ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2018 – 2019

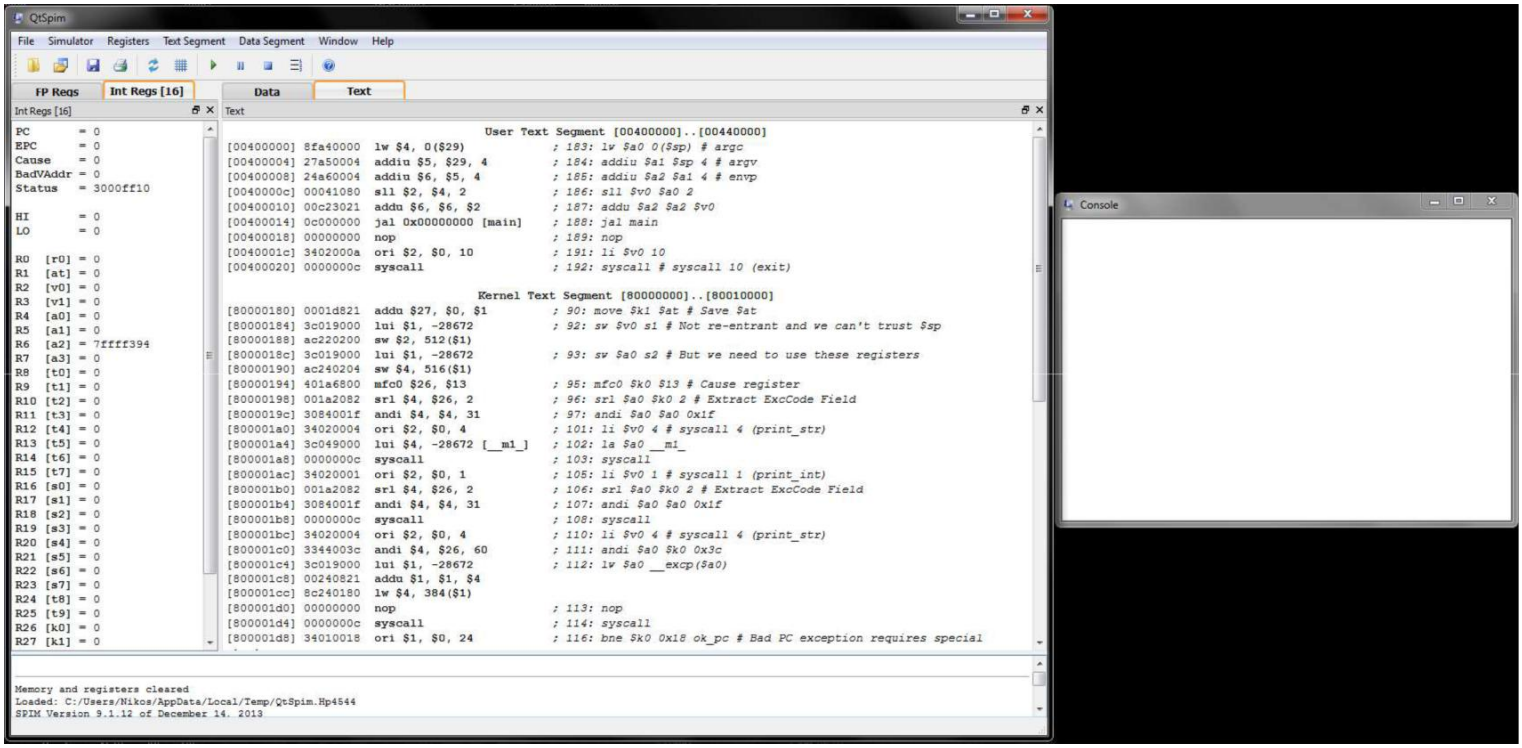
ΕΡΓΑΣΤΗΡΙΟ 1

Το εργαστήριο αυτό είναι εισαγωγικό και έχει σκοπό την εξοικείωση του φοιτητή με τον προσομοιωτή SPIM π ου θα χρησιμοποιηθεί κατά τη διάρκεια του εξαμήνου.

Το QtSPIM είναι το πρόγραμμα που θα χρησιμοποιήσουμε και αποτελεί ένα προσομοιωτή ενός επεξεργαστή MIPS. Κατά τη διάρκεια του εργαστηρίου θα μάθετε να προγραμματίζετε σε γλώσσα Assembly του επεξεργαστή MIPS και τα προγράμματά σας θα εκτελούνται στο QtSPIM, το οποίο μπορείτε να το κατεβάσετε από τη σελίδα του μαθήματος στο e-class.

Το QtSPIM:

Όταν ανοίξετε το QtSPIM εμφανίζεται στην οθόνη σας το παράθυρο που φαίνεται στην παρακάτω εικόνα.



Όπως βλέπετε χωρίζεται στις εξής περιοχές:

1. **Γραμμή Μενού Επιλογών:** Εδώ υπάρχουν τα διάφορα μενού που είναι διαθέσιμα από το πρόγραμμα.
2. **Γραμμή Εικονιδίων Μενού Εντολών:** Εδώ υπάρχουν εικονίδια που αντιστοιχούν σε εντολές που βρίσκονται στα διάφορα μενού και χρησιμοποιούνται πιο συχνά από τους χρήστες.
3. **Περιοχή Παρουσίασης Καταχωρητών:** Εδώ παρουσιάζονται τα περιεχόμενα των καταχωρητών. Υπάρχουν δυο tabs. Το ένα αφορά τους καταχωρητές που χρησιμοποιούνται όταν προγραμματίζουμε με integer arithmetic (Int Regs) και το άλλο όταν προγραμματίζουμε με floating point arithmetic (FP Regs).
4. **Περιοχή Παρουσίασης Μνήμης:** Εδώ εμφανίζονται τα περιεχόμενα της μνήμης. Θεωρείται ότι η μνήμη χωρίζεται σε δυο κομμάτια, των δεδομένων και του προγράμματος. Αντίστοιχα υπάρχουν και δυο tabs. Το Data tab μας δείχνει τις διευθύνσεις και τα περιεχόμενα στα κομμάτια Data και Stack της μνήμης. Το Text tab μας δείχνει τις εντολές MIPS που έχουν φορτωθεί στη μνήμη για να εκτελεστούν. Αριστερά εμφανίζεται η διεύθυνση μνήμης μιας εντολής, στη συνέχεια τα περιεχόμενα της μνήμης σε δεκαεξαδική μορφή, η πραγματική εντολή MIPS που εκτελείται και μετά το ; εμφανίζεται ο κώδικας Assembly και οποιοδήποτε σχόλιο γράψατε στον κώδικά σας.
5. **Περιοχή Ενημέρωσης:** Εδώ εμφανίζονται οι ενέργειες που εκτελούνται από τον προσομοιωτή.

Το πρώτο πρόγραμμα σε Assembly για τον MIPS και πως εκτελείται με το QtSPIM:

Για να προγραμματίζετε σε Assembly για τον MIPS (από εδώ και στο εξής μόνο Assembly) θα ακολουθείτε πάντα τα παρακάτω βήματα:

1. Η ανάπτυξη του κώδικα θα γίνεται σε έναν απλό κειμενογράφο, π.χ. το Notepad.
2. Τα αρχεία θα τ α σώζετε με κατάληξη .s και θα φροντίζετε να έχει αποθηκευθεί χωρίς ιδιότητες αρχείου κείμενου. (π.χ. στο Notepad στην επιλογή Save as type επιλέγουμε All files και όχι Text document).
3. Ανοίγετε το QtSPIM και φορτώνετε το αρχείο με τον κώδικά σας πατώντας το κουμπί Load File (από το μενού File).
4. Εκτελείτε το πρόγραμμα είτε κανονικά πατώντας το κουμπί Run (από το μενού Simulator) είτε σε debugging mode πατώντας το κουμπί Single Step (από το μενού Simulator).

<u>Μερικά σχόλια σχετικά με το πρόγραμμα:</u>

- Ό,τι ακολουθεί το σύμβολο # είναι σχόλια.
- Ό,τι ακολουθεί το .data αποτελεί δεδομένα που αποθηκεύονται στο data segment της μνήμης και εμφανίζονται στο tab data.
- Ό,τι ακολουθεί το .text αποτελεί κώδικα, δηλαδή εντολές που αποθηκεύονται στο κομμάτι εντολών της μνήμης και εμφανίζονται στο tab text.
- Στο πρόγραμμα αυτό γίνεται παρουσίαση των βασικών System Calls του MIPS. Τα System Calls (Κλήσεις Συστήματος) είναι ειδικές συναρτήσεις που χρησιμοποιούνται –κυρίως– για την επικοινωνία με τον χρήστη, όπως για παράδειγμα εμφάνιση μηνύματος, εκτύπωση ακεραίου, λήψη ακεραίου από το πληκτρολόγιο, κλπ.

Στη συνέχεια ακολουθεί ένας συνοπτικός πίνακας των system calls που θα πρέπει να θυμάστε ή να έχετε μαζί σας όταν γράφετε ένα πρόγραμμα.

Service	System Call Code	Arguments	Result
Print_int	1	$a0 = integer	
Print_float	2	$f12 = float	
Print_double	3	$f12 = double	
Print_string	4	$a0 = string	
Read_int	5		Integer (in $v0)
Read_float	6		Float (in $f0)
Read_double	7		Double (in $f0)
Read_string	8	$a0 = buffer, $a1 = length	
Sbrk	9	$a0 = amount	Address (in $v0)
exit	10		

Στη συνέχεια ακολουθεί ένα πρόγραμμα σε Assembly. Αντιγράψτε το σε έναν κειμενογράφο και ακολουθήστε τα παραπάνω βήματα για να τρέξετε το πρόγραμμα στο QtSPIM.

```
     .data
# Constant strings to be output to the terminal
promptInt:   .asciiz "Please input an integer: "
resultInt:   .asciiz "Next integer is: "
linefeed:    .asciiz "\n"
enterkey:    .asciiz "Press any key to end program."

     .text
main:
# prompt for an integer
      li    $v0, 4            # code for print_string
      la    $a0, promptInt    # point $a0 to prompt string
      syscall

# get an integer from the user
      li    $v0, 5            # code for read_int
      syscall                 # get int from user → return in $v0
      move  $t0, $v0          # move the resulting int to $t0

# compute the next integer
      addi  $t0, $t0, 1       # t0 ← t0 + 1

# print out text for the result
      li    $v0, 4            # code for print_string
      la    $a0, resultInt    # point $a0 to result string
      syscall                 #print the result string

# print out the result
      li    $v0, 1            # code for print_int
      move  $a0, $t0          # put result in $a0
      syscall                 # print out the result
```

```
# print out a line feed
      li    $v0, 4            # code for print_string
      la    $a0, linefeed     # point $a0 to linefeed string
      syscall                 # print linefeed

# wait for the enter key to be pressed to end program
      li    $v0, 4            # code for print_string
      la    $a0, enterKey     # point $a0 to enterKey string
      syscall                 # print enterKey

# wait for input by getting an integer from the user (integer is
ignored)
      li    $v0, 5            # code for read_int
      syscall                 # get int from user → returned in $v0

# All done, thank you!
      li    $v0, 10           # code for exit
      syscall                 # exit program
```