ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
Οργάνωση Υπολογιστών

Ομάδα: LAB31235453
Μπαλαμπάνης Ηλίας 2014030127
Μποκαλίδης Αναστάσιος 2014030069

Αναφορά Εργαστηρίου 2

Περιγραφή Άσκησης

Α) Μελετήστε την κωδικοποίηση των εντολών του CHARIS
Σε αυτό το ερώτημα πρέπει να αναγνωρίσουμε και να αποκωδικοποιήσουμε κάθε εντολή. Οι μαθηματικές και οι λογικές πράξεις καθώς και κάποιες εντολές ολίσθησης, είναι R τύπου και διακρίνονται από το function που έχουν. Τις υπόλοιπες εντολές που είναι τύπου I και J τις διακρίνουμε με τον αν χρειάζεται να κάνουμε SignExtend το Immed ή ZeroFill το Immed στους αντίστοιχους καταχωρητές.

Β) Σχεδιασμός και Υλοποίηση Κύριας Μνήμης 2048x32
Σε αυτό το ερώτημα απλώς πήραμε το κώδικα από την εκφώνηση και υλοποιήσαμε την μνήμη που ζητείται. Μπορούμε να αναφέρουμε ότι δεν είναι όλη η μνήμη μόνο για δεδομένα ή εντολές. Οι θέσεις από 0-1023 είναι για εντολές και οι υπόλοιπες για δεδομένα.

Γ) Σχεδιασμός και υλοποίηση βαθμίδας ανάκλασης εντολών (IFSTAGE)

Σε αυτό το ερώτημα μας ζητήθηκε να υλοποιήσουμε ένα module με το οποίο θα ελέγχουμε ποιες εντολές θα ανακτούμε από την μνήμη μας. Το σύστημα αποτελείται από έναν καταχωρητή των 32 bits που είναι ο PC (Program Counter). Κάθε φορά που εκτελείται μια εντολή η διεύθυνση του PC αυξάνεται κατά 4 πράγμα που σημαίνει ότι στον επόμενο κύκλο θα δείχνει στην επόμενη διεύθυνση της μνήμης και θα ανακτήσουμε μια νέα εντολή. Πέρα από την μνήμη και τον PC υλοποιούμε και ένα αθροιστή στον οποίο προστίθεται η διεύθυνση του PC, αυξημένη κατά 4, μαζί με μια είσοδο, την PC_Immed η οποία είναι 32 bits. Αυτό γίνεται έτσι ώστε όταν θα έχουμε μια εντολή branch τότε να προστίθεται στην διεύθυνση του PC η τιμή Immediate της εντολής branch και να κατευθύνεται στην επιθυμητή διεύθυνση της μνήμης. Τέλος έχουμε και ένα πολυπλέκτη με τον οποίο ελέγχουμε για το ποια πληροφορία θα περάσει στον PC (η προηγούμενη διεύθυνση του +4 ή η προηγούμενη διεύθυνση του

+Immediate). Πρέπει να επισημάνουμε ότι στην είσοδο διεύθυνσης της μνήμης δεν περνάμε την 32 bit πληροφορία του PC αλλά τα bit (12 downto 2)

Δ) Σχεδιασμός και υλοποίηση βαθμίδας αποκωδικοποίησης εντολών (DECSTAGE)

Σε αυτό το ερώτημα μας ζητήθηκε να υλοποιήσουμε ένα module με το οποίο θα αποκωδικοποιούμε τις εντολές που παίρνουμε από την μνήμη και ελέγχουμε ποια δεδομένα θα εισέρχονται στους καταχωρητές της REGISTER FILE που υλοποιήσαμε στο προηγούμενο εργαστήριο. Αρχικά χωρίζουμε την εντολή μας που ανακτούμε από την μνήμη σε 4 διαφορετικά σήματα. Το πρώτο σήμα αποτελείται από τα 25-21 bits της εντολής και μας δείχνει ποιόν καταχωρητή να διαβάσουμε από την $1^{η}$ θύρα ανάγνωσης της REGISTER FILE. Το δεύτερο σήμα αποτελείται από τα 15-11 bits της εντολής και μας δείχνει ποιόν καταχωρητή να διαβάσουμε από την $2^{η}$ θύρα ανάγνωσης της REGISTER FILE. Το τρίτο σήμα αποτελείται από τα 20-16 bits της εντολής και μας δείχνει σε ποιο καταχωρητή της REGISTER FILE να γράψουμε ή να διαβάσουμε(αυτή την επιλογή την ελέγχουμε με ένα πολυπλέκτη όπως φαίνεται στο διάγραμμα της βαθμίδας). Το τέταρτο και τελευταίο σήμα αποτελείται από τα 15-0 bits της εντολής και μέσω του Immed_calculator που φτιάξαμε παράγουμε το Immed που είναι 32 bits . Στο Immed_calculator πήραμε όλες τις περιπτώσεις εντολών που μας δώθηκαν στον αρχικό πίνακα της εκφώνησης και σύμφωνα με το opcode κάθε εντολής ρυθμίζουμε αν χρειάζεται sign extend (Immed) ή zerofill (Immed). Σε αυτή την βαθμίδα χρησιμοποιήσαμε και ένα ακόμα πολυπλέκτη με το οποίο διαλέγουμε από πού θέλουμε να εισάγουμε δεδομένα στους καταχωρητές, είτε από την μνήμη είτε από την ALU.

