ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Ο Ρ Γ Α Ν Ω Σ Η Υ Π Ο Λ Ο Γ Ι Σ Τ Ω Ν

(Η Ρ Υ 302)

Ε Ρ Γ Α Σ Ι Α #2

ΚΑΤΣΙΜΠΑΣ ΠΕΤΡΟΣ

2016030038

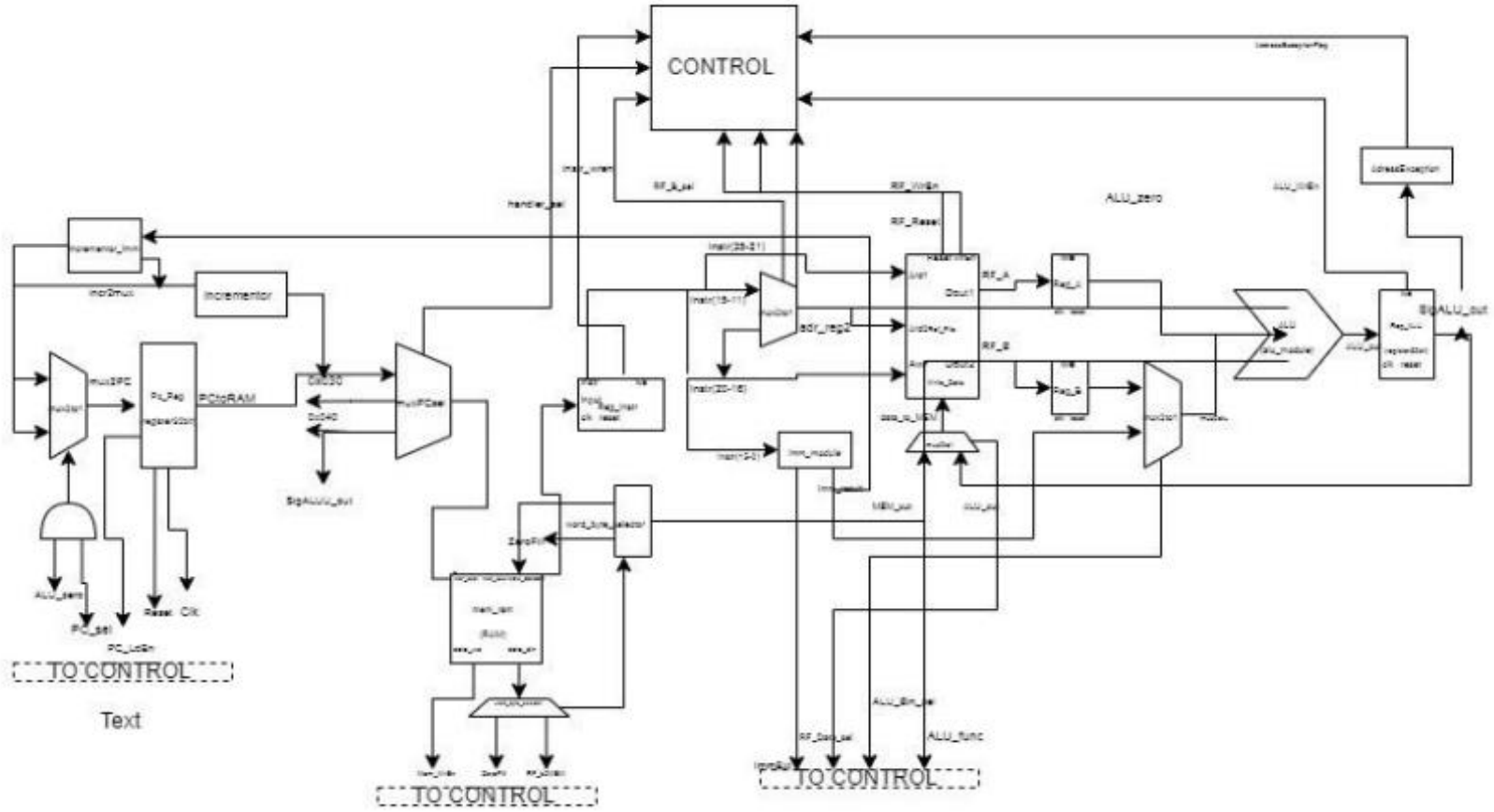
Σκοπός της Εργαστηριακής Άσκησης

Ο σκοπός της 2ης εργασίας χωρίστηκε σε δύο φάσεις, οι οποίες αφορούσαν την μετατροπή του επεξεργαστή μονού κύκλου (PROC_SC) σε επεξεργαστή πολλαπλών κύκλων (PROC_MC) καθώς και τη τελική του δημιουργία του σε pipeline (PROCESSOR_PIPELINE).

