6ο Εργαστήριο Αρχιτεκτονικής Η/Υ: Προσομοίωση επεξεργαστή

Α. Ευθυμίου

Παραδοτέο: Τρίτη 5 Απρίλη 2016, 23:00

Ο σκοπός αυτής της άσκησης είναι η εμβάθυνση της κατανόησης λειτουργίας ενός απλού επεξεργαστή, χρησιμοποιώντας τα εργαλεία Quartus και Modelsim.

Σας δίνεται ένας ολόκληρος επεξεργαστής σχεδόν ίδιος με αυτόν του συγγράμματος που παρουσιάστηκε στις διαλέξεις. Επιπλέον της jump, που περιγράφεται στο σύγγραμμα, έχουν προστεθεί οι εντολές lui, ori, addi, addiu[1].

Η δουλειά σας είναι να γράψετε ένα πρόγραμμα σε assembly που χρησιμοποιεί όλες τις υλοποιημένες εντολές του επεξεργαστή ώστε να βεβαιωθείτε ότι έχει πράγματι υλοποιηθεί σωστά.

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 Verilog και να θυμάστε βασικές αρχές ψηφιακής σχεδίασης.

Μή ξεχάσετε να επιστρέψετε τα παραδοτέα που αναφέρονται στο τέλος του κειμένου για να πάρετε βαθμό γι'αυτή την εργαστηριακή άσκηση!

1 Η άσκηση

Για να ξεκινήσετε θα χρειαστείτε όλα τα αρχεία του εργαστηρίου δίνοντας τις εξής εντολές:

```
git remote add lab06_starter https://github.com/UoI-CSE-MYY402/lab06_starter.git
git fetch lab06_starter
git merge lab06_starter/master -m "Fetched lab06 starter files"
```

Ξεκινήστε το Quartus σύμφωνα με το σύστημά σας. Ανοίξτε το Quartus project με όνομα, mips.qpf, που υπάρχει έτοιμο. Κάντε διπλό κλίκ 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. Επίσης πρόσθεσα ονόματα σε διάφορα καλώδια γιατί αλλιώς παίρνουν αυτόματα κάποια ονόματα και μετά είναι δύσκολο να τα παρακολουθήσει κανείς.

2 Μετατροπή προγραμμάτων σε γλώσσα μηχανής

Για να προσομοιώσετε τον επεξεργαστή θα χρειαστείτε προγράμματα σε γλώσσα μηχανής και αρχικές τιμές στη μνήμη δεδομένων. Τα modules 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 θα δείτε δεκαεξαδικούς αριθμούς.

Για να δημιουργήσετε τα δικά σας αρχεία θα χρησιμοποιήσετε τον MARS. Ανοίξτε το αρχείο lab06.asm στον Mars:

```
   .data
number:
```

[1] Αντίθετα με τις προδιαγραφές του MIPS, οι addi, addiu έχουν υλοποιηθεί ώστε να είναι ακριβώς όμοιες. Γενικά οι εντολές με απρόσημους αριθμούς δεν έχουν υλοποιηθεί. Η addiu είναι εξαίρεση γιατί χρησιμοποιείται μερικές φορές από τις ψευτοεντολές li, la που μας είναι χρήσιμες.

```
   .word  15

   .globl main

   .text
main:
    add $t1, $zero, $zero
```

Παρατηρήστε ότι υπάρχει μόνο μια γραμμή κώδικα και ότι λείπει το συνηθισμένο τέλος προγραμμάτων assembly με την εντολή syscall. Αυτό γίνεται γιατί δεν είναι υλοποιημένη η syscall στον επεξεργαστή. Πρέπει να θυμάστε ότι και στα υπόλοιπα προγράμματά που θα τρέχουν στον επεξεργαστή 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 τη Verilog 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.v, dmem.v περιμένουν τα συγκεκριμένα αρχεία για να δουλέψουν.

Για κάποιο λόγο (bug) το αρχείο δεδομένων που γράφει το Mars περιέχει 1024 γραμμές. Ο προσομοιωτής αργότερα διαμαρτύρεται, αλλά η προσομοίωση γίνεται κανονικά.

