

1ο Εργαστήριο Αρχιτεκτονικής Η/Υ: MIPS assembly, γνωριμία με τον MARS

A. Ευθυμίου

Παραδοτέο: Παρασκευή 16 Οκτώβρη, 23:59

Μή ξεχάσετε να επιστρέψετε, μέσω GitHub, το παραδοτέο που αναφέρεται στο τέλος του κειμένου για να πάρετε βαθμό γι'αυτή την εργαστηριακή άσκηση!

Με αυτή την εργαστηριακή άσκηση θα εξοικειωθείτε με τον MARS, έναν προσομοιωτή γλώσσας assembly του MIPS, που θα χρησιμοποιήσετε σε πολλές επόμενες εργαστηριακές ασκήσεις. Θα πρέπει να έχετε μελετήσει το πρώτο μάθημα για τη γλώσσα assembly του MIPS (3η διάλεξη του μαθήματος) που αντιστοιχεί στις ενότητες 2.1-2.3 του βιβλίου.

Δευτερεύων, αλλά πολύ 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. Μόνο η τελευταία παράδοση βαθμολογείται.

Χρησιμοποιείτε το Piazza για ανταλλαγή συμβουλών, πληροφοριών, καλών πρακτικών χρήσης κτλ, αλλά μη δίνετε πληροφορίες που φανερώνουν τη λύση των ασκήσεων. Οι βοηθοί και ο διδάσκοντας θα συμμετέχουν στις συζητήσεις αυτές. Στο Piazza 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 MARS, που αναπτύχθηκε από τους Pete Sanderson και Ken Vollmar, εκτός από προσομοιωτής είναι ένα πλήρες γραφικό περιβάλλον ανάπτυξης και εξασφάλισης προγραμμάτων (IDE). Στο εργαστήριο 0 δόθηκαν οδηγίες για την εγκατάστασή του. Επειδή είναι γραμμένος σε Java τρέχει σε υπολογιστές με οποιοδήποτε λειτουργικό σύστημα αρκεί να υπάρχει εγκατεστημένη η κατάλληλη έκδοση του Java J2SE. Στους υπολογιστές των εργαστηρίων του Τμήματος θα τον βρείτε στο `~myy505/bin/MarsMTT505_4_5.jar`.

1 Βασικές πληροφορίες για MIPS 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 $v0,$zero,10` και `syscall`. 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.
- Δεν επιτρέπεται να γράψετε 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 τελεστές 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 MARS

Κάντε τα παρακάτω βήματα:

1. Ακολουθήστε τον σύνδεσμο <https://classroom.github.com/a/McV9Mfkc>. 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 κλικ 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.

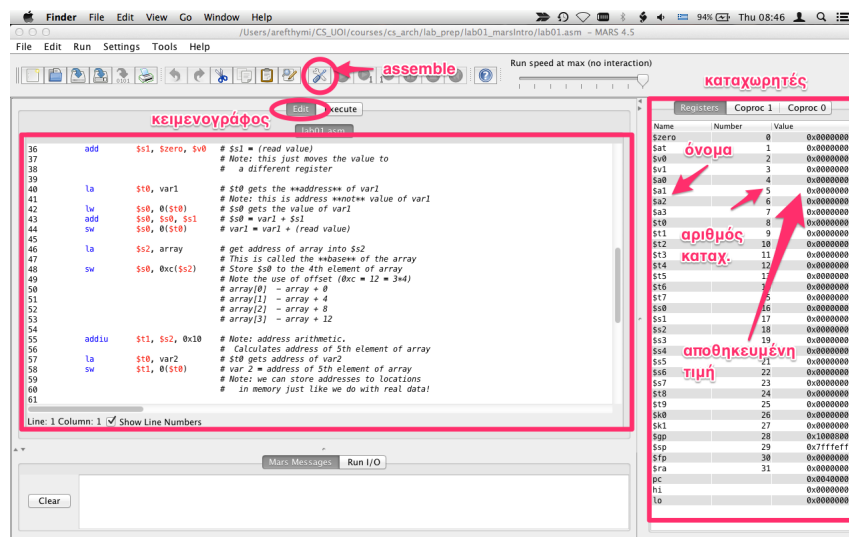
Κλωνοποιείτε το στον υπολογιστή σας (εργαστήριο ή σπίτι ή και τα δύο) με την εντολή:

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

Για να πάρετε τα αρχεία της εργαστηριακής άσκησης, μεταβείτε στον κατάλογο που θα δημιουργηθεί από το παραπάνω βήμα και θα έχει το ίδιο όνομα με το αποθετήριο (αλλάζετε το ghUsername με το όνομα χρήστη): Εκεί θα βρείτε το αρχείο lab01.asm, το πρώτο σας πρόγραμμα σε MIPS assembly.

