Εργασία 1 – Αναφορά

Μιχαήλ Κρατημένος

ΑΜ: 2018030104

Σκοπός εργασίας: Η υλοποίηση ενός single-cycle επεξεργαστή CHARIS, ο οποίος είναι αντίστοιχος με τον επεξεργαστή του MIPS.

1η Φάση

Σε αυτήν τη φάση έγινε υλοποίηση των οντοτήτων ALU και Register File (RF). Πιο συγκεκριμένα, η ALU εκτελεί αριθμητικές και λογικές πράξεις, ανάλογα με το opcode που δίνεται, ενώ η register file περιέχει τους 32 καταχωρητές που χρησιμοποιούνται στην αρχιτεκτονική μας.

ALU: κύκλωμα που εκτελεί αριθμητικές και λογικές πράξεις χρησιμοποιώντας συμπληρώματα ως προς 2. Με τη χρήση της εντολής case, καθώς και του opcode εκτελούνται οι απαιτούμενες εντολές που ζητούνται στην εκφώνηση.

Το σήμα **Zero** ενεργοποιείται όταν η έξοδος (σήμα **Out1**) είναι μηδέν, ενώ το σήμα **Ovf** ενεργοποιείται όταν υπάρχει υπερχείλιση (overflow), δηλαδή όταν προστίθενται 2 ομόσημοι αριθμοί και προκύπτει ετερόσημος, καθώς και όταν αφαιρούνται 2 αριθμοί και το αποτέλεσμα είναι ομόσημο με το 2ο τελεστή. Για το σήμα **Cout** χρησιμοποιήθηκε ένα ενδιάμεσο σήμα 33 bit κρατώντας εν τέλει το MSB το οποίο είναι το carry out.

RF: Το register file είναι το κύκλωμα των τριών modules (decoder, 32 registers και 2 multiplexers 32 εισόδων) που ζητήθηκε στην εκφώνηση με τη διαφορά ότι το write enable καθώς και οι πύλες AND υλοποιήθηκαν μέσα στον αποκωδικοποιητή, ενώ στον καταχωρητή 0 δεν επιτρέπεται να γράψει κανείς γιαυτό το write enable είναι πάντα 0. Επίσης για τις εισόδους των πολυπλεκτών χρησιμοποιήθηκε μια διάταξη 32 32-bitων θέσεων, μέσω της δημιουργίας πακέτου (**MUX_IN_PACK**).

2η Φάση

RAM: Έγινε χρήση του έτοιμου κώδικα που μας δόθηκε στην εκφώνηση και αφού ελέγχθηκε μέσω testbench, έγινε ένωση με τα modules IFSTAGE και MEMSTAGE, όπου ουσιαστικά έπαιξε το ρόλο της μνήμης στα top modules που προέκυψαν (IFSTAGE_TOPMODULE, MEMSTAGE_TOPMODULE).

IFSTAGE: Στο κύκλωμα αυτό ενημερώνεται ο program counter (PC) ανάλογα με την κάθε εντολή που δίνεται στο instruction set (Instr). Πιο αναλυτικά, περιέχει τον καταχωρητή PC, έναν αθροιστή που αυξάνει κατά 4 τον PC (ADDER_INCREMENTOR) καθώς και έναν αθροιστή για τις εντολές διακλάδωσης (ADDER_INC_AND_IMMED) ο οποίος προσθέτει και το immediate. Τέλος έχουμε και έναν πολυπλέκτη (MUX2TO1) ο οποίος επιλέγει ποιανού αθροιστή την τιμή να περάσει αναλόγως με το αν θέλουμε διακλάδωση ή όχι, μέσω του σήματος PC_Sel.

DECSTAGE: Το κύκλωμα αυτό αποκωδικοποιεί τις εντολές. Πιο συγκεκριμένα, δέχεται το instruction από τη RAM και τα πιο σημαντικά bits του πάνε στο RF για τις εντολές τύπου R, ενώ για τις εντολές immediate έχει υλοποιηθεί το συννεφάκι (CLOUD_UNIT) και σε αυτό πάνε τα λιγότερο σημαντικά bits του instruction. Ειδικότερα, το συννεφάκι αυτό επεκτείνει το immediate με 4 διαφορετικούς τρόπους, σύμφωνα με τις απαιτήσεις των κωδικοποιήσεων των εντολών για τις αντίστοιχες τιμές του ImmExt ως εξής:

