1ο Εργαστήριο Αρχιτεκτονικής Η/Υ: Risc-V assembly, γνωριμία με τον Ripes

Α. Ευθυμίου

Παραδοτέο: Τρίτη 15 Οκτώβρη, 23:59

Μή ξεχάσετε να επιστρέψετε, μέσω GitHub, το παραδοτέο που αναφέρεται στο τέλος του κειμένου!

Με αυτή την εργαστηριακή άσκηση θα εξοικειωθείτε με τον Ripes, έναν προσομοιωτή γλώσσας assembly του Risc-V, που θα χρησιμοποιήσετε σε πολλές επόμενες εργαστηριακές ασκήσεις. Θα πρέπει να έχετε μελετήσει το πρώτο μάθημα για τη γλώσσα assembly του Risc-V (2η διάλεξη του μαθήματος) που αντιστοιχεί στις ενότητες 2.1-2.3 του βιβλίου.

Δευτερεύων, αλλά σημαντικός, στόχος της άσκησης είναι η εξοικίωσή σας με τον τρόπο παράδοσης των ασκήσεων του μαθήματος μέσω του GitHub. Το git απαιτεί κάποιο χρόνο εκμάθησης γιατί είναι αρκετά διαφορετικό από ό,τι, οι περισσότεροι, έχετε συνηθίσει μέχρι τώρα. **Είναι σημαντικό να μην περιμένετε μέχρι την τελευταία στιγμή για να παραδώσετε την άσκηση.** Κάντε μερικές δοκιμές νωρίς ώστε να μπορείτε να αντιμετωπίσετε τυχόν προβλήματα. Δεν χρειάζεται να έχετε ολοκληρώσει την άσκηση για να κάνετε μια δοκιμαστική παράδοση. Μια μικρή αλλαγή σε ένα σχόλιο του προγράμματος είναι αρκετή για να ελέγξετε ότι μπορείτε να παραδώσετε την άσκηση σωστά.

Χρησιμοποιείστε το Edstem για ανταλλαγή συμβουλών, πληροφοριών, καλών πρακτικών χρήσης κτλ, αλλά μή δίνετε πληροφορίες που φανερώνουν τη λύση των ασκήσεων. Οι βοηθοί και ο διδάσκοντας θα συμμετέχουν στις συζητήσεις αυτές. Στο Edstem υπάρχει μια εκτενής ανάρτηση για τη χρήση του git,GitHub στις εργαστηριακές ασκήσεις του μαθήματος και τα πιο συνηθισμένα προβλήματα που μπορεί να παρουσιαστούν.

Ο Ripes, που αναπτύχθηκε από τον Morten Borup Petersen, εκτός από προσομοιωτής είναι ένα πλήρες γραφικό περιβάλον ανάπτυξης και εκσφαλμάτωσης προγραμμάτων (IDE). Στο εργαστήριο 0 δόθηκαν οδηγίες για την εγκατάστασή του. Στους υπολογιστές των εργαστηρίων του Τμήματος θα τον βρείτε στο `~myy505/bin/Ripes-v2.2.6-linux-x86_64.AppImage`.

1 Βασικές πληροφορίες για Risc-V assembly

- Τα προγράμματα assembly αποθηκεύονται σε απλά αρχεία κειμένου και οι καταλήξεις τους είναι συνήθως `.asm` ή `.s`.
- Τα προγράμματα πρέπει να τελειώνουν με τις εξής δύο εντολές: `addi a7, zero,10` και `ecall`. Με αυτές, όταν ολοκληρωθεί η εκτέλεση του προγράμματος, ο έλεγχος περνάει ξανά στο λειτουργικό σύστημα.
- Τα σχόλια ξεκινούν με το σύμβολο `#` και τελειώνουν στο τέλος της γραμμής. Στην Risc-V έκδοση του συγγράμματος, χρησιμοποιείται το `//` για την αρχή σχολίων, αλλά ο Ripes δεν το δέχεται. Χρησιμοποιείστε το `#`.
- Δεν είναι καλό να γράφετε πάνω από μία εντολή ανά γραμμή.
- Οι ετικέτες (labels) πρέπει να τελειώνουν με το σύμβολο `:`
- Ο assembler, το πρόγραμμα που διαβάζει κώδικα assembly, είναι εξαιρετικά απλός. Μπορεί να μην επιτρέπει για παράδειγμα οι εντολές να «σπάνε» σε ξεχωριστές γραμμές και τα μηνύματα λάθους μπορεί να μην είναι πάντα αρκετά κατατοπιστικά. Ακολουθήστε το πρότυπο των προγραμμάτων που σας δίνονται για να γράφετε σωστό κώδικα που ακολουθεί το στυλ με το οποίο είναι εξοικειωμένοι οι περισσότεροι προγραμματιστές στον κόσμο. Οι βασικές οδηγίες είναι:
  - οι ετικέτες γράφονται τέρμα αριστερά και είναι σχετικά μικρές.

