MILE Simulator

Version 1.0

User's Manual

Νοέμβριος, 2011

Περιεχόμενα

1. Εισαγωγή στον προσομοιωτή

Το έγγραφο αυτό αποτελεί σύντομη εισαγωγή στο πρόγραμμα **MILE (MIPS Interactive Learning Environment)**, έναν user friendly προσομοιωτή του επεξεργαστή MIPS. Ο προσομοιωτής δημιουργήθηκε ως εκπαιδευτικό εργαλείο για το μάθημα "**Αρχιτεκτονικής Υπολογιστών**" του τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Ξεκίνησε ύστερα από την σχετική πρόταση του Computing Systems Laboratory (CSLab), υπό την εποπτεία του κ. Κωνσταντίνου Νίκα.

Ο προσομοιωτής αυτός προσφέρει στο χρήστη τη δυνατότητα να αναπτύξει και να μεταγλωττίσει προγράμματα σε MIPS assembly language. Κατ' επέκταση, προσομοιώνει την εκτέλεση των προγραμμάτων αυτών σε MIPS architecture. Η αρχιτεκτονική και το σύνολο των εντολών του MIPS processor, αναλύονται εκτενέστερα στα πλάισια του μαθήματος.

1.1 Εγκατάσταση

Η παρούσα version 1.0 του simulator, είναι διαθέσιμη στο:

http://www.cslab.ece.ntua.gr/courses/comparch/assign.go

όπου υπάρχουν και οι 3 επιμέρους εκδόσεις, ανάλογα με το μηχάνημά σας:

- Για **Windows.**
- Για **Linux 32-bit.**
- Για **Linux 64-bit.**

Από τους αντίστοιχους συνδέσμους, κατεβάστε το φάκελο, που περιέχει το εκτελέσιμο αρχείο και κάποιες κατάλληλες για την εκτέλεση βιβλιοθήκες. Επιλέγοντας απλά την εφαρμογή με το όνομα **msim**, μπορείτε άμεσα να ξεκινήσετε την χρήση του προσομοιωτή. Για τις linux εκδόσεις, συνίσταται να εκτελέσετε αρχικά το run.sh, σε περίπτωση που δεν είστε βέβαιοι ότι έχετε τις βιβλιοθήκες της Qt.

