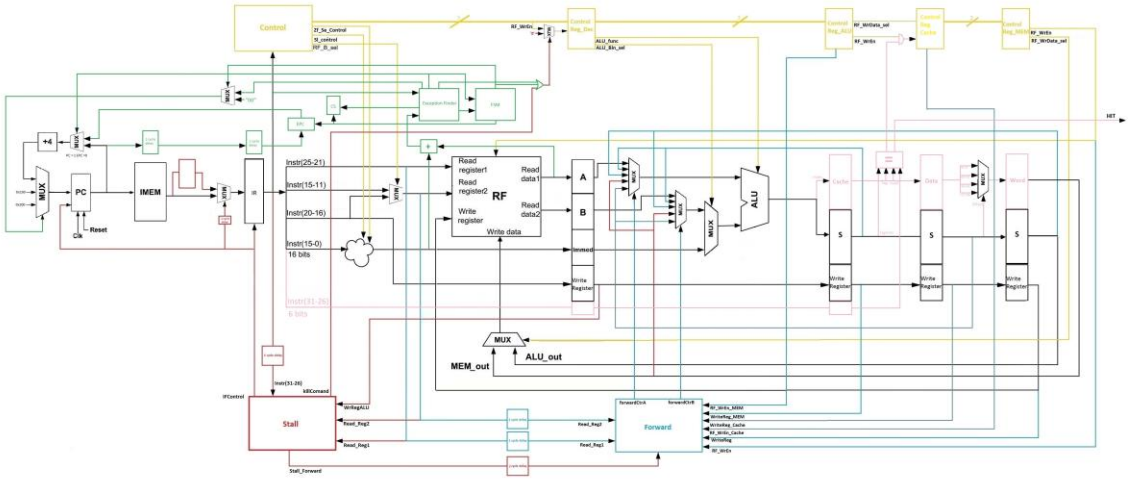
10/5/2017

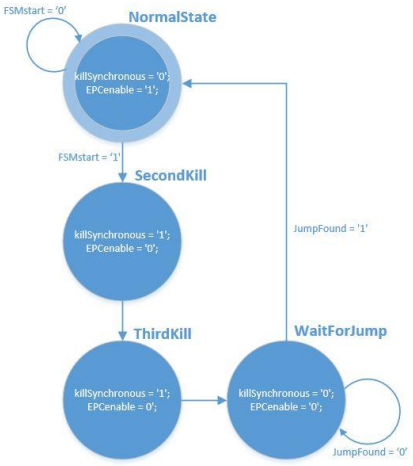
Αναφορά Εργαστηρίου 6

LAB31231454

Μανώλης Πετράκος
Μιχαήλ Δακανάλης

Προεργασία





Περιγραφή Άσκησης

Το πρώτο μέρος της άσκησης ήταν να δημιουργηθεί μία μονάδα που θα αναγνωρίζει και θα αντιμετωπίζει τα exceptions στέλνοντας το πρόγραμμα στον exception handler. Οι εξαιρέσεις που αντιμετωπίζονται είναι οι λανθασμένες εντολές και η πρόσβαση σε θέσεις μνήμης που δεν υπάρχουν.

Καταρχήν πρέπει στο decode να υπάρχει ο PC της εντολής που βρίσκεται στο συγκεκριμένο επίπεδο. Αποθηκεύουμε την τιμή στον καταχωρητή EPC ενώ την έχουμε καθυστερήσει κατάλληλα, ώστε να συμβαδίζει με την εντολή. Επίσης υπάρχει και ένα ασύγχρονο κύκλωμα που αναγνωρίζει τα exceptions κοιτώντας την εντολή και την διεύθυνση μνήμης που

παράγεται. Σε περίπτωση που εμφανιστεί ένα exception κάνει τέσσερις διαφορετικές λειτουργίες. Σκοτώνει την λανθασμένη εντολή, βάζει την κατάλληλη τιμή στον cause register, θέτει στον PC την διεύθυνση του κατάλληλου handler και ενεργοποιεί την FSM.

Στην κανονική λειτουργία του κυκλώματος η FSM έχει το enable των EPC και Cause Register ενεργό. Σε περίπτωση που εμφανιστεί ένα exception θα το απενεργοποιήσει ώστε να μείνουν οι τιμές της εντολής που προκάλεσε το πρόβλημα. Επίσης θα σκοτώσει τις επόμενες δύο εντολές που έχουν ήδη φορτωθεί στο pipeline. Τέλος θα μείνει σε μία κατάσταση όπου οι εντολές θα εκτελούνται κανονικά αλλά θα κρατάει τους EPC και Cause Register σταθερούς μέχρι να εμφανιστεί μία jump_epc.

