18/4/2019

2ο Εργαστήριο Οργάνωσης Υπολογιστών
ΣΧΕΔΙΑΣΗ ΒΑΣΙΚΩΝ ΒΑΘΜΙΔΩΝ ΤΟΥ DATAPATH ΕΝΟΣ ΑΠΛΟΥ ΕΠΕΞΕΡΓΑΣΤΗ
8/3/2019

Ομάδα LAB31239632

Αλέξανδρος Πουπάκης 2016030061
Βαγγέλης Κιούλος 2016030056

Σκοπός εργαστηριακής άσκησης

Στη παρούσα εργαστηριακή άσκηση έπρεπε να σχεδιάσουμε τις βασικές βαθμίδες του datapath ενός απλού επεξεργαστή, χρησιμοποιώντας το ALU καθώς και το Register File που αναπτύξαμε στο 1ο εργαστήριο.

Προεργασία

Έπρεπε να σχεδιαστούν 5 modules, η RAM, το IFSTAGE, το DECSTAGE, ALUSTAGE, MEMSTAGE.

Για την RAM χρησιμοποιήσαμε τον δοθέντα κώδικα και την αρχικοποιήσαμε με το αρχείο ram.data από τα έγγραφα του μαθήματος.

Για το IFSTAGE κατασκευάσαμε αρχικά ένα module με την λογική της μονάδας ανάκλησης εντολών και έπειτα σε ένα top module το συνδέσαμε με τη RAM, σύμφωνα με το σχεδιάγραμμα της εκφώνησης.

Για το DECSTAGE υλοποιήσαμε την απαραίτητη λογική και την συνδέσαμε με το Register File του 1ου εργαστηρίου. Επίσης παρατηρήσαμε ότι η λογική για την παραγωγή της εξόδου Immed χρειαζόταν κάποια επιπλέον σήματα ελέγχου και τα οποία προσθέσαμε.

Για το ALUSTAGE, ομοίως με το DECSTAGE, υλοποιήσαμε την απαραίτητη λογική και τη συνδέσαμε με το υπάρχον ALU.

Στο MEMSTAGE γίνεται το offset των διευθύνσεων κατά 0x400 και συνδέονται τα σήματα με την μνήμη.

Περιγραφή

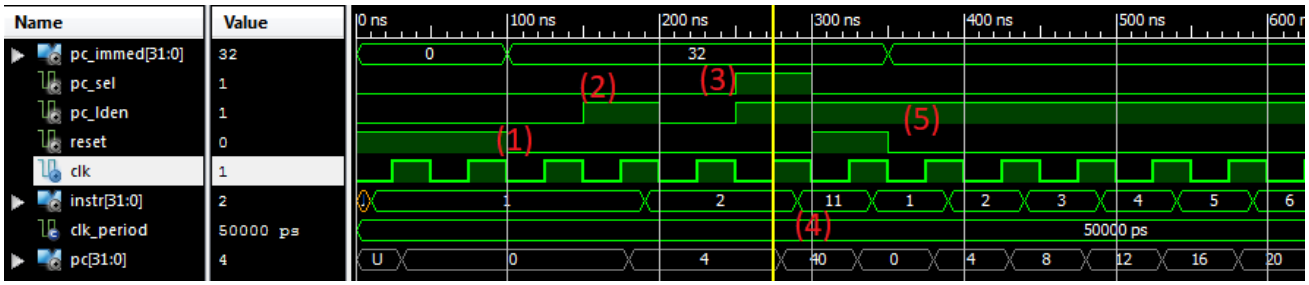
Η ανάπτυξη του VHDL κώδικα που υλοποιεί τις παραπάνω μονάδες ήταν αρκετά απλή και για αυτό δεν χρήζει εξήγησης. Αναφέρουμε μόνο ότι στο DECSTAGE κατασκευάσαμε τρία processes, ένα για την επιλογή του δεύτερου καταχωρητή προς ανάγνωση, ένα για την επιλογή των δεδομένων εγγραφής (είτε το ALU_out είτε το MEM_out) και τέλος ένα για την μετατροπή του 16-bit immediate ακεραίου σε 32-bit ακέραιο. Για το τελευταίο process, προστέθηκαν τα σήματα SignExtend και Shift ως είσοδοι στο DECSTAGE module, είναι

ενεργά στο ‘1’ και ελέγχουν την επέκταση προσήμου και ολίσθηση κατά 2 bit του Immediate αντίστοιχα.

Επισημαίνουμε επίσης ότι στο IFSTAGE, χρησιμοποιήθηκαν τα bit 12 έως 2 του καταχωρητή PC για τον προσδιορισμό της διεύθυνσης ανάγνωσης εντολής, αφού η μνήμη είναι ευθυγραμμισμένη ανά 4 byte (οπότε διαβάζουμε εντολές από διευθύνσεις που είναι ακέραια πολλαπλάσια του 4). Η επιλογή των 11 bit οφείλεται στο ότι η μνήμη μας είναι 2 kB, οπότε δεν χρειάζονται τα πάνω 19 bit.

