

# 4ο Εργαστήριο Αρχιτεκτονικής Η/Υ: Προσομοίωση και διόρθωση επεξεργαστή

Α. Ευθυμίου

Παραδοτέο: Παρασκευή 28 Νοέμβρη 2021, 23:59

Ο σκοπός της άσκησης είναι η εμβάθυνση της κατανόησης λειτουργίας ενός απλού επεξεργαστή, χρησιμοποιώντας τα εργαλεία Quartus και Modelsim. Αυτό επιτυγχάνεται με προσομοίωση προγραμμάτων, παρακολουθώντας πώς αλλάζουν τα διάφορα σήματα του επεξεργαστή, ανάλογα με την εντολή που εκτελείται κάθε φορά, και επιβεβαιώνοντας ότι τα αποτελέσματα είναι πάντα σωστά. Θα πρέπει να τρέξετε ένα πρόγραμμα στον επεξεργαστή και, ελέγχοντας τα αποτελέσματα της εκτέλεσης, να βρείτε και να διορθώσετε τα δύο σφάλματα που έχει.

Θα πρέπει να έχετε μελετήσει τα μαθήματα για την υλοποίηση του MIPS σε ένα κύκλο ρολογιού που αντιστοιχούν μέχρι την ενότητα 4.4 του βιβλίου. Πρέπει επίσης να κάνετε μια γρήγορη επανάληψη στη γλώσσα VHDL, του σχεδιαστικού εργαλείου Quartus, και να θυμάστε βασικές αρχές ψηφιακής σχεδίασης.

Τρέχοντας ένα πρόγραμμα για να επιβεβαιώσετε ότι ο επεξεργαστής δουλεύει σωστά, θα αναγκαστείτε να εξετάσετε τα περισσότερα, αν όχι όλα, τα σήματα δεδομένων και ελέγχου και θα διερευνήσετε ακραίες περιπτώσεις λειτουργίας, τις οποίες ο επεξεργαστής θα πρέπει να χειρίζεται σωστά. Έτσι θα καταλάβετε σε βάθος πως δουλεύει. Θα χρησιμοποιήσετε τον MARS ως το λεγόμενο Golden model: το μοντέλο που θεωρούμε ότι είναι πάντα σωστό και με αυτό συγκρίνουμε τα αποτελέσματα της προσομοίωσης. Αν παίρνετε τις ίδιες τιμές με τον MARS, ο επεξεργαστής έχει υλοποιηθεί σωστά. Διαφορετικά, θεωρείτε ότι ο MARS «έχει δίκιο» και το λάθος είναι στο μοντέλο VHDL του επεξεργαστή.

Σας δίνεται ένας ολόκληρος επεξεργαστής MIPS, σχεδόν ίδιος με αυτόν του συγγράμματος που παρουσιάστηκε στις διαλέξεις. Επιπλέον της jump, που περιγράφεται στο σύγγραμμα, έχουν προστεθεί οι εντολές lui, ori. Έχει επίσης υλοποιηθεί και η addiu, αλλά ως όμοια της addi ώστε να λειτουργούν οι ψευτοεντολές la, li. Δεν είναι αυτό ένα από τα δύο λάθη! Για να πάρετε τον κώδικα, δημιουργήστε το αποθετήριο σας στο GitHub, ακολουθώντας το σύνδεσμο

<https://classroom.github.com/a/TBC>

## 1 Ο επεξεργαστής στο Quartus

Ξεκινήστε το Quartus σύμφωνα με το σύστημά σας. Για οδηγίες εγκατάστασης του Quartus, δείτε το φυλλάδιο εργαστηρίου 0. Όλα τα προβλήματα σχετικά με Quartus-modelsim συγκεντρώνονται σε ένα piazza post.

Όσοι δεν έχετε χρησιμοποιήσει το Quartus, δείτε το εισαγωγικό μέρος του εργαστηρίου του μαθήματος Ψηφιακή Σχεδίαση 2 στο ecourse. Στο ίδιο μάθημα υπάρχουν και σημειώσεις για VHDL.

Ανοίξτε το Quartus project με όνομα, mips.qpf, που υπάρχει έτοιμο. Κάντε διπλό κλικ στο mips για να δείτε το σχηματικό του επεξεργαστή. Εξερευνήστε το σχέδιο, δείτε τα περιεχόμενα των διαφόρων block/symbols και παρατηρήστε καλά τα ονόματα των σημάτων, γιατί θα τα χρειαστείτε για την μετέπειτα προσομοίωση.