Καλή απόλαυση... ☺

1.2 Βοήθεια - Διευκρινήσεις

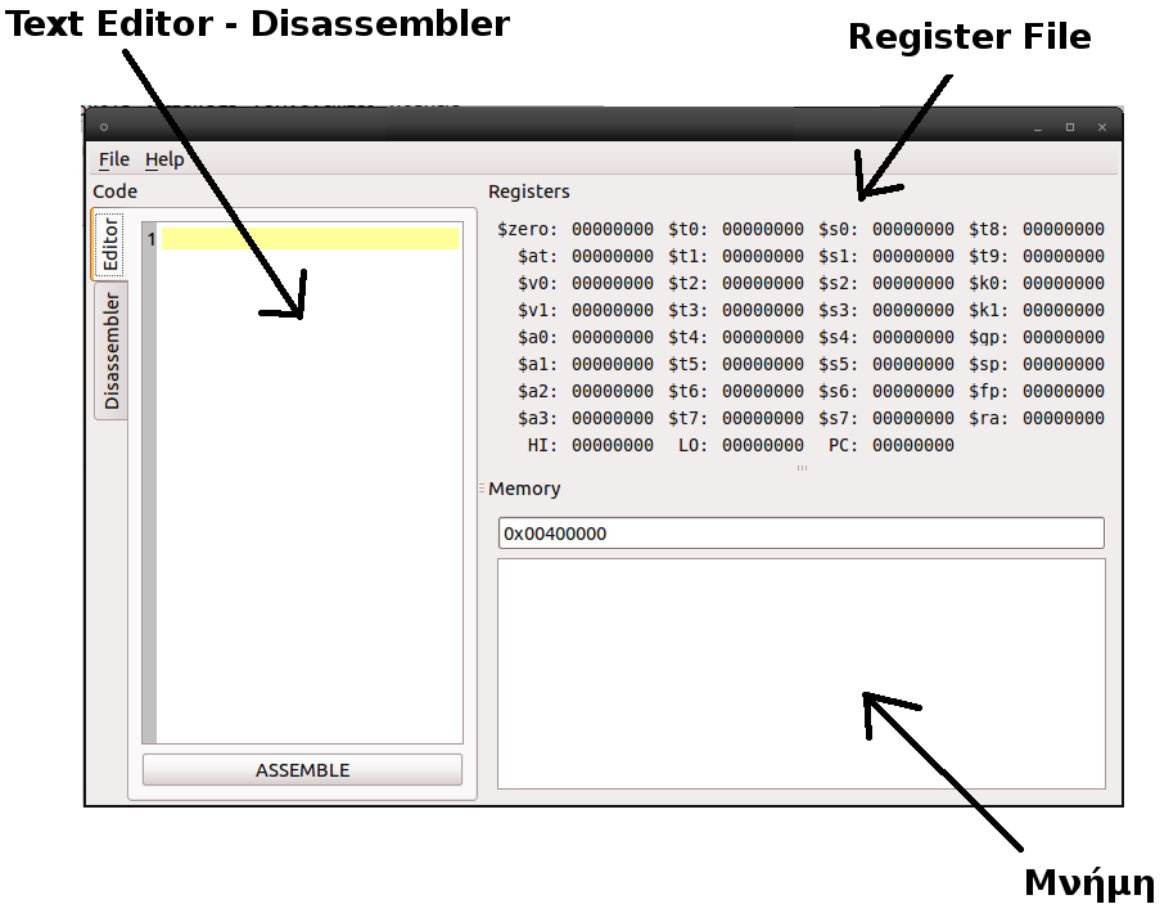
Για περισσότερες διευκρινήσεις και απορίες, απευθυνθείτε στο mail:

mile-dev@cslab.ece.ntua.gr

2. Ξεκινώντας με τον προσομοιωτή

Ανοίγοντας τον προσομοιωτή, θα έχετε μπροστά σας την παρακάτω εικόνα (*Εικόνα 1*). Στο παράθυρο αυτό, βλέπουμε τα βασικά τμήματα του:

- Text Editor – Disassembler
- Register file
- Μνήμη

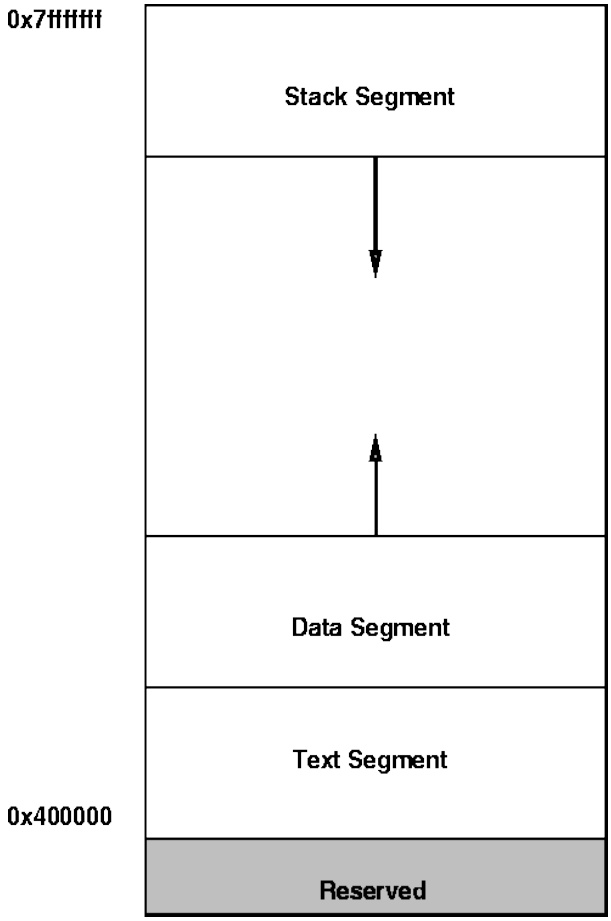


Εικόνα 1 – Το κεντρικό παράθυρο του προσομοιωτή

2.1 Το memory layout του MIPS

Ο προσομοιωτής ακολουθεί τον τρόπο οργάνωσης της μνήμης του MIPS (*Εικόνα 2*) .