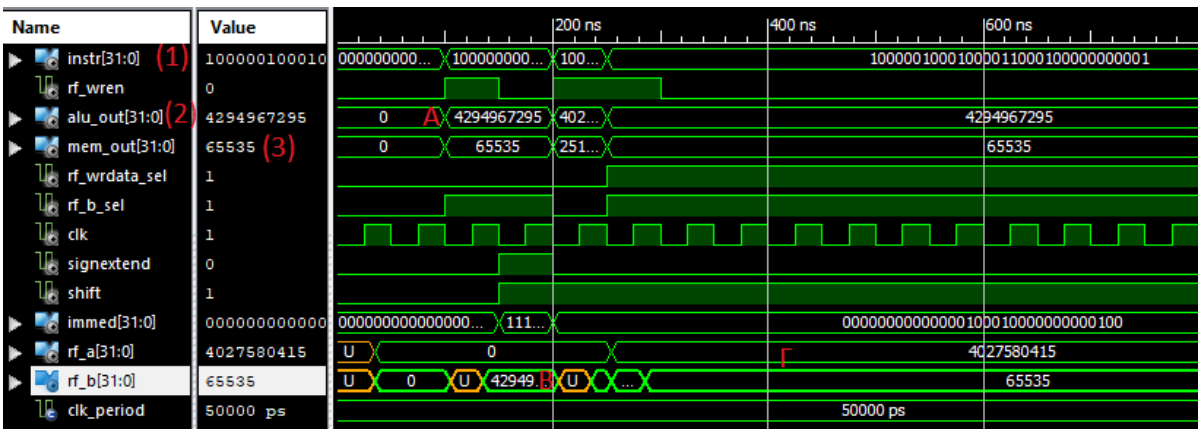
Κυματομορφές-Προσομοίωση

IFSTAGE:



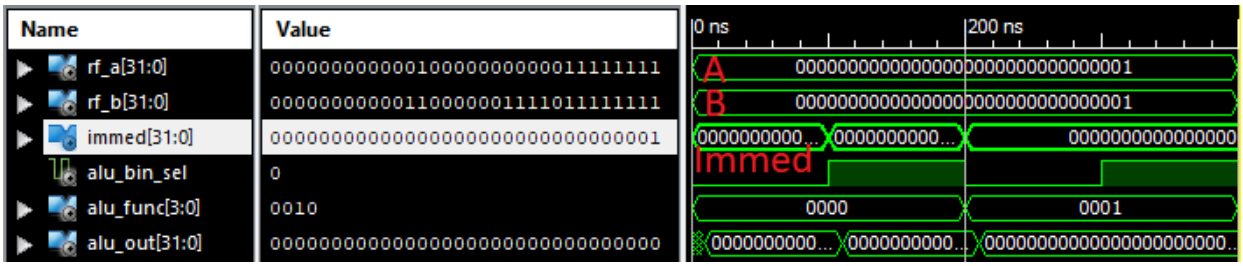
Αρχικά, ανοίγοντας το ram.data σε έναν editor παρατηρούμε ότι στις διευθύνσεις 0 έως 30 βρίσκονται ακολουθίες bit που αντιστοιχούν στους δεκαδικούς αριθμούς 1 ως 31 αντίστοιχα. Επίσης, στην παραπάνω κυματομορφή εμφανίζεται (με γκρι) το εσωτερικό σήμα PC που δείχνει την διεύθυνση ανάγνωσης της μνήμης. Παρατηρούμε ότι μετά το Reset, το instr έχει την τιμή 1 (ένδειξη 1), όπως θα έπρεπε αφού διαβάζουμε τη διεύθυνση 0 και έπειτα με ένα load στον καταχωρητή PC αυξάνεται η διεύθυνση κατά 4 και διαβάζουμε το 2 στην έξοδο (ένδειξη 2). Ύστερα, φορτώνουμε μια φορά το PC_Immed στον PC μέσω του PC_sel = ‘1’ (ένδειξη 3) και η διεύθυνση αυξάνεται κατά 4 και την τιμή του PC_Immed, δηλ. 4 + 4 + 32 = 40 (ένδειξη 4), όπου σωστά διαβάζουμε την τιμή 11 στο Instr. Στη συνέχεια αφήνουμε το PC_LdEn στο ‘1’ και ο PC αυξάνεται μονίμως κατά 4 (ένδειξη 5).