Προσπάθησα να κρατήσω τα ονόματα των σημάτων ίδια με αυτά του συγγράμματος, αλλά σε κάποιες περιπτώσεις οι κανόνες ονομάτων του Quartus με υποχρέωσαν να αλλάξω λίγο κάποια ονόματα. Στο σχηματικό υπάρχουν κάποια σήματα ελέγχου που δεν χρησιμοποιούνται στον επεξεργαστή, ενώ άλλα (εισόδου σε κάποιους πολυπλέκτες) έχουν επιπλέον bits. Αυτό έχει γίνει για να μπορούν να προστεθούν επιπλέον εντολές, αλλά σε αυτό το εργαστήριο δεν θα χρησιμοποιηθούν. Επίσης πρόσθεσα ονόματα σε διάφορα καλώδια γιατί αλλιώς παίρνουν αυτόματα κάποια ονόματα και μετά είναι δύσκολο να τα παρακολουθήσει κανείς.

### 1.1 Μετατροπή σχηματικού σε VHDL

Ο προσομοιωτής (modelsim) δεν μπορεί να χειριστεί σχηματικά απευθείας, οπότε πρέπει πρώτα να μετατρέψουμε τα σχηματικά σε VHDL. Με το tab του σχηματικού επιλεγμένο η μετατροπή αυτή γίνεται με: File > Create/Update > Create HDL Design File from Current File. Ελέγξτε το όνομα αρχείου που θα δημιουργηθεί και δώστε VHDL ως File type. Μετά πατήστε OK και περιμένετε μέχρι να ολοκληρωθεί η μετατροπή. Θα εμφανιστεί ένα καινούριο tab με τα αποτελέσματα, που, αν όλα πάνε καλά, δε θα χρειαστείτε άλλο και μπορείτε να κλείσετε. **Προσοχή αν αλλάξετε ένα σχηματικό, θα πρέπει να κάνετε ξανά τη μετατροπή πριν την προσομοίωση. Αλλιώς ο προσομοιωτής, που γνωρίζει μόνο τα αρχεία VHDL, θα χρησιμοποιεί την προηγούμενη έκδοση του κυκλώματος.** Όπως πάντα, αν κατά τη μετατροπή σχηματικού σε VHDL δείτε μηνύματα λάθους, θα πρέπει να διορθώσετε το κύκλωμα πριν συνεχίσετε.

## 2 Μετατροπή προγραμμάτων σε γλώσσα μηχανής

Για να προσομοιώσετε τον επεξεργαστή θα χρειαστείτε προγράμματα σε γλώσσα μηχανής και αρχικές τιμές στη μνήμη δεδομένων. Οι οντότητες dmem και imem, οι μνήμες δεδομένων και εντολών του επεξεργαστή, έχουν τη δυνατότητα να φορτώσουν τα περιεχόμενα των αρχείων data\_memfile.dat και instr\_memfile.dat στις αντίστοιχες μνήμες. Αν ανοίξετε τα αρχεία αυτά (όταν υπάρχουν) θα δείτε δεκαεξαδικούς αριθμούς.

Τα αρχεία αυτά δημιουργούνται με τον MARS. Ανοίξτε το αρχείο lab04.asm στον Mars. Παρατηρήστε ότι λείπει το συνηθισμένο τέλος προγραμμάτων assembly με την εντολή syscall. Αυτό γίνεται γιατί δεν είναι υλοποιημένη η εντολή syscall στον επεξεργαστή. Πρέπει να θυμάστε ότι και στα υπόλοιπα προγράμματά που θα τρέχουν στον επεξεργαστή μέσω προσομοίωσης VHDL δεν θα πρέπει να έχουν αυτή την εντολή.

Μετατρέψτε το σε γλώσσα μηχανής πατώντας το κουμπί “Assemble” (F3). Τώρα μπορείτε να πάρετε το αντίστοιχο πρόγραμμα σε γλώσσα μηχανής, αλλά σε 2 χωριστά αρχεία: για εντολές και δεδομένα. Από το μενού File, επιλέξτε Dump Memory. Θα εμφανιστεί ένα μικρό παράθυρο. Μπορείτε να επιλέξετε μεταξύ “.text” για το πρόγραμμα και “.data” για τα δεδομένα. Η μορφή που μπορεί να διαβαστεί από τον προσομοιωτή είναι “Hexadecimal Text” Τα ονόματα αρχείων πρέπει να είναι data\_memfile.dat για δεδομένα και instr\_memfile.dat για εντολές. Μη χρησιμοποιήσετε άλλα ονόματα. Τα imem.vhd, dmem.vhd περιμένουν τα συγκεκριμένα αρχεία για να δουλέψουν.