  - οι εντολές ξεκινούν δεξιότερα για να ξεχωρίζουν από τις ετικέτες και είναι στοιχισμένες.
  - οι τελεσταίοι των εντολών επίσης απέχουν μερικά κενά από τα ονόματα των εντολών.

- Ο assembler, εκτός από εντολές assembly, ετικέτες και σχόλια, δέχεται και κάποιες ειδικές λέξεις ως οδηγίες που λέγονται directives. Χρησιμοποιούνται για να ξεχωρίσουν το πρόγραμμα από τα δεδομένα και για να δεσμεύσουν χώρο στη μνήμη για δεδομένα. Τα directives ξεκινούν πάντα με μία τελεία. Θα μάθετε μερικές απαραίτητες directives από το πρώτο πρόγραμμα assembly. Τα σχόλια του προγράμματος εξηγούν τι σημαίνει η κάθε μία.

2 Εξοικείωση με τον Ripes

Κάντε τα παρακάτω βήματα:

1. Αρχικοποίηση δομής καταλόγων και αποθετηρίων

   (α′) Ακολουθήστε τον σύνδεσμο που υπάρχει στο ecourse για την άσκηση αυτή. Κάνοντας κλικ στον σύνδεσμο, δημιουργείται ένα νέο αποθετήριο στον οργανισμό του μαθήματος. Μερικές φορές στέλνεται και ένα email στη διεύθυνση που έχετε δώσει στο GitHub. Μπορείτε να δείτε το νέο αποθετήριο είτε αμέσως μετά το κλίκ στον σύνδεσμο, ή ακολουθώντας τους συνδέσμους του email ή απλά στα αποθετήρια https://github.com/UoI-CSE-MYY505 που είναι ορατά σε εσάς. Το URL του αποθετηρίου θα έχει τη μορφή https://github.com/UoI-CSE-MYY505/lab01-ghUsername, όπου ghUsername το όνομα χρήστη που έχετε στο GitHub.

   (β′) Δημιουργήστε ένα κατάλογο για όλες τις εργαστηριακές ασκήσεις του μαθήματος στον υπολογιστή σας και μεταβείτε σε αυτόν. Ονομάζω αύτον τον κατάλογο MYY505Labs, για να μπορώ να αναφερθώ σε αυτόν παρακάτω, αλλά μπορείτε να χρησιμοποιήσετε ό,τι όνομα θέλετε.

   (γ′) Ανοίξτε ένα τερματικό και μεταβείτε στον παραπάνω κατάλογο.

   (δ′) Κλωνοποιείστε το *βοηθητικό αποθετήριο* του εργαστηρίου (είναι διαφορετικό από αυτό που δημιουργήθηκε αυτόματα παραπάνω) με την εντολή:

   ```
   git clone https://github.com/UoI-CSE-MYY505/myy505Utils.git
   ```

   (ε′) Κλωνοποιείστε το αποθετήριο της άσκησης με την εντολή (αλλάζετε το ghUsername με το όνομα χρήστη):

   ```
   git clone https://github.com/UoI-CSE-MYY505/lab01-ghUsername.git
   ```

   Θα πρέπει να έχετε μια δομή καταλόγων της μορφής:

   ```
   MYY505Labs
   |- myy505Utils
   |- lab01-ghUsername
   ```

   Μεταβείτε στον κατάλογο `lab01-ghUsername`. Εκεί θα βρείτε το αρχείο `lab01.s`, το πρώτο σας πρόγραμμα σε Risc-V assembly.

2. Ξεκινήστε τον Ripes, π.χ. σε τερματικό με την εντολή `Ripes-v2.2.6-linux-x86_64.AppImage`. Μπορεί να χρειαστεί να τροποποιήσετε το `PATH` για να το βρεί ο διερμηνέας του τερματικού.