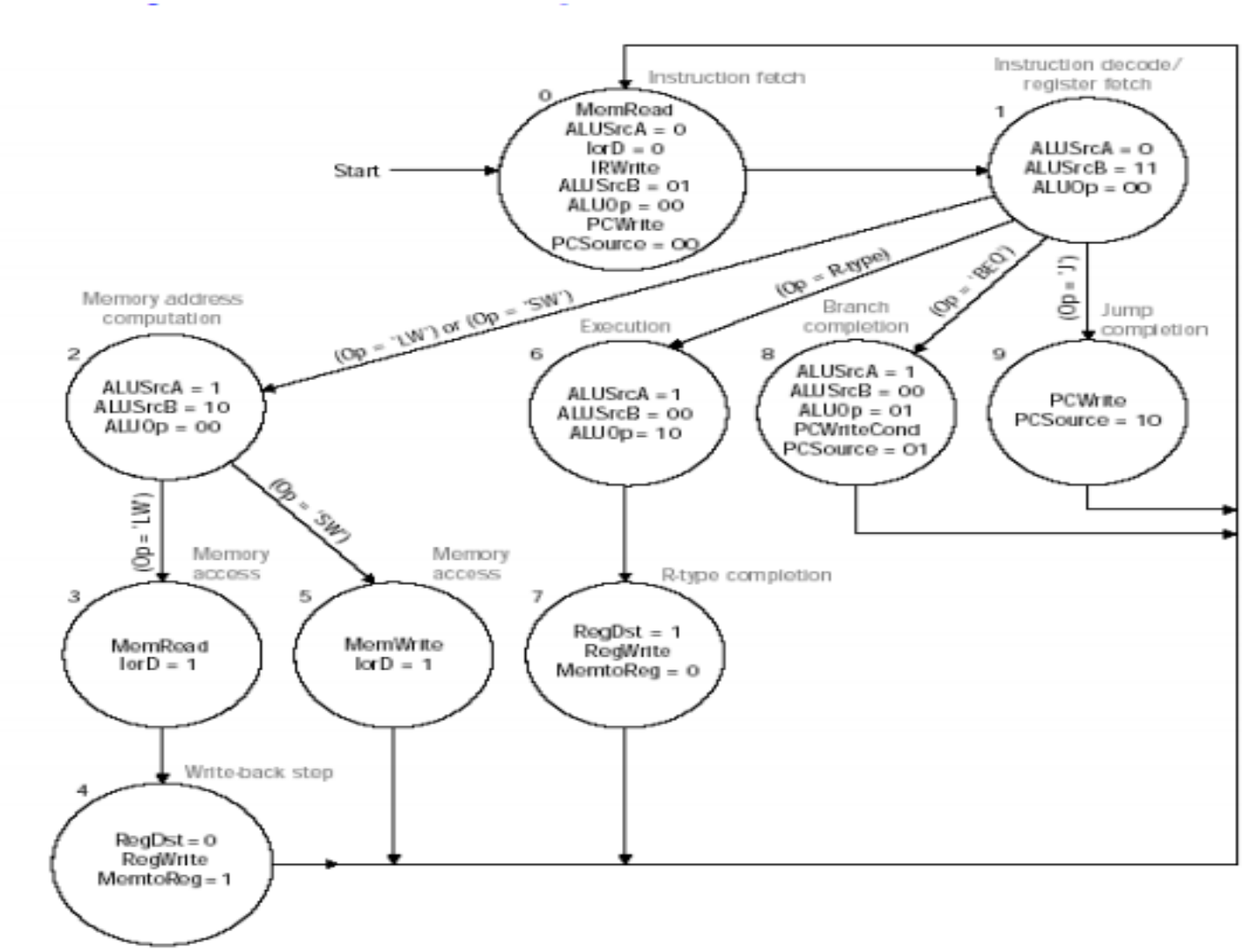
4η Φάση εργασίας(MultiCycle_Processor)