3 Προσομοίωση του επεξεργαστή

Αφού έχετε δημιουργήσει τα αρχεία που φορτώνονται στις μνήμες, μπορείτε να τρέξετε με προσομοίωση στον επεξεργαστή.

Ξεκινήστε τον προσομοιωτή με την εντολή vsim. Αν το Modelsim «θυμάται» το παλιό modelsim project του lab05, κλείστε το με File > Close. Το Modelsim μπορεί να θυμάται και τον προηγούμενο κατάλογο από όπου το τρέξατε, οπότε προσέξτε σε διάφορα παράθυρα/διαλόγους μήπως και δεν έχετε δώσει το σωστό κατάλογο.

Ξεκινήστε ένα νέο modelsim project με όνομα `mips_sim`. Προσθέστε όλα τα αρχεία Verilog (.v) **εκτός από** τα `lpm_constant4_bb.v lpm_constant4_syn.v`.

Κάντε Compile > Compile All

Ξεκινήστε την προσομοίωση: Simulate > Start Simulation. Διαλέξτε το top.v από τη βιβλιοθήκη Work.

Προσοχή, φαίνεται πως στα Windows το Modelsim τρέχει την προσομοίωση σε κατάλογο διαφορετικό από αυτό που βρίσκονται τα αρχεία του lab06. 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). Δυστυχώς, **η μή εύρεση των αρχείων είναι απλά ένα warning για το Modelsim** και έτσι συνεχίζει την προσομοίωση διαβάζοντας x (η τιμή της verilog για μη αρχικοποιημένες τιμές) για εντολές και έτσι όλα τα σήματα γίνονται τελικά x. Αν δείτε το παράθυρο με τα μηνύματα του προσομοιωτή θα δείτε 2 warnings.

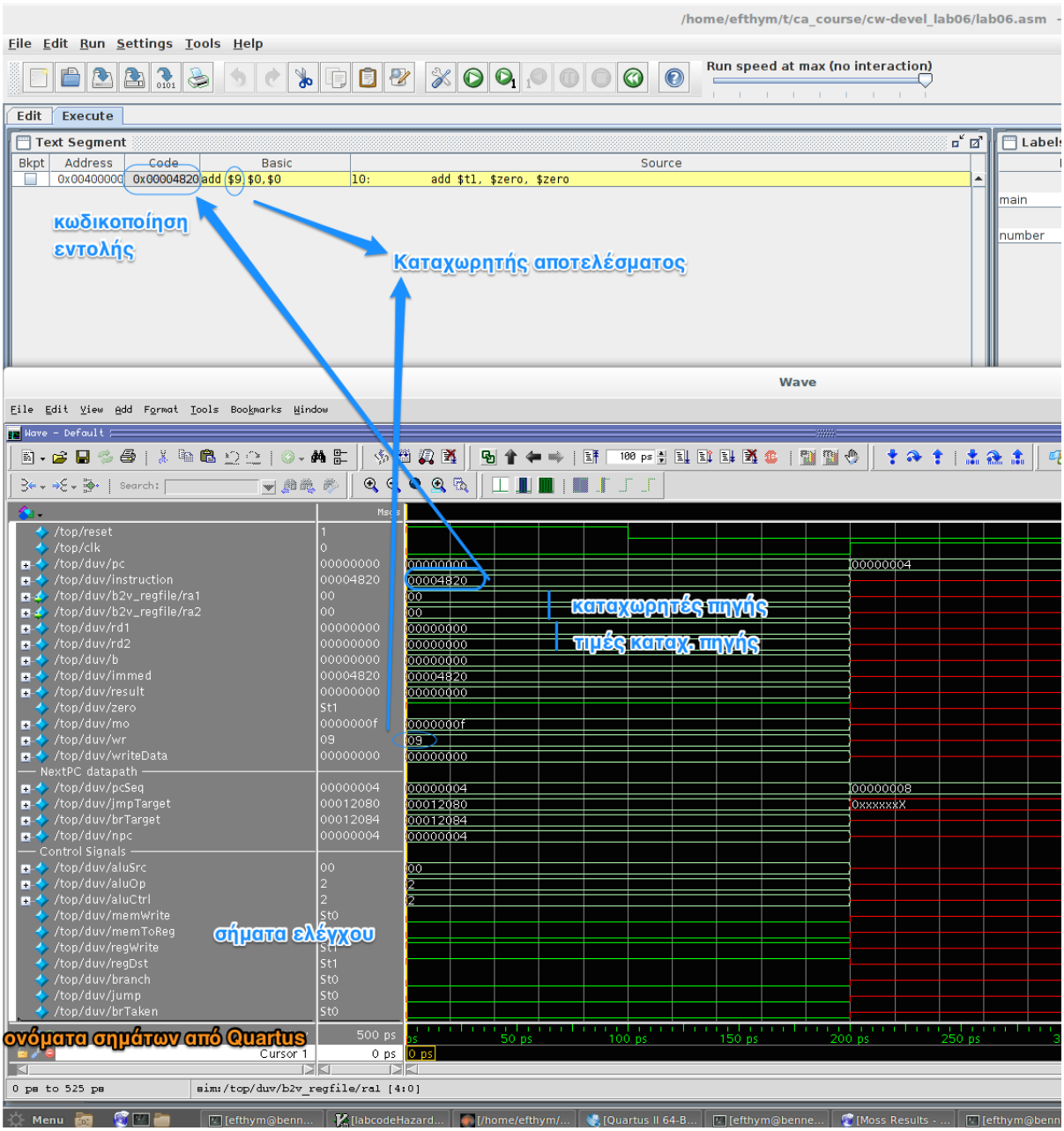
Η λύση είναι να αλλάξετε τις εντολές readmemh στα αρχεία imem.v dmem.v ώστε να έχουν το πλήρες μονοπάτι των αρχείων instr_memfile.dat και data_memfile.dat