Ε) Σχεδιασμός και υλοποίηση βαθμίδας Εκτέλεσης Εντολών (ALUSTAGE)
Σε αυτό το ερώτημα προεκτείναμε στην ουσία την ALU με ένα πολυπλέκτη. Σε αυτόν τον πολυπλέκτη επιλέγουμε ανάλογα με την εντολή τον $2^{ο}$ αριθμό με τον οποίο θέλουμε να κάνουμε πράξη. Αν είναι τύπου R η εντολή θα περνάμε σαν είσοδο στην ALU το αποτέλεσμα από ένα καταχωρητή διαφορετικά περνάμε το Immed που παράγουμε παρακάτω.

ΣΤ) Σχεδιασμός και υλοποίηση βαθμίδας Πρόσβασης Μνήμης (MEMSTAGE)

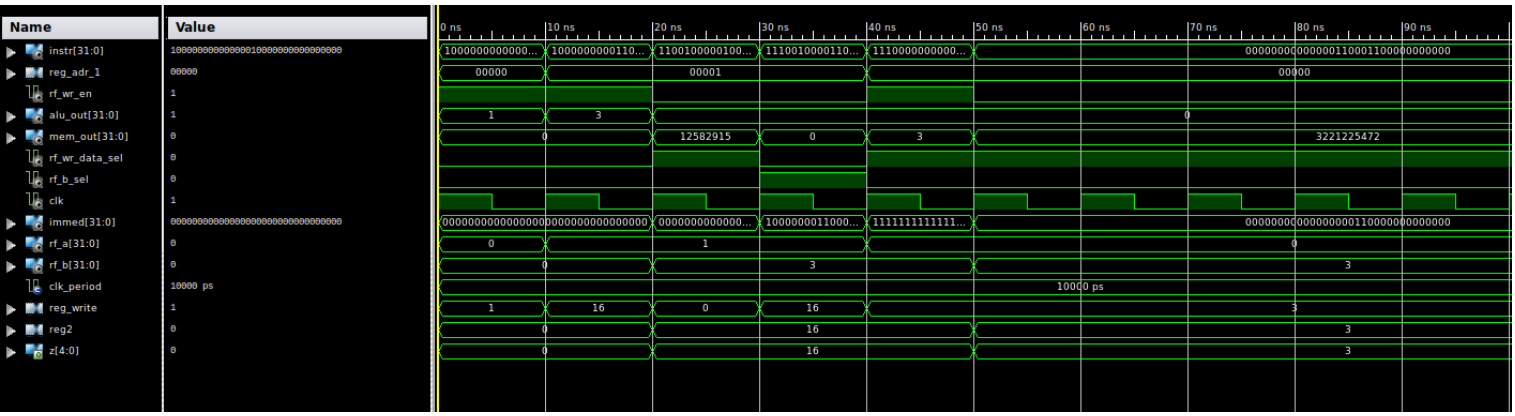
Σε αυτό το ερώτημα υλοποιήσαμε το κομμάτι της μνήμης όπου αποθηκεύουμε τα δεδομένα. Στην ουσία την διεύθυνση εγγραφής δεδομένων , η οποία είναι από την ALU, την προσθέτουμε κατά 0x400. Αυτό το κάνουμε διότι χωρίζουμε την μνήμη μας σε 2 χώρους, ένα για εγγραφή-ανάγνωση εντολών που είναι από 0-1023 θέση και ένα για εγγραφή-ανάγνωση δεδομένων που είναι από 1024-2048 θέση. Οπότε όταν προσθέτουμε την τιμή 0x400 κατευθείαν ξεκινάμε από την θέση 1024 που αρχίζει η περιοχή των δεδομένων.

Κυματομορφές

IFSTAGE κυματομορφή :

Αρχικά κάνουμε reset τον PC. Μετά επιλέγουμε από τον πολυπλέκτη να εισάγεται στον PC η διεύθυνση PC+4. Αργότερα ζητάμε να εισάγουμε στον PC το Immed που είναι το 16 και συνεχίζουμε κάπως ανάλογα το τεστ.