Για την επίτευξη του επεξεργαστή πολλαπλών κύκλων χρειάστηκε να επέμβουμε κατάλληλα στο DATAPATH του επεξεργαστή της αρχικής φάσης αλλά και να υλοποιήσουμε και να ενσωματώσουμε σε αυτόν μία μηχανή πεπερασμένων καταστάσεων (FSM) για τον έλεγχο (control) του DATAPATH και των σημάτων ελέγχου του. Συγκεκριμένα, χρειάστηκε να προσθέσουμε 5 καταχωρητές ανάμεσα στις δομικές βαθμίδες της σχεδίασής μας ώστε να υποστηρίζετε με επιτυχία ένα multicycle DATAPATH. Οι καταχωρητές αυτοί είναι οι Reg_Instr μετά τη βαθμίδα Instruction Fetch, Reg_A και Reg_B μετά τη βαθμίδα Instruction Decode ο Reg_ALU μετά τη βαθμίδα Instruction EX_DEC, προκειμένου να κρατούνται οι απαραίτητες πληροφορίες για τουλάχιστον 1 κύκλο ώστε όταν η επόμενη βαθμίδα τις χρειαστεί να έχει πρόσβαση σε αυτές και τέλος ο Reg_Mem, από όπου περνάει η έξοδος της RAM ώστε να πάει στον πολυπλέκτη πριν τον Register_File.

Ακολουθεί η σχηματική απεικόνιση του διαμορφωμένου Datapath σύμφωνα με τα παραπάνω:

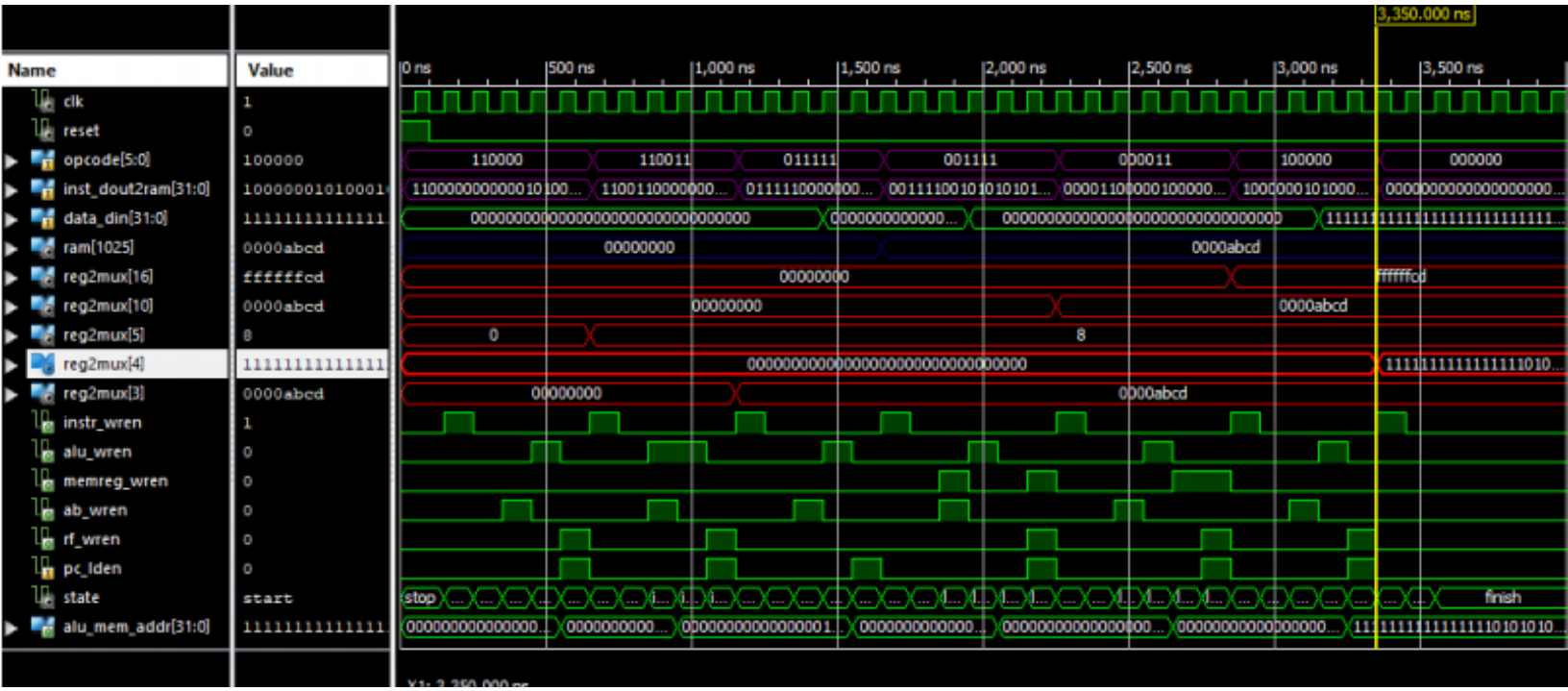


Είναι εξίσου αναγκαίο να αποτυπώσουμε το ολοκληρωμένο, για το Multi-Cycle Processor, διάγραμμα καταστάσεων:

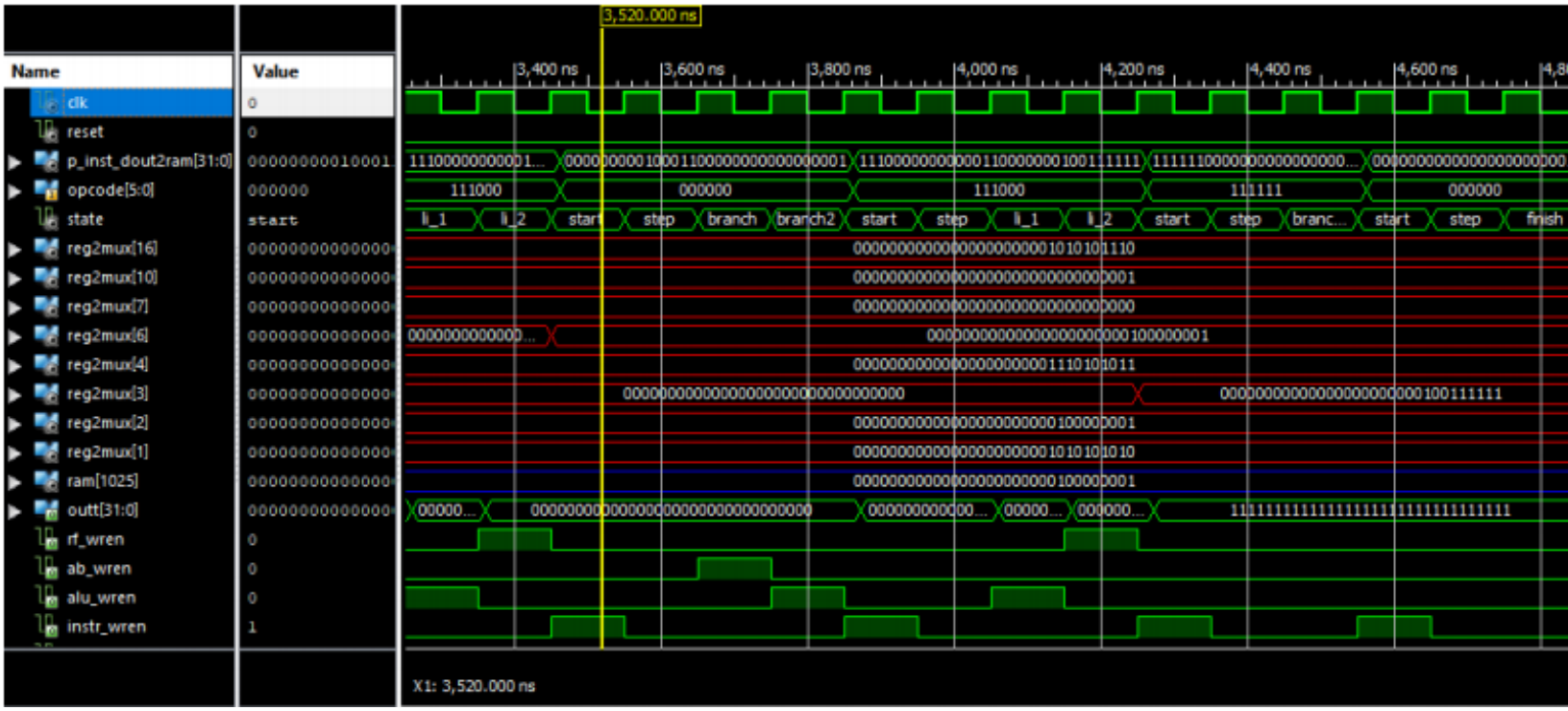


Waveforms

Ακολουθεί κυματομορφή με το πρόγραμμα αναφοράς που μας δόθηκε.(report_program1.data)



Ακολουθεί κυματομορφή με το πρόγραμμα που δημιουργήσαμε(με όνομα MC_custom.data στον φάκελο καθώς και το .txt αρχείο με τις εντολές Assembly).