00: Zero filling, 01: Zero filling και 16 αριστερές ολισθήσεις

10: Sign extension, 11: Sign extension και 2 αριστερές ολισθήσεις (πολλαπλασιασμός με 4)

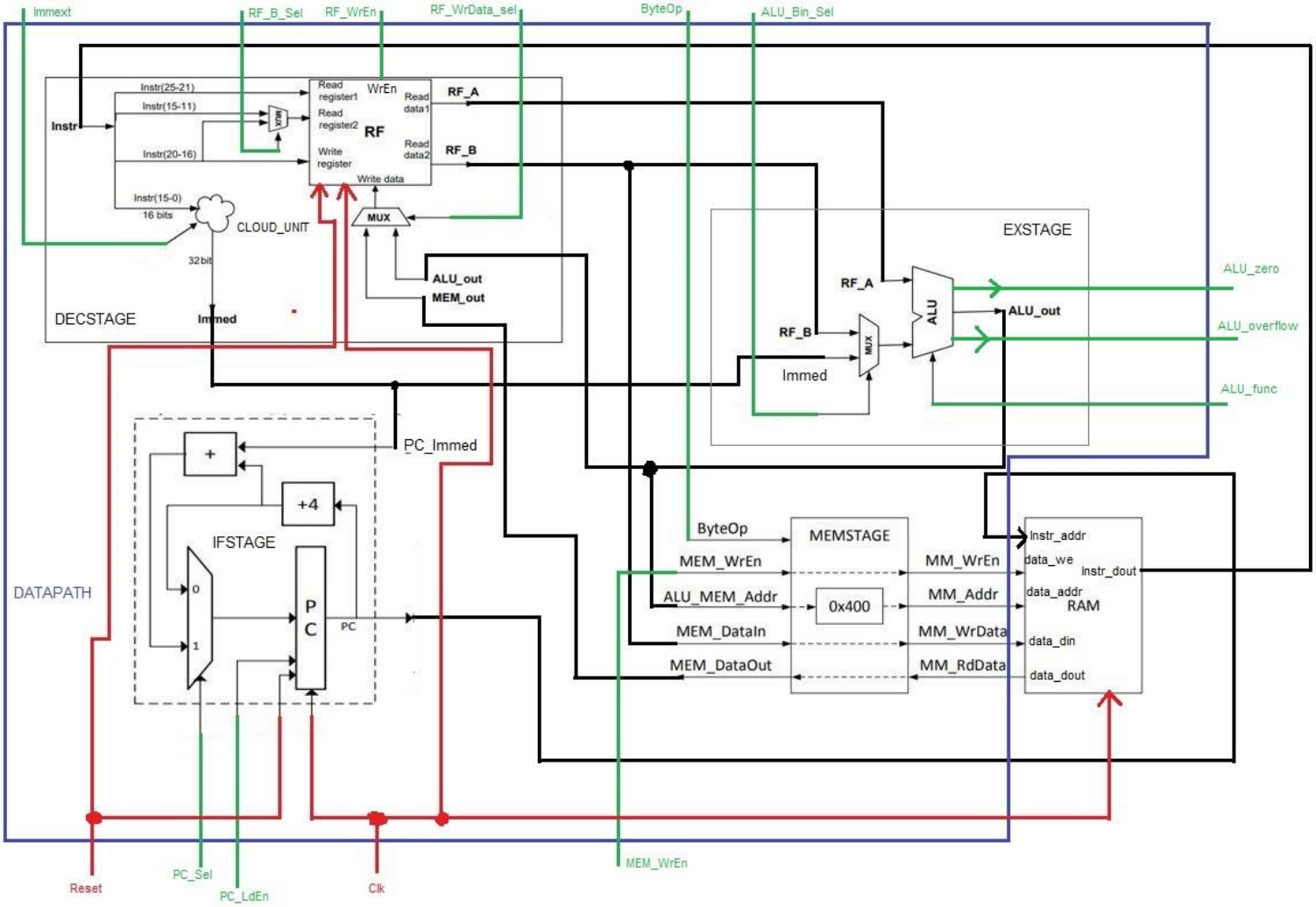
EXSTAGE: Το κύκλωμα αυτό είναι η βαθμίδα εκτέλεσης αριθμητικών και λογικών εντολών, γιαυτό και εμπεριέχει την ALU από την πρώτη φάση, καθώς και έναν πολυπλέκτη που επιλέγει από ποια βαθμίδα να πάρει σήμα αναλόγως τι τύπου εντολή έχουμε. Ως έξοδοι υπάρχουν τα σήματα ALU_zero και ALU_overflow που χρησιμοποιούνται στο CONTROL αργότερα, αλλά και στο τελικό simulation.

