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

1η ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ ΣΤΗΝ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ
Ακ. έτος 2018-2019, 5ο εξάμηνο, ΣΗΜΜΥ

ΤΜΗΜΑ 2ο (Μ - Ω)
Ημερομηνία παράδοσης: 11/11/2018
Απορίες στο: ca2018-2019-tmima2@cslab.ece.ntua.gr

ΜΕΡΟΣ Ι

Δίνεται η ακόλουθη συνάρτηση γραμμένη σε C, καθώς και η μετάφρασή της σε assembly του MIPS. Ο καταχωρητής **$s0** έχει τιμή **2**. Η εντολή `div rs, rt` εκτελεί διαίρεση του περιεχομένου του καταχωρητή `rs` με το περιεχόμενο του καταχωρητή `rt` και το πηλίκο τοποθετείται στον καταχωρητή `lo`, ενώ το υπόλοιπο στον καταχωρητή `hi`. Η εντολή `mfhi rd` μεταφέρει τα περιεχόμενα του καταχωρητή `hi` στον καταχωρητή `rd`[1]. Να συμπληρωθούν κατάλληλα τα κενά.

```
int odd_sum(int *a, int size) {
  int sum = 0, i, *p;
  for (p = a, i = 0; i < size; p++, i++) {
    if (*p % 2 != 0) {
      sum += *p;
    }
  }
  return sum;
}
```

```
odd_sum:  add $t0, ___, ___
          add $t1, $a0, ___
          add $t2, ___, $zero
loop:     slt ___, $t2, ___
          beq $t3, $zero, ___
          lw $t4, _____
          div ___, ___
          mfhi ___
          beq $t5, ___, cont
          add $t0, $t0, ___
cont:
          addi $t1, $t1, ___
          addi $t2, $t2, ___
          j ___
exit:     add ___, $zero, ___
          jr ___
```

[1]Για τις δύο αυτές εντολές που δεν περιλαμβάνονται στον βασικό πυρήνα των εντολών του MIPS μπορείτε να μάθετε περισσότερα ανατρέχοντας στο Παράρτημα Β του συγγράματος του μαθήματος.

ΜΕΡΟΣ ΙΙ

Να γραφεί ρουτίνα σε assembly του MIPS που να πραγματοποιεί **in-place** αντιστροφή της θέσης των στοιχείων πίνακα ακέραιων αριθμών **32-bit**. Για παράδειγμα, όταν δοθεί ως όρισμα ο πίνακας [-5, 13, 100, 0, -10], ο κώδικάς σας θα πρέπει να τον μετασχηματίζει σε [-10, 0, 100, 13, -5]. Τα ορίσματά της θα είναι η διεύθυνση του πρώτου στοιχείου καθώς και το μέγεθος του πίνακα (σε αριθμό στοιχείων — **όχι** σε bytes).

ΜΕΡΟΣ ΙΙΙ

Οι παρακάτω C συναρτήσεις υλοποιούν τον αλγόριθμο ταξινόμησης **heapsort**[2] για ακέραιους αριθμούς **32-bit**. Υλοποιήστε τις αντίστοιχες ρουτίνες σε assembly του MIPS.

```c
void swap(int *m1, int *m2) {
  int tmp = *m1;
  *m1 = *m2;
  *m2 = tmp;
}

// To heapify a subtree rooted with node i which is an index in arr[].
// n is the size of heap
void heapify(int arr[], int n, int i) {
  int largest = i;    // initialize largest as root
  int l = 2 * i + 1;  // left = 2 * i + 1
  int r = 2 * i + 2;  // right = 2 * i + 2

  // if left child is larger than root
  if (l < n && arr[l] > arr[largest]) {
    largest = l;
  }

  // if right child is larger than largest so far
  if (r < n && arr[r] > arr[largest]) {
    largest = r;
  }

  // if largest is not root
  if (largest != i) {
    swap(&arr[i], &arr[largest]);
    heapify(arr, n, largest);   // recursively heapify the affected sub-tree
  }
}

// main function to implement heapsort
void heapsort(int arr[], int n) {
  int i;
  // build heap (rearrange array)
  for (i = n / 2 - 1; i >= 0; i--) {
    heapify(arr, n, i);
  }
  // one by one extract an element from heap
  for (i = n - 1; i >= 0; i--) {
    swap(&arr[0], &arr[i]);     // move current root to end
    heapify(arr, i, 0);         // call max heapify on the reduced heap
  }
}
```

[2] https://en.wikipedia.org/wiki/Heapsort

* * *

Σημείωση: Για διευκόλυνση, σας προτρέπουμε να χρησιμοποιήσετε το προσομοιωτή Qtspim (`http://spimsimulator.sourceforge.net`). Εδώ μπορείτε να βρείτε ένα λεπτομερές εγχειρίδιο χρήσης του.

Παραδοτέο της άσκησης θα είναι ένα **ηλεκτρονικό κείμενο** (κατά προτίμηση **pdf**, για λόγους συμβατότητας) που θα περιέχει τις απαντήσεις των τριών μερών. Το έγγραφο πρέπει να φέρει στην αρχή του τα στοιχεία σας (όνομα, επώνυμο και αριθμό μητρώου). Οι κώδικες που θα παραδοθούν οφείλουν να είναι σε ευανάγνωστη μορφή και να συνοδεύονται από επαρκή σχόλια.