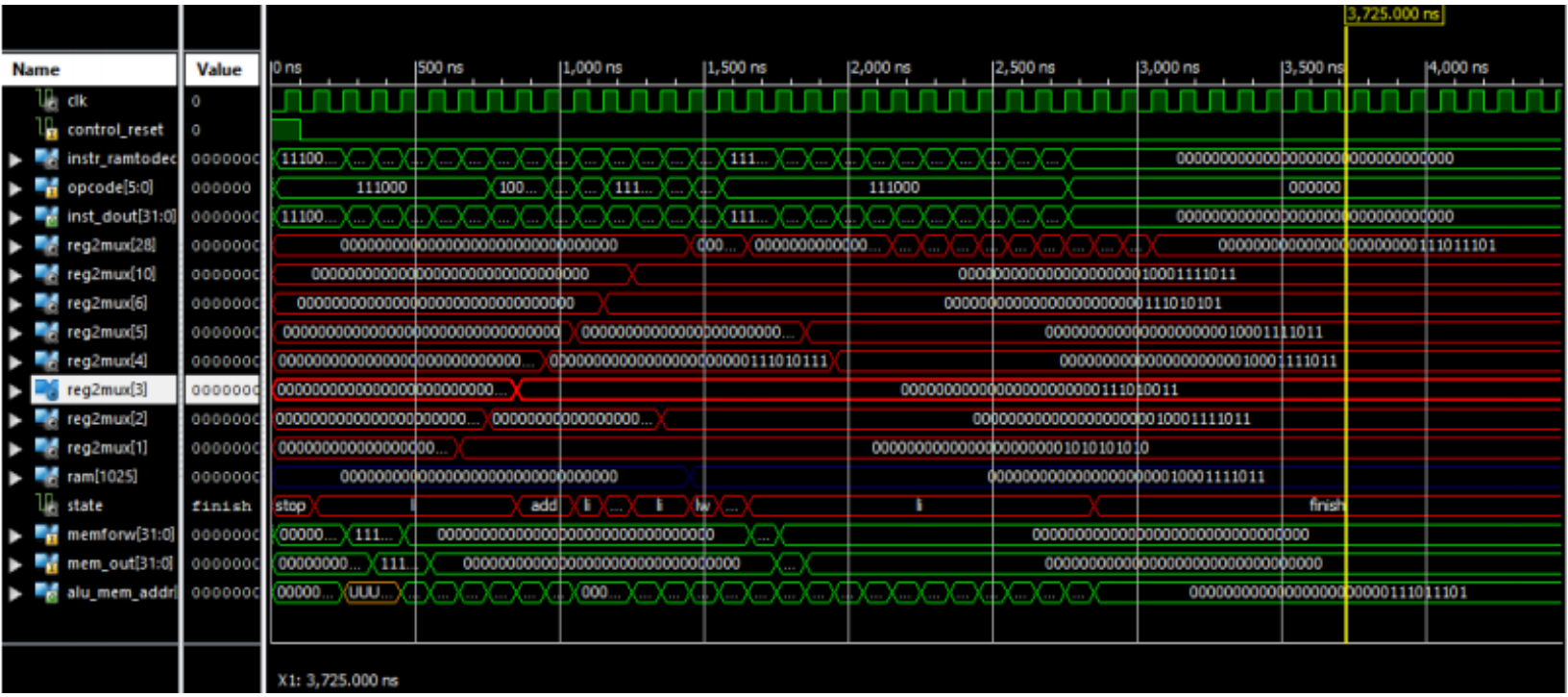
Συμπεράσματα

Παρατηρήσαμε ότι η δημιουργία MultiCycle Processor εκτελούσε τις εντολές του δοθέντος προγράμματος με μεγάλη καθυστέρηση από ότι ο επεξεργαστής του μονού κύκλου αφού κάθε βήμα από component σε component διαρκούσε ένα κύκλο ρολογιού. Επίσης, για την δημιουργία χρειάστηκε μόνο μια ALU η οποία ικανοποιούσε και τις πράξεις των Registers αλλά και για τον υπολογισμό διευθύνσεων σε τύπου branch εντολές. Όλα αυτά έγιναν όμως για να προχωρήσουμε στη δημιουργία της τελικής και βελτιωμένης έκδοσης του επεξεργαστή pipeline όπως θα δούμε στη συνέχεια.

5η Φάση εργασίας(Pipeline_Processor)

Προκειμένου να υποστηρίζει pipeline ο επεξεργαστής μας έπρεπε να γίνουν διάφορες δομικές αλλαγές στη σχεδίαση του. Συγκεκριμένα, προστέθηκαν κατάλληλοι pipeline registers ενδιάμεσα στα διάφορα στάδια του control path (Inst. Fetch Stage, Inst. Decode Stage, Execute Stage, Memory Stage) ώστε οι εντολές που υλοποιήθηκαν (add, li, sw, lw) να έχουν συγκεκριμένο αριθμό κύκλων ώστε να μπορεί να επιτευχθεί συγχρονισμός κατά το pipelining αλλά και τα δεδομένα/έξοδοι κάθε stage να είναι διαθέσιμα κατά βούληση στον κατάλληλο κύκλο. Όσο αφορά τους κύκλους κάθε εντολής, οι add, li και sw σχεδιάστηκαν ώστε να πραγματοποιούνται σε 5 κύκλους, ενώ η lw σε 4 κύκλους και με κατάλληλο forwarding και stalling να συγχρονίζεται με τις υπόλοιπες τρείς εντολές. Για την πραγματοποίηση του forwarding και των stalls υλοποιήθηκαν δύο δομικές μονάδες (modules) που εμπεριέχουν την κατάλληλη λογική, όπως συγκρίσεις ισότητας της εισόδου και εξόδου των διαφόρων σταδίων του επεξεργαστή αλλά και έλεγχος διαφόρων σημάτων, ώστε να πραγματοποιούνται όπου είναι αναγκαία για την την αντιμετώπιση των data hazards. Οποιαδήποτε control hazards, που αφορούν κυρίως branches, παραλήφθηκαν σύμφωνα με την εκφώνηση, καθώς δεν υλοποιήθηκαν branch/jump type εντολές. Η μονάδα (module) του control άλλαξε ριζικά από το 4ο εργαστήριο και ως προς τα states από τα οποία αποτελείται αλλά και από τα σήματα εισόδου/εξόδου της, καθώς έπρεπε να συντονίζει τους νέους pipeline registers συν τοις άλλοις. Τα states τα οποία την αποτελούσαν ήταν τέσσερα και αντιπροσώπευαν κάθε εντολή που υλοποιήθηκε, δηλαδή add (r-type), li, lw, sw, και ένα reset state για την αρχικοποίησή της. Το πρόγραμμα που θα έτρεχε στον επεξεργαστή, όπως στα προηγούμενα εργαστήρια φορτώνεται στη RAM και τρέχει αυτόματα με τη βοήθεια του control module/FSM.

