ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
www.cslab.ece.ntua.gr

1η ΑΣΚΗΣΗ ΣΤΗΝ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ

Ακ. έτος 2012-2013, 5ο Εξάμηνο, Σχολή ΗΜ&ΜΥ

Τελική Ημερομηνία Παράδοσης: 09/12/2012

ΜΕΡΟΣ Α

Δίνονται τα παρακάτω δύο προγράμματα γραμμένα σε C, καθώς και η αντίστοιχη μετάφραση τους σε assembly MIPS. Συμπληρώστε τα κενά. Σας υπενθυμίζουμε ότι ο καταχωρητής $0 είναι πάντα μηδέν, ενώ οι εντολές sll και srl πραγματοποιούν λογικό shift αριστερά ή δεξιά αντίστοιχα, τόσες θέσεις όσες ορίζει ο τελευταίος τελεστέος της εντολής. ***(Θέμα επαναληπτικής εξεταστικής Σεπτεμβρίου 2012)***

i)
```
int i = 0;
while (i < (n-1)){
   if(arr[i+1] < arr[i]){
        int temp = arr[i];
        arr[i] = arr[i+1];
        arr[i+1] = temp;
   }
   i = i+1;
}
```

```
        add  $3, $0, $0
        addi $7, $4, ___
loop:   bge  $3, $7, ___
        addi $8, $3, 1
        sll  $8, $8, ___
        sll  $9, $3, ___
        lw   $10, 400($9)
        lw   $11, ___($8)
        bge  ___, ___, skip
        sw   $10, 400(___)
        sw   $11, 400(___)
skip:   addi $3, $3, ___
        j    loop
done:   halt

* /
```

ii)
```
while ( ((x >> 1) & 0x1) != 0)
  {
     x = x >> 1;
     count ++;
  }

/*
 * Δίνεται ότι η μεταβλητή x
 * είναι στον $1 και η count στον
 * $2
/*
```

```
        addi $3, $0, __
loop:   srl  $4, $1, __
        and  $5, __, $3
        beq  $5, __, done
        add  $1, __, $0
        addi $2, __, __
        beq  __, $0, __
done:   halt
```

ΜΕΡΟΣ Β

(i) Δίνεται ο παρακάτω πίνακας με τα δεδομένα που βρίσκονται στη μνήμη για τις θέσεις 0x1000-0x108F.

Διεύθυνση	**Bytes**							
0x1000	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
0x1008	0x00	0x00	0x00	0x02	0x02	0x00	0x00	0x00
0x1010	0x00	0x00	0x03	0x00	0x00	0x00	0x00	0x00
0x1018	0x00	0x00	0x00	0x00	0x03	0x00	0x00	0x00
0x1020	0x00	0x00	0x00	0x03	0x03	0x03	0x03	0x00
0x1028	0x00	0x00	0x00	0x04	0x00	0x00	0x00	0x00
0x1030	0x00	0x00	0x00	0x00	0x00	0x00	0x04	0x00
0x1038	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
0x1040	0x00	0x04	0x00	0x00	0x00	0x00	0x00	0x00
0x1048	0x00	0x00	0x00	0x00	0x04	0x00	0x00	0x00
0x1050	0x00	0x05	0x05	0x05	0x05	0x05	0x00	0x00
0x1058	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
0x1060	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
0x1068	0x01	0x02	0x01	0x03	0x04	0x07	0x03	0x06
0x1070	0x03	0x07	0x03	0x08	0x03	0x09	0x06	0x04
0x1078	0x07	0x05	0x09	0x09	0x08	0x08	0x07	0x06
0x1080	0x06	0x06	0x06	0x05	0x06	0x07	0x07	0x07
0x1088	0x05	0x04	0x04	0x03	0x03	0x02	0x08	0x07

Οι θέσεις μνήμης 0x1000-0x1063 περιέχουν έναν πίνακα 10x10, ο οποίος αντιπροσωπεύει το πεδίο μάχης του γνωστού παιχνιδιού της ναυμαχίας. Σε αυτόν τον τετραγωνικό πίνακα 100 θέσεων είναι τοποθετημένα (οριζόντια, κάθετα ή διαγώνια) τα εξής πλοία:

- 1 αεροπλανοφόρο μήκους 5 (συμβολίζεται στον πίνακα με το 0x05)
- 1 φρεγάτα μήκους 4 (συμβολίζεται στον πίνακα με το 0x04)
- 2 υποβρύχια μήκους 3 (συμβολίζονται στον πίνακα με το 0x03)
- 1 αντιτορπιλικό μήκους 2 (συμβολίζεται στον πίνακα με το 0x02)

