερικά	Διακοπή
Κλήση συστήματος	Εσωτερικά	Ειδική περίπτωση
Αριθμητική υπερχείλιση	Εσωτερικά	Ειδική περίπτωση
Μη ορισμένη εντολή	Εσωτερικά	Ειδική περίπτωση
Βλάβη υλικού	Εσωτερικά ή εξωτερικά	Ειδική περίπτωση ή διακοπή

Πολλές από τις ανάγκες στην υποστήριξη μιας ειδικής περίπτωσης δημιουργούνται από την ίδια τη φύση της ειδικής περίπτωσης. Γι' αυτό και θα αναφερθούμε ξανά σε ειδικές περιπτώσεις όταν μιλήσουμε αναλυτικά για την ιεραρχία μνήμης και για περιφερειακές συσκευές, ώστε να κατανοήσουμε καλύτερα τα κίνητρα για την υποστήριξη της κάθε ειδικής περίπτωσης. Προς το παρόν θα ασχοληθούμε με την υποστήριξη στη ΜΕ δύο ειδικών περιπτώσεων, που προέρχονται από εκείνο το μέρος του συνόλου εντολών και της υλοποίησης της ΜΕΔ που έχουμε ήδη αναλύσει.

Η ανίχνευση των ειδικών περιπτώσεων και η αντιμετώπισή τους γίνεται συχνά στο όριο του χρόνου του κύκλου μηχανής, κι επομένως μπορεί να επηρεάσει την απόδοση του επεξεργαστή. Αν δε δώσουμε την πρέπουσα σημασία στις ειδικές περιπτώσεις κατά τη διάρκεια σχεδίασης της ΜΕ, μια εκ των υστέρων προσθήκη αυτών σε μια ήδη πολύπλοκη ΜΕ μπορεί τόσο να μειώσει την απόδοση του επεξεργαστή, όσο και να δυσκολέψει την ορθή υλοποίησή τους.

Πώς γίνεται ο χειρισμός των ειδικών περιπτώσεων

Οι δύο τύποι ειδικών περιπτώσεων που εμφανίζονται στην υλοποίησή μας είναι η ειδική περίπτωση από μη ορισμένη εντολή, και η ειδική περίπτωση από αριθμητική υπερχείλιση. Η βασική ενέργεια του επεξεργαστή σε μια ειδική περίπτωση είναι η αποθήκευση της διεύθυνσης της εντολής που προκάλεσε την ειδική περίπτωση σε έναν καταχωρητή ειδικού σκοπού, τον μετρητή προγράμματος ειδικών περιπτώσεων (EPC), και η μεταφορά του ελέγχου ροής σε κατάλληλη διεύθυνση στον κώδικα του λειτουργικού συστήματος.

Το λειτουργικό σύστημα μπορεί να χειριστεί την ειδική περίπτωση με κατάλληλη ενέργεια, όπως κάποια λειτουργία εξυπηρέτησης του προγράμματος εφαρμογής, είτε κάποια συγκεκριμένα βήματα σε απάντηση στην ειδική περίπτωση, είτε τον τερματισμό του προγράμματος με κατάλληλο μήνυμα σφάλματος. Αν δεν τερματίσει το πρόγραμμα εφαρμογής, το λειτουργικό σύστημα θα συνεχίσει την εκτέλεσή του μετά το χειρισμό της ειδικής περίπτωσης, από διεύθυνση που καθορίζεται από το περιεχόμενο του EPC.

Για να μπορέσει το λειτουργικό σύστημα να χειριστεί μια ειδική περίπτωση, πρέπει εκτός από την εντολή που την προκάλεσε να γνωρίζει και τον τύπο την ειδικής περίπτωσης, το λόγο δηλαδή που αυτή εμφανίστηκε. Υπάρχουν δύο κυρίως μέθοδοι για τη γνωστοποίηση του τύ-

που μιας ειδικής περίπτωσης. Η αρχιτεκτονική MIPS χρησιμοποιεί έναν ειδικό καταχωρητή κατάστασης, τον Καταχωρητή Αιτίου, όπου αποθηκεύεται ο τύπος της ειδικής περίπτωσης που εμφανίστηκε.

