ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ & ΥΛΙΚΟΥ
ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ ΓΙΑ ΤΟ ΜΑΘΗΜΑ: ΗΡΥ 312
ΟΡΓΑΝΩΣΗ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2017
Εργαστήριο 4

Υλοποίηση σύνθετων εντολών σε έναν επεξεργαστή πολλαπλών κύκλων

Σκοπός του Εργαστηρίου

1. Ο ορισμός της αρχιτεκτονικής ενός συνόλου σύνθετων εντολών.
2. Η σχεδίαση και ολοκλήρωση ενός επεξεργαστή πολλαπλών κύκλων που υλοποιεί σύνθετες εντολές και θα βασίζεται σε προηγούμενο παραδοτέο.
3. Η βαθύτερη κατανόηση λειτουργίας ενός επεξεργαστή πολλαπλών κύκλων.

Αρχιτεκτονική Σύνθετων Εντολών

Καλείστε να υλοποιήσετε έναν non-pipelined επεξεργαστή βασισμένο σε ένα extended υποσύνολο της αρχιτεκτονικής συνόλου εντολών ECHARIS (Extended CHAnia Risc Instruction Set, Έκδοση 2), ο οποίος **εκτός από τις εντολές που υλοποιούσε στα προηγούμενα εργαστήρια**, θα υλοποιεί και το παρακάτω σύνολο εντολών:

Custom εντολές:

- cmov (conditional move)
- add_MMX_byte
- branch_equal_reg_mem
- branch_not_equal_mem
- byte_pack_mem
- byte_unpack_mem

Οι παραπάνω εντολές ανάλογα με την λειτουργία τους θα μπορούν να ανήκουν σε κάποιο από τα δύο format εντολών που σας δόθηκε στο 2ο εργαστήριο.

Η διευθυνσιοδότηση **<u>όλων των μνημών</u>** του νέου σας datapath θα γίνεται με διευθύνσεις bytes, ενώ οι λέξεις στις μνήμες θα πρέπει να είναι ευθυγραμμισμένες σε πολλαπλάσια των 4 Bytes.

Η κωδικοποίηση των εντολών καθώς και η λειτουργία τους θα γίνεται σύμφωνα με τον ακόλουθο πίνακα:

Opcode	FUNC	ΕΝΤΟΛΗ	ΠΡΑΞΗ	Περιγραφή

100001	100000	cmov	if(RF[rt] != 0)
			RF[rd] ← RF[rs]
			Else
			Nop
				Αν η τιμή του rt είναι true αντιγράφει στον καταχωρητή rd τον καταχωρητή rs.

100011	101000	add_MMX_byte	RF[rd] ← RF[rt](31 downto 24) + RF[rs](31 downto 24) & RF[rt](23 downto 16) + RF[rs](23 downto 16) & RF[rt](15 downto 8) + RF[rs](15 downto 8) & RF[rt](7 downto 0) + RF[rs](7 downto 0)	"Βλέπει" τους καταχωρητές σαν 4 ανεξάρτητα bytes. Προσθέτει τα αντίστοιχα bytes των καταχωρητών rt και rs και τα αποτελέσματα τα ενώνει δημιουργώντας μία λέξη 32 bits, η οποία αποθηκεύεται στον καταχωρητή rd.

100111	-	branch_equal_reg_mem	if(RF[rt] == MEM[SingExtend(Imm << 2)])
			PC ← PC + 4 + RF[rs]
			else
			PC ← PC + 4
				Κάνει σύγκριση μίας τιμής της μνήμης με την τιμή του καταχωρητή rt. Αν είναι ίσα η εκτέλεση του προγράμματος συνεχίζει σε συγκεκριμένη θέση μνήμης διαφορετικά συνεχίζεται κανονικά η ροή του προγράμματος.

101111	-	branch_not_equal_mem	if(MEM[RF[rs]] != MEM[RF[rt]])
			PC ← PC + 4 + Immed
			else
			PC ← PC + 4
				Κάνει σύγκριση των δεδομένων δύο θέσεων μνήμης με διευθύνσεις τα περιεχόμενα των καταχωρητών rs και rt. Αν οι τιμές δεν είναι ίσες το πρόγραμμα διακλαδίζεται, διαφορετικά η ροή του προγράμματος συνεχίζεται κανονικά.

111100	-	byte_pack_mem	base_addr = RF[rs] + SignExtend(Imm) RF[rd] (7 downto 0) ← MEM[base_addr] (7 downto 0)) RF[rd] (15 downto 8) ← MEM[base_addr + 4] (7 downto 0)) RF[rd] (23 downto 16) ← MEM[base_addr + 8] (7 downto 0)) RF[rd] (31 downto 24) ← MEM[base_addr +12] (7 downto 0))	Φορτώνει τα λιγότερο σημαντικά bytes τεσσάρων διαδοχικών θέσεων μνήμης και με αυτά φτιάχνει μια λέξη την οποία γράφει στον καταχωρητή rd. Το byte της μικρότερης διεύθυνσης αντιστοιχεί στο λιγότερο σημαντικό byte του αποτελέσματος
111110	-	byte_unpack_mem	base_addr = RF[rs] + SignExtend(Imm) MEM[base_addr] ←SignExtend (RF[rd] (7downto0)) MEM[base_addr + 4] ←SignExtend (RF[rd] (15 downto 8)) MEM[base_addr +8] ←SignExtend (RF[rd] (23downto16)) MEM[base_addr + 12] ←SignExtend (RF[rd] (31 downto 24))	Αντίθετη της byte_pack_mem, αποθηκεύει τα τέσσερα bytes του rd σε τέσσερις διαδοχικές θέσεις μνήμης (με την απαραίτητη επέκταση προσήμου). Το byte της μικρότερης διεύθυνσης αντιστοιχεί στο λιγότερο σημαντικό byte του αποτελέσματος

Διεξαγωγή

A. Μελετήστε την κωδικοποίηση των custom εντολών

Μελετήστε την κωδικοποίηση των εντολών και ορίστε για κάθε μία από αυτές το format στο οποίο ανήκει.

