3ο Εργαστήριο Αρχιτεκτονικής Η/Υ: MIPS assembly: Υπολογισμός γινομένου στοιχείων συνδεδεμένης λίστας με αναδρομή

Α. Ευθυμίου

Παραδοτέο: Τρίτη 7 Νοέμβρη, 23:59

Το αντικείμενο αυτής της άσκησης είναι ένα πρόγραμμα που υπολογίζει το γινόμενο των αριθμών (θετικών ακέραιων) που είναι αποθηκευμένοι σε μία συνδεδεμένη λίστα. Σε αυτή την εργαστηριακή άσκηση θα γράψετε ένα προγράμμα assembly που χρησιμοποιεί υπορουτίνες και αναδρομή. Θα πρέπει να έχετε μελετήσει τα μαθήματα για τη γλώσσα assembly του MIPS που αντιστοιχούν μέχρι και την ενότητα 2.8 του βιβλίου (διάλεξη 7).

Έχετε 2 εβδομάδες διαθέσιμες γι'αυτή την άσκηση. Μη την αφήσετε όμως για την τελευταία στιγμή! Η χρήση της στοίβας για υπορουτίνες είναι αρκετά περίπλοκη και θέλει σκέψη και πιθανόν πολλές δοκιμές για να υλοποιήσετε σωστά αναδρομικές συναρτήσεις.

Για να ξεκινήσετε, ακολουθήστε τον σύνδεσμο https://classroom.github.com/a/rZ1QZ2iz. Κλωνοποιήστε το αποθετήριο που θα δημιουργηθεί και συνεχίστε όπως στις προηγούμενες ασκήσεις.

1 Μέρος 1: Υπορουτίνα πολλαπλασιασμού mulproc

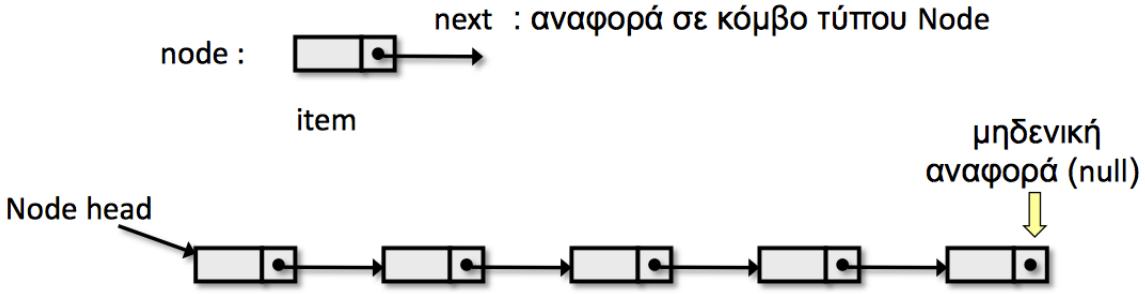
Η άσκηση απαιτεί τον υπολογισμό του γινομένου 2 αριθμών. Επειδή η έκδοση του MIPS που χρησιμοποιούμε δεν έχει εντολή πολλαπλασιασμού, θα πρέπει να υλοποιηθεί ως υπορουτίνα, με όνομα mulproc, η οποία κάνει τον υπολογισμό με επαναλαμβανόμενες προσθέσεις. Μην χρησιμοποιήσετε τις (ψευτο-)εντολές πολλαπλασιασμού του MIPS, ούτε ολισθήσεις. Υποθέστε ότι οι αριθμοί είναι απρόσημοι (θετικοί ή μηδέν) και αγνοήστε τυχόν υπερχειλίσεις.

Στον σκελετό προγράμματος που δίνεται, προσθέστε τον κώδικά σας στο σημείο που δείχνει το label mulproc.

Ελέγξτε ότι η υπορουτίνα mulproc δουλεύει σωστά πρίν προχωρήσετε στην επόμενη υπορουτίνα γιατί είναι απλούστερη ως προς τη χρήση της στοίβας αφού δεν καλεί άλλες υπορουτίνες. Θα είναι πολύ πιο δύσκολο να καταλάβετε τυχόν λάθη αν αφήσετε τον έλεγχο όλης της άσκησης για το τέλος. Το πρώτο τεστ του Lab03Test.java ελέγχει αυτή την υπορουτίνα μόνη της.