2. Ξεκινήστε τον MARS, π.χ. σε τερματικό με την εντολή `java -jar MarsMYY505_4_5.jar`. Σε αυτή την άσκηση το αρχείο MarsMYY505_4_5.jar περιλαμβάνεται στο αποθετήριο για ευκολία. Στις επόμενες, δεν θα περιλαμβάνεται.
3. Επειδή δεν θα χρειαστείτε τους συν-επεξεργαστές (co-processors) πριν από τη διάλεξη για τις εξαιρέσεις και διακοπές, μπορείτε να σείρετε το δεξί υπό-παράθυρο προς τα δεξιά ώστε να φαίνεται μόνο το tab “Registers”. Έτσι θα υπάρχει περισσότερος χώρος για να βλέπετε τον κώδικα, κυρίως κατά την εκτέλεσή του (στο tab “execute”). Αν τα γράμματα σας φαίνονται πολύ μικρά, μπορείτε να αλλάξετε το μέγεθός τους (και ό,τι άλλο θέλετε) από τις ρυθμίσεις στο Settings→Editor. Προτείνω τη ρύθμιση “Font Family: Monospaced” και tab size 4.
4. Φορτώστε το αρχείο lab01.asm χρησιμοποιώντας είτε το εικονίδιο (2ο από αριστερά) ή από το μενού: File→Open, ή, ακόμα καλύτερα, με τη συντόμευση από το πληκτρολόγιο (Ctrl-O). Το παράθυρο του MARS θα μοιάζει με το σχήμα 1.
Διαβάστε προσεκτικά τον κώδικα (και τα σχόλια) στο “Edit” tab. Παρατηρήστε ότι οι εντολές, ετικέτες και σχόλια χρωματίζονται αυτόματα (code highlighting) και ότι υπάρχει η δυνατότητα αυτόματης συμπλήρωσης εντολών (completion suggestion).
5. Όταν πλέον καταλαβαίνετε καλά τον κώδικα, ετοιμάστε τον για εκτέλεση (αγγλικός όρος assemble) χρησιμοποιώντας το μενού (Run→Assemble), ή το εικονίδιο με το κατσαβίδι και το κλειδί ή με το πλήκτρο F3.

Αυτόματα θα αλλάξει ο MARS στο tab “Execute”, όπου φαίνονται 3 παράθυρα στη θέση του κώδικα: η μνήμη εντολών (text segment), η μνήμη δεδομένων (data segment) και η αντιστοίχιση



Σχήμα 1: Ο κειμενογράφος του MARS.

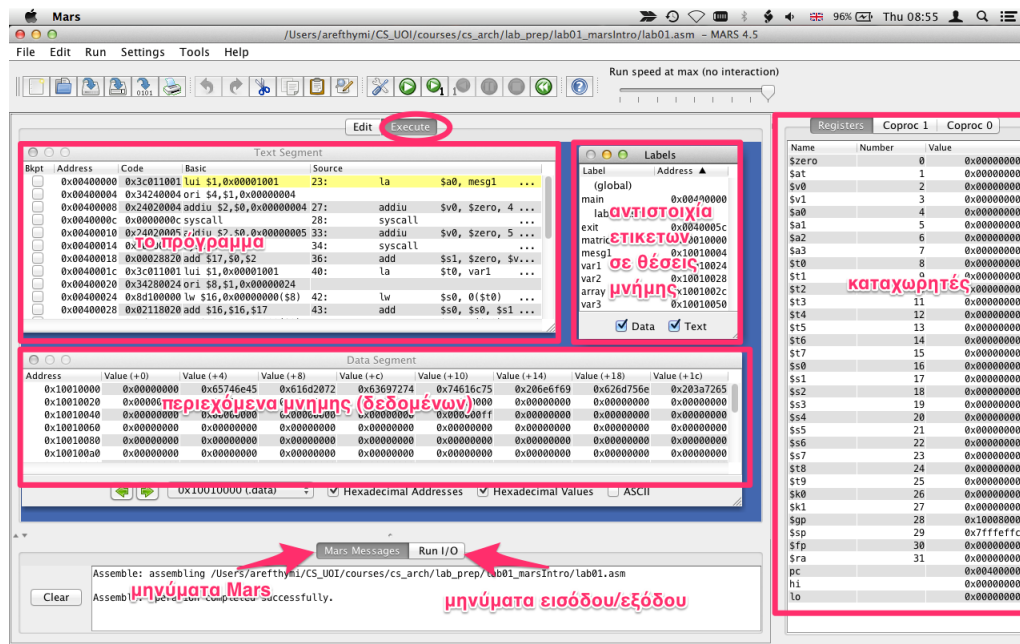
ετικετών σε διευθύνσεις (labels). Στα δεξιά θα παραμείνει το παράθυρο με τους καταχωρητές. Το παράθυρο του MARS θα μοιάζει με το σχήμα 2.