3. Στο παράθυρο που θα ανοίξει, πατήστε το εικονίδιο που μοιάζει με ολοκληρωμένο κύκλωμα, επάνω αριστερά. (Δεν υπάρχει αντίστοιχη επιλογή στο κεντρικό μενού. Μόνο πατώντας το εικονίδιο, μπορείτε να κάνετε αυτή τη ρύθμιση.) Θα ανοίξει ένα μικρό παράθυρο με τίτλο, Select Processor. Επιλέξτε το RISC-V, 32-bit, Single-cycle processor.

   Οι υπόλοιπες επιλογές θα πρέπει να είναι: ISA Exts. M μόνο, όχι C, Register initialization:
   `x2 (sp) = 0x7ffffff0, x3 (gp) = 0x10000000`

   Πατήστε OK για να συνεχίσετε.

4. Επιλέξτε το Editor tab, πρώτο, επάνω από τα εικονίδια στα αριστερά του κεντρικού παραθύρου. Φορτώστε το αρχείο `lab01.s` χρησιμοποιώντας το μενού: File→Load Program, ή, καλύτερα, με τη συντόμευση από το πληκτρολόγιο (Ctrl-O). Για File type επιλέξτε Source file και αφού πατήσετε το Open..., διαλέξτε το παραπάνω αρχείο.

   Ο κώδικας του `lab01.s` εμφανίζεται στο αριστερό τμήμα του παραθύρου. Αυτό το τμήμα είναι ο αληθινός Editor όπου μπορείτε να κάνετε αλλαγές και όπου εμφανίζονται σημάδια για συντακτικά λάθη κλπ.

   Στο μεσαίο τμήμα είναι η έξοδος του assembler. Σε αυτό το τμήμα δεν μπορείτε να κάνετε απευθείας αλλαγές. Το View mode, από επάνω του, έχει κάποιες επιλογές. Το Binary εμφανίζει μόνο τον κώδικα μηχανής, ενώ το Disassembled (επιλέξτε αυτό) δείχνει και τις εντολές assembly. Υπάρχουν διαφορές ανάμεσα στον κώδικα assembly του Editor (αριστερό τμήμα) και στο Disassembled, γιατί οι εντολές la li είναι ψευτοεντολές: διασπώνται σε δύο αληθινές εντολές η κάθε μια. (Θα δούμε τι ακριβώς συμβαίνει λίγο παρακάτω στο μάθημα). Στο disassembled mode η πρώτη στήλη είναι η διεύθυνση της εντολής (δεκαεξαδική μορφή), η δεύτερη στήλη είναι ο κώδικας μηχανής (δεκαεξαδική μορφή), και η τρίτη η εντολή σε assembly. Επιπλέον φαίνονται τα labels του κώδικα και σε ποιες διευθύνσεις αντιστοιχούν. Παρατηρήστε ότι τα labels δεν καταλαμβάνουν χώρο στη μνήμη. Για παράδειγμα το prog φαίνεται στη διεύθυνση 0x20 και η παρακάτω εντολή είναι επίσης στη διεύθυνση 0x20.

   Διαβάστε προσεκτικά τον κώδικα (και τα σχόλια). Παρατηρήστε ότι οι εντολές, ονόματα καταχωρητών, ετικέτες και σχόλια χρωματίζονται αυτόματα (code highlighting). Συντακτικά λάθη σημειώνονται με κόκκινη, κυματιστή υπογράμμιση. Η φαρδιά ροζ-κόκκινη γραμμή δείχνει την επόμενη εντολή που θα εκτελέσει ο προσομοιωτής.

5. Όταν πλέον καταλαβαίνετε καλά τον κώδικα, ετοιμάστε τον για εκτέλεση (αγγλικός όρος assemble) πατώντας το εικονίδιο με το σφυρί ή με με τη συντόμευση πληκτρολογίου ctlr-B.

   Καλό είναι να μάθετε τις συντομεύσεις πληκτρολογίου. Κερδίζετε χρόνο.

   Όταν φορτώνετε ένα νέο πρόγραμμα, ο Ripes το περνάει από τον assembler αυτόματα. Μερικές φορές το εικονίδιο με το σφυρί ειναι απενεργοποιημένο γιατί έχει ήδη γίνει η διαδικασία.