B. Επεκτείνετε την λειτουργικότητα του CHARIS

B1. Υλοποίηση του datapath του επεξεργαστή ECHARIS

Μελετήστε το datapath που υλοποιήσατε στο προηγούμενο εργαστήριο. Βρείτε και υλοποιήστε τις αλλαγές που απαιτούνται στο datapath, πχ. πολυπλεκτών, καταχωρητών, μετρητών, για την υλοποίηση των παραπάνω εντολών αλλά και όλων των εντολών του προηγούμενου εργαστηρίου.

ΠΡΟΣΟΧΗ!!! Για όλες τις εντολές θα χρησιμοποιήσετε τα βασικά modules που υλοποιήσατε στα προηγούμενα εργαστήρια.

B2. Μελέτη και υλοποίηση του control του επεξεργαστή ECHARIS

Παρατηρήστε ξανά όλο το Control που κατασκευάσατε από το προηγούμενο εργαστήριο και κάντε τις αλλαγές-προσθήκες που χρειάζονται για να ελέγξετε τη ροή εκτέλεσης της κάθε νέας εντολής.

ΠΡΟΣΟΧΗ!!! Μετά τις αλλαγές στο datapath του ECHARIS ίσως χρειαστεί να γίνουν ξανά αλλαγές στο control του επεξεργαστή σας για την σωστή υλοποίηση των εντολών του προηγούμενου εργαστηρίου.

Τέλος, ενώστε το datapath του ECHARIS με το control που υλοποιήσατε και προσομοιώστε την σωστή λειτουργία όλων των εντολών.

Εκτέλεση

1. **ΠΡΟΣΟΧΗ!!!!! Πριν από οποιαδήποτε αλλαγή στον κώδικα του προηγούμενου εργαστηρίου κρατήστε ένα backup του κώδικα του Εργαστηρίου 3 διότι θα τον χρειαστούμε σε επόμενο εργαστήριο.**
2. Για τον έλεγχο της ορθής λειτοργίας του συγκεκριμένου εργαστηρίου θα πρέπει να λειτουργούν ορθά **τουλάχιστον οι εντολές li και sw** από το προηγούμενο εργαστήριο.
3. Αλλάξτε τον κώδικα VHDL που υλοποιεί το datapath του επεξεργαστή CHARIS ώστε να μπορεί να υλοποιεί τις νέες εντολές.
4. Προσθέστε ό,τι επιπλέον λογική χρειαστείτε στο datapath (καταχωρητές, μετρητές, πολυπλέκτες κτλ.).
5. Αλλάξτε τον κώδικα VHDL που υλοποιεί την μονάδα ελέγχου του επεξεργαστή CHARIS σύμφωνα με τις νέες ανάγκες και ονομάστε το αρχείο σας: Extended_control.vhd

6. Συνδέστε το datapath με το CONTROL ώστε να υλοποιήσετε την πλήρη λειτουργία ενός επεξεργαστή πολλαπλών κύκλων. Ονομάστε το αρχείο σας: ECHARIS.vhd
7. Επιβεβαιώστε την ορθή λειτουργία του επεξεργαστή δίνοντας πραγματικές εντολές σαν είσοδο και ελέγχοντας τις τιμές των σημάτων εξόδου σε κάθε κατάσταση της FSM.
8. Προσομοιώστε και επιβεβαιώσετε την λειτουργία του επεξεργαστή εκτελώντας εντολές μια-μια (με το χέρι).
9. Προσομοιώστε και επιβεβαιώσετε την λειτουργία του επεξεργαστή **εκτελώντας το πρόγραμμα που σας δόθηκε και που χρησιμοποιεί το extended instruction set του CHARIS (test.data)** με το οποίο θα αρχικοποιήσετε την IMEM.

Παραδοτέα

1. Αρχεία κώδικα VHDL (πηγαίος).
2. Σχηματικό διάγραμμα της μηχανής πεπερασμένων καταστάσεων του control **σημειώνοντας τις αλλαγές** που κάνατε σε σχέση με το προηγούμενο εργαστήριο.
3. Σχηματικό διάγραμμα του datapath **σημειώνοντας τις αλλαγές** που υλοποιήσατε σε σχέση με το προηγούμενο εργαστήριο.
4. Κυματομορφές προσομοίωσης (καλύψτε όλες τις περιπτώσεις).
5. Σύντομη αναφορά στη διαδικασία σχεδίασης και υλοποίησης (μαζί με σχόλια και ενδεχόμενα προβλήματα που παρατηρήθηκαν για μελλοντική του βελτίωση του εργαστηρίου).

ΚΑΛΗ ΕΠΙΤΥΧΙΑ