Αφού σιγουρευτείτε ότι η προσομοίωση γίνεται κανονικά, επιλέξτε τα σήματα που θέλετε να παρακολουθήσετε στις κυματομορφές. Θα πρέπει να κοιτάτε το σχηματικό στο Quartus για να βρίσκετε τα ονόματα των σημάτων που θέλετε. Για ευκολία, σας δίνεται ένα αρχείο με τα κύρια σήματα: wave.do. Για να το χρησιμοποιήσετε, επιλέξτε File > Load 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

Τρέξτε την προσομοίωση, με το κατάλληλο εικονίδιο ή το F9. Για το παραπάνω πρόγραμμα της μίας εντολής, αρκούν 300ps. Κάντε Zoom-full (μεγεθυντικός φακός με σκούρο εσωτερικό ή πλήκτρο F), για να δείτε καλύτερα τις τιμές των σημάτων. Οι κυματομορφές θα μοιάζουν με το σχήμα 1.

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.) Τα ονόματα των σημάτων στις κυματομορφές είναι τα ίδια με αυτά του σχηματικού του επεξεργαστή στο Quartus.

Παρατηρήστε ποιοι καταχωρητές διαβάζονται (ra1, ra2), τις τιμές τους (rd1, rd2), την πράξη που κάνει



Σχήμα 1: Κυματομορφές και Mars.

η ALU (aluCtrl), το αποτέλεσμα (result), τον αριθμό καταχωρητή στον οποίο γράφεται το αποτέλεσμα (wr), κλπ. Βεβαιωθείτε ότι δουλεύει σωστά.

Στην επόμενη ακμή του ρολογιού, οι περισσότερες κυματομορφές κοκκινίζουν γιατί παίρνουν τιμές X, την ειδική τιμή της Verilog που δείχνει ότι δεν γνωρίζει αν ένα bit είναι 1 ή 0. Αυτό συμβαίνει γιατί η επόμενη εντολή που διαβάζεται από τη μνήμη (PC = 4) 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, σε πιο σύνθετα προγράμματα, 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 συνήθως γίνεται σε σήματα που δεν είναι χρήσιμα για κάποιες εντολές, για παράδειγμα η τιμή του 2ου καταχωρητή, rt (σήμα rd2 στις κυματομορφές), όταν αυτός δεν χρησιμοποιείται για ανάγνωση αλλά είναι ο καταχωρητής προορισμού (π.χ. σε μια lw).