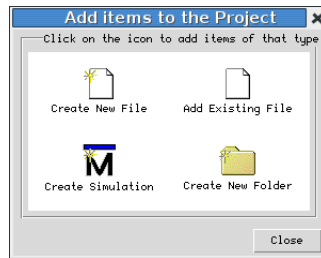
Για κάποιο λόγο (**bug**) το αρχείο δεδομένων (data\_memfile.dat) που γράφει το Mars περιέχει 1024 γραμμές. Αργότερα, αυτό μπορεί να προκαλέσει πρόβλημα στην προσομοίωση, με το modelsim εμφανίζει ένα σχετικό error. Για να το αποφύγετε, ανοίξτε το data\_memfile.dat με έναν απλό κειμενογράφο, σβήστε τις επιπλέον γραμμές, π.χ. από την γραμμή 50 και κάτω, και αποθηκεύστε το με το ίδιο όνομα.

**Προσοχή** κάντε το Dump Memory της μνήμης δεδομένων **πριν** τρέξετε το πρόγραμμα στον MARS. Διαφορετικά θα γράφει στο αρχείο τα δεδομένα όπως έχουν αλλαχθεί από την εκτέλεση του προγράμματος!

Στα Windows, υπάρχει και ένα άλλο πρόβλημα σχετικό με αυτά τα αρχεία. Εξηγείται παρακάτω στην Προσομοίωση 3.3.

## 3 Προσομοίωση του επεξεργαστή

Αφού έχετε δημιουργήσει τα αρχεία που φορτώνονται στις μνήμες, μπορείτε να τρέξετε το πρόγραμμα με προσομοίωση στον επεξεργαστή. Το όνομα του εκτελέσιμου είναι vsim και σε Linux χρειάζεται να του υποδείξετε το path για τις δυναμικές βιβλιοθήκες πριν ξεκινήσει: σε bash αυτό γίνεται με `export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${QUARTUS_TOP}/modelsim_ase/lib32`, με το QUARTUS\_TOP το path που εγκαταστήσατε το quartus. Σε Windows θα πρέπει να βρείτε που βρίσκεται το αντίστοιχο πρόγραμμα.



Σχήμα 1: Το παράθυρο για το νέο ModelSim project

### 3.1 Δημιουργία ModelSim project

Από το μενού, επιλέξτε File > New > Project... Δώστε το όνομα mips\_sim στο project, αφήστε τις υπόλοιπες επιλογές ως έχουν και πατήστε OK.

**Σημείωση:** Με τις τυπικές ρυθμίσεις το Modelsim «θυμάται» πάντα το προηγούμενο Modelsim project. Αυτό είναι βολικό όταν έχει κανείς ένα μόνο project, αλλά θα πρέπει να το προσέξετε στις επόμενες ασκήσεις ή αν τρέξετε δικές σας προσομοιώσεις. Αν θέλετε να αλλάξετε project, κλείστε το παλιό με File > Close Project και ανοίξτε νέο. Επειδή θυμάται και τον προηγούμενο κατάλογο από όπου το τρέξατε (Project Location), προσέξτε να δώσετε το σωστό, νέο κατάλογο.

Θα εμφανιστεί ένα μικρό παράθυρο σαν αυτό του σχήματος 1. Αφού υπάρχουν ήδη τα αρχεία για το κύκλωμα και το testbench, επιλέξτε Add Existing File. Στο νέο παράθυρο που εμφανίζεται (συνήθως πίσω από το προηγούμενο και ίσως να μην το προσέξετε αμέσως), πατήστε το Browse.. Επιλέξτε όλα τα αρχεία VHDL (.vhd), κρατώντας το πλήκτρο Ctrl μαζί με κλικ για να διαλέξετε πολλαπλά αρχεία. Πατήστε OK για την επιλογή αρχείων και ξανά OK για το παράθυρο Add Existing File. Θα δείτε τα ονόματα των αρχείων να εμφανίζονται στο επάνω τμήμα του παραθύρου του ModelSim. Μπορείτε τώρα να κλείσετε το παράθυρο Add items to the Project.

### 3.2 Compilation

Σε αυτό το βήμα ελέγχονται τα αρχεία VHDL για λάθη και δημιουργούνται κάποια προσωρινά δεδομένα που χρειάζονται για τη προσομοίωση. Από το μενού, επιλέξτε Compile > Compile All.

Στα αρχικά αρχεία του αποθετηρίου δεν θα συμβεί, αλλά, όταν κάνετε αλλαγές αργότερα, αν δείτε λάθη σε αυτό το βήμα, θα πρέπει να τα διορθώσετε πριν προχωρήσετε. Η διόρθωση μπορεί να γίνει στον Editor του Quartus, αν το πρόβλημα είναι σε αρχεία VHDL, ή αλλάζοντας το σχηματικό, για αρχεία που προέρχονται από σχηματικό. Στη δεύτερη περίπτωση θα πρέπει να ανανεωθεί το αντίστοιχο αρχείο VHDL από το αλλαγμένο σχηματικό όπως περιγράφηκε προηγουμένως.