Σύμφωνα με το **MIPS memory layout,** οι εντολές τοποθετούνται στο Text Segment (address 0x0040000). Παρατηρείστε λοιπόν ότι και ο κώδικας που κάθε φορά μεταγλωττίζετε ξεκινά και τοποθετείται από τη διεύθυνση αυτή.



Εικόνα 2 – MIPS memory layout

2.2 Οι κλήσεις συστήματος του MIPS

Ο MIPS περιλαμβάνει ένα σύνολο system services, τα **Systems Calls,** για την υλοποίηση πρόσθετων λειτουργιών, όπως είσοδος / έξοδος δεδομένων από το χρήστη, τερματισμός της εκτέλεσης κλπ.

Τα βήματα που ακολουθούνται σε μια κλήση συστήματος είναι:

i. Τοποθέτηση στον καταχωρητή $v0 του κωδικού της επιθυμητής κλήσης συστήματος
ii. Εκτέλεση της εντολής syscall

Ο προσομοιωτής χρησιμοποιεί την κλήση συστήματος **exit**, για τον τερματισμό της εκτέλεσης (**exit – terminate execution**) του κώδικα, η οποία υλοποιείται με τις εντολές:

```
ori $v0, $zero, 10   # Φορτώστε το 10 στον καταχωρητή
syscall              # Πραγματοποιήστε την κλήση συστήματος
```

Ο ρόλος της exit system call αναλύεται στη συνέχεια (βλέπε επόμενη ενότητα 2.3του user's manual) .

2.3 Ξεκινήστε και τερματίστε τον κώδικά σας

Για να οριστούν κατάλληλα τα σημεία “εκκίνησης“ και “τερματισμού” της εκτέλεσης του κώδικά σας, ο μεταγλωττιστής εξ ' ορισμού τοποθετεί στην αρχή του Test Segment, τις εξής 3 εντολές:

jal main

ori $v0, $zero, 10

syscall

Η **jal main** μεταφέρει την εκτέλεση στην αρχή του κώδικά σας, μέσω “jump and link”, και στον καταχωρητή της return address αποθηκεύεται η διεύθυνση της επόμενης εντολής, της **ori** ($ra = 0x0040004). Η **ori** και η **syscall,** υλοποιούν την κλήση συστήματος για τον τερματισμό της εκτέλεσης του προγράμματός

<u>Το “σημείο εκκίνησης” του κώδικά σας:</u>

Τοποθετώντας την ετικέτα “main”, προσδιορίστε την αρχή του προγράμματός σας. Σε διαφορετική περίπτωση, η εκτέλεση θα ξεκινήσει από την πρώτη εντολή του text editor.

<u>Το “σημείο τερματισμού” του κώδικά σας:</u>

Με δεδομένο ότι ο $ra έχει τη διεύθυνση της **ori**, και η τιμή του δεν μεταβάλλεται μέσα στο πρόγραμμα, προσδιορίστε το τέλος του προγράμματός σας τοποθετώντας την εντολή:

jr $ra

Προφανώς, αν χρησιμοποιείτε επιπλέον εντολές jal του κώδικα, μπορείτε να διατηρήσετε την τιμή $ra = 0x0040004 σε έναν άλλον καταχωρητή και με αυτόν να κάνετε την jr:

jr $(όνομα καταχωρητή)