4 Συγγραφή και προσομοίωση προγράμματος επιβεβαίωσης σωστής λειτουργίας

Τώρα που γνωρίζετε τη διαδικασία συγγραφής προγράμματος, μετατροπής του σε γλώσσα μηχανής και προσομοίωσης, ήρθε η στιγμή να γράψετε ένα προγραμμα που επαληθεύει τη λειτουργία του επεξεργαστή. Αλλάξτε το lab06.asm για το σκοπό αυτό.

Για την ώρα ο επεξεργαστής υλοποιεί το υποσύνολο των εντολών MIPS: add, sub, and or, slt, lw, sw, beq, j, lui, addi, addiu, ori, Οπότε θα πρέπει να γράψετε ένα πρόγραμμα που περιλαμβάνει όλες αυτές τις εντολές.

Σκεφτείτε προσεκτικά πώς μπορεί να γίνει αυτό. Για παράδειγμα θα πρέπει να υπάρχουν αρκετές εκτελέσεις εντολών beq: μία που ακολουθείται (branch taken) και μία που δεν ακολουθείται, ώστε να δοκιμαστούν και οι δύο περιπτώσεις. Επιπλέον οι διευθύνσεις προορισμού της beq μπορεί να είναι σε επόμενες εντολές ή σε προηγούμενες.

Επίσης, θα πρέπει να χρησιμοποιήσετε μεγάλη ποικιλία τιμών δεδομένων. Όχι συνέχεια 0 και μικρούς, θετικούς ακέραιους αλλά και αρνητικούς, μεγάλους αριθμούς κλπ. Από την άλλη είναι αδύνατο να ελέγξετε 2^{32} δυνατές τιμές για κάθε εντολή. Επομένως σκεφτείτε αρκετές διαφορετικές περιπτώσεις και δοκιμάστε τες, αλλά μην αποπειραθείτε να κάνετε εξαντλητικό έλεγχο όλων των δυνατών συνδιασμών τιμών.

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

Οι μνήμες έχουν χώρο για 128 εντολές και 128 λέξεις δεδομένων. Μη γράψετε προγράμματα που ξεπερνούν αυτά τα όρια. Θα προσέξετε ότι μόνο τα 9 λιγότερο σημαντικά bit των διευθύνσεων χρησιμοποιούνται, έτσι οι τιμές διευθύνσεων που βλεπετε 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.

Όταν ολοκληρώσετε την προσομοίωση, πάρτε ένα screenshot το παράθυρο με τις κυματομορφές με Full Zoom, που να δείχνει ότι λειτουργεί σωστά ο επεξεργαστής με όλες τις εντολές. Θα πρέπει να φαίνονται τουλάχιστον τα σήματα του wave.do που αναφέρθηκε παραπάνω. Μην προσθέσετε επιπλέον χρόνο στο τέλος, ούτε να κόψετε μέρος των κυμματομορφών. Δεν χρειάζεται να έχει υψηλή ανάλυση και να φαίνονται όλες οι τιμές γιατί θα είναι αρκετά zoomed out. **Θα πρέπει όμως να μπορεί να μετρηθεί ο αριθμός κύκλων ρολογιού και να φαίνονται καθαρά περιπτώσεις όπου αλλάζουν τιμές σημαντικά σήματα ελέγχου (π.χ. εγγραφή στη μνήμη).** Το screenshot θα χρησιμοποιηθεί ως μια οπτική επιβεβαίωση ότι το lab06.asm που παραδώσατε παράγει την ίδια κυματομορφή με αυτή που θα δουν οι βαθμολογητές όταν το τρέξουν. Ονομάστε το αρχείο screenshot και δώστε την κατάλληλη κατάληξη για να μπορούμε να το διαβάσουμε.