Παρατηρήστε τις πληροφορίες στα παράθυρα. Ο αρχικός κώδικας (και τα σχόλια) φαίνονται ακόμη μαζί με άλλες πληροφορίες όπως η διεύθυνση κάθε εντολής και η κωδικοποίησή της σε γλώσσα μηχανής (που θα είναι το αντικείμενο επόμενου μαθήματος), όπως στο σχήμα 3. Θα δείτε ότι η πρώτη γραμμή του κώδικα έχει χρωματιστεί με κίτρινο χρώμα: αυτό υποδεικνύει την εντολή που πρόκειται να εκτελεστεί.

Στο παράθυρο μνήμης δεδομένων (Σχήμα 4) μπορεί κανείς να δει τα δεδομένα ως δεκαεξαδικούς αριθμούς, ως δεκαδικούς αριθμούς, ή ως συμβολοσειρά χαρακτήρων με κωδικοποίηση ASCII (1 byte ανά χαρακτήρα). Η μορφή παρουσίασης αλλάζει χρησιμοποιώντας τα tick boxes στο κάτω μέρος του παραθύρου.

Παρατηρήστε ότι εμφανίζονται 8 λέξεις των 32 bit ανά γραμμή. Η διεύθυνση είναι ο αριθμός στην αρχή της κάθε γραμμής και αντιστοιχεί στην πρώτη (στα αριστερά) λέξη. Για παράδειγμα η διεύθυνση της πρώτης γραμμής είναι 0x10010000 με τις αρχικές ρυθμίσεις του MARS. Η επόμενη λέξη βρίσκεται στη διεύθυνση που φαίνεται στην αρχή της γραμμής + 4, δηλαδή 0x10010004. Θυμηθείτε ότι κάθε λέξη είναι 32 bit, δηλαδή 4 byte, και οι διευθύνσεις της μνήμης αναφέρονται σε κάθε ένα byte. Έτσι οι διευθύνσεις διαδοχικών λέξεων διαφέρουν κατά 4. Η διεύθυνση της τελευταίας λέξης της πρώτης γραμμής είναι 0x1001001c, ίση με τη διεύθυνση της πρώτης λέξης + 28_{ten} (0x1c). Η απόσταση της διεύθυνσης κάθε λέξης της γραμμής από την αρχική φαίνεται στην επικεφαλίδα της αντίστοιχης στήλης του παραθύρου: είναι ο αριθμός μέσα στην παρένθεση και είναι σε δεκαεξαδική μορφή.

- Εκτελέστε το πρόγραμμα εντολή προς εντολή χρησιμοποιώντας το Run→Step ή το αντίστοιχο εικονίδιο (πράσινο τρίγωνο με τον αριθμό 1) ή πλήκτρο σύντμευσης (βρείτε το!). Παρατηρήστε τις αλλαγές στους καταχωρητές και στη μνήμη δεδομένων. Ο MARS δείχνει με χρώμα τις τελευταίες αλλαγές. Σε αντίθεση με το παράθυρο μνήμης εντολών όπου το χρώμα δείχνει την επόμενη εντολή που θα εκτελεστεί, στα παράθυρα καταχωρητών και μνήμης το χρώμα δείχνει την αλλαγή που μόλις έγινε.
- Εξερευνήστε όλα τα μενού εκτός του Tools και προσπαθείστε να μάθετε τις συντμήσεις πλήκτρων. Θα χρησιμοποιείτε το MARS σε πολλές εργαστηριακές ασκήσεις: όσο περισσότερο γνωρίζετε



Σχήμα 2: Ο MARS σε κατάσταση εκτέλεσης προγράμματος.