Εικόνα 3

Εναλλακτικά, σε οποιαδήποτε σημείο του κώδικα επιθυμείτε να τερματίσετε, μπορείτε ανεξάρτητα να χρησιμοποιήσετε απευθείας την κλήση συστήματος:

ori $v0, $zero, 10

syscall

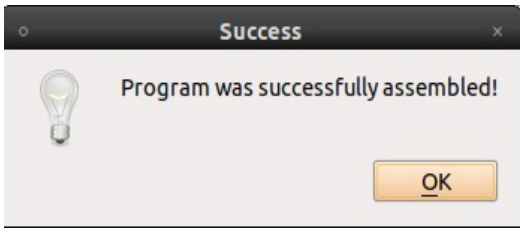
2.4 Μεταγλωττίστε τα προγράμματα σας

Για να γράψετε ένα assembly program επιλέξτε στα αριστερά του παραθύρου την καρτέλα **editor**, ώστε να μεταβείτε στον text editor του προσομοιωτή. Εκεί μπορείτε να αναπτύξετε τον επιθυμητό κώδικα σε assembly.

Επιπλέον, μπορείτε να δημιουργήσετε και να προσπελάσετε assembly files, επέκτασης **.s** . Τις ανάλογες ενέργειες μπορείτε να τις επιλέξετε από το αντίστοιχο μενού "**File**".

Μπορείτε να μεταγλωττίσετε το πρόγραμμά σας με το αντίστοιχο κουμπί "ASSEMBLE". Σε περίπτωση ανεπιτυχούς μεταγλώττισης, σας επιστρέφετε το σχετικό μήνυμα με τη γραμμή του συντακτικού λάθους (*Εικόνα 4*).

Εικόνα 4 – Μήνυμα λάθους



Εικόνα 5 – Επιτυχής μεταγλώττιση

Σε διαφορετική περίπτωση, ενημερώνεστε με το κατάλληλο μήνυμα (*Εικόνα 5*), και μεταφέρεστε στην καρτέλα του **disassembler** , από όπου μπορείτε να πραγματοποιήσετε την εκτέλεση (βλέπε επόμενη ενότητα 3.1 του user's manual) .

3. Εκτέλεση προγραμμάτων

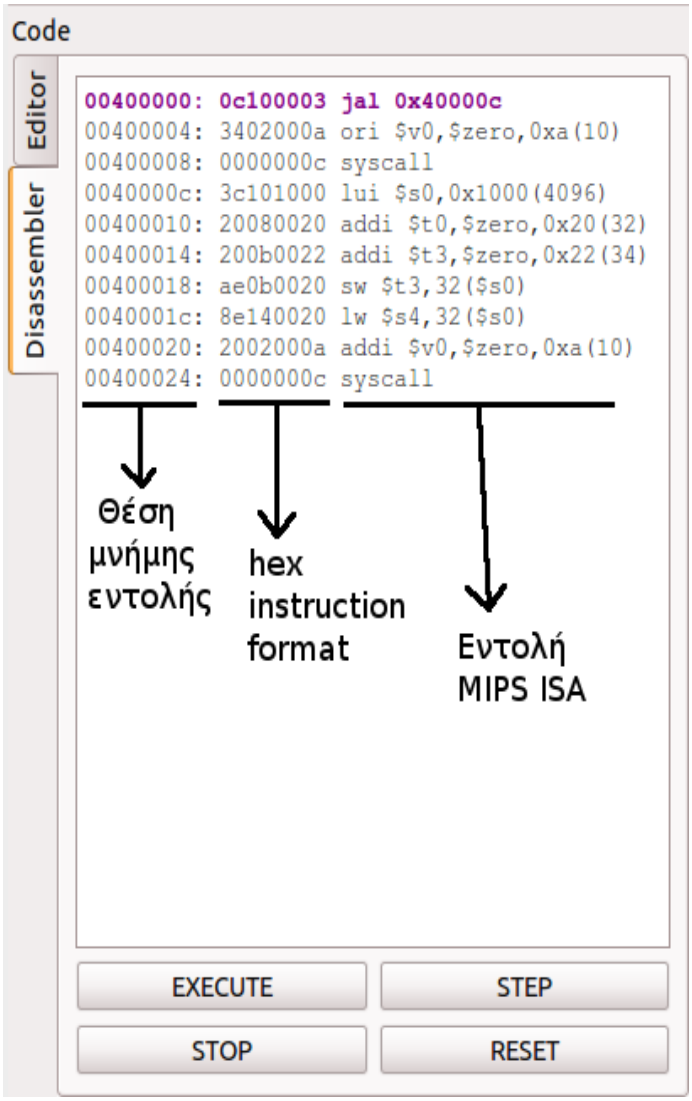
Αφού ολοκληρώσατε με επιτυχία τη μεταγλώττιση του κώδικά σας, έχετε μεταφερθεί στην καρτέλα του **disassembler.** (*Εικόνα 6*) Από εδώ, μπορείτε να εκτελέσετε το πρόγραμμά σας και να πάρετε τα αναμενόμενα αποτελέσματα.