2 Συνδεδεμένη λίστα

Η συνδεδεμένη λίστα είναι μία από τις πιο γνωστές και απλές δομές δεδομένων. Ομαδοποιεί σε μια δομή (συχνά ονομάζεται struct, ή class), έναν αριθμό από δεδομένα και έναν **δείκτη** στο επόμενο στοιχείο της λίστας. Ο δείκτης παίρνει την τιμή 0 (null) για να δείξει ότι δεν υπάρχει επόμενο στοιχείο στη λίστα. Στο σχήμα 1, από τις διαλέξεις του μαθήματος "Δομές Δεδομένων", φαίνεται σχηματικά μια λίστα.



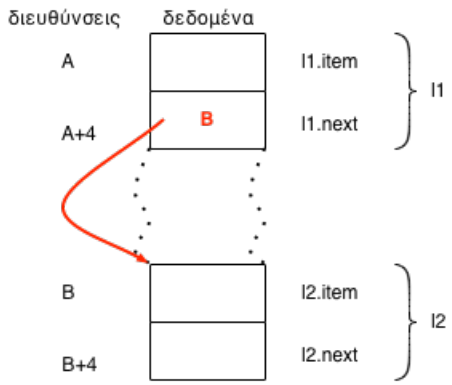
Σχήμα 1: Σχηματική αναπαράσταση λίστας.

Στην άσκηση αυτή, κάθε στοιχείο της λίστας θα περιέχει μόνο έναν απρόσημο (θετικό), ακέραιο αριθμό 32 bit και, φυσικά, τον δείκτη στο επόμενο στοιχείο. Σε C θα γράφαμε:

```
struct Node {
  unsigned int item;  // data
  struct Node  *next; // pointer to next element
}
```

Όταν ο μεταφραστής (compiler) οργανώνει πως θα τοποθετηθούν στη μνήμη τα δεδομένα μιας δομής, κάνει κάτι αντίστοιχο με τους πίνακες: τοποθετεί τα δεδομένα και το δείκτη σε συνεχόμενες θέσεις μνήμης. Έτσι αν βάλει το πεδίο/μέλος item στη διεύθυνση A, ο δείκτης next θα τοποθετηθεί στη διεύθυνση A+4 και η διεύθυνση A+8 θα είναι διαθέσιμη για άλλα δεδομένα. Στους παραπάνω υπολογισμούς υποθέτουμε 32 bit ακέραιους (int) και διευθύνσεις μνήμης των 32 bit, όπως παντού στο μάθημα. Επίσης, για τον MIPS, υποθέτουμε ότι η αρχική διεύθυνση A είναι πολλαπλάσιο του 4, ώστε τα δεδομένα να είναι ευθυγραμμισμένα (aligned) στη μνήμη.

Η διεύθυνση ενός αντικειμένου/μεταβλητής της δομής είναι η ίδια με τη διεύθυνση του πρώτου πεδίου του. Έτσι όταν ένας δείκτης δείχνει στο επόμενο στοιχείο της λίστας, η τιμή τού δείκτη είναι η διεύθυνση του πρώτου πεδίου δεδομένων του επόμενου στοιχείου της λίστας. Πιο συγκεκριμένα (βλ. σχήμα 2), υποθέστε ότι έχουμε 2 στοιχεία λίστας της παραπάνω δομής: `l1, l2`. Αν το `l1` βρίσκεται στη διεύθυνση A, το `l1.item` βρίσκεται στη διεύθυνση A και το `l1.next` βρίσκεται στη διεύθυνση A+4. Αν το `l2` βρίσκεται στη διεύθυνση B, το `l2.item` βρίσκεται στη διεύθυνση B και το `l2.next` βρίσκεται στη διεύθυνση B+4. Η τιμή του `l1.next` είναι B.



Σχήμα 2: Οργάνωση λίστας στη μνήμη.