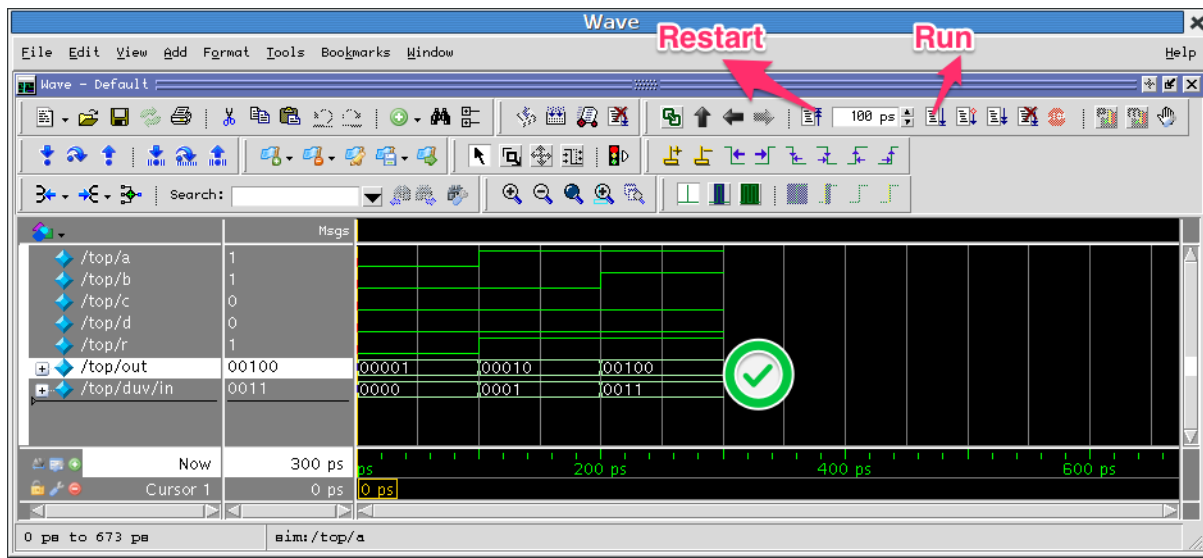
### 3.3 Προσομοίωση

Για να ξεκινήσετε την προσομοίωση, από το μενού επιλέξτε Simulate > Start Simulation... Θα εμφανιστεί ένα παράθυρο με πολλές «βιβλιοθήκες» (libraries). Βρείτε τη βιβλιοθήκη **work** που ήταν η προεπιλογή όταν είχατε δημιουργήσει το ModelSim project. Πατήστε το + δίπλα στη παραπάνω βιβλιοθήκη και θα δείτε τα περιεχόμενά της. Επιλέξτε το top. Αυτό είναι η «υψηλότερη» VHDL οντότητα που περιλαμβάνει τις υπόλοιπες, κάτι σαν την main στη Java. Πατήστε OK.

Θα δείτε ότι το παράθυρο του ModelSim αλλάζει μορφή. Πάνω αριστερά θα δείτε ένα παράθυρο που ονομάζεται Sim και στα δεξιά του άλλα δύο παράθυρα με ονόματα Objects και Processes.

Το παράθυρο Sim περιέχει την ιεραρχία των οντοτήτων. Συγκεκριμένα θα δείτε πρώτα το top, από κάτω του το dun (device under verification), που αντιστοιχεί στον mips, κλπ. Κάνοντας κλικ σε μία οντότητα εμφανίζονται, στο παράθυρο Objects, τα ονόματα των σημάτων που υπάρχουν μέσα σε αυτό.

Στην προσομοίωση θέλουμε να μπορούμε να βλέπουμε τις τιμές που έχουν διάφορα σήματα, είσοδοι, έξοδοι και εσωτερικοί κόμβοι, ώστε να μπορούμε να καταλάβουμε αν το κύκλωμα δουλεύει σωστά ή



Σχήμα 2: Το παράθυρο κυματομορφών.

να βρούμε την αιτία του λάθους. Επειδή οι τιμές των σημάτων μεταβάλλονται με το χρόνο, οι αναπαραστάσεις των τιμών τους λέγονται κυματομορφές, **waveforms**.

Πριν ξεκινήσει η προσομοίωση, πρέπει να πούμε στον προσομοιωτή, για ποιά σήματα να δημιουργήσει κυματομορφές. Αυτό γίνεται επιλέγοντας το όνομα του σήματος στο παράθυρο Objects και από το μενού **Add > To Wave > Selected Signals**, ή με δεξιά κλικ και αντίστοιχες επιλογές. Μπορείτε να επιλέξετε πολλά σήματα μαζί πατώντας το **Ctrl** όταν κάνετε κλικ στο όνομα σήματος. Αν θέλετε να προσθέσετε κυματομορφές από σήματα που βρίσκονται σε άλλη οντότητα, προσθέστε τα σήματα του ενός, διαλέξτε την άλλη οντότητα από το παράθυρο **sim** και μετά συνεχίστε τη διαδικασία όπως παραπάνω.