Δεν θα πρέπει να υπάρχουν λάθη στον επεξεργαστή. Αν βρείτε κάποιο λάθος και είστε σίγουροι ότι δεν φταίει το προγραμμά σας, ειδοποιήστε τον διδάσκοντα!

5 Αξιολόγηση

Ο κύριος σκοπός της άσκησης είναι η σε βάθος κατανόηση της λειτουργίας ενός επεξεργαστή. Αυτό επιτυγχάνεται με προσομοίωση προγραμμάτων, παρακολουθώντας πως αλλάζουν τα διάφορα σήματα του επεξεργαστή, ανάλογα με την εντολή που εκτελείται κάθε φορά και επιβεβαιώνοντας ότι τα αποτελέσματα είναι πάντα σωστά.

Γράφωντας ένα πρόγραμμα για να επιβεβαιώσετε ότι ο επεξεργαστής δουλεύει σωστά, θα αναγκαστείτε να εξετάσετε τα περισσότερα, αν όχι όλα, τα σήματα και θα διερευνήσετε ακραίες περιπτώσεις λειτουργίας, τις οποίες ο επεξεργαστής θα πρέπει να χειρίζεται σωστά. Μπορείτε να 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 στον επεξεργαστή.

Είναι αρκετά δύσκολο να αξιολογήσει κανείς πόσο καλό είναι ένα πρόγραμμα σαν αυτό που θα γράψετε. Ο πιο συνηθισμένος τρόπος είναι να μετρήσει κανείς την κάλυψη (coverage) του κώδικα Verilog του επεξεργαστή που πετυχαίνει το πρόγραμμα. Υπάρχουν ειδικά εργαλεία σχεδίασης κυκλωμάτων που κάνουν αυτή τη δουλειά. Δυστυχώς το Modelsim που δίνεται δωρεάν μαζί με το Quartus δεν έχει αυτή τη δυνατότητα: υπάρχει σε κάποιο μενού αλλά όταν το τρέξει κανείς ζητάει άδεια (license) η οποία δεν είναι δωρεάν.

Έχουμε όμως τέτοια εργαλεία στο Εργαστήριο Συστημάτων VLSI και Αρχιτεκτονικής Υπολογιστών και θα χρησιμοποιηθούν για την επιβράβευση όσων έχουν προσπαθήσει να πιάσουν ακραίες περιπτώσεις με το πρόγραμμά τους (περίπου 1.5 βαθμός). Δεν μπορώ να δώσω πρόσβαση σε αυτά γιατί μπορούν να χρησιμοποιηθούν μόνο από ένα χρήστη κάθε φορά και γιατί χρειάζονται πολύ χρόνο να μάθει κανείς να τα χρησιμοποιεί.

Κριτήρια αξιολόγησης των παραδοτέων είναι η "εμπειρική" ποιότητα του προγράμματος ως μέσου επιβεβαίωσης ορθής λειτουργίας (π.χ. χρήση όλων των διαφορετικών εντολών, σε πολλές καταστάσεις, μεγάλο εύρος τιμών δεδομένων) και οι μετρήσεις κάλυψης.

6 Παραδοτέα

Τα παραδοτέα της άσκησης είναι τα αρχεία `lab06.asm` και το screenshot των κυματομορφών. Η παράδοση θα γίνει μέσω GitHub, όπως πάντα. **Το screenshot είναι ιδιαίτερα σημαντικό γιατί αποδεικνύει ότι έγινε το εργαστήριο από εσάς.** Πρέπει να ακολουθεί τις προδιαγραφές που αναφέρονται παραπάνω. Θα διερευνυθεί αν αντιστοιχεί στο πρόγραμμα που στάλθηκε και αν μοιάζει με άλλα screenshots που παραδώθηκαν.

Όπως και στο προηγούμενο παραδοτέο μήν 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 για παράδοση στο GitHub, μήν στείλετε όλο τον κατάλογο lab05, αλλά τα επιμέρους αρχεία.

7 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 περιτά αρχεία. Κρατήστε μόνο ό,τι υπάρχει στο αποθετήριο του GitHub το lab06.asm και το screenshot.