Μιχάλης Ψυχής
el22094, michalis.psychis@gmail.com
1η Σειρά Ασκήσεων, Αρχιτεκτονική Υπολογιστών. Τμήμα 3

Μέρος Α

```
MAIN:
addi    $sp, $sp, -8
sw      $s1, 4($sp)
sw      $s0, 0($sp)
lw      $s0, 0($a0)
lw      $s1, 0($a1)
add     $t3, $zero, $zero
subi    $a2, $a2, 1

LOOP:
slt     $t4, $t3, $a2
beq     $t4, $zero, END
addi    $a0, $a0, 4
addi    $a1, $a1, 4
lw      $t5, 0($a0)
lw      $t6, 0($a1)
slt     $t4, $t5, $s0
beq     $t4, $zero, CHECK2
add     $s0, $t5, $zero

CHECK2:
slt     $t4, $t6, $s1
beq     $t4, $zero, NEXT
add     $s1, $t6, $zero

NEXT:
addi    $t3, $t3, 1
j       LOOP

END:
add     $v0, $s0, $s1
lw      $s0, 0($sp)
lw      $s1 , 4($sp)
addi    $sp, $sp, 8
jr      $ra
```

Μέρος Β

```
# Κώδικας για να τρέχει στο QtSpim
.data
input:  .word 65535             # int input, διάλεξα τυχαία το 0xffff, αναμενόμενο
                                # αποτέλεσμα = 1559699
.text
.globl main

main:
    la      $a0, input          # Load input into $a0
    jal     cslab_hash          # Jump to function, result will be in $v0
    # Print the result
    move    $a0, $v0            # Move result to $a0
    li      $v0, 1              # Syscall code for printing integer
    syscall                     # Make syscall
    # Exit program
    li          $v0, 10         # Syscall code for exit
    syscall                     # Make syscall

#Κώδικας προγράμματος
cslab_hash:
    addi    $sp, $sp, -8
    sw      $s0, 4($sp)
    sw      $ra, 0($sp)
    lw      $s0, 0($a0)             # φορτώνουμε την input στον $s0
    ori     $t0, $zero, 0x1505      # φορτώνουμε στο $t0 το hash=5381 => 0x1505(hex)

LOOP2:
    beq     $s0, $zero, END2    # Εάν το input ($s0) είναι 0, πήγαινε στο end
    andi    $t1, $s0, 0xFF      # input AND 1111 1111 => store στον $t1 (c)
    sll     $t2, $t0, 4         # Κάνε sll 4 το hash ($t0) και αποθήκευσε το στον $t2
    add     $t3, $t2, $t0       # Πρόσθεσε στον $t2 το hash ($t0)
    add     $t0, $t3, $t1       # Πρόσθεσε το c και τον $t3. Αυτό είναι το νέο hash
    srl     $s0, $s0, 8         # Κάνε shift right logical το input.
    j       LOOP2

END2:
    or      $v0, $zero, $t0     # επιστρέφουμε την hash ($t0) στον $v0
    lw      $ra, 0($sp)
    lw      $s0, 4($sp)
    addi    $sp, $sp, 8
    jr      $ra
```

Μέρος Γ

```
.data
data_array: .word 10, 20, 30, 40, 50, 60, 70, 80      # Ο πίνακας μας με το δέντρο.
