Αρχιτεκτονική Υπολογιστών

5ο Εξάμηνο 2019-2020

1η Σειρά Ασκήσεων

Χρήστος Τσούφης – 03117176

Μέρος Α

```
        lw $s0, 0($s3)          #$s0 = pivot
LOOP:   slt $t0, $s1, $s2
        beq $t0, $zero, EXIT
I_LOOP: addi $s1, $s1, 1        #$s1 = left
        sll $t0, $s1, 2
        add $t0, $s3, $t0
        lw $t1, 0($t0)
        slt $t0, $t1, $s0
        bne $t0, $zero, I_LOOP
J_LOOP: addi $s2, $s2, -1       #$s2 = right
        sll $t0, $s2, 2
        add $t0, $s3, $t0
        lw $t1, 0($t0)
        slt $t0, $s0, $t1
        bne $t0, $zero, J_LOOP
        slt $t0, $s1, $s2
        beq $t0, $zero, LOOP
        sll $a0, $s1, 2
        add $a0, $a0, $s3
        sll $a1, $s2, 2
        add $a1, $a1, $s3
        jal swap
        j LOOP
EXIT:   add $s4, $s2, $zero
```

Μέρος Β

```
        #s = $a0
        #&A[0] = 0($a0)
        #&B[0] = 8($a0)
        #&C[0] = 16($a0)
        #&D[0] = 20($a0)
        #0xdeadbeef = 11011110101011011011111011101111

FOO:    lui $t0, 1101111010101101
        ori $t0, $t0, 1011111011101111          #a = $t0 = 0xdeadbeef
        add $t1, $zero, $zero                   #b = $t1 = 0
        add $t2, $zero, $zero                   #c = $t2 = 0
        lw $t3, 24($a0)                         #d = $t3 = s->D[1]
        add $t4, $zero, $zero                   #i = $t4 = 0

        addi $t6, $zero, 2                      #$t6 = 2
LOOP_A: sll $t5, $t4, 2                         #$t5 = 4*i
        add $t5, $t5, $a0                       #$t5 = &A[i]
        lw $t5, 0($t4)                          #$t5 = s->A[i]
        add $t0, $t0, $t5                       #a += s->A[i]
        addi $t4, $t4, 1                        #i++ (size of char = 1 byte)
        bne $t4, $t6, LOOP_A
        add $t4, $zero, $zero                   #i=0

        addi $t6, $zero, 8                      #$t6 = 8
LOOP_B: add $t5, $t4, $a0                       #$t5 = s + i
        lb $t5, 8($t4)                          #$t5 = s->B[i]
        add $t1, $t1, $t5                       #b += s->B[i]
        addi $t4, $t4, 1                        #i++
        bne $t4, $t6, LOOP_B
        add $t4, $zero, $zero                   #i=0

        addi $t6, $zero, 2                      #$t6 = 2
LOOP_C: sll $t5, $t4, 1                         #$t5 = 2*i
        add $t5, $t5, $a0                       #$t5 = s + 2*i
        lh $t5, 16($t4)                         #$t5 = s->C[i]
        add $t2, $t2, $t5                       #c += s->C[i]
        addi $t4, $t4, 1                        #i++
        bne $t4, $t6, LOOP_C

        addi $t5, $zero, 30                     #$t5 = 30
        div $t0, $t5                            #hi = a%30, lo = a/30
        mfhi $t5                                #$t5 = a%30
        slti $t5, $t5, 10                       #$t5 = 1 if a%30 < 10
        beq $t5, $zero, NEXT
        mflo $t5                                #$t5 = a/30
        add $t0, $t5, $zero                     #a = a/30

NEXT:   sb $t1, 8($a0)                          #s->B[0] = b
        sh $t2, 16($a0)                         #s->C[0] = c;
        add $v0, $t0, $zero                     #return value is $v0
        jr $ra
```