Μια δεύτερη μέθοδος γνωστοποίησης του τύπου της ειδικής περίπτωσης είναι μέσω *διανυσματικών διακοπών*. Σε μια διανυσματική διακοπή, η διεύθυνση στην οποία μεταφέρεται ο έλεγχος ροής προκύπτει κατ' ευθείαν από το αίτιο της ειδικής περίπτωσης, μέσα δηλαδή από το υλικό, σαν ένα στοιχείο ενός διανύσματος διευθύνσεων. Για παράδειγμα, για την υποστήριξη των δύο παραπάνω τύπων ειδικών περιπτώσεων, μπορούμε να ορίσουμε το ακόλουθο διάνυσμα:

Τύπος ειδικής περίπτωσης	**Διεύθυνση εξυπηρέτησης**
Μη ορισμένη εντολή	$C0000000_{16}$
Αριθμητική υπερχείλιση	$C0000020_{16}$

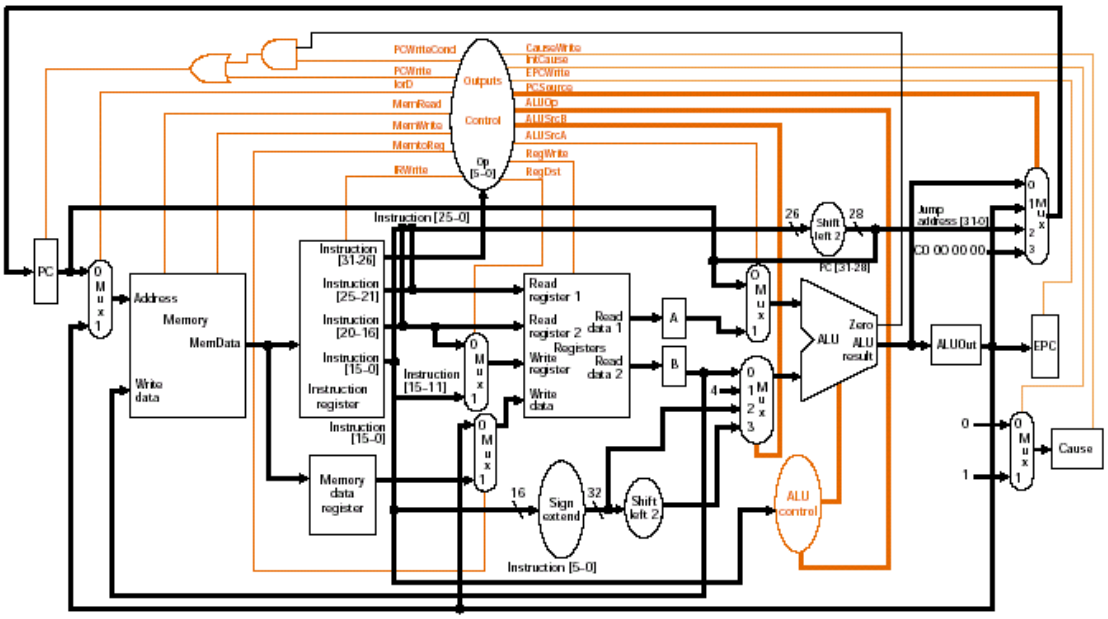
Το λειτουργικό σύστημα γνωρίζει τον τύπο της ειδικής περίπτωσης από τη διεύθυνση στην οποία μεταβαίνει ο έλεγχος, όταν αυτή εμφανίζεται. Οι παραπάνω διευθύνσεις απέχουν μεταξύ τους κατά 32 bytes ή 8 εντολές, μέσα στις οποίες το λειτουργικό σύστημα μπορεί να κάνει την αναγνώριση του τύπου και κάποια στοιχειώδη ενέργεια. Αν οι ειδικές περιπτώσεις δεν είναι διανυσματικές, η διεύθυνση στην οποία μεταβαίνει ο έλεγχος σε μια ειδική περίπτωση είναι μοναδική, οπότε η αναγνώριση του τύπου αυτής πρέπει να γίνει μέσω του καταχωρητή αιτίου.

