ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

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

1η ΑΣΚΗΣΗ ΣΤΗΝ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ

Ακ. έτος 2019-2020, 5ο Εξάμηνο, Σχολή ΗΜ&ΜΥ
Τελική Ημερομηνία Παράδοσης: **24/11/2019**

ΜΕΡΟΣ Α

Δίνεται το παρακάτω πρόγραμμα σε C καθώς και μια μετάφραση του σε assembly MIPS. Η διεύθυνση του πρώτου στοιχείου του πίνακα A είναι αποθηκευμένη στον καταχωρητή $s3. Συμπληρώστε τα κενά. Σας υπενθυμίζουμε ότι ο καταχωρητής $0 (ή $zero) είναι πάντα μηδέν.

```
int A[N];
int left = -1, int right = N;
int pivot, res;

pivot = A[0];

while (left < right) {
  while (A[++left] < pivot);
  while (A[--right] > pivot);

  if (left < right)
      swap(&A[left], &A[right]);
}

res = right;
```

```
            lw   $s0, ____
LOOP:       slt  $t0, ____, ____
            ___  $t0, $zero, EXIT
I_LOOP:     addi ___, $s1, 1
            sll  $t0, $s1, ____
            add  ___, $s3, $t0
            lw   ___, 0($t0)
            slt  $t0, $t1, ____
            ___  $t0, $zero, ____
J_LOOP:     addi ___, $s2, -1
            sll  $t0, $s2, ____
            add  ___, $s3, $t0
            lw   ___, 0($t0)
            slt  $t0, ___, $t1
            ____ $t0, $zero, ____
            slt  $t0, ___, ____
            ___  $t0, $zero, ____
            ___  $a0, $s1, ____
            ___  $a0, $a0, $s3
            sll  ___, ___, 2
            add  $a1, $a1, ____
            ___  swap
            j    LOOP
EXIT:       add  $s4, ___, $zero
```

ΜΕΡΟΣ Β

Υλοποιήστε την παρακάτω ρουτίνα σε assembly του MIPS. Δίνονται τα μεγέθη των τύπων char 8 bits, short int 16 bits και int 32 bits.

```
struct S {
  int A[2];
  char B[8];
  short int C[2];
  char *D[2];
};

int foo(struct S *s) {
  int a, i;
  char b;
  short int c;
  char *d;

  a = 0xdeadbeef; b = 0; c = 0;
  d = s->D[1];

  for (i=0; i < 2; i++) a += s->A[i];
  for (i=0; i < 8; i++) b += s->B[i];
  for (i=0; i < 2; i++) c += s->C[i];

  if (a % 30 < 10)
    a = a / 30;

  s->B[0] = b;
  s->C[0] = c;
  return a;
}
```

ΜΕΡΟΣ Γ

Παρακάτω δίνεται η δομή που αναπαριστά έναν κόμβο μίας συνδεδεμένης λίστας, η δομή που αναπαριστά μία συνδεμένη λίστα καθώς και οι συναρτήσεις για την αναζήτηση, εισαγωγή και διαγραφή ενός κλειδιού (key) μέσα στη λίστα. Επίσης, δίνεται και η βοηθητική συνάρτηση *traverse().* Υλοποιήστε τις ρουτίνες traverse, lookup, insert και delete σε assembly του MIPS. Θεωρείστε ότι η συνάρτηση *create_node()* υπάρχει και επιστρέφει έναν καινούριο κόμβο με το δεδομένο κλειδί.

```
struct node {
  int key
  struct node *next;
};

struct list {
  struct node *head;
};
```

```
/*
 * Helper function that traverses list 'l'.
 * Returns a pointer to the node which contains 'key' or to the first
 *   node with a key greater than 'key'.
 * On return *prev points to the previous node of the one that
 *   is being returned.
 * If both the returned node and *prev are NULL, the list is empty.
 * When only the returned node is NULL 'key' is not is the list and its
 *   position would be in the end of the list.
 * When only *prev is NULL 'key' is either present in l->head or its
 *   position would be in the beginning of the list.
 */

struct node *traverse(struct list *l, int key, struct node **prev) {
  struct node *curr;

  curr = l->head;
  *prev = NULL;
  while (curr != NULL && curr->key < key) {
    *prev = curr;
    curr = curr->next;
  }
  return curr;
}


int lookup(struct list *l, int key) {
  struct node *curr, *prev;

  curr = traverse(l, key, &prev);
  if (curr == NULL || curr->key != key) return 0;
  else                                  return 1;
}

int insert(struct list *l, int key) {
  struct node *curr, *prev, *new;

  curr = traverse(l, key, &prev);
  if (curr != NULL && curr->key == key) return 0;

  new = create_node(key);
  new->next = curr;
  if (prev != NULL) prev->next = new;
  else              l->head = new;
  return 1;
}


int delete(struct list *l, int key) {
  struct node *curr, *prev;

  curr = traverse(l, key, &prev);
  if (curr == NULL || curr->key != key) return 0;

  if (prev != NULL) prev->next = curr->next;
  else              l->head = curr->next;
  return 1;
}
```

Για την υλοποίηση της άσκησης μπορείτε να χρησιμοποιήσετε τον ***MILE*** *(manual.pdf)* , ένα MIPS emulator που 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 MILE μπορείτε να τον κατεβάσετε από τη σελίδα των ασκήσεων του site του μαθήματος:

http://www.cslab.ece.ntua.gr/courses/comparch/assign.go

*Παραδοτέο της άσκησης θα είναι ένα ηλεκτρονικό κείμενο (**pdf**, **docx** ή **odt**) που θα περιέχει τους κώδικες assembly και των 3 μερών της άσκησης. 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 στο mycourses.

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.