Μέρος Γ

```
                #$a0 = l, $a1 = key, $a2 = prev, $v0 = return value
TRAVERSE:   lw $t0, 0($a0)                          #$t0 = curr = l->head
            sw $zero, 0($a2)                        #*prev = NULL
LOOP:       beq $t0, $zero, EXIT_LOOP               #if(curr != NULL)
            lw $t1, 0($t0)                          #$t1 = curr->key
            slt $t1, $t1, $a1                       #$t1 = ($t1 < key)
            beq $t1, $zero, EXIT_LOOP               #if(curr->key < key)
            sw $t0, 0($a2)                          #*prev = curr
            lw $t0, 4($t0)                          #curr = curr->next
            j LOOP
EXIT_LOOP:  add $v0, $t0, $zero                     #return curr
            jr $ra

            #$a0 = l, $a1 = key, $v0 = return value
LOOKUP:     addi $sp, $sp, -8
            sw $ra, 4($sp)
            sw $s0, 0($sp)                          #push $ra, $s0
            add $s0, $a1, $zero                     #$s0 = key
            jal TRAVERSE                            #$v0 = curr
            add $t0, $v0, $zero                     #$t0 = curr
            lw $t1, 0($v0)                          #$t1 = curr->key
            sub $s0, $t1, $s0                       #$s0 = $t1 - $s0
            beq $t0, $zero, R_ZERO                  #if(curr!=NULL)
            bne $s0, $zero, R_ZERO                  #if(curr->key==key)
            addi $v0, $zero, 1                      #return 1
            j EXIT_L
R_ZERO:     add $v0, $zero, $zero                   #return 0
EXIT_L:     lw $ra, 4($sp)                          #pop $ra
            lw $s0, 0($sp)                          #pop $s0
            addi $sp, $sp, 8
            jr $ra

            #$a0 = l, $a1 = key, $v0 = return value
INSERT:     addi $sp, $sp, -20 sw $ra, 16($sp)      #push $ra
            sw $s0, 12($sp)                         #push $s0
            sw $s1, 8($sp)                          #push $s1
            sw $s2, 4($sp)                          #push $s2
            sw $s3, 0($sp)                          #push $s3
            add $s0, $a1, $zero                     #$s0 = key
            add $s2, $a0, $zero                     #$s2 = l
            jal TRAVERSE                            #$v0 = curr
            lw $s3, 0($a3)                          #$s3 = prev
            add $s1, $v0, $zero                     #$s1 = curr
            lw $t0, 0($s1)                          #$t0 = curr->key
            sub $t1, $s0, $t0                       #$t1 = key– curr->key
            beq $s1, $zero, CONTINUE                #if(curr!=NULL)
            bne $t1, $zero, CONTINUE                #if(curr->key==key)
            add $v0, $zero, $zero                   #return 0

            j EXIT_C
```

```
CONTINUE:  add $a0, $s0, $zero                  #$a0 = key
           jal CREATE_NODE                      #$v0 = new
           add $t0, $v0, $zero                  #$t0 = new
           sw $s1, 4($t0)                       #new->next=curr
           addi $v0, $zero, 1                   #return 1
           beq $s3, $zero, ELSE_I               #if(prev!=NULL)
           sw $t0, 4($s3)
           j EXIT_C
ELSE_I:    sw $t0, 0($s2)
EXIT_C:    lw $ra, 16($sp)                      #pop $ra
           lw $s0, 12($sp)                      #pop $s0
           lw $s1, 8($sp)                       #pop $s1
           lw $s2, 4($sp)                       #pop $s2
           lw $s3, 0($sp)                       #pop $s3
           addi $sp, $sp, 20
           jr $ra

           #$a0 = l, $a1 = key, $v0 = return value
DELETE:    addi $sp, $sp, -12
           sw $ra, 8($sp)                       #push $ra
           sw $s0, 4($sp)                       #push $s0
           sw $s1, 0($sp)                       #push $s1
           add $s0, $a0, $zero                  #$s0 = l
           add $s1, $a1, $zero                  #$s1 = key
           jal TRAVERSE                         #$v0 = curr
           add $t0, $v0, $zero                  #$t0 = curr
           lw $t1, 0($a2)                       #$t1 = *$a2 = prev
           lw $t2, 0($t0)                       #$t2 = curr->key
           lw $t3, 4($t0)                       #$t3 = curr->next
           beq $t0, $zero, D_ZERO               #if(curr != NULL)
           bne $t2, $s1, D_ZERO                 #if(curr->key == key)
           beq $t1, $zero, ELSE_D               #if(prev != NULL)
           sw $t3, 4($t1)                       #prev->next = #curr->next
           addi $vo, $zero, 1                   #return 1
           j EXIT_D
ELSE_D:    sw $t3, 0($s0)                       #l->head = curr->next
           addi $vo, $zero                      #return 1
           j EXIT_D
D_ZERO:    add $v0, $zero, $zero                #return 0
EXIT_D:    lw $ra, 8($sp)                       #pop $ra
           lw $s0, 4($sp)                       #pop $s0
           lw $s1, 0($sp)                       #pop $s1
           addi $sp, $sp, 12
           jr $ra
```

Σημείωση: Όπου υπάρχει “if (condition)” σημαίνει ότι επαληθεύεται η συνθήκη οπότε προχωράει στην επόμενη γραμμή, αλλιώς πηγαίνει στο αναγραφόμενο LABEL.