Μπορούμε να εκτελέσουμε τις λειτουργίες που απαιτούνται για τις δύο ειδικές περιπτώσεις, προσθέτοντας στη βασική ΜΕΔ που έχουμε υλοποιήσει κάποιους καταχωρητές ειδικού σκοπού και σήματα ελέγχου, με κάποια αντίστοιχη μικρή τροποποίηση της μηχανής καταστάσεων. Ας υποθέσουμε ότι η μέθοδος γνωστοποίησης του τύπου της ειδικής περίπτωσης που υλοποιούμε είναι αυτή που χρησιμοποιεί η αρχιτεκτονική MIPS (αν και η μέθοδος των διανυσματικών διακοπών δεν είναι πιο δύσκολη). Θα προσθέσουμε έτσι στη ΜΕΔ τους εξής δύο καταχωρητές:

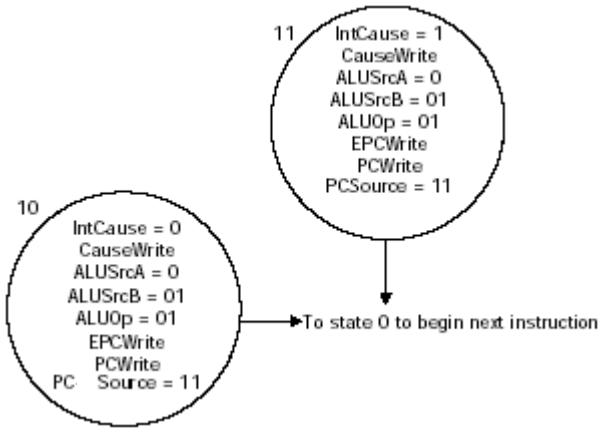
- *EPC*: Ένας καταχωρητής μεγέθους 32 bits, ο οποίος θα αποθηκεύει τη διεύθυνση της εντολή που προκαλεί την ειδική περίπτωση. (Ενας τέτοιος καταχωρητής θα ήταν απαραίτητος ακόμα και αν υλοποιούσαμε διανυσματικές ειδικές περιπτώσεις.)
- *Καταχωρητής Αιτίου*: Ένας καταχωρητής, ο οποίος αποθηκεύει τον τύπο της ειδικής περίπτωσης. Στην αρχιτεκτονική MIPS ο καταχωρητής αυτός έχει μέγεθος 32 bits, παρ' ό,τι κάποια ψηφία δε χρησιμοποιούνται. Θα υποθέσουμε στη δική μας ΜΕΔ ότι το λιγότερο σημαντικό ψηφίο του καταχωρητή αιτίου κωδικοποιεί τον τύπο της ειδικής περίπτωσης: μη ορισμένη εντολή = 0, αριθμητική υπερχείλιση = 1.

Θα χρειαστούμε δύο σήματα ελέγχου για τον έλεγχο εγγραφής των δύο παραπάνω καταχωρητών, έστω *EPCWrite* και *CauseWrite*. Επιπλέον, θα χρειαστούμε ένα σήμα μεγέθους 1 bit, για να θέτουμε κατάλληλη τιμή στο λιγότερο σημαντικό ψηφίο του καταχωρητή αιτίου. Τέλος, πρέπει να μπορούμε να εισάγουμε στον PC κάποια ειδική διεύθυνση εισόδου στο λειτουργικό σύστημα, που να αντιστοιχεί στην αρχή του κώδικα χειρισμού ειδικών περιπτώσεων. Ας υποθέσουμε ότι η διεύθυνση αυτή είναι η $C0000000_{16}$. Μέχρι τώρα, ο PC λαμβάνει τιμή από την έξοδο ενός πολυπλέκτη τριών εισόδων που ελέγχεται από το σήμα PCSource. Μπορούμε εύκολα να προσθέσουμε τη σταθερή τιμή $C0000000_{16}$ σαν τέταρτη είσοδο στον πολυπλέκτη. Η τιμή αυτή θα επιλέγεται ως είσοδος του PC με τιμή 11_2 στο σήμα PCSource.