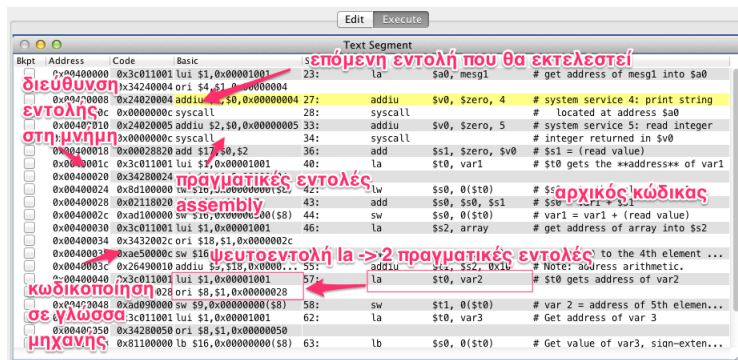
τις δυνατότητές του τόσο πιο παραγωγικοί θα είστε, συνεπώς θα τελειώνετε τις ασκήσεις γρηγορότερα.

Ενδεικτικά αναφέρονται οι παρακάτω χρήσιμες λειτουργίες.

Υπάρχει η δυνατότητα να "πάτε προς τα πίσω" (backstep) μία εντολή κάθε φορά (εικονίδιο: μπλέ τρίγωνο με τον αριθμό 1), να επανεκκινήσετε το πρόγραμμα (reset), καθώς και να καθορίσετε την ταχύτητα εκτέλεσης (slider επάνω δεξιά) ώστε να παρακολουθείτε τις αλλαγές που συμβαίνουν όταν εκτελείτε, μέχρι τέλους, ένα μεγάλο πρόγραμμα (Run→Go).

Κατά τη διάρκεια της εκτέλεσης, μπορείτε, εκτός από το να παρατηρείτε τιμές καταχωρητών και μνήμης, να τις αλλάξετε με διπλό κλικ στο πεδίο τιμής (value).

Κάνοντας κλικ στο όνομα μιας ετικέτας (label) στο παράθυρο ετικετών, εμφανίζεται ένα λεπτό μπλε πλαίσιο στο παράθυρο δεδομένων στη λέξη που αντιστοιχεί στην ετικέτα αυτή.



Σχήμα 3: Το υποπαράθυρο προγράμματος.

Data Segment									
Address	Value (+0)	Value (+4)	Value (+8)	Value (+C)	Value (+10)	Value (+14)	Value (+18)	Value (+1C)	
0x10000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
0x10010020	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
0x10010040	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
0x10010060	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
0x10010080	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
0x100100A0	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
0x100100C0	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
0x100100E0	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
0x10010100	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
0x10010120	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
0x10010140	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
0x10010160	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
0x10010180	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
0x100101A0	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
0x100101C0	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
0x100101E0	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	

0x10010000 (.data)

Hexadecimal Addresses

Hexadecimal Values

ASCII

to τμήμα της μνήμης που βλέπουμε

Σχήμα 4: Το υποπαράθυρο μνήμης δεδομένων.

3 Παραδοτέο

Αποθηκεύστε τον αριθμό μητρώου σας (matriculation number - matric) στη μνήμη γράφοντάς τον στη θέση της ετικέτας matric: χρησιμοποιώντας τον editor του MARS, αντικαταστήστε το 0 με τον αριθμό μητρώου σας στην παραπάνω ετικέτα. Αλλάξτε την γραμμή 30 του lab01.asm (li \$s0, 100) με μία εντολή assembly, ώστε να φορτώνει στον καταχωρητή \$s0 την τιμή που είναι αποθηκευμένη στην διεύθυνση μνήμης που αντιστοιχεί στο label matric. Η διεύθυνση του matric υπάρχει στον καταχωρητή \$a0 ήδη από την εντολή της γραμμής 18. Ξανακάνετε assemble το πρόγραμμά σας αφού τώρα πια έχει αλλάξει και ελέγξτε το.