3 Υπορουτίνα υπολογισμού γινομένου δεδομένων λίστας

Για το δεύτερο μέρος της άσκησης θα υλοποιήσετε μια υπορουτίνα που υπολογίζει, **αναδρομικά**, το γινόμενο των δεδομένων μιας συνδεδεμένης λίστας, σε assembly. Παρακάτω δίνεται η υλοποίηση σε ψευτοκώδικα:

```
struct Node {
  unsigned int item;  // data
  struct Node  *next; // pointer to next element
}

unsigned int listProd (Node nptr) {
  if (nptr == NULL)  // NULL == 0
      return 1;
```

```
  else
      return mulproc(nptr.item, listProd(nptr.next));
}
```

Η υπορουτίνα θα πρέπει **να χρησιμοποιεί αναδρομή υποχρεωτικά** και **να ακολουθεί τις συμβάσεις του MIPS ως προς τη χρήση καταχωρητών και στοίβας**. Ακόμη και αν τα αποτελέσματα είναι σωστά θα χάσετε πολύ μεγαλο μέρος των βαθμών αν δεν ακολουθήσετε τα παραπάνω.

Στο lab03.asm υπάρχει η ετικέτα της υπορουτίνας listProd, όπου και πρέπει να γράψετε κώδικα.

4 Εξέταση περιεχομένων στοίβας

Η στοίβα παίζει σημαντικό ρόλο στις κλήσεις συναρτήσεων και ιδιαίτερα στην αναδρομή. Στον MARS για να παρατηρείτε τί συμβαίνει στην περιοχή της μνήμης που αντιστοιχεί στη στοίβα, αλλάξτε την επιλογή στο κάτω μέρος του παραθύρου Data Segment σε "current $sp".

Οι φοιτητές του τμήματος Γιώργος Ζάχος (προπτ.) και Πέτρος Μανούσης (υποψ. διδάκτορας), έχουν αναπτύξει ένα εργαλείο που παρουσιάζει πολύ καλύτερα τη περιοχή της μνήμης που χρησιμοποιείται ως στοίβα. Φαίνονται καθαρά η τρέχουσα θέση του δείκτη στοίβας (sp), με κίτρινο χρώμα, και γράφονται τα ονόματα των καταχωρητών που αποθηκεύθηκαν σε κάθε θέση της στοίβας. Μελλοντικά θα φαίνονται και τα τμήματα της στοίβας που αντιστοιχούν σε κάθε κλήση υπορουτίνας.

Για να το χρησιμοποιήσετε θα πρέπει να κατεβάσετε μια (ξανά-)αλλαγμένη έκδοση του MARS από τη σελίδα του μαθήματος στο ecourse (απαιτεί Java SDK 11) και στο μενού Tools, να επιλέξτε Stack Visualizer. Ο απευθείας σύνδεσμος είναι http://ecourse.uoi.gr/mod/resource/view.php?id=60736. Πρίν τρέξετε το πρόγραμμα σας θα πρέπει να πατήσετε το Connect to MIPS στο παράθυρο που θα εμφανιστεί.

Θα ανοίξω ένα θέμα στο Piazza όπου μπορείτε να γράφετε παρατηρήσεις, αναφορές προβλημάτων σχετικες με το εργαλείο αυτό.

5 Παραδοτέο

Το παραδοτέο της άσκησης είναι το αρχείο lab03.asm που περιέχει το πρόγραμμά σας.

Πρέπει να κάνετε commit τις αλλαγές σας και να τις στείλετε (push) στο GitHub repository για να βαθμολογηθούν πριν από την καταληκτική ημερομηνία!

Το πρόγραμμά σας θα βαθμολογηθεί για την ορθότητά του, την ποιότητα σχολίων και την συνοπτικότητά του. Οπως τονίστηκε παραπάνω, είναι εξαιρετικά σημαντικό να ακολουθήσετε τους κανόνες του MIPS για την σωστή χρήση των καταχωρητών στις κλήσεις υπορουτινών (ποιοί πρέπει να διατηρηθούν, ποιοί δεν μπορούμε να είμαστε σίγουροι ότι διατηρούνται, κλπ). Το 1/3 των βαθμών θα αντιστοιχεί στο 1ο μέρος, τη ρουτίνα multproc.