Επειδή ο PC αυξάνεται στον πρώτο κύκλο μηχανής κάθε εντολής, δε μπορούμε να περνάμε στον EPC απ' ευθείας την τιμή του PC, επειδή η τιμή που θα γραφεί θα είναι η αυξημένη τιμή. Μπορούμε όμως να χρησιμοποιήσουμε την ΑΛΜ, για να αφαιρέσουμε τη σταθερά 4 από τον PC και να αποθηκεύσουμε το αποτέλεσμα στον EPC. Κάτι τέτοιο δεν απαιτεί πρόσθετα σήματα ελέγχου ή διαδρομές στη ΜΕΔ, εφ' όσον η ΑΛΜ μπορεί ήδη να εκτελέσει αφαίρεση, ενώ η σταθερά 4 αποτελεί ήδη επιλογή εισόδου της ΑΛΜ. Η είσοδος δεδομένων του EPC επομένως, θα συνδεθεί στην έξοδο της ΑΛΜ. Η διορθωμένη ΜΕΔ πολλαπλών κύκλων μηχανής ανά κύκλο εντολής που υλοποιεί ειδικές περιπτώσεις δίνεται στο διάγραμμα της επόμενης σελίδας.



Χρησιμοποιώντας την παραπάνω ΜΕΔ, οι ενέργειες που λαμβάνονται για καθένα τύπο ειδικής περίπτωσης καθορίζονται μέσα από μία νέα κατάσταση. Για κάθε ειδική περίπτωση, η αντίστοιχη κατάσταση πρέπει να θέτει τιμή στον καταχωρητή αιτίου, να υπολογίζει και να αποθηκεύει τη διεύθυνση της εντολής που προκάλεσε την ειδική περίπτωση στον EPC, και τέλος να θέτει στον PC τη διεύθυνση στην οποία αρχίζει ο κώδικας χειρισμού ειδικών περιπτώσεων. Έτσι, για το χειρισμό των δύο τύπων ειδικών περιπτώσεων που εξετάζουμε, προσθέτουμε τις δύο καταστάσεις που δίνονται στο ακόλουθο διάγραμμα.



Για να συνδέσουμε τη νέα μηχανή καταστάσεων με αυτήν της κυρίως ΜΕ, πρέπει να καθορίσουμε πώς ανιχνεύουμε τις ειδικές περιπτώσεις, και να προσθέσουμε τόξα μετάβασης, που μεταφέρουν τον έλεγχο από την κυρίως ΜΕ στην παραπάνω μηχανή καταστάσεων χειρισμού ειδικών περιπτώσεων.

Πώς η ΜΕ ελέγχει για ειδικές περιπτώσεις

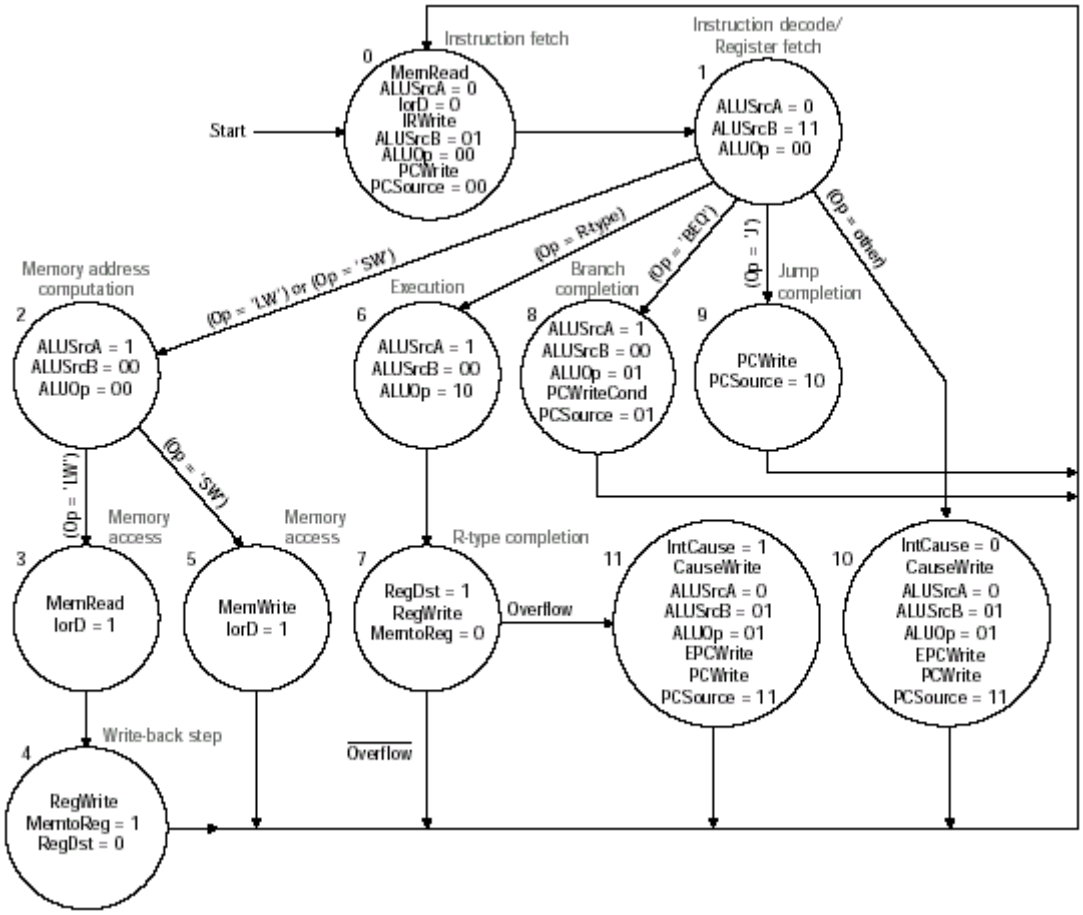
Τώρα απομένει να σχεδιάσουμε μια μέθοδο ανίχνευσης των ειδικών περιπτώσεων, ώστε ο έλεγχος να μεταφέρεται σε μία από τις καταστάσεις της παραπάνω μηχανής καταστάσεων. Οι ειδικές περιπτώσεις καθενός από τους δύο τύπους ειδικών περιπτώσεων ανιχνεύονται διαφορετικά:

- *Μη ορισμένη εντολή*: Αυτός ο τύπος ειδικής περίπτωσης ανιχνεύεται, όταν δεν έχει οριστεί επόμενη κατάσταση από την κατάσταση 1 της κυρίως ΜΕ, για τη συγκεκριμένη τιμή του κωδικού λειτουργίας που εμφανίζεται. Ορίζουμε έτσι ως επόμενη κατάσταση

της 1 για κάθε τιμή κωδικού λειτουργίας διαφορετική από 35 (εντολές lw), 43 (εντολές sw), 0 (R-εντολές), 4 (εντολές beq) και 2 (εντολές j), την παραπάνω κατάσταση 10. Συνδέουμε έτσι τις δύο καταστάσεις 1 και 10 με ένα κατευθυνόμενο τόξο με ετικέτα τον κωδικό λειτουργίας που αναγράφεται συμβολικά ως “other”.

- *Αριθμητική υπερχείλιση*: Η αριθμητική υπερχείλιση εμφανίζεται στην ΑΛΜ. Η ΜΕ ανιχνεύει αυτόν τον τύπο ειδικής περίπτωσης με τη βοήθεια του σήματος αριθμητικής υπερχείλισης που παρέχεται ως έξοδος της ΑΛΜ. Η μηχανή καταστάσεων χρησιμοποιεί έτσι αυτό το σήμα, για να καθορίσει ως νέα πιθανή επόμενη κατάσταση της 7, την παραπάνω κατάσταση 11.

Το διάγραμμα της νέας συνολικής μηχανής καταστάσεων δίνεται παρακάτω, και περιλαμβάνει τόσο την υλοποίηση του συνόλου εντολών MIPS που εξετάζουμε, όσο και τις δύο ειδικές περιπτώσεις που μελετήσαμε παραπάνω. Υπενθυμίζεται ότι η πρόκληση στη σχεδίαση της ΜΕ ενός πραγματικού επεξεργαστή είναι ο χειρισμός μιας μεγάλης ποικιλίας αλληλεπιδράσεων μεταξύ εντολών καθώς και ειδικών περιπτώσεων, έτσι ώστε η ΜΕ να παραμένει ταυτόχρονα μικρή και γρήγορη. Οι πιο πολύπλοκες αλληλεπιδράσεις που μπορούν να εμφανιστούν στην πραγματικότητα είναι αυτές που κάνουν τη ΜΕ το δυσκολότερο κομμάτι της σχεδίασης υλικού.



