4ο Εργαστήριο Αρχιτεκτονικής Η/Υ: MIPS assembly: Υπολογισμός γινομένου στοιχείων συνδεδεμένης λίστας με αναδρομή

Α. Ευθυμίου

Παραδοτέο: Τρίτη 22 Μάρτη, 23:00

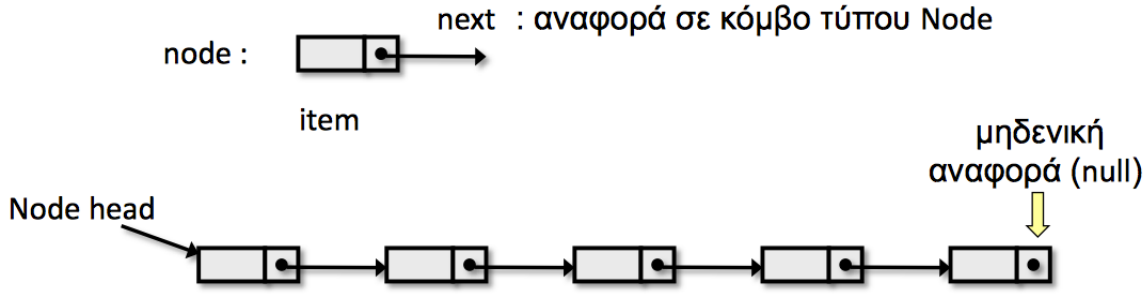
Το αντικείμενο αυτής της άσκησης είναι ένα πρόγραμμα που υπολογίζει το γινόμενο των αριθμών που είναι αποθηκευμένοι σε μία συνδεδεμένη λίστα. Σε αυτή την εργαστηριακή άσκηση θα γράψετε ένα προγράμμα assembly που χρησιμοποιεί υπορουτίνες και αναδρομή. Θα πρέπει να έχετε μελετήσει τα μαθήματα για τη γλώσσα 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.

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".

1 Συνδεδεμένη λίστα

Η συνδεδεμένη λίστα είναι μία από τις πιο γνωστές και απλές δομές δεδομένων. Ομαδοποιεί σε μια δομή (συχνά ονομάζεται struct, ή class σε γλώσσες υψηλού επιπέδου), έναν αριθμό από δεδομένα και έναν **δείκτη** (ή αναφορά σε κάποιες γλώσσες) 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.

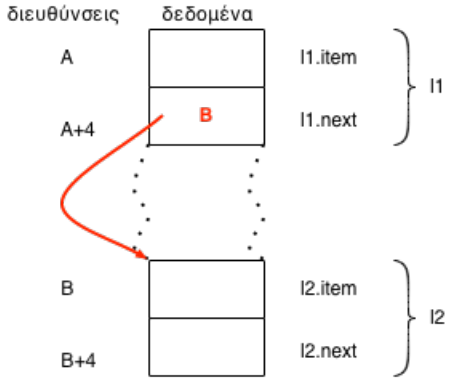
Στην άσκηση αυτή, κάθε στοιχείο της λίστας θα περιέχει μόνο έναν απρόσημο (θετικό), ακέραιο αριθμό 32 bit και, φυσικά, τον δείκτη στο επόμενο στοιχείο. Σε Java θα γράφαμε:

```
class Node {
  int  item;  // data
  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 Α, ο δείκτης next θα τοποθετηθεί στη διεύθυνση Α+4 και η διεύθυνση Α+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.

2 Η άσκηση

Για την άσκηση θα υλοποιήσετε μια υπορουτίνα που υπολογίζει, αναδρομικά, το γινόμενο των δεδομένων μιας συνδεδεμένης λίστας, σε assembly. Η υπορουτίνα θα πρέπει να ακολουθεί τις συμβάσεις του MIPS ως προς τη χρήση καταχωρητών και στοίβας. Παρακάτω δίνεται η υλοποίηση σε ψευτοκώδικα:

```
class Node {
  int  item;  // data
  Node next;  // pointer to next element
}

int listProd (Node nptr) {
  if (nptr == NULL)  // NULL == 0
      return 1;
  else
  return mult(nptr.data, listProd(nptr.next));
}
```

Για να υπολογίσετε το γινόμενο 2 αριθμών, που χρειάζεται η listProd, θα γράψετε άλλη μία υπορουτίνα, με όνομα mult, η οποία κάνει τον υπολογισμό με επαναλαμβανόμενες προσθέσεις. Η υπορουτίνα αυτή θα πρέπει να είναι ξεχωριστή από την listProd και να ακολουθεί τις συμβάσεις του MIPS για την χρήση καταχωρητών. Υποθέστε ότι οι αριθμοί είναι απρόσημοι (θετικοί ή μηδέν) και αγνοήστε τυχόν υπερχειλίσεις. Μην χρησιμοποιήσετε τις εντολές πολλαπλασιασμού του MIPS, ούτε ολισθήσεις.

Συνιστάται να γράψετε και να ελέγξετε πρώτα την υπορουτίνα που υπολογίζει το γινόμενο (mult) και μετά την listProd γιατί η mult είναι απλούστερη ως προς τη χρήση της στοίβας αφού δεν καλεί άλλες υπορουτίνες.

Για να πάρετε τα αρχεία της εργαστηριακής άσκησης, μεταβείτε στον κατάλογο εργασίας που είχατε κλωνοποιήσει από το αποθετήριό σας στο GitHub. (cd <όνομα χρήστη GitHub>-labs). Μετά, κάνετε τα παρακάτω βήματα:

```
git remote add lab04_starter https://github.com/UoI-CSE-MYY402/lab04_starter.git
git fetch lab04_starter
git merge lab04_starter/master -m "Fetched lab04 starter files"
```

Στον κατάλογο lab04 σας δίνεται το αρχείο lab04.asm. Υπάρχουν οι ετικέτες των υπορουτινών mult, listProd, όπου και πρέπει να γράψετε κώδικα. Επίσης υπάρχει η main που καλεί τέσσερεις φορές την υπορουτίνα listProd δίνοντας ως παράμετρο (στον `$a0`) τη διεύθυνση του πρώτου στοιχείου της λίστας. Μετά από κάθε κλήση αποθηκεύει το αποτέλεσμα σε διαφορετικό καταχωρητή. Όταν τελειώσει η εκτέλεση **ολόκληρου** του προγράμματος οι τιμές αυτών των καταχωρητών θα πρέπει να είναι οι αναμενόμενες. Προσοχή θα πρέπει να έχετε βάλει το κατάλληλο setting στον MARS ώστε να ξεκινά την εκτέλεση προγράμματος από την main και όχι από την πρώτη εντολή του lab04.asm. Δειτε το lab01.pdf για λεπτομέρειες.

Μην αλλάξετε τον κώδικα της main ούτε τις θέσεις ή την σειρά των ετικετών δεδομένων, γιατί χρησιμοποιούνται από τον αυτόματο έλεγχο. Δοκιμάστε το πρόγραμμα με άλλους αριθμούς αν θέλετε, αλλά στο τέλος παραδώστε το με τα αρχικά δεδομένα.

3 Παραδοτέο

Το παραδοτέο της άσκησης είναι το αρχείο lab04.asm που περιέχει το πρόγραμμά σας. Στον κατάλογο test το lab04_tester.py περιέχει έλεγχο για τα πρόγραμμα χωρίς να αλλάζει τα δεδομένα, γι'αυτό και δεν πρέπει να αλλάξετε τον κώδικα της main και τις υπάρχουσες τιμές δεδομένων. Δεν χρειάζεται να κάνετε αλλαγές στο lab04_tester.py, παρά μόνο να επιβεβαιώσετε ότι το πρόγραμμά σας περνάει τον έλεγχο.

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.