   Σπάνια ο Ripes δίνει compilation error χωρίς να περιγράφει τα λάθη που υπάρχουν. Σε αυτή την περίπτωση δεν υπάρχουν συντακτικά λάθη! Πατήστε OK και συνεχίστε κανονικά.

   Από αυτό το σημείο αξίζει να παρατηρείτε τι συμβαίνει στο δεξιότερο τμήμα του παραθύρου (GPR - General Purpose Registers). Εκεί φαίνονται οι τρέχουσες τιμές των καταχωρητών του Risc-V. Πρώτα είναι το αλήθινο όνομα - αριθμός του καταχωρητή (ξεκινάει με x), μετά το βολικό όνομα που χρησιμοποιούμε (a, s, t, και αριθμός) και τέλος η τρέχουσα τιμή. Η μορφή της τιμής μπορεί να αλλαχθεί από το Display type pull-down menu στο κάτω μέρος. Αυτό είναι πολύ χρήσιμο γιατί οι καταχωρητές χρησιμοποιούνται για διάφορα είδη δεδομένων. Τέλος, μπορεί κανείς να αλλάξει την τιμή κάποιου καταχωρητή με διπλό κλικ στην τιμή του. Αυτό είναι χρήσιμο κατά την εκσφαλμάτωση γιατί μπορεί κανείς να δει τι θα συμβεί στο πρόγραμμα με διαφορετική τιμή καταχωρητή, χωρίς να αλλάξει τον κώδικα.

   Για να δείτε τα περιεχόμενα της μνήμης, επιλέξτε το εικονίδιο που μοιάζει με πλακέτα μνήμης DIMM (πράσινο), το Memory tab. Στο κάτω μέρος υπάρχει το Go to section pull-down menu με το οποίο μπορείτε να επιλέξετε το τμήμα της μνήμης που θέλετε να δείτε. Για την ώρα μας ενδιαφέρει μόνο το .data που είναι το τμήμα των δεδομένων.

   Το κύριο τμήμα του παραθύρου έχει ως πρώτη στήλη τη διεύθυνση, και ακολουθούν η τιμή της λέξης που αντίστοιχεί στη διεύθυνση και οι τιμές των τεσσάρων bytes που απαρτίζουν τη λέξη, από το λιγότερο σημαντικό προς το περισσότερο σημαντικό. Οι διευθύνσεις των λέξεων αυξάνονται από το κάτω μέρος του παραθύρου προς το επάνω. Όπου η μνήμη δεν έχει αποθηκευμένη κάποια τιμή, φαίνεται γκρί και έχει ως τιμή το X. Στο κάτω μέρος μπορείτε να διαλέξετε τη μορφή της τιμής (Display type), όπως με τους καταχωρήτές. Αντίθετα από τους καταχωρητές όμως, δεν μπορείτε να αλλάξετε τις τιμές μνήμης.

Στο Memory tab δεν φαίνονται τα label δεδομένων για να ξεχωρίζετε που βρίσκονται τα επιμέρους δεδομένα. Για να δειτε την αντιστοίχιση label - διεύθυνσης, επιστρέψτε στο Editor tab, και πατήστε το εικονίδιο που μοιάζει με πυξίδα, επάνω δίπλα στο παράθυρω των καταχωρητών. Δυστυχώς το παράθυρο αυτό δεν μπορεί να μείνει ανοιχτό όσο δουλεύετε. Επομένως σημειώστε κάπου τις διευθύνσεις που σας ενδιαφέρουν και κλείστε το. Προσοχή, αν κάνετε αλλαγές, οι διευθύνσεις που αντιστοιχούν στα labels μπορεί να αλλάξουν. Πρέπει να έχετε πάντα τις τρέχουσες αντιστοιχίσεις για να μην κοιτάζετε σε λάθος θέση της μνήμης!

Όσο είστε στο Memory tab, ψάξτε να βρείτε στο .data section τις αρχικές τιμές που αποθηκεύει εκεί το πρόγραμμα. Βρείτε για παράδειγμα που είναι το 30 (δεκαδική μορφή) του matric, πώς αποθηκεύονται οι τιμές του array και το string str1.