DECSTAGE:



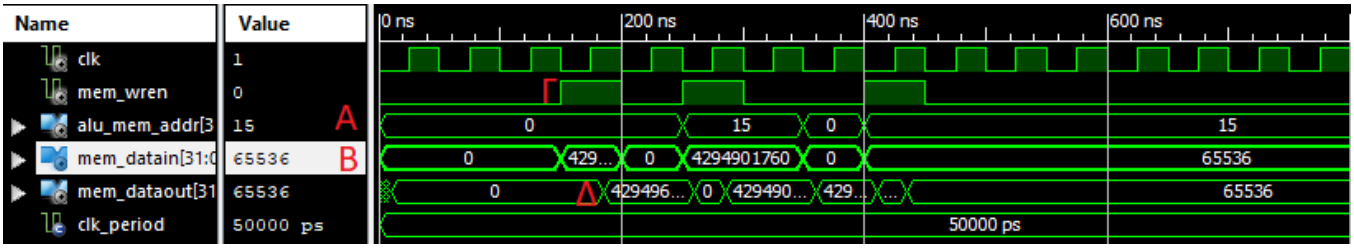
Στις παραπάνω κυματομορφές, στην είσοδο “instr” (1), εισάγουμε την εντολή που θέλουμε, ενώ στις εισόδους “alu_out” (2) και “mem_out” (3) εισάγουμε τα δεδομένα που θέλουμε να αποθηκεύσουμε στους καταχωρητές. Στην πρώτη περίπτωση, διαβάζουμε τα δεδομένα του καταχωρητή 0, οπότε το rs (bits 25-21 της εντολής) είναι “00000” και γράφουμε τα δεδομένα που υπάρχουν στην είσοδο “alu_out”, δηλαδή την τιμή “0x0000FFFF” στον καταχωρητή 1 καθώς το rd (bits 20-16 της εντολής) είναι “00001” (ένδειξη Α). Στην έξοδο RF_A, παρατηρούμε το “0x00000000”, και στην έξοδο RF_B, παρατηρούμε τα δεδομένα που υπήρχαν στην είσοδο alu_out (ένδειξη Β). Στην δεύτερη περίπτωση, διαβάζουμε τον καταχωρητή 0 καθώς το rs είναι “00000” και επομένως στην έξοδο RF_A έχουμε την τιμή “0x00000000” και γράφουμε την τιμή “0xF00FFFFF” στον καταχωρητή 17 καθώς το rd είναι “10001”. Προφανώς, στην έξοδο RF_B έχουμε την τιμή “0xF00FFFFF”, δηλαδή την τιμή που ήταν αποθηκευμένη στον καταχωρητή 17. Με την τελευταία εντολή, διαβάζουμε τα δεδομένα του καταχωρητή 17 καθώς το rs είναι “10001” άρα, στην έξοδο RF_A έχουμε την τιμή “0xF00FFFFF” και γράφουμε στον καταχωρητή 1 (rd = 00001) την τιμή που υπάρχει στην είσοδο “mem_out” δηλαδή την τιμή “0x0000FFFF”. Η τιμή “0x0000FFFF” εμφανίζεται στην έξοδο RF_B (ένδειξη Γ).

ALUSTAGE:



Στις παραπάνω κυματομορφές, παρατηρούμε την λειτουργία της μονάδας “ALUSTAGE”. Στις εισόδους, RF_A και RF_B, εισάγουμε τους τελεστέους της πράξης που θέλουμε να κάνουμε ενώ στην είσοδο “alu_func” εισάγουμε τον κωδικό της πράξης που θέλουμε να γίνει. Εάν η είσοδος “alu_bin_sel” είναι '1', τότε η πράξη θα γίνει ανάμεσα στα δεδομένα των εισόδων RF_A και Immed. Στο τέλος, στην έξοδο “alu_out” εμφανίζεται το αποτέλεσμα της πράξης.

MEMSTAGE:



Παραπάνω, παρατηρούμε τις κυματομορφές που προκύπτουν από την προσομοίωση της μονάδας “MEMSTAGE”. Στην είσοδο “alu_mem_addr” (ένδειξη Α) εισάγουμε την διεύθυνση στην οποία θέλουμε να αποθηκεύσουμε τα δεδομένα μας. Οι διευθύνσεις που εισάγουμε αποθηκεύονται από την διεύθυνση 1024 και μετά. Στην συνέχεια, στην είσοδο “mem_data_in” (ένδειξη Β) εισάγουμε τα δεδομένα που θέλουμε να αποθηκεύσουμε στην μνήμη. Επιπλέον, όταν η είσοδος “mem_wren” (ένδειξη Γ) είναι ΄1΄, τότε, μπορούμε να εισάγουμε τα δεδομένα μας στην μνήμη. Στην έξοδο “mem_dataout” (ένδειξη Δ) παρατηρούμε τα δεδομένα που έχουμε αποθηκεύσει.

Συμπεράσματα

- Στην υλοποίηση μας, στις διευθύνσεις 0 έως 1023, αποθηκεύονται οι εντολές, ενώ στις διευθύνσεις από 1024 έως 2047, αποθηκεύονται τα δεδομένα.

- Πρέπει να λάβουμε υπόψη ότι ανάλογα με την εντολή, πρέπει να κάνουμε διαφορετική επέκταση πρόσημου.