Στο παράθυρο το **disassembler,** έχουμε ανά γραμμή (από αριστερά προς τα δεξιά) :

- Την **εντολή** σε MIPS ISA.
- Τα **32 bits της εντολής** (instruction format) σε hexadecimal μορφή.
- Τη **θέση της μνήμης** που είναι αποθηκευμένη η εντολή.

Για την εκτέλεση, μπορείτε να επιλέξετε ανάμεσα σε 4 διαφορετικές ενέργειες:

- ***EXECUTE :*** Εκτελεί τον κώδικα μέχρι το τέλος, εμφανίζοντας την τελική κατάσταση της μνήμης και των καταχωρητών
- ***STEP :*** Εκτελεί μια – μια τις εντολές. Κάθε φορά, **η εντολή που πρόκειται να εκτελεστεί, καταδεικνύεται με μωβ χρώμα.**
- ***STOP :*** Διακόπτει την εκτέλεση του προγράμματος.
- ***RESET :*** Επαναφέρει τον προσομοιωτή στην αρχική κατάσταση, ώστε να ξαναρχίσει η εκτέλεση από την αρχή.

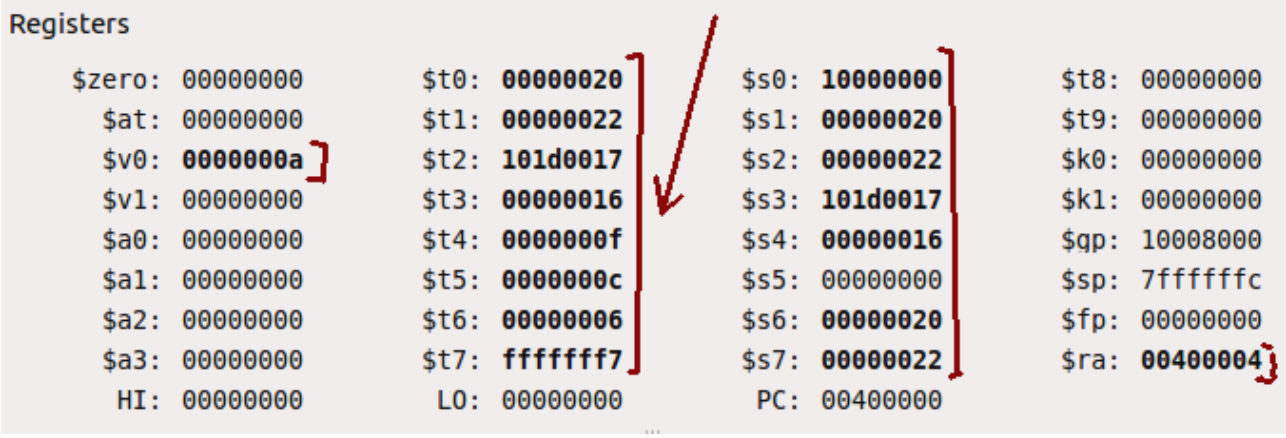


Εικόνα 6 - Disassembler

Τέλος, κάθε φορά που επιθυμείτε να αλλάξετε πρόγραμμα ή να τροποποιήσετε το πρόγραμμα που ήδη εκτελείτε, απλά μεταβείτε στον editor, πραγματοποιήστε τις επιθυμητές αλλαγές και κάνετε ξανά assemble.