tree:       .space  60                # Ένας άδειος πίνακας 15 τετράμπιτων αριθμών
newline:    .asciiz "\n"              # Εντολή για new line στο command window

.text
.globl main

main:
    # Load arguments
    la      $a0, data_array           # Address of data_array
    li      $a1, 8                    # Size of data_array
    la      $a2, tree                 # Address of tree
    jal     create_leaves             # Καλώ την create_leaves, στον πίνακα tree υπάρχουν
                                      # πλέον τα hashes των data_array
    li      $a1, 8                    # $a1 = num_of_leaves
    jal     create_Merkle_Tree        # Καλούμε την Merkle_Tree
    move    $t3, $v0                  # Μεταφέρουμε την διεύθυνση του root_hash στο $t3

    # Exit program
    li      $v0, 10                   # Syscall code for exit
    syscall

# Κώδικας Προγγράματος
create_leaves:
    addi    $sp, $sp, -20       # Αποθηκεύουμε στο stack τις $s0-$s3 και $ra
    sw      $s0, 16($sp)
    sw      $s1, 12($sp)
    sw      $s2, 8($sp)
    sw      $s3, 4($sp)
    sw      $ra, 0($sp)

    move    $s0, $a0            # Μεταφέρουμε τα Inputs στους καταχωρητές
    move    $s1, $a1
    move    $s2, $a2
    li      $s3, 0              # i=0;

loop:
    beq     $s3, $s1, exit      # Εάν i = array_size => exit;
    move    $a0, $s0            # Μεταφέρουμε στον $a2 το address του $s0 (data_array)
    jal     cslab_hash          # Καλούμε την cslab_hash, όπως ακριβώς στο μέρος Β.
    sw      $v0, 0($s2)         # Αντιγράφουμε την τιμή του $v0 (hash($s0)), στον πίνακα
                                # $s2 (tree)
```

```
    addi    $s0, $s0, 4         # Αυξάνουμε κατά 4 τις διευθύνσεις των πινάκων.
    addi    $s2, $s2, 4         # Πάμε ένα τετράμπιτο αριθμό παρακάτω.
    addi    $s3, $s3, 1         # i++
    j       loop

exit:
    lw      $ra, 0($sp)          #Επαναφέρουμε το περιεχόμενο των $s0-$s3
    lw      $s3, 4($sp)
    lw      $s2, 8($sp)
    lw      $s1, 12($sp)
    lw      $s0, 16($sp)
    addi    $sp, $sp, 20
    jr      $ra                  #Επιστρέφουμε στην main

create_Merkle_Tree:
    addi    $sp, $sp, -24
    sw      $s4, 20($sp)
    sw      $s3, 16($sp)
    sw      $s2, 12($sp)
    sw      $s1, 8($sp)
    sw      $s0, 4($sp)
    sw      $ra, 0($sp)

    move    $t0, $a1                # Αποθηκεύουμε τον $a1 (num_of_leaves) στον $t0
    move    $s1, $a2                # Μεταφέρουμε την διεύθυνση του $a2 (tree) στον $s1
    li      $s2, 0                  # i=0;
    srl     $s3, $t0, 1             # $s3 (level_ops) = $t0/2
    # unsigned int * tail = &tree[num_of_leaves];
    sll     $t1, $t0, 2             # $t1 = $t0 * 4
    add     $t1, $s1, $t1           # $t1 = $s1 + $t1
    move    $s4, $t1                # Αντιγράφουμε την διεύθυνση του $t1 στον $s4 (tail)

while_loop:
    slt     $t0, $zero, $s3         # Εάν $s3 = 0 (level_ops), πήγαινε στο exit2
    beq     $t0, $zero, exit2
    li      $s0, 0                  # $s0 (j) = 0

while_loop2:
    slt     $t0, $s0, $s3           # Εάν $s3 = $s0 (level_ops = j), jump στην divide
    beq     $t0, $zero, divide
    sll     $t0, $s2, 2             # t0 = $s2*4 (t0 = i*4)
    addi    $t1, $s2, 1             # t1 = $s2 + 1 (t1 = i+1)
    sll     $t2, $t1, 2             # t2 = $t1*4 (t2 = (i+1)*4)
    add     $t3, $s1, $t0           # t3 = &trees[4i]
```

```
    add     $t4, $s1, $t2           # t4 = &trees[4(i+1)]
    lw      $t1, 0($t3)             # t1 = trees[4i]
    lw      $t2, 0($t4)             # t2 = trees[4(i+1)]
    xor     $t5, $t1, $t2           # xored_val = tree[i] ^ tree[i+1]

    sw      $t5, 0($a0)
    jal     cslab_hash              # Καλούμε την cslab_hash, όπως ακριβώς στο μέρος Β.