5.7 Συμπεράσματα

Όπως είδαμε, τόσο η ΜΕΔ όσο και η ΜΕ ενός επεξεργαστή μπορούν να σχεδιαστούν, ξεκινώντας από την αρχιτεκτονική του συνόλου εντολών και την κατανόηση των βασικών χαρακτηριστικών της τεχνολογίας υλοποίησής τους. Όσο αφορά τη ΜΕΔ, ξεκινήσαμε τη σχεδίαση μιας απλοποιημένης ΜΕΔ MIPS με την απόφαση να την υλοποιήσουμε με απλό κύκλο μηχανής ανά κύκλο εντολής, χρησιμοποιώντας κατάλληλες υπομονάδες της αρχιτεκτονικής. Βέ-

βαια, στη σχεδίαση μιας ΜΕΔ, η διαθέσιμη τεχνολογία επίσης επηρεάζει πολλές αποφάσεις μας, υποδεικνύοντας τι υπομονάδες μπορούμε να έχουμε, ακόμα και εάν η υλοποίηση του απλού κύκλου μηχανής έχει νόημα. Με τον ίδιο τρόπο, είδαμε μετά πώς η απόφασή μας να χρησιμοποιήσουμε συντομότερο κύκλο μηχανής και να διαχωρίσουμε τον κύκλο εντολής σε βήματα, οδήγησε στη ΜΕΔ των πολλαπλών κύκλων μηχανής. Και στις δύο υλοποιήσεις, η οργάνωση ανώτερου επιπέδου – απλού κύκλου ή πολλαπλών κύκλων μηχανής – καθώς και η αρχιτεκτονική του συνόλου εντολών, καθορίζουν τα χαρακτηριστικά της ΜΕΔ.

Παρόμοια, η ΜΕ καθορίζεται σε μεγάλο βαθμό από την αρχιτεκτονική του συνόλου εντολών, την οργάνωση ανώτερου επιπέδου, και από το σχέδιο της ΜΕΔ. Στην πρώτη υλοποίηση της ΜΕΔ, αυτά τα τρία στοιχεία καθόρισαν ουσιαστικά τη μορφή και τις τιμές των σημάτων ελέγχου. Στη δεύτερη υλοποίηση, η ακριβής αντιστοίχηση των λειτουργιών στους διαφορετικούς κύκλους μηχανής, κάτι που επίσης καθορίζεται από την αρχιτεκτονική του συνόλου εντολών και το σχέδιο της ΜΕΔ, είναι αυτή που ορίζει τον έλεγχο της ΜΕΔ.

Ο έλεγχος είναι μια από τις μεγαλύτερες προκλήσεις της αρχιτεκτονικής υπολογιστών. Ο βασικός λόγος γι' αυτό, είναι ότι η σχεδίαση της ΜΕ απαιτεί ακριβή γνώση όλων των τμημάτων του επεξεργαστή, και πώς αυτά αλληλεπιδρούν. Για τη σχεδίαση της ΜΕ μελετήσαμε δύο μεθόδους αναπαράστασης του ελέγχου: μηχανές πεπερασμένων καταστάσεων και μικροπρογραμματισμός. Οι δύο αυτές μέθοδοι μας επέτρεψαν να διαχωρίσουμε την περιγραφή του ελέγχου από τις λεπτομέρειες υλοποίησης της ΜΕ. Η αφαιρετικότητα στην ανάλυση είναι ο κύριος τρόπος αντιμετώπισης της αυξανόμενης πολυπλοκότητας των υπολογιστικών συστημάτων.

Έχοντας σχεδιάσει μια αναπαράσταση του ελέγχου, μπορούμε εύκολα να προχωρήσουμε στην απεικόνιση της ΜΕ σε υλικό. Η ακριβής υλοποίηση εξαρτάται τόσο από την πολυπλοκότητα του ελέγχου, όσο και από την τεχνολογία που χρησιμοποιούμε για το σκοπό αυτό. Ο διαχωρισμός της περιγραφής του ελέγχου από την υλοποίηση της ΜΕ έχει το επιπλέον πλεονέκτημα ότι η πρώτη δεν εξαρτάται από τις μεταβολές στην τεχνολογία, όπως εξαρτάται η δεύτερη.