Όταν έρθει μία jump_epc το ασύγχρονο κύκλωμα την αναγνωρίζει, θέτει στον PC την τιμή του EPC και γυρίζει την FSM στην αρχική της κατάσταση. Δημιουργήθηκε ένα πρόβλημα όπου το πρόγραμμα πήγαινε δύο φορές στον exception handler και μετά επέστρεφε στην κανονική λειτουργία. Για την αντιμετώπισή του, όταν η FSM βρίσκεται στην τελική κατάσταση, ο PC δεν παίρνει τις θέσεις των handler με την χρήση ενός πολυπλέκτη.

Το δεύτερο μέρος της άσκησης ζητάει να αντικατασταθεί η μνήμη δεδομένων με μία cache. Επίσης ο έλεγχος του hit/miss έπρεπε να γίνεται σε διαφορετικό κύκλο από την εύρεση των δεδομένων. Γι' αυτό προστέθηκε ένα παραπάνω επίπεδο στο Datapath και στο Control. Επίσης επεκτάθηκαν τα forward και τα stall ώστε να δουλεύουν και για τον επιπλέον κύκλο.

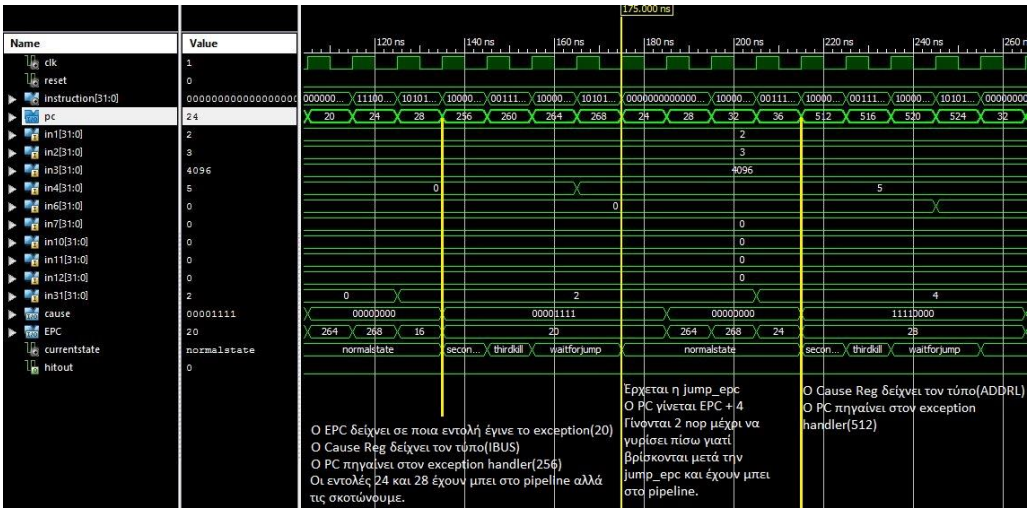
Η Cache δέχεται το index που έχει δημιουργηθεί από την εντολή και βγάζει το κατάλληλο block. Μετά υπάρχει ένα κύκλωμα το οποίο δέχεται το tag και το valid από την cache, το tag και το opcode της εντολής και παράγει το hit. Σε περίπτωση που η εντολή είναι load, αλλά το valid είναι 0 ή υπάρχει ασυμφωνία μεταξύ των tags υπάρχει miss και σκοτώνουμε την εντολή.

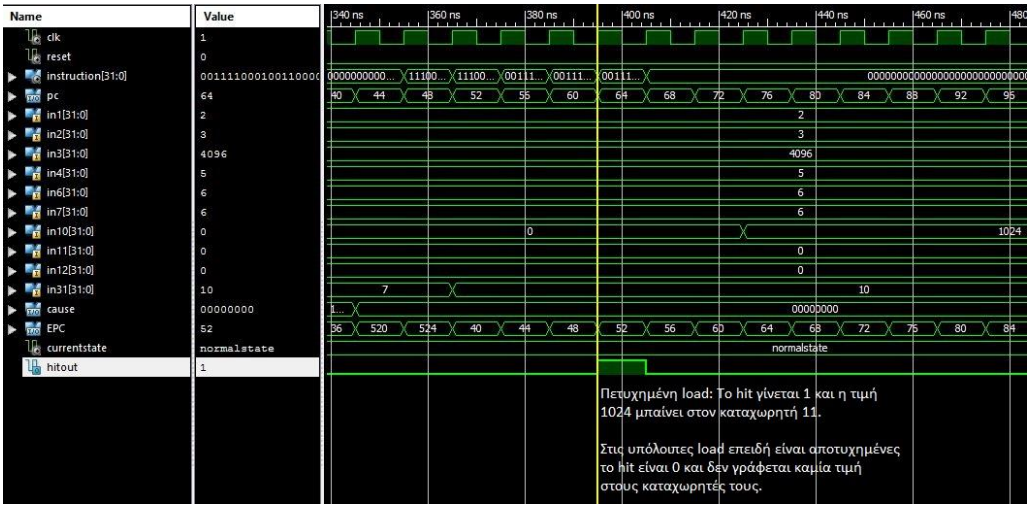
Στον επόμενο κύκλο από το Data της cache διαλέγεται η λέξη που υποδεικνύει το word offset και τέλος γράφεται στον Register File.