MEMSTAGE: Στο κύκλωμα αυτό γίνεται εγγραφή (store) και ανάγνωση (load) μνήμης. Αρχικά επιβεβαιώνεται ότι η διεύθυνση της μνήμης που θα διαβάζεται πρέπει να είναι η διεύθυνση που στέλνει η ALU προστιθέμενη κατά 0x400 (offset), έτσι ώστε τα δεδομένα που χρειάζεται να αποθηκευτούν στη μνήμη να αποθηκεύονται στο σωστό μέρος της μνήμης.

Με τη χρήση του σήματος ByteOp, όταν είναι 0 δεν υπάρχουν αλλαγές στις εξόδους, άρα λειτουργούν οι εντολές lw/sw. Σε αντίθετη περίπτωση, για lb/sb, στις εξόδους μένουν τα 8 λιγότερο σημαντικά bit (1 byte) και στα πιο σημαντικά γίνεται zero filling, καθώς δεν μπορούν να χρησιμοποιηθούν.

3η Φάση

DATAPATH: Με τη σύνδεση όλων των βαθμίδων από τη 2η φάση δημιουργήθηκε το datapath. Επίσης, μόνο στο testbench έγινε σύνδεση και της RAM, για να είναι σίγουρη η ορθή λειτουργία του κυκλώματος όπως φαίνεται παρακάτω:



CONTROL: Στη βαθμίδα αυτή ουσιαστικά γίνεται έλεγχος του τι θα δείχνουν τα σήματα εξόδου σύμφωνα με το Instruction που δέχεται ως είσοδο, το οποίο καθορίζει το Opcode (τα 6 πιο σημαντικά bits), το FUNC (τα 6 λιγότερο σημαντικά bits) για εντολές τύπου R, αλλά και το immediate (τα 16 λιγότερο σημαντικά bits) για εντολές τύπου I. Τα υπόλοιπα bits καθορίζουν τους καταχωρητές που χρησιμοποιούνται. Ειδικότερα, οι τιμές του Opcode καθορίζουν το τι τιμές θα πάρουν τα σήματα εξόδου όντας οι κωδικοί των εντολών που δίνονται, τα οποία μετά κατευθύνουν το Datapath.

Οι τιμές των σημάτων εξόδου έχουν καθοριστεί ως εξής:

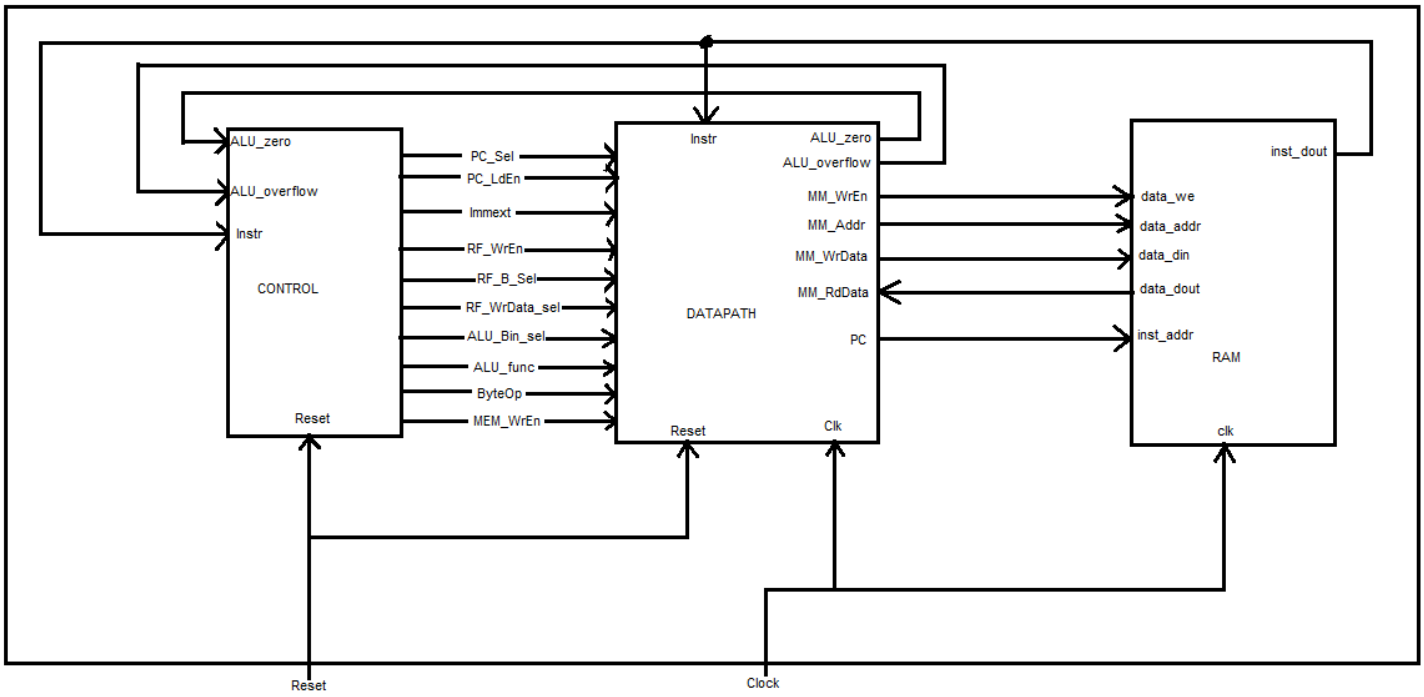
- Το PC_Sel είναι ενεργό (1) όταν έχουμε εντολές branch, αλλιώς είναι κλειστό.
- Το RF_WrEn είναι ενεργό όταν πρέπει να γίνει εγγραφή σε κάποιον καταχωρητή.
- Το RF_WrData_sel είναι ενεργό μόνο για τις εντολές lb, lw όπου και χρειάζεται να φορτωθεί το αποτέλεσμα της μνήμης και αντίστοιχα το MEM_WrEn είναι ενεργό μόνο για τις εντολές sb, sw όπου και χρειάζεται να γίνει καταχώρηση στη μνήμη.

Οι τιμές της ALU_func (FUNC) για τις εντολές τύπου I δίνονται ως εξής:

- 0000, για τις εντολές li, lui, addi, lb/sb και lw/sw, αφού χρησιμοποιούν την ALU ως αθροιστή.
- 0001, για τις εντολές beq, bne, αφού εδώ η ALU χρησιμοποιείται για αφαίρεση φορτώνεται στον PC_Sel το ALU_zero, όπου ανάλογα το αποτέλεσμα της πράξης επιστρέφεται true ή false.
- 0011 και 0101 για τις εντολές ori και nandi αντίστοιχα, όπου εδώ η ALU χρησιμοποιείται για τις αντίστοιχες πράξεις.
- Για την εντολή b δεν χρησιμοποιήθηκε η ALU, καθώς δεν ήταν απαραίτητη.

