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

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2018-19
Εργαστήριο 4

Ολοκλήρωση ενός επεξεργαστή πολλαπλών κύκλων (datapath & FSM Ελέγχου), Προσθήκη εξαιρέσεων

Σκοπός του Εργαστηρίου

Η σχεδίαση της μονάδας Ελέγχου (Control) του Datapath και η ολοκλήρωση ενός επεξεργαστή πολλαπλών κύκλων.

Α. Σχεδιασμός και υλοποίηση της μονάδας ελέγχου (65%)

Σχεδιάστε την Μηχανή Πεπερασμένων Καταστάσεων (FSM) που να γεννά τα επιθυμητά σήματα ελέγχου σε κάθε κύκλο ρολογιού ανάλογα με την εντολή που εκτελείται. Η FSM έχει σαν είσοδο την εντολή (opcode και func), πιθανώς flags όπως το Zero, κ.λ.π., και σαν εξόδους όλα τα σήματα ελέγχου του Datapath.

Παρατηρήστε το datapath που κατασκευάσατε στο προηγούμενο εργαστήριο και σχεδιάστε τη μονάδα ελέγχου που χρειάζεται για να ελέγξετε τη ροή εκτέλεσης της κάθε εντολής. Θα χρειαστεί να προσθέσετε καταχωρητές μεταξύ των βαθμίδων ώστε να κρατάτε τις τιμές για κάποια σήματα που παράγονται από μια βαθμίδα και πρέπει να χρησιμοποιηθούν σε επόμενη κατάσταση σε την επόμενη βαθμίδα.

Β. Υλοποίηση Εξαιρέσεων (35%)

Προσθέστε εξαιρέσεις στον επεξεργαστή σας. Παρόμοια με τον MIPS, χρησιμοποιούνται δύο επιπλέον καταχωρητές, ο EPC (που κρατάει την διεύθυνση της εντολής η οποία παρήγαγε το exception) και ο Cause Register (ο οποίος αποθηκεύει το λόγο της εξαίρεσης). Τα exceptions που καλείστε να υλοποιήσετε, οι αντίστοιχες διευθύνσεις στις οποίες θα μεταβαίνει η εξαιρέση του προγράμματος όταν αυτά συμβούν, καθώς και οι τιμές του cause register παρουσιάζονται στον ακόλουθο πίνακα:

Ονομα	Αιτιολογία	Διεύθυνση handler	Τιμή Cause Register
ILLEGAL INSTR	Άγνωστη εντολή (Άγνωστο Opcode/Func)	0x030	0x00000111
ADDR_RANGE	Λανθασμένη διεύθυνση ανάγνωσης (*τελική* διεύθυνση λέξης εκτός της ζώνης [0..2K), η αντίστοιχα διεύθυνση byte εκτός της ζώνης [0..8K).	0x040	0x00111000

Για να είναι εφικτή η επιστροφή από τον exception handler και η συνέχιση του προγράμματος, καθώς και ο έλεγχος του λόγου της εξαίρεσης προστέθηκαν οι εντολή jump_epc και move_cause που φαίνονται στον ακόλουθο πίνακα:

Opcode	FUNC	ΕΝΤΟΛΗ	ΠΡΑΞΗ
000010	-	jump_epc	PC ← EPC + 4 (I-type)
100000	000010	move_cause	rd ← Cause (R-type)

Ο EPC και ο Cause Register είναι καταχωρητές 32 bit που θα βρίσκονται στη βαθμίδα Decode. Στην περίπτωση που συμβεί μια εξαίρεση, η τιμή του PC της εντολής που προκάλεσε το exception γράφεται στον καταχωρητή EPC, ο κωδικός της εξαίρεσης που προκλήθηκε γράφεται στον Cause Register, και το PC να πάρει την κατάλληλη διεύθυνση του handler σύμφωνα με τον παραπάνω πίνακα.

Εκτέλεση

1. Επεκτείνετε το datapath του επεξεργαστή που υλοποιήσατε στο 3ο Εργαστήριο. Προσθέστε ό,τι επιπλέον λογική χρειαστείτε στο datapath (καταχωρητές, κτλ).
2. Γράψτε τον κώδικα VHDL που υλοποιεί την μονάδα ελέγχου του datapath. Ονομάστε το αρχείο σας CONTROL.vhd
3. Συνδέστε το datapath με το CONTROL ώστε να υλοποιήσετε την πλήρη λειτουργία ενός επεξεργαστή πολλαπλών κύκλων. Ονομάστε το αρχείο σας processor.vhd. Η κύρια μνήμη θα πρέπει να βρίσκεται εκτός αυτού του module.
4. Επιβεβαιώστε την ορθή λειτουργία του επεξεργαστή δίνοντας πραγματικές εντολές σαν είσοδο και ελέγχοντας τις τιμές των σημάτων εξόδου σε κάθε κατάσταση της FSM.