Waveforms



Οι πρώτες έξι εντολές αξιοποιούν την εντολή li για να φορτωθούν ενδεικτικές τιμές στους καταχωρητές 1-6. Είναι προφανές πως δεν προκαλείται κάποιου είδους hazard καθώς η εντολή li δεν χρησιμοποιεί σαν τελεσταίους, καταχωρητές. Τις δύο αμέσως επόμενες εντολές απαρτίζουν ένα ζευγάρι add εκ των οποίων η πρώτη εντολή γίνεται χωρίς να παρουσιαστούν κίνδυνοι δεδομένων (χρησιμοποιώντας σαν τελεσταίους καταχωρητές που έχουν έγκυρες τιμές, καταχωρητές 1 και 2). Το αποτέλεσμα της πρώτης πρόσθεσης αποθηκεύεται στον καταχωρητή 10. Η αμέσως επόμενη εντολή add συντάχθηκε έτσι ώστε να προκληθεί hazard. Πιο συγκεκριμένα δόθηκε σαν ένας απ'τους τελεσταίους να είναι ο καταχωρητής 10. Υπό φυσιολογικές συνθήκες το instruction decode αυτής της εντολής θα έπαιρνε εσφαλμένη τιμή για τον καταχωρητή 10 καθώς η προηγούμενη εντολή τον έχει αλλάξει σε αυτό το στάδιο αλλά δεν έχει προλάβει να γίνει η καταχώρηση του. Για να αντιμετοπιστεί αυτό το φαινόμενο εισήχθη στο κύκλωμα μια νέα μονάδα με όνομα forwarding unit. Δουλειά της, να συγκρίνει τους τελεσταίους του επιπέδου instruction fetch προς instruction decode με τον καταχωρητή rd των επιπέδων memory stage αλλά και write back stage και ύστερα να προωθεί κατάλληλα την τιμή του αντίστοιχου rd σε μία από τις εισόδους της ALU προς εγγραφή (δεδομένου πως τα rd των δύο αυτών επιπέδων συνοδεύονται από θετικό σήμα εγγραφής στην register file. Αμέσως μετά τις δύο προσθέσεις προστέθηκε μια εντολή li στον καταχωρητή 28 που χρησιμοποιείται και αργότερα για να συνεχιστεί η λειτουργία του προγράμματος. Μετά από αυτήν βρίσκεται μια εντολή sw που επιθυμεί να γράψει την τιμή του καταχωρητή 10 στην πρώτη θέση μνήμης. Υπό άλλες συνθήκες η εγγραφή αυτή θα έφερνε ως αποτέλεσμα μια μηδενική τιμή στη θέση μνήμης καθώς μέχρι και δύο κύκλους μετά του επιπέδου instruction fetch δεν είχε καταγραφεί η αλλαγή της τιμής του καταχωρητή 10. Με κατάλληλο εμπλουτισμό της μονάδας forwarding αντιμετωπίζεται και αυτό το θέμα data hazard. Πιο συγκεκριμένα η μονάδα αυτή παίρνει σαν είσοδο το καθυστερημένο op code και ελέγχοντας αν το rd του επιπέδου

instruction fetch είναι ίδιο με το rd του επιπέδου execute stage ή memory stage καθώς επίσης τα rd αυτά συνοδεύονται από θετικές τιμές εγγραφής στην register file, παράγει τα κατάλληλα σήματα ελέγχου στις εισόδους της ALU που καθορίζουν αν θα γίνει προώθηση ή όχι. Ύστερα αφού γίνουν άλλες δύο εντολές li στον καταχωρητή 28, γίνεται η εντολή lw. Εδώ αξίζει να σημειωθεί πως λόγω των προβλημάτων που μπορούν να προκύψουν από τη χρήση της lw το κύκλωμα ήρθε στην ανάγκη προσθήκης άλλης μιας μονάδας που θα διαχειρίζεται τα stalls που πρέπει να γίνουν για να συνεχιστεί σωστά η ροή του προγράμματος. Αυτό το κύκλωμα δέχεται σαν εισόδους τα σήματα που χρησιμοποιεί σαν συνθήκες ανίχνευσης stall καθώς και τα σήματα που θα απενεργοποιήσει μόλις εντοπιστεί. Έτσι ελέγχεται αν βρισκόμαστε σε εντολή lw καθυστερημένη κατά ένα κύκλο καθώς και αν οι τελεσταίοι του επιπέδου instruction fetch είναι ίσοι με τον rd του επιπέδου instruction decode που σημαίνει πως η επόμενη εντολή θα χρησιμοποιούσε την τιμή που θα διαβαζόταν από την μνήμη.