6. Εκτελέστε το πρόγραμμα εντολή προς εντολή χρησιμοποιώντας το εικονίδιο που μοιάζει με το >, ή πλήκτρο σύντμευσης (βρείτε το!). Μπορείτε να πάτε μερικές εντολές πίσω (με το <) ή να αρχίσετε την εκτέλεση από την αρχή με το εικονίδιο με τα δύο τόξα σε ένα κύκλο. Στις ψευτοεντολές li, la προχωρήστε μέχρι να κοκκινίσει η επόμενη εντολή στον Editor (δηλαδή εκτελέστε 2 αληθινές εντολές), πριν να δείτε τα περιεχόμενα των καταχωρητών.

   Παρατηρήστε τις αλλαγές στους καταχωρητές και στη μνήμη δεδομένων μετά την εκτέλεση κάθε εντολής. Με αυτό τον τρόπο θα δείτε τι συμβαίνει και θα μπορέσετε να λύσετε πιθανές απορίες μόνοι σας. Αυτό είναι το πλεονέκτημα της χρήσης προσομοίωσης: ο εύκολος πειραματισμός! Αν κάτι δεν σας φαίνεται σωστό, πάτε ένα βήμα πίσω και ξαναεκτελέστε την εντολή, προσέχοντας τις αλλαγές στις τιμές καταχωρητών ή μνήμης. Αλλάξτε τον κώδικα και δείτε τι προκαλούν οι αλλαγές σας. (Για την παράδοση της άσκησης όμως πρέπει να κάνετε μόνο ό,τι ζητείται! Κρατήστε αντίγραφο του κώδικα για πειραματισμό).

7. Εξερευνήστε τα υπόλοιπα εικονίδια. Θα χρησιμοποιείσετε τον Ripes σε πολλές εργαστηριακές ασκήσεις: όσο περισσότερο γνωρίζετε τις δυνατότητές του τόσο πιο παραγωγικοί θα είστε, συνεπώς θα τελειώνετε τις ασκήσεις γρηγορότερα.

   Πολύ χρήσιμη είναι η δυνατότητα εκτέλεσης μέχρι μια εντολή. Αυτό ονομάζεται breakpoint. Για να ενεργοποιήσετε ένα breakpoint πριν από την εκτέλεση μιας εντολής, κάνετε κλικ στο μπλε περιθώριο πριν από την εντολή που θέλετε (στο disassembled view). Θα εμφανιστεί ένα κόκκινο σημάδι. Αν εκτελέσετε συνεχόμενες εντολές, με το εικονίδιο >>, η εκτέλεση θα σταματήσει ακριβώς πριν την εκτέλεση της συγκεκριμένης εντολής.

   Με αυτό τον τρόπο μπορείτε να περνάτε στα γρήγορα εντολές που ξέρετε ότι λειτουργούν σωστά και να εκτελείτε βήμα-βήμα μόνο το τμήμα του προγράμματος που σας ενδιαφέρει.

3 Παραδοτέο

Αλλάξτε το πρόγραμμα ώστε αντί του 30, να υπάρχει ο αριθμός μητρώου σας στην μνήμη στην θέση της ετικέτας matric.

Αλλάξτε την γραμμή `li s0, 100` με μία εντολή assembly, που να φορτώνει στον καταχωρητή `s0` την τιμή που είναι αποθηκευμένη στην διεύθυνση μνήμης που αντιστοιχεί στο label matric. Η διεύθυνση του matric υπάρχει ήδη στον καταχωρητή `a0`, επομένως μία εντολή αρκεί.

Ξανακάνετε assemble το πρόγραμμά σας αφού τώρα πια έχει αλλάξει και ελέγξτε το.

Υπάρχει μια ακόμα αλλαγή που πρέπει να γίνει. Διαβάστε το παρακάτω κείμενο.

3.1 Προαιρετικό: Αυτόματος έλεγχος ορθότητας

Στο repository θα βρείτε το αρχείο Lab01Test.py το οποίο, σε συνδιασμό με κώδικα που υπάρχει στο βοηθητικό αποθετήριο myy505Utils, δίνει τη δυνατότητα αυτόματου ελέγχου του προγραμμάτος. Ο έλεγχος γίνεται θέτοντας τιμές σε καταχωρητές και μνήμη πρίν την εκτέλεση του προγράμματος και συγκρίνοντας τιμές καταχωρητών και μνήμης με προϋπολογισμένες τιμές που θα πρέπει να δίνει ένα σωστό πρόγραμμα.

Για να τρέξετε το τεστ, εκτελέστε το Lab01Test.py μέσα στον κατάλογο της άσκησης. Θα πρέπει να έχετε τη δομή καταλόγων που περιγράφεται παραπάνω, δηλαδή να υπάρχει το myy505Utils στον γονικό κατάλογο του τρέχοντος καταλόγου lab01-ghUsername.