Παραδοτέα

1. Αρχεία κώδικα VHDL (πηγαίος).
2. Σχηματικό διάγραμμα του ολοκληρωμένου datapath δείχνοντας τις μεταξύ των βαθμίδων συνδέσεις καθώς και διάγραμμα της μηχανής πεπερασμένων καταστάσεων του control path.
3. Προσομοίωση του κυκλώματος σας. Πρέπει να είστε σε θέση να κάνετε επαλήθευση των αποτελεσμάτων που βλέπετε στο simulation σε σχέση με τα αναμενόμενα.
4. Σύντομη αναφορά στη διαδικασία σχεδίασης και υλοποίησης (μαζί με σχόλια και ενδεχόμενα προβλήματα που παρατηρήθηκαν για μελλοντική του βελτίωση του εργαστηρίου).

Ενδεικτικό **πρόγραμμα ελέγχου εξαίρεσης**

```
00: addi r5, r0, 8
04: // illegal opcode (111100) => exception, goto handler at addr 0x30
08: sw   r5, 4(r0)      // executed after exception handler:  στην διεύθυνση 0x4 => 0x404 το περιεχόμενο του r5 (0x111 από το cause)
0C: lw  r10, 8192(r0)  // address out of range => exception, goto handler at addr 0x40
10: nand r0, r0, r0
14: nand r0, r0, r0
…
2C: nand r0, r0, r0
30: // First instruction of illegal opcode handler
    move_cause r5  // copy cause into r5 (was 8)
34: jump_epc       // return to *next* instruction from the exception
38: nand r0, r0, r0
3C: nand r0, r0, r0
40: // First instruction of Memory Address Error handler
    jump_epc       // do nothing, just return to *next* instruction from the exception
44: nand r0, r0, r0
48: nand r0, r0, r0
```

Ροή εκτέλεσης: PC (σε δεκαεξαδικό): 00, 04, ↯30, 34, 08, 0C, ↯40, 10, 14, … όπου ↯σημαίνει εξαίρεση και μετάβαση στον handler

Εντολή	Opcode	rs	rd	rt	func	Immed	Binary	Hex
00: addi r5,r0,8	110000	00000	00101	-	-	0000000000001000	1100 00|00 000|0 0101| 0000000000001000	C005 0008
04: illegal	111100	00000	00000	-	-	0000000000000000	1111 00|00 000|0 0000| 0000000000000000	F000 0000
08: sw r5,4(r0)	011111	00000	00101	-	-	0000000000000100	0111 11|00 000|0 0101| 0000000000000100	7C05 0004
0C: lw r10,8192(r0)	001111	00000	01010	-	-	0010000000000000	0011 11|00 000|0 1010| 0010000000000000	3C0A 2000
10: nand r0,r0,r0	100000	00000	00000	00000	110010	-	1000 00|00 000|0 0000| 0000 0|000 00|11 0010	8000 0032
14: nand r0,r0,r0	100000	00000	00000	00000	110010	-	1000 00|00 000|0 0000| 0000 0|000 00|11 0010	8000 0032
… MORE nand …								
2C: nand r0,r0,r0	100000	00000	00000	00000	110010	-	1000 00|00 000|0 0000| 0000 0|000 00|11 0010	8000 0032
30: move_cause r5	100000	-	00101	-	000010	-	1000 00|00 000|0 0101| 0000 0|000 00|00 0010	8005 0002
34: jump_epc	000010	-	-	-	-	-	0000 10|00 000|0 0000| 0000000000000000	0800 0000
38: nand r0,r0,r0	100000	00000	00000	00000	110010	-	1000 00|00 000|0 0000| 0000 0|000 00|11 0010	8000 0032
3C: nand r0,r0,r0	100000	00000	00000	00000	110010	-	1000 00|00 000|0 0000| 0000 0|000 00|11 0010	8000 0032
40: jump_epc	000010	-	-	-	-	-	0000 10|00 000|0 0000| 0000000000000000	0800 0000
44: nand r0,r0,r0	100000	00000	00000	00000	110010	-	1000 00|00 000|0 0000| 0000 0|000 00|11 0010	8000 0032
48: nand r0,r0,r0	100000	00000	00000	00000	110010	-	1000 00|00 000|0 0000| 0000 0|000 00|11 0010	8000 0032