Συμπεράσματα

Από τις δύο τελευταίες φάσεις του εργαστηρίου συμπεράναμε πως για να υποστηρίζει ένας επεξεργαστής multicycle εντολές θα πρέπει να γίνουν ορισμένες δομικές αλλαγές στη single cycle σχεδίασή του. Αυτό οφείλεται στο ότι ένα multicycle DATAPATH δε μπορεί να πραγματοποιήσει στοιχειώδη αποθήκευση των πληροφοριών που θα χρειαστεί η επόμενη βαθμίδα εκτέλεσης της εντολής, στον επόμενο κύκλο εκτέλεσης της εντολής. Αντιλαμβανόμαστε, λοιπόν, πως η μετατροπή του επεξεργαστή σε multicycle DATAPATH με control που υποστηρίζει multicycle εντολές είναι αναγκαία και ικανή συνθήκη για να υποστηρίξει ο επεξεργαστής στο μέλλον παραλληλισμό μέσω pipelining ώστε να αξιοποιεί πλήρως τις δυνατότητες που έχει και παρέχει στο χρήστη. συμπεραίνουμε πως αν και το pipelining αποτελεί μέθοδο που επαναπροσδιορίζει τις ικανότητες του επεξεργαστή, δεν είναι τόσο εύκολη η υλοποίηση του και απαιτεί δομικές αλλαγές στον επεξεργαστή για να το υποστηρίζει. Πιο συγκεκριμένα, η προσθήκη πάρα πολλών καταχωρητών ώστε τα διάφορα σήματα του DATAPATH να καθυστερούν ή να αποθηκεύονται για κάποιο χρονικό διάστημα, δημιουργεί την ανάγκη για πολύ συγκεκριμένο χειρισμό όλων των σημάτων αναλόγως το stage στο οποίο «βρίσκονται» μέσα στον επεξεργαστή αλλά και από την εντολή η οποία εκτελείται και από την εντολή η οποία θα εκτελεστεί αμέσως μετά. Συμπεραίνουμε, λοιπόν, πως όλα αυτά απαιτούν ένα εξειδικευμένο και σύνθετο στη σύλληψη control το οποίο διαχειρίζεται όλα τα παραγόμενα σήματα σε κάθε ξεχωριστό stage της διέλευσης μιας εντολής. Ταυτόχρονα, πρέπει να ληφθούν υπόψη οι καθυστερήσεις που είναι αναγκαίες να προστεθούν σε κάθε stage χρησιμοποιώντας τους pipeline registers, ελέγχοντας τα σήματα ελέγχου τους, ώστε να λειτουργούν αρμονικά οι εντολές που εκτελεί σειριακά ο επεξεργαστής. Εδώ αντιλαμβανόμαστε τη σημασία των data hazards καθώς είναι εκείνα που μας προτρέπουν στην κατάλληλη διαχείριση των σημάτων ελέγχου του DATAPATH αλλά και των pipeline registers ώστε τα δεδομένα που απαιτεί μία εντολή σε κάθε stage να είναι διαθέσιμα και να είναι τα σωστά. Επομένως, συμπεραίνουμε ότι είναι αναγκαία η χρήση μεθόδων όπως το forwarding και το stalling των δεδομένων των pipeline registers στα κατάλληλα stages, όπου είναι αναγκαίο. Προκειμένου όμως να αξιολογηθούν οι περιπτώσεις κατά τις οποίες δημιουργούνται data hazards συμπεραίνουμε πως χρειάζονται δομικές μονάδες που να αντιμετωπίζουν αυτό το πρόβλημα, των οποίων η υλοποίηση είναι μία δύσκολη και σύνθετη διαδικασία.