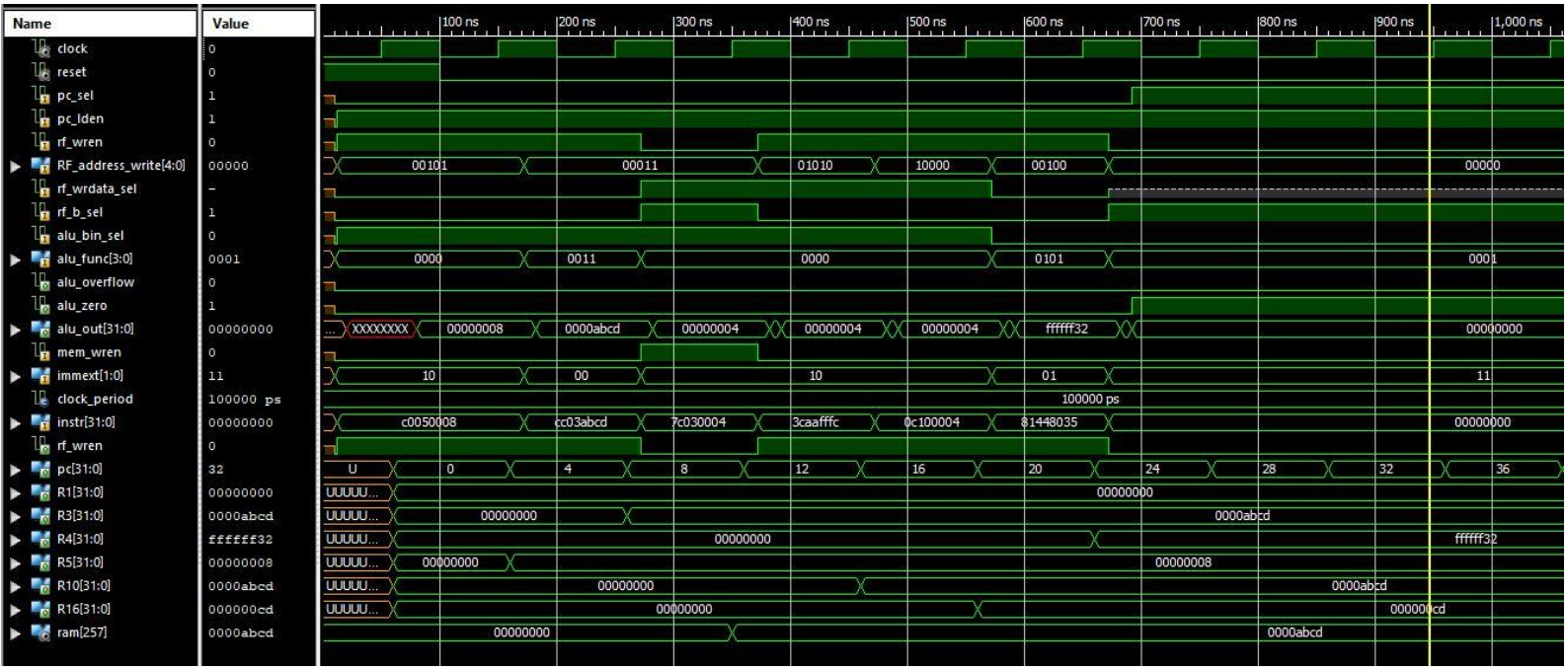
Για τις εντολές τύπου R η ALU_func παίρνει τα 4 λιγότερο σημαντικά bits του Instruction, τα οποία συμβολίζουν την πράξη που πρέπει να γίνει

<u>PROC_SC:</u> Το κύκλωμα αυτό αποτελεί τον επεξεργαστή ενός κύκλου όπου ζητήθηκε να σχεδιαστεί σε αυτήν τη φάση και επιτελείται με τη σύνδεση των modules: Datapath, Control και RAM:



Προγράμματα αναφοράς

1)



Παρατηρήθηκε ότι οι εντολές που ζητήθηκαν να γίνουν σε αυτό το στάδιο έχουν εκτελεστεί σωστά, αφού στον R5 έχει φορτωθεί η τιμή 8 (**addi r5, r0, 8**) στα 100 – 200 ns. Στον R3 φορτώνεται η τιμή xABCD (**ori r3, r0, ABCD**) στα 200 – 300 ns, ενώ η ALU_func έχει πολύ σωστά την τιμή 0011 για το OR. Στα 300 – 400 ns παρατηρείται ότι φορτώνεται σε θέση της RAM η τιμή της R3 (**sw r3, 4(r0)**). Στα 400 – 500 ns γράφεται σωστά στον R10 η τιμή από τη RAM (**lw r10, -4(r5)**). Στα 500 – 600 ns γράφεται σωστά στον R16 η τιμή x00CD, αφού φορτώνουμε μόνο ένα byte (**lb r16, 4(r0)**) από την ίδια διεύθυνση μνήμης με πριν. Τέλος στα 600 – 700 ns γράφεται σωστά στον R4 η το αποτέλεσμα της πράξης nand (**nand r4, r10, r16**)

2)

Εδώ έπρεπε να επιτευχθεί ένα infinite loop, κάτι που σημαίνει ότι δεν εκτελείται ποτέ η εντολή addi r1, r0, 1 κάτι που φαίνεται από τις τιμές του instr αλλά και του R1 που είναι πάντα 0.

Παρατηρήσεις

- Χρησιμοποιήθηκε ακόμη ένα rom αρχείο (rom3) για την επαλήθευση της λειτουργίας του Datapath. Από ότι παρατηρήθηκε λειτουργεί σωστά.
- Επειδή έχουν οριστεί καθυστερήσεις στα κυκλώματα, η περίοδος του ρολογιού έχει οριστεί στα 100 ns για να προλαβαίνουν οι τιμές να αλλάζουν.
- Τα testbenches των IFSTAGE, MEMSTAGE έγιναν σε σύνδεση με τη μνήμη.