Για ευκολία, σας δίνεται ένα αρχείο με τα κύρια σήματα: **wave.do**. Για να το χρησιμοποιήσετε, επιλέξτε **File > Load > Macro File** και στο παράθυρο που θα εμφανιστεί διαλέξτε το παραπάνω αρχείο. Μπορείτε να αλλάξετε θέση στα σήματα τραβώντας τα με το ποντίκι. Επίσης μπορείτε να τοποθετείτε διαχωριστικά (όπως στο **wave.do**) με **Add > Divider**

Θα δείτε ένα καινούριο παράθυρο, **Wave** να εμφανίζεται δίπλα στο **Objects**. Επειδή αυτό το παράθυρο θέλουμε να το αλλάζουμε θέση και μέγεθος ελεύθερα, πατήστε το κουμπί **Undoc** στη πάνω δεξιά μεριά του ώστε να γίνει ξεχωριστό παράθυρο από αυτό του **ModelSim**.

Το παράθυρο **Wave** έχει στα αριστερά τα ονόματα των σημάτων που διαλέξαμε, ένα χώρο στη μέση όπου θα εμφανίζονται οι τιμές των σημάτων και ένα μεγάλο μέρος δεξιά που θα εμφανιστούν οι κυματομορφές αλλά, για την ώρα, είναι άδειο.

Για να τρέξει η προσομοίωση βρείτε πάνω δεξιά τα εικονίδια που μοιάζουν με φύλλα χαρτιού με μερικές γραμμές πάνω τους και ένα βέλος δίπλα στο χαρτί. Πατήστε το εικονίδιο **Run** (ή πατήστε **F9**). Θα δείτε ότι οι κυματομορφές αρχίζουν να εμφανίζονται. Επειδή το προκαθορισμένο βήμα προσομοίωσης είναι 100ps, αλλάξτε το σε 200ns για να "τρέχει" γρηγορότερα. Πατήστε το αρκετές φορές ακόμη μέχρι να σιγουρευτείτε ότι οι κυματομορφές, εκτός του ρολογιού (**clk**) και του **pc**, δεν αλλάζουν άλλο και τα περισσότερα σήματα γίνονται κόκκινα, δηλαδή παίρνουν άγνωστες τιμές (συμβολίζονται με **X**), περίπου μέχρι 7000ns. Θα πρέπει να βλέπετε κάτι σαν το σχήμα 2, αλλά με πολύ περισσότερες κυματομορφές.

**Προσοχή**, φαίνεται πως στα **Windows** το **Modelsim** τρέχει την προσομοίωση σε κατάλογο διαφορετικό από αυτό που βρίσκονται τα αρχεία του **lab04**. Έτσι δεν βρίσκει το **instr\_memfile.dat** και το **data\_memfile.dat** που περιέχουν τα αρχικά δεδομένα της μνήμης εντολών (το πρόγραμμα σε γλώσσα μηχανής) και μνήμης δεδομένων (αρχικές τιμές). Αν δείτε στο παράθυρο με τα μηνύματα του

προσομοιωτή (το κάτω μέρος του Modelsim) Error (vsim-7) που αναφέρουν ότι δεν βρέθηκε κάποιο από τα αρχεία XX\_memfile.dat, τότε κάντε τα παρακάτω:

Στα αρχεία imem.vhd dmem.vhd αλλάξτε τα ονόματα των .dat αρχείων ώστε να έχουν το πλήρες μονοπάτι των αρχείων instr\_memfile.dat και data\_memfile.dat. Ίσως να πρέπει να χρησιμοποιήσετε το / του Unix ως το διαχωριστικό ονομάτων καταλόγων και όχι το \ των Windows! Φυσικά θα χρειαστεί ξανά compile αφού κάνατε αλλαγές.

Αν είχατε ξεχάσει να προσθέσετε ένα σήμα στις κυματομορφές, μπορείτε να το κάνετε αργότερα, αλλά θα δείτε ότι δεν θα φαίνεται η κυματομορφή του για το προηγούμενο χρόνο προσομοίωσης, θα εμφανιστεί μόνο για τη συνέχεια της προσομοίωσης. Αν θέλετε να δείτε τη κυματομορφή από την αρχή, πατήστε το εικονίδιο Restart και ξανατρέξτε την προσομοίωση από την αρχή.