DECSTAGE κυματομορφή :

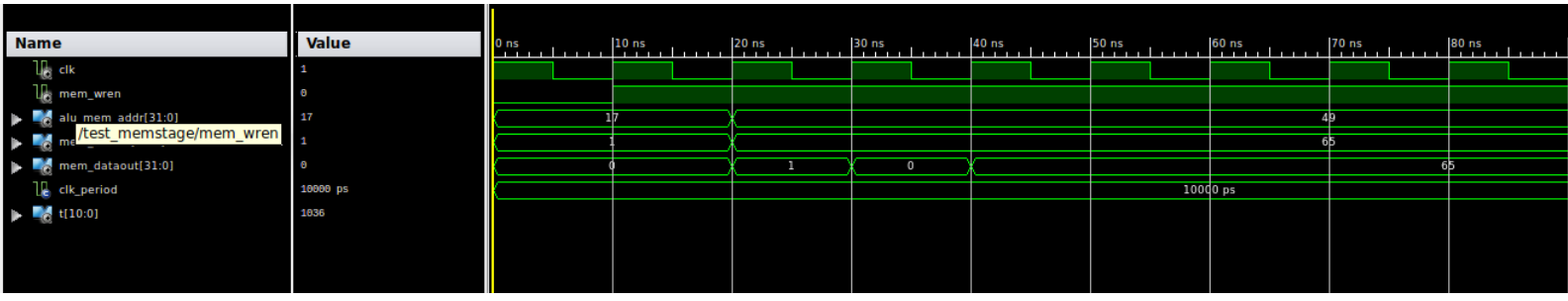


1ος κύκλος: Γράφουμε το ALU_OUT στον REG1

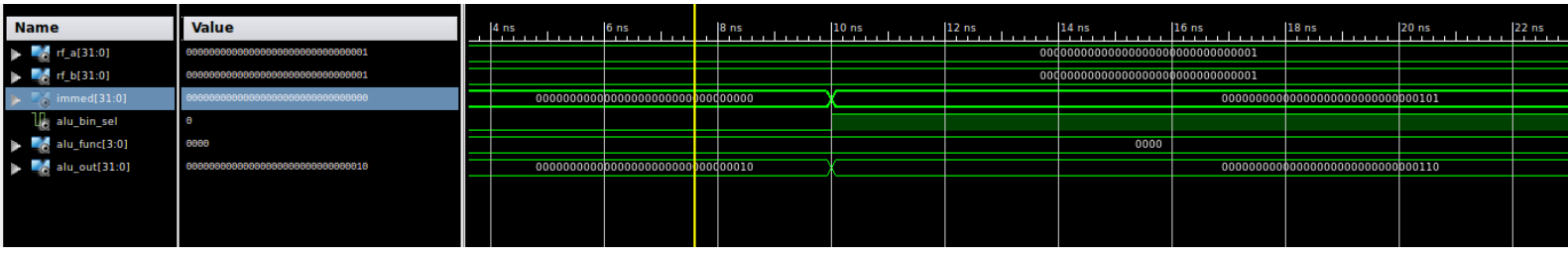
2ος κύκλος: Γράφουμε το ALU_OUT στον REG16

Στους **επόμενους κύκλους** παίζουμε με τα immed και αναλόγως γίνεται sign extend η zero fill.

MEMSTAGE κυματομορφή:



ALUSTAGE κυματομορφή:



Συμπεράσματα

Σε αυτό το εργαστήριο μάθαμε για κάποιες από τις πιο βασικές μονάδες του επεξεργαστή mips. Συνδιάσαμε τον Regsiter File και την Alu από το προηγούμενο εργαστήριο με κάποιες νέες μονάδες. Πλέον γίνονται όλες οι πράξεις του επεξεργαστή μέσω αυτών των μονάδων. Καταλάβαμε επίσης πως οι εντολές που μάθαμε στο μάθημα των ψηφιακών υπολογιστών αποκωδικοποιούνται σε bits και ανάλογα με το τι πληροφορία περιέχουν γίνονται οι αντίστοιχες διεργασίες-πράξεις στον επεξεργαστή. Επιπλεόν είδαμε την επικοινωνία του επεξεργαστή με την μνήμη, δηλαδή πως ανακτά και πότε εντολές και τι δεδομένα αποθηκεύει ανάλογα με αυτές τις εντολές. Τέλος είδαμε πως λειτουργεί πραγματικά ο καταχωρητής PC που μάθαμε σε προηγούμενο μάθημα και πως ακριβώς αλλάζει σε περιπτώσεις που έχουμε εντολές branch-jump.