3.1 Έλεγχος ορθότητας

Στο repository θα βρείτε το αρχείο Lab01Test.java καθώς και το munit.jar. Με αυτά υπάρχει η δυνατότητα αυτόματου ελέγχου προγραμμάτων assembly. Ο έλεγχος γίνεται θέτοντας τιμές σε καταχωρητές και μνήμη πριν την εκτέλεση του προγράμματος και συγκρίνοντας τιμές καταχωρητών και μνήμης με προϋπολογισμένες τιμές που θα πρέπει να δίνει ένα σωστό πρόγραμμα. Το munit.jar προέρχεται από τον Zachary Kurmas του Grand Valley State University με λίγες μικροαλλαγές από τον διδάσκοντα. Χρησιμοποιεί το JUnit και τον MARS για να δίνει τιμές εισόδου, να τρέχει προγράμματα και να ελέγχει την μνήμη και τις τιμές καταχωρητών στο τέλος της εκτέλεσης. Περισσότερες πληροφορίες μπορείτε να δείτε σε αυτόν τον σύνδεσμο.

Για να τρέξετε το τεστ, δώστε τις εντολές:

```
javac -cp munit.jar Lab01Test.java
java -jar munit.jar lab01.asm Lab01Test.class
```

Η πρώτη εντολή μεταγλωττίζει το αρχείο Lab01Test.java και η δεύτερη τρέχει το τέστ στο πρόγραμμα lab01.asm. Αν όλα είναι εντάξει θα δείτε το μήνυμα All tests (1) passed. Ο αριθμός στην παρένθεση είναι ο αριθμός των τεστ, που σε αυτή την άσκηση είναι 1.

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

```
Failure: verify_lab01(Lab01Test): $s0 should be 0 expected:<0> but was:<100>
Tests run: 1, Failures: 1
```

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

Ως μέρος του παραδοτέου αυτής της άσκησης θα πρέπει να αλλάξετε την τιμή της μεταβλητής `matricNumber` (γραμμή 12 στο `Lab01Test.java`) ώστε να είναι ο δικό σας αριθμός μητρώου. Με αυτή την αλλαγή και αυτές στο αρχείο `.asm` που περιγράφηκαν παραπάνω, βεβαιωθείτε ότι το τέστ πλέον περνάει με επιτυχία.

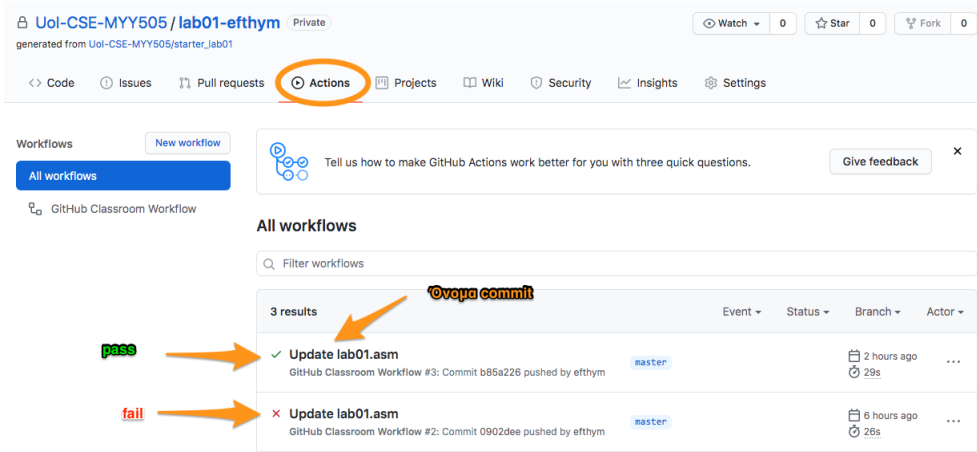
Αντίστοιχα αρχεία labXXTest.java θα υπάρχουν και για τις επόμενες εργαστηριακές ασκήσεις. Γενικά δεν θα χρειάζεται να κάνετε πολλές αλλαγές σε αυτά, ούτε είναι απαραίτητο να καταλαβαίνετε πλήρως

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 εξετάζονται οι ασκήσεις που παραδίδετε. 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!