    sw      $v0, 0($s4)             # tail[] = hashed value

    addi    $s2, $s2, 2             # i = i+2
    addi    $s0, $s0, 1             # j++
    addi    $s4, $s4, 4             # tail[i++]
    j       while_loop2

divide:
    srl     $s3, $s3, 1             # $s3 = $s3/2 ( level_ops /= 2;)
    j       while_loop

exit2:
    addi    $v0, $s4, -4            # tails[i--]
    lw      $ra, 0($sp)
    lw      $s0, 4($sp)
    lw      $s1, 8($sp)
    lw      $s2, 12($sp)
    lw      $s3, 16($sp)
    lw      $s4, 20($sp)
    addi    $sp, $sp, 24
    jr      $ra
```

Μέρος Δ

```
.data
data_array: .space  32      #Ο πίνακας με το δέντρο.
tree:       .space  60      #Ένας άδειος πίνακας 15 τετράμπιτων αριθμών
str: .asciiz    "Root hash: "

.text
.globl main

main:
    # Load Arguments
    la      $a0, data_array     # Address of data_array
    li      $a1, 8              # Size of data_array
    la      $a2, tree           # Address of tree

# Φορτώνουμε στον πίνακα data_array[] την φράση "094_NTUA_ECE_CA_2024_1106_2004_MIPS"
    # data_array[0]
    andi    $t0, $t0, 0         # $t0 = 0
    or      $t0, $t0, 0x303934  # "094"
    sw      $t0, 0($a0)         # Αποθηκεύουμε στον $a0 την λέξη

    # data_array[1]
    addi    $a0, $a0, 4             # data_array[i++]
    andi    $t0, $t0, 0             # $t0 = 0
    or      $t0, $t0, 0x4e545541    # "NTUA"
    sw      $t0, 0($a0)             # Αποθηκεύουμε στον $a0 την λέξη

    # data_array[2]
    addi    $a0, $a0, 4             # data_array[i++]
    andi    $t0, $t0, 0             # $t0 = 0
    or      $t0, $t0, 0x454345      # "ECE"
    sw      $t0, 0($a0)             # Αποθηκεύουμε στον $a0 την λέξη

    # data_array[3]
    addi    $a0, $a0, 4             # data_array[i++]
    andi    $t0, $t0, 0             # $t0 = 0
    or      $t0, $t0, 0x4341        # "CA"
    sw      $t0, 0($a0)             # Αποθηκεύουμε στον $a0 την λέξη

    # data_array[4]
    addi    $a0, $a0, 4             # data_array[i++]
    andi    $t0, $t0, 0             # $t0 = 0
    or      $t0, $t0, 0x32303234    # "2024"
    sw      $t0, 0($a0)             # Αποθηκεύουμε στον $a0 την λέξη
```

```
# data_array[5]
addi    $a0, $a0, 4             # data_array[i++]
andi    $t0, $t0, 0             # $t0 = 0
or      $t0, $t0, 0x31313036    # "1106"
sw      $t0, 0($a0)             # Αποθηκεύουμε στον $a0 την λέξη

# data_array[6]
addi    $a0, $a0, 4             # data_array[i++]
andi    $t0, $t0, 0             # $t0 = 0
or      $t0, $t0, 0x32303034    # "2004"
sw      $t0, 0($a0)             # Αποθηκεύουμε στον $a0 την λέξη

# data_array[7]
addi    $a0, $a0, 4             # data_array[i++]
andi    $t0, $t0, 0             # $t0 = 0
or      $t0, $t0, 0x4d495053    # "MIPS"
sw      $t0, 0($a0)             # Αποθηκεύουμε στον $a0 την λέξη

addi    $a0, $a0, -28
jal     create_leaves       # Καλώ την create_leaves, στον πίνακα tree υπάρχουν πλέον
                            # τα hashes των data_array
li      $a1, 8              # num_of_leaves
jal     create_Merkle_Tree
move    $t3, $v0

la      $a0, str            # Τυπώνουμε στην κονσόλα "Root hash: "
li      $v0, 4
syscall

lw      $a0, 0($t3)         # Τυπώνουμε στην κονσόλα το root_hash της λέξης μας
li      $v0, 1
syscall

# Exit program
li      $v0, 10             # Syscall code for exit
syscall
```