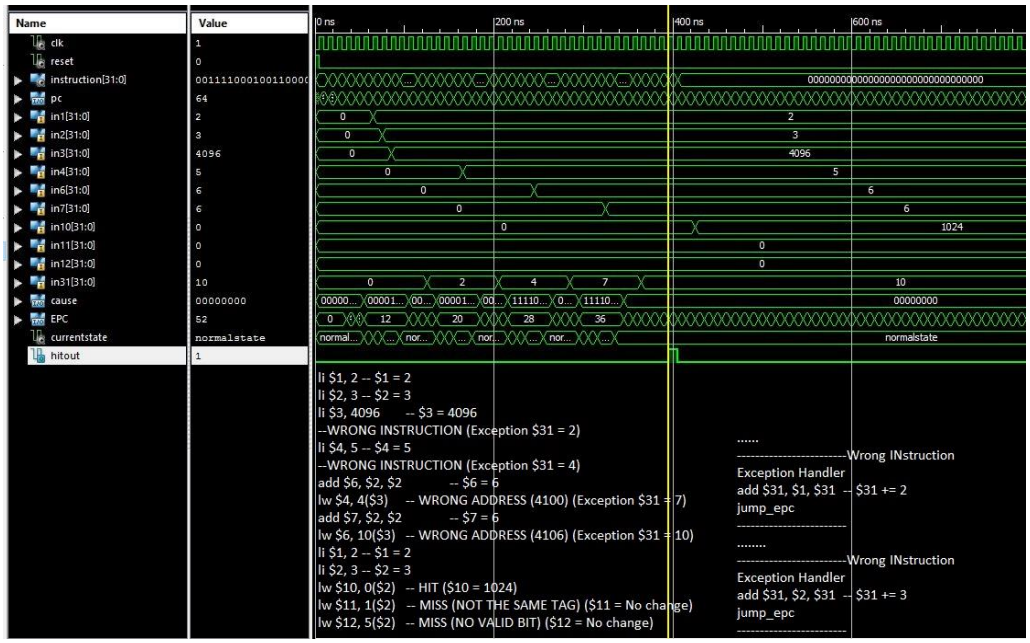
Η cache αποτελείται από τρία επιμέρους πεδία, το valid, το tag και το data. Το valid είναι 1 bit και δείχνει αν υπάρχουν έγκυρα δεδομένα στο block. Το tag αποτελείται από 3 bit και δείχνει από ποια θέση μνήμης προέρχονται τα δεδομένα. Το μέγεθός του εξαρτάται από την συνάρτηση: #address_bits – $\log_2$(cache_size/associativity). Το τελευταίο πεδίο είναι το data, έχει μέγεθος 128 bit και περιέχει 4 λέξεις.

Η διεύθυνση που παράγουν οι εντολές είναι 12 bytes και αποτελείται από 4 πεδία, το index τα byte και word offset και το tag. Το index αποτελείται από 5 bits ($\log_2$(size/associativity)) και υποδεικνύει σε ποια θέση της cache θέλει να έχει πρόσβαση η εντολή. Το word offset αποτελείται από 2 bits και δείχνει ποια από τις 4 λέξεις του block θέλει η εντολή. Το byte offset δείχνει ποιο byte της λέξης χρειάζεται. Τέλος το tag αποτελείται από τα 3 bits που υπολείπονται και χρησιμοποιείται για τον έλεγχο ύπαρξης του σωστού block στην cache.

Κυμματομορφές







Συμπεράσματα

Μάθαμε πως να διαχειριζόμαστε exceptions στο hardware και πως να δημιουργήσουμε αλλά και να προσθέσουμε στον επεξεργαστή μία μνήμη cache.