3.2 Αυτόματος έλεγχος ασκήσεων

Κάθε φορά που προωθείτε (push) αλλαγές στο GitHub σε κάποια άσκηση assembly, θα τρέχουν αυτόματα τέστ ελέγχου. Αυτό γίνεται με κάποιους αυτοματισμούς, actions, που παρέχει το GitHub. Έτσι μετά από κάθε προώθηση μπορεί κανείς να έχει μια γρήγορη εκτίμηση αν η άσκηση που παρέδωσε περνάει τα βασικά τεστ.

Για να δείτε τα αποτελέσματα, κάνετε κλικ στο tab Actions και θα εμφανιστεί κάτι σαν την εικόνα 5, όπου φαίνονται τα τελευταία στιγμιότυπα (commit) και στα αριστερά τους ένα εικονίδιο που εξηγεί αν περνάει ή όχι τον αυτόματο έλεγχο: κόκκινο αν αποτυγχάνει, πράσινο αν περνάει και πορτοκαλί για όσο διάστημα το τεστ τρέχει.



Σχήμα 5: GitHub Actions.

Λεπτομέρειες μπορείτε να δείτε κάνοντας κλικ σε ένα στιγμιότυπο οπότε εμφανίζεται κάτι σαν την εικόνα 6. Για την ακρίβεια πρέπει να κάνετε κλικ στο Autograding για να εμφανιστεί το σκούρο πλαίσιο με τα αποτελέσματα. Μετά κάνετε κλικ στο "Run education/autograding@v1" για να δείτε τα μηνύματα που δίνει το munit.

Κατά την βαθμολόγηση της άσκησης μπορεί να χρησιμοποιηθούν επιπρόσθετα τεστ που δεν ήταν έτοιμα κατά την ανακοίνωση της άσκησης, και επιπλέον εξετάζεται και η ποιότητα των προγραμμάτων, οπότε το πράσινο σήμα δεν σημαίνει και βαθμό 10!

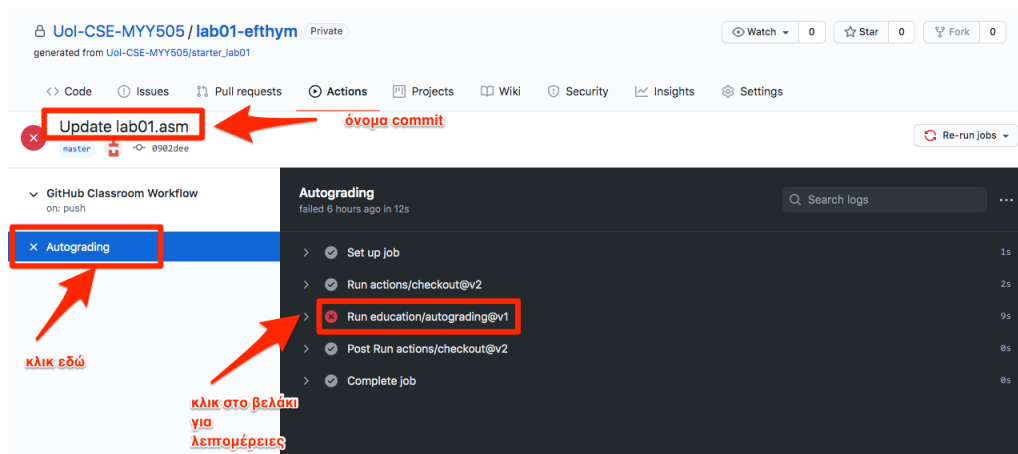
3.3 Παράδοση της άσκησης

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

```
git add lab01.asm Lab01Test.java
git commit -m "Completed lab01"
git push origin master
```

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 και στην ανάρτηση στο Piazza.

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



Σχήμα 6: Μηνύματα αυτόματου ελέγχου.

μητρώου κλπ). Χρησιμοποιώ προγράμματα που τρέχουν τις ασκήσεις αυτόματα και απαιτούν τα συγκεκριμένα ονόματα αρχείων και καταλόγων που δίνονται!

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 βαθμολογηθεί! Επιπλέον, αν το αποθετήριο είναι public, θα είναι ορατό σε οποιονδήποτε που θα μπορεί να το αντιγράψει. Εννοείται ότι δεν επιτρέπεται να έχετε τις λύσεις σε public αποθετήριο, ακόμη και ως αντίγραφο.