Αντίστοιχα, στις θέσεις μνήμης 0x1068-0x108F είναι αποθηκευμένες οι συντεταγμένες των βολών του αντιπάλου σας. Η κάθε βολή αποτελείται από ένα ζεύγος bytes που συμβολίζουν αντίστοιχα τη γραμμή και τη στήλη όπου θα χτυπήσει η βολή. Θεωρήστε ότι οι συντεταγμένες του πρώτου στοιχείου της πρώτης γραμμής του τετραγωνικού πίνακα είναι (0x00, 0x00).

Υλοποιήστε τη διαδικασία (procedure) *battleship_shots* σε assembly MIPS, η οποία εκτελεί τις βολές του αντιπάλου. Για κάθε βολή, η διαδικασία υπολογίζει αν είναι πετυχημένη ή όχι σώνοντας σε μια συμβολοσειρά (null-terminated ASCII string) το χαρακτήρα H (Hit) ή M (Miss) αντίστοιχα. Ταυτόχρονα, αν η βολή είναι επιτυχημένη μειώνει κατά 1 τις διαθέσιμες "ζωές" του παίκτη (αρχικά είναι ίσες με το συνολικό μήκος των πλοίων, δηλαδή 17).

Η διαδικασία που θα υλοποιήσετε θα πρέπει να δέχεται τα εξής ορίσματα:

- στον $a0 ένα δείκτη στη μνήμη στο σημείο όπου είναι αποθηκευμένος ο τετραγωνικός πίνακας των 100 θέσεων,
- στον $a1 ένα δείκτη στη μνήμη στο σημείο όπου βρίσκονται αποθηκευμένες οι συντεταγμένες της κάθε βολής,
- στον $a2 τον αριθμό των βολών που θα εκτελεστούν,

- στον $a3 ένα δείκτη στη μνήμη στο σημείο όπου θα αποθηκευτεί η συμβολοσειρά των αποτελεσμάτων των βολών.

Τέλος, η διαδικασία θα επιστρέφει ως αποτέλεσμα στον καταχωρητή $v0 τον αριθμό των "ζωών" που έχουν απομείνει στον παίκτη μετά τις βολές του αντιπάλου.

(ii) Εκτελέστε την διαδικασία που υλοποιήσατε με βάση τα περιεχόμενα της μνήμης που δίνονται στον πιο πάνω πίνακα. Δώστε την τιμή που επιστρέφει η συνάρτηση καθώς και τη συμβολοσειρά που παράγεται.

Για την υλοποίηση της άσκησης μπορείτε να χρησιμοποιήσετε τον ***MSIM***, ένα emulator του MIPS οποίος αναπτύχθηκε από συμφοιτητές σας και διατίθεται από το εργαστήριο Υπολογιστικών Συστημάτων (CSLab). Στον emulator αυτό, μπορείτε να γράφετε MIPS assembly και να την εκτελείτε παρακολουθώντας τα περιεχόμενα των καταχωρητών και της μνήμης καθιστώντας έτσι ευκολότερη την παραγωγή και τον έλεγχο του απαιτούμενου κώδικα. Τον MSIM μπορείτε να τον κατεβάσετε από τη σελίδα των ασκήσεων του site του μαθήματος.

Παραδοτέο της άσκησης θα είναι ένα ηλεκτρονικό κείμενο (pdf, docx ή odt) που θα περιέχει τις απαντήσεις του Α μέρους καθώς και τον κώδικα assembly του Β μέρους μαζί με τα αποτελέσματα της εκτέλεσης. Ο κώδικας θα πρέπει να περιέχει αναλυτικά σχόλια για την κατανόηση της λύσης σας από τους διδάσκοντες. Στο ηλεκτρονικό κείμενο να αναφέρετε στην αρχή τα στοιχεία σας (Όνομα, Επώνυμο, ΑΜ).

Η άσκηση θα παραδοθεί ηλεκτρονικά στην ιστοσελίδα:

http://www.cslab.ece.ntua.gr/courses/comparch/submit

Δουλέψτε ατομικά. Έχει ιδιαίτερη αξία για την κατανόηση του μαθήματος να κάνετε μόνοι σας την εργασία. Μην προσπαθήσετε να την αντιγράψετε από άλλους συμφοιτητές σας.