Αν όλα είναι εντάξει, δεν θα δείτε κανένα μήνυμα. Διαφορετικά θα δείτε μηνύματα που περιγράφουν το λάθος και τις αναμενόμενες και πραγματικές τιμές.

Για την ώρα το πρόγραμμα που σας δόθηκε δεν περνάει τον έλεγχο και εμφανίζεται το μήνυμα:

```
s0 should be equal to your matric number. Got 100, expected 0 (0)
s1 should be equal to your matric number + 1. Got 101, expected 1 (1)
matricplus1 should be matric + 1 (in memory). Got 268435480, expected 1 (1)
Unexpected value in array[3]. Got 100, expected 0 (0)
```

Αυτό γίνεται γιατί το Lab01Test.py θεωρεί ότι το matric έχει την τιμή 0.

Αλλάξτε την τιμή της μεταβλητής matricNumber (γραμμή 12 στο Lab01Test.py) ώστε να είναι ο δικό σας αριθμός μητρώου. Με αυτή την αλλαγή και αυτές στο αρχείο .s που περιγράφηκαν παραπάνω, βεβαιωθείτε ότι το τέστ πλέον περνάει με επιτυχία.

Αντίστοιχα αρχεία labXXTest.py θα υπάρχουν και για τις επόμενες εργαστηριακές ασκήσεις. Γενικά δεν θα χρειάζεται να κάνετε πολλές αλλαγές σε αυτά, ούτε είναι απαραίτητο να καταλαβαίνετε πλήρως τον κώδικα σε αυτά. Δίνονται για να συνηθίσετε στο ότι πρέπει πάντα να ελέγχετε ό,τι πρόγραμμα γράφετε και για να έχετε μια εικόνα των εργαλείων με τα οποία ελέγχονται οι ασκήσεις που βαθμολογούνται. Βέβαια δεν είναι σίγουρο ότι θα μπορούν όλα τα τεστ που θα χρησιμοποιηθούν για την αξιολόγηση των ασκήσεων να είναι έτοιμα την ώρα της ανάρτησης, επομένως ακόμα και ασκήσεις που περνούν όλα τα τεστ δεν είναι σίγουρο ότι δεν έχουν λάθη! Στην αξιολόγηση λαμβανονται υπόψη και η ποιότητα των σχολίων, η σαφήνεια του κώδικα, το μέγεθός του και η ταχύτητα εκτέλεσης. Επομένως αν μια άσκηση περνάει το τεστ δεν σημαίνει ότι ο βαθμός της θα είναι 10.

3.2 Παράδοση της άσκησης

Για να παραδώσετε την άσκηση δίνετε τις παρακάτω εντολές σε τερματικό:

```
git add lab01.s Lab01Test.py
git commit -m "Completed lab01"
git push origin main
```

Το μήνυμα της εντολής git commit είναι ενδεικτικό, μπορείτε να γράψετε κάτι άλλο. Η εντολή git push μπορεί να γίνει συντομότερη, όπως εξηγήθηκε στις διαλέξεις .

Προσοχή, μην αλλάζετε τα ονόματα αρχείων ή καταλόγων (π.χ. προσθέτοντας όνομα, αριθμό μητρώου κλπ).

Ελέγξτε αν η άσκηση παραδόθηκε σωστά: Με έναν browser συνδεθείτε κατευθείαν στο αποθετήριό σας στον οργανισμό του μαθήματος: `https://github.com/UoI-CSE-MYY505/lab01-ghUsername`. Θα πρέπει τα αρχεία εκεί να έχουν αλλάξει και να είναι όμοια με αυτά του τοπικού σας αποθετηρίου.

Προσοχή στο παραπάνω URL πρέπει να υπάρχει το UoI-CSE-MYY505! Πολύ συνηθισμένο λάθος είναι να χρησιμοποιηθεί ένα προσωπικό αποθετήριο στο GitHub, εκτός του οργανισμού του μαθήματος. Αυτά τα αποθετήρια, δεν περιέχουν το UoI-CSE-MYY505 στο URL τους. Σε αυτή την περίπτωση δεν θα παραλειφθεί το παραδοτέο σας και δεν θα πάρετε τον αντίστοιχο βαθμό!