Πολύ χρήσιμο είναι το Zoom Full ώστε να δείτε τις κυματομορφές στην πλήρη ανάπτυξή τους. Μετά μπορείτε να κάνετε zoom in ώστε να μπορείτε να βλέπετε καθαρά τις τιμές των σημάτων. Με κλικ σε ένα χρονικό σημείο, εμφανίζεται μια κίτρινη γραμμή και δίπλα στα ονόματα των σημάτων θα βλέπετε τις αντίστοιχες τιμές σε αυτό το χρονικό σημείο. Το παράθυρο κυματομορφών θα μοιάζει με το κάτω μέρος του σχήματος 3, αλλά θα έχει πολύ περισσότερες αλλαγές στις τιμές των σημάτων.

### 3.4 Προσομοίωση επεξεργαστή

Παρατηρήστε ποιοι καταχωρητές διαβάζονται (ra1, ra2), τις τιμές τους (rd1, rd2), την πράξη που κάνει η ALU (aluCtrl), το αποτέλεσμα (result), τον αριθμό καταχωρητή στον οποίο γράφεται το αποτέλεσμα (wr), κλπ. Τα ονόματα των σημάτων στις κυματομορφές είναι τα ίδια με αυτά του σχηματικού του επεξεργαστή στο Quartus. Μπορείτε να βρείτε την κωδικοποίηση μιας εντολής σε γλώσσα μηχανής από το πεδίο Code του Mars, καθώς επίσης και τους αριθμούς (όχι ονόματα) καταχωρητών, από το πεδίο Basic. (Ο κώδικας που γράψατε φαίνεται στο πεδίο Source.)

Στην τελευταία ακμή του ρολογιού, οι περισσότερες κυματομορφές κοκκινίζουν γιατί παίρνουν τιμές X, την ειδική τιμή της VHDL που δείχνει ότι δεν γνωρίζει αν ένα bit είναι 1 ή 0. Αυτό συμβαίνει γιατί η επόμενη εντολή που διαβάζεται από τη μνήμη δεν είναι καθορισμένη και παριστάνεται με X. Ακολούθως τα X εξαπλώνονται σχεδόν σε όλα τα σήματα. Αυτό είναι φυσιολογικό να συμβαίνει στο τέλος του προγράμματος, αλλά αν το δείτε να συμβαίνει κατά τη διάρκεια της εκτέλεσης στα περισσότερα σήματα, κάποιο λάθος θα πρέπει να έχει συμβεί. Μερικές φορές, μεμονωμένα σήματα γίνονται X χωρίς να υπάρχει πρόβλημα. Αυτό γίνεται σε σήματα που δεν είναι χρήσιμα για κάποιες εντολές, για παράδειγμα η τιμή του 2ου καταχωρητή, r1 (σήμα rd2 στις κυματομορφές), όταν αυτός δεν χρησιμοποιείται για ανάγνωση αλλά είναι ο καταχωρητής προορισμού (π.χ. σε μια lw).

Στον κώδικα VHDL του επεξεργαστή έχουν προστεθεί καθυστερήσεις σε βασικά κυκλώματα (π.χ. ALU, αρχείο καταχωρητών, μνήμες). Έτσι σε κάθε ακμή του ρολογιού οι νέες τιμές εμφανίζονται με μια καθυστέρηση.<sup>1</sup> Παρατηρήστε τις τιμές λίγο αργότερα, περίπου στην κατερχόμενη ακμή του ρολογιού, όταν οι τιμές θα έχουν σταθεροποιηθεί. Προσοχή όμως ο πρώτος κύκλος είναι «μισός»: όταν πέσει το σήμα reset στο 0, ο PC παίρνει την τιμή 0 και ξεκινάει την εκτέλεση της πρώτης εντολής, που ολοκληρώνεται πριν την πρώτη ανοδική ακμή του ρολογιού. Από εκεί και έπειτα, κάθε εντολή εκτελείται σε έναν πλήρη κύκλο.

### 3.5 Διαφορές από τον MARS

Οι καταχωρητές, εκτός του 0 (zero) δεν είναι αρχικοποιημένοι και περιέχουν X. Μη στηρίζεστε στο γεγονός ότι έχουν 0 όπως στον MARS.