3.1 Παρατηρώντας το register file

Στο πάνω δεξιά τμήμα του παραθύρου, στο πεδίο **Registers**, απεικονίζεται σε δεκαεξαδική μορφή το περιεχόμενο των καταχωρητών του Register File. Για τη διευκόλυνση του χρήστη, σε κάθε βήμα της εκτέλεσης (επιλογή step), οι καταχωρητές που αλλάζουν τιμή, επισημαίνονται κατάλληλα. Με τον ίδιο τρόπο, κατά την συνολική εκτέλεση (επιλογή execute), επισημαίνονται όλοι οι καταχωρητές των οποίων η τιμή διαφοροποιήθηκε. (*Εικόνα 7*)

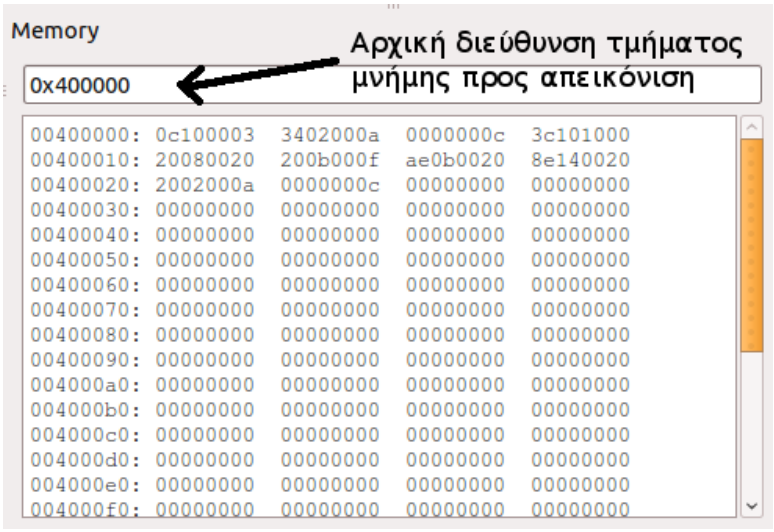


Εικόνα 7 – Αλλαγές στο περιεχόμενο των καταχωρητών

3.2 Παρατηρώντας τη μνήμη

Στο πεδίο **Memory** του προσομοιωτή (*Εικόνα 8*), απεικονίζεται τμήμα της μνήμης του προγράμματος.

Στο κυλιόμενο παράθυρο τυπώνονται **100 διαδοχικές θέσεις** μνήμης, με αρχή την επιλεγμένη από το χρήστη διεύθυνση στο πλαίσιο. Κατά συνέπεια, αν δώσετε στο πλαίσιο αυτό την κατάλληλη διεύθυνση (hex μορφή), θα τυπωθεί το περιεχόμενο του τμήματος της μνήμης που επιθυμείτε να προβάλλετε.



Εικόνα 8 – Περιεχόμενα μνήμης

4. Developers - Contributors

Ο προσομοιωτής **MILE** είναι αποτέλεσμα της συνεργασίας των φοιτητών του τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του ΕΜΠ:

- **Νικόλαος Βάθης**
- **Δημήτριος Σταμούλης**
- **Νικόλαος Τσιρώνης**
- **Χαράλαμπος Χαλιός**

Για τυχόν παρατηρήσεις σας, αναφορά σφαλμάτων, καθώς και επιθυμία συμμετοχής στην ομάδα ανάπτυξης του προσομοιωτή, επικοινωνήστε μαζί μας στο mile-dev@cslab.ece.ntua.gr