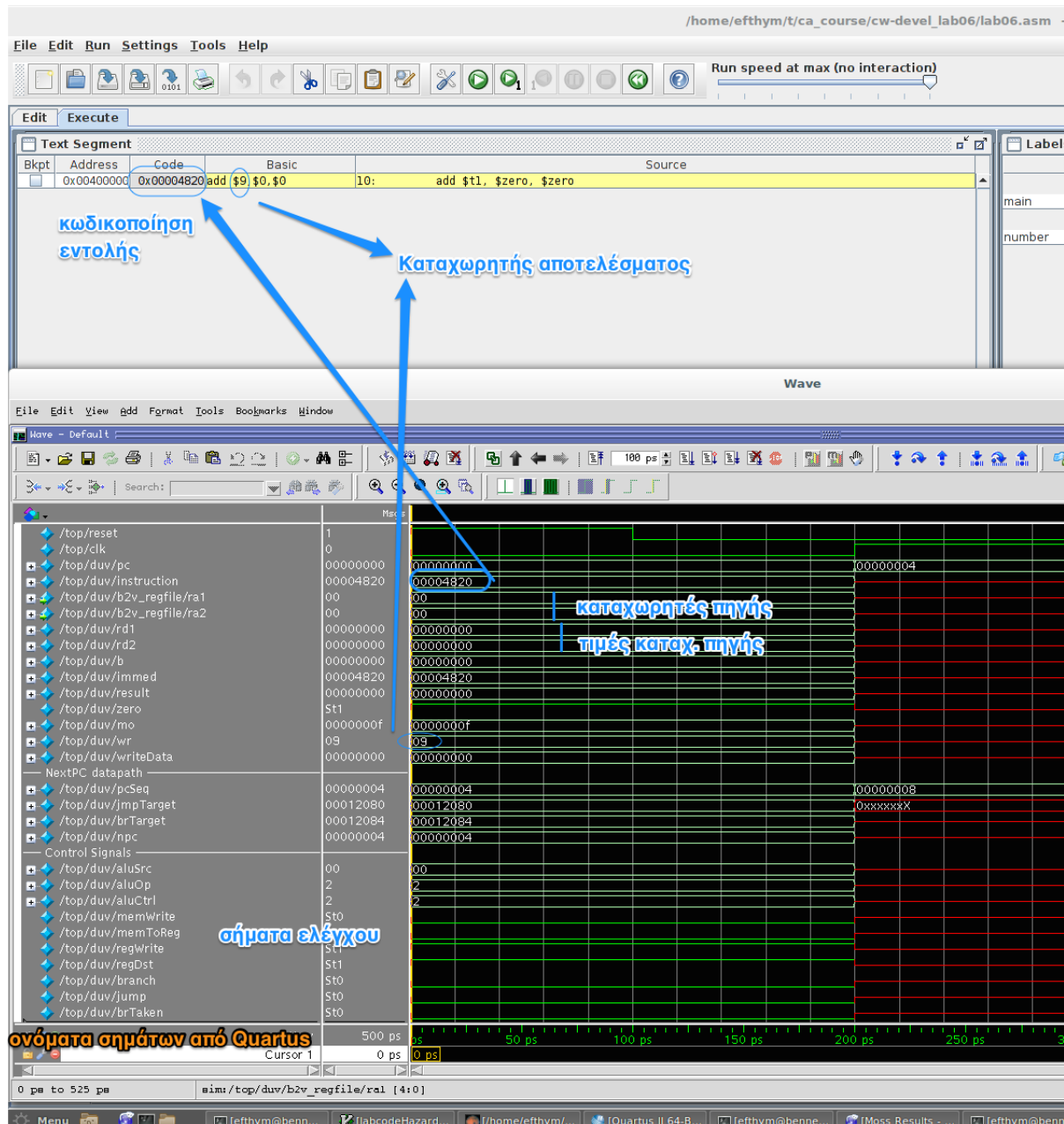
Οι μνήμες έχουν χώρο για 128 εντολές και 128 λέξεις δεδομένων. Θα προσέξετε ότι μόνο τα 9 λιγότερο σημαντικά bit των διευθύνσεων χρησιμοποιούνται (τα 2 τελευταία είναι πάντα 0 γιατί χρησιμοποιούμε μόνο 32 bit λέξεις), έτσι οι τιμές διευθύνσεων που βλέπετε στον MARS είναι κάπως διαφορετικές από αυτές του Modelsim. Για παράδειγμα τα δεδομένα στον MARS ξεκινούν από τη διεύθυνση 0x10010000 ενώ στο Modelsim θα βλέπετε να ξεκινούν από τη διεύθυνση 0. Παρόμοια θα παρατηρήσετε ότι ενώ ο PC ξεκινάει με την τιμή 0, όταν εκτελεστεί η πρώτη j θα γίνει 0x004000030. Τα πιο σημαντικά

<sup>1</sup>Αυτό δεν φαίνεται στο σχήμα 3 γιατί το screenshot πάρθηκε από προηγούμενη έκδοση του κώδικα.

bit των διευθύνσεων δεν συνδέονται πουθενά και έτσι το 0x004000030 θα αντιστοιχεί στη διεύθυνση 0x000000030. Το ίδιο φυσικά θα ισχύει και για τις υπόλοιπες διευθύνσεις που ξεκινούν από 0x004 και παρόμοια για την μνήμη δεδομένων.

#### 4 Η άσκηση: Προσομοίωση προγράμματος και έλεγχος σωστής λειτουργίας του επεξεργαστή

Τώρα που γνωρίζετε τη διαδικασία συγγραφής προγράμματος, μετατροπής του σε γλώσσα μηχανής και προσομοίωσης, ήρθε η στιγμή να επαληθεύσετε τη λειτουργία του επεξεργαστή χρησιμοποιώντας το



Σχήμα 3: Κυματομορφές και Mars.

lab04.asm.

Για την ώρα ο επεξεργαστής υλοποιεί το υποσύνολο των εντολών MIPS: add, sub, and or, slt, lw, sw, beq, j, lui, addi, ori. Υπάρχουν όμως 2 λάθη στην υλοποίηση: κάποιες εντολές έχουν υλοποιηθεί λανθασμένα.

Θα πρέπει να βρείτε τα λάθη κοιτάζοντας τα αποτελέσματα του επεξεργαστή στις κυματομορφές και συγκρίνοντάς τα με τα αποτελέσματα του MARS. Το αποτέλεσμα μιας εντολής εξαρτάται από το είδος της:

- οι αριθμητικές-λογικές εντολές (συμπεριλαμβανομένης και της slt) και η lw (ελέγξτε και τη διεύθυνση αν είναι σωστή), γράφουν αποτέλεσμα σε ένα καταχωρητή
- οι sw γράφουν στην κατάλληλη θέση μνήμης (ελέγξτε τη διεύθυνση, την τιμή του καταχωρητή που αποθηκεύεται, τις τιμές στην μνήμη)
- οι beq, j έχουν ως αποτέλεσμα την αλλαγή ροής προγράμματος, άρα θα πρέπει να ελέγξετε αν η επόμενη εντολή που εκτελείται είναι η σωστή.

Αφού βρείτε τις εντολές που δίνουν λάθος αποτέλεσμα, ψάξτε τον λόγο για τον οποίο γίνεται το λάθος. Ελέγξτε αν οι τιμές εισόδου είναι σωστές. Π.χ. για μια add ελέγξτε τις τιμές των καταχωρητών πηγής, ενώ για μία addi αν η τιμή του καταχωρητή πηγής και της σταθεράς είναι σωστές. Ελέγξτε αν τα αποτελέσματα γράφονται στο σωστό καταχωρητή ή θέση μνήμης. Τέλος ελέγξτε αν κάποιο σήμα ελέγχου είναι λάθος, π.χ. λάθος πράξη, ή δεν γίνεται εγγραφή σε καταχωρητή (το σήμα regWrite είναι 0 αντί για 1). **Προσοχή.** Μερικές φορές το λάθος μπορεί να έχει συμβεί σε προηγούμενη εντολή αλλά το παρατηρείτε σε επόμενη. Να είστε προετοιμασμένοι και γι' αυτό το ενδεχόμενο.

Αφού βρείτε ακριβώς τι φταίει, διορθώστε το. **Δεν θα χρειαστούν αλλαγές στο σχηματικό του επεξεργαστή.** παρά μόνο μικρές αλλαγές σε αρχεία VHDL. Επιβεβαιώστε ότι πλέον ο επεξεργαστής δουλεύει σωστά, ελέγχοντας ότι τα αποτελέσματα είναι πλέον ίδια με τον MARS.

## 5 Παραδοτέο

Το παραδοτέο της άσκησης είναι τα αρχεία VHDL του επεξεργαστή που αλλάξατε για να διορθώσετε το λάθος. Η παράδοση θα γίνει μέσω GitHub, όπως πάντα.

**Προσοχή:** Μην ανεβάσετε στο GitHub κανένα άλλο αρχείο, γιατί πιάνουν πολύ χώρο και υπάρχει κίνδυνος να πετύχετε τα όρια του GitHub για χώρο αποθήκευσης αποθετηρίου. Ειδικά αν χρησιμοποιείτε drag&drop ή upload για παράδοση στο GitHub, μην στείλετε όλο τον κατάλογο lab04-XXX, αλλά τα επιμέρους αρχεία.

## 6 Καθαρισμός αρχείων

Στους υπολογιστές των εργαστηρίων, επειδή ο διαθέσιμος χώρος σας στο δίσκο είναι περιορισμένος (quota), και τα εργαλεία που χρησιμοποιήσατε δημιουργούν πολλά και μεγάλα αρχεία, όταν τελειώσετε με την άσκηση, σβήστε όλα τα περιττά αρχεία. Κρατήστε μόνο τα αρχεία που προϋπήρχαν στο αποθετήριο όταν το κλωνοποιήσατε (με τις αλλαγές τους) και το ModelSim project file (κατάληξη .mpf). Με αυτά τα αρχεία μπορείτε οποιαδήποτε στιγμή να ξαναδείτε τη δουλειά σας στο Quartus ή να ξανακάνετε προσομοιώσεις στο Modelsim.