Ιωάννης Τσαντήλας 03120883

ΜΕΡΟΣ Α

Δίνεται το παρακάτω πρόγραμμα σε C καθώς και μια μετάφραση του σε assembly MIPS. Θεωρήστε ότι ο καταχωρητής $s0 περιέχει τη διεύθυνση του πρώτου στοιχείου του πίνακα array. Συμπληρώστε τα κενά. Σας υπενθυμίζουμε ότι ο καταχωρητής $0 (ή $zero) είναι πάντα μηδέν.

```
int array[20];
int *p, N;

p = &array[3];

while (*p != 0) {
   if (*p < 50) *p = *p % N;
   else *p = *p / N;
   p++;
}
```

```
        addi ____, $s0, ____
LOOP:   lw   ____, 0($s1)
        ____ $t0, ____, ____
        div  ____, $s2
        ____ ____, $t0, 50
        ____ $t1, ____, ____
        ____ $t0
        jmp  ____
ELSE:   ____ ____
NEXT:   sw   ____, ____
        addi ____, ____, ____
        jmp  ____
END:
```

Ο κώδικας σε Assembly (έχω γράψει με **bold** τα κενά που έπρεπε να συμπληρωθούν) είναι:

```
# s0 = &array[0]
# s1 = p (διεύθυνση)
# s2 = N
        addi $s1, $s0, 12       # s1 = s0 + 12 <=> p = &array[3]
                                # Κάθε ακέραιος έχει μήκος 4 bytes, άρα 3*4=12.
LOOP:   lw $t0, 0($s1)          # t0 = array[3] <=> t0 = *p.
        beq $t0, $zero, END     # Συνθήκη του while: if(*p!=0) <=> if(t0==0) break;
        div $t0, $s2            # Lo =*p/N  (πηλίκο διαίρεσης)
                                # Hi =*p%N  (υπόλοιπο διαίρεσης)
        slti $t1, $t0, 50       # if(t0<50) <=> if(*p<50) t1=1;
                                # else t1=0;
        beq $t1, $zero, ELSE    # if(t1==0) <=> if(*p>=50) Else();
        mfhi $t0                # t0 = *p = Hi = *p%N
        j NEXT
ELSE:   mflo $t0                # t0 = *p = Lo = *p/N
NEXT:   sw $t0, 0($s1)          # Αποθηκεύουμε το περιεχόμενο του t0 (*p)
                                # στην διεύθυνση που δείχνει ο p (s1)
        addi $s1, $s1, 4        # s1 += 4 <=> p++; (κάθε ακέραιος έχει μήκος 4 bytes)
        j LOOP
END:
```

ΜΕΡΟΣ Β

Οι παρακάτω ρουτίνες σε C υλοποιούν τους αλγορίθμους ταξινόμησης **bubble sort**, **cocktail sort** και **insertion sort** ενός πίνακα ακεραίων με N στοιχεία. Υλοποιήστε τις ρουτίνες σε assembly του MIPS.

```
void bubbleSort(int *A, int N) {
      int i, j;
      bool swapped;

      for (i = 0; i < N-1; i++) {
            swapped = false;
            for (j = 0; j < N-i-1; j++)
                  if (A[j] > A[j+1]) {
                        swap(&A[j], &A[j+1]);
                        swapped = true;
                  }
            }
            if (!swapped) break;
      }
}
```

```
void swap (int *x, int * y) {
      int temp = *x;

      *x = *y;
      *y = temp;
}
```

############ Αρχή των Bubble_Sort & Swap ############

 # s0 = &A[0] *# t1 == slt purposes*

 # s1 = N *# t2 == &A[j]*

 # s2 = i *# t8 = A[j]*

 # s3 = swapped (0=F) *# t9 = A[j+1]*

 # s4 = j

 # s5 = N-1-i

Θεωρώ πως στην κλήση της συνάρτησης Bubble_Sort: a0 = &A[0], a1 = N

Bubble_Sort:

```
#### Αποθηκεύω πραγματίδια στην στοίβα ####
addi $sp, $sp, -28
sw $ra, 24($sp)
sw $s0, 20($sp)
sw $s1, 16($sp)
sw $s2, 12($sp)
sw $s3, 8($sp)
sw $s4, 4($sp)
sw $s5, 0($sp)
#### Αρχικοποίηση πραγματιδίων ####
add $s0, $a0, $zero     # s0 = &A[0]
addi $s1, $a1, -1       # s1 = N-1
add $s2, $zero, $zero   # s2 = i = 0
```

```
        j For_I
For_I:
        #### Συνθήκη For_I ####
        beq $s2, $s1, Finale    # if(s2==s1) <=> if(i==N-1) break;
        #### Ισχύει i<N-1 ####
        add $s3, $zero, $zero # s3 = swapped = 0 (F)
        add $s4, $zero, $zero # s4 = j = 0
        sub $s5, $s1, $s2       # s5 = N-1-i
        j For_J
Next_I:
        addi $s2, $s2, 1        # i++
        beq $s3, $zero, Finale # if(!swapped) break;
        j For_I
For_J:
        #### Συνθήκη For_J ####
        beq $s4, $s5, Next_I   # if(j == N-1-i) Next_I();
        #### Ισχύει j < N-1-i ####
        sll $t2, $s4, 2         # t2 = s4*4 = j*4
        add $t2, $t2, $s0       # t2 += s0 = &A[j]
        lw $t8, 0($t2)          # t8 = A[j]
        lw $t9, 4($t2)          # t9 = A[j+1]
        addi $s4, $s4, 1        # j++
        slt $t1, $t9, $t8       # if(t9<t8) <=> if(A[j+1]<A[j]) t1=1;
                                # else t1=0;
        beq $t1, $zero, For_J  # if(t1==0) <=> if(A[j+1]>=A[j]) For_J();
        addi $s3, $zero, 1      # s3 = swapped = 1 (T)
        #### Καλώ την Swap(*x, *y) ####
        add $a0, $t2, $zero     # a0 = t2 = &A[j]
        addi $a1, $t2, 4        # a1 = t2+4 = &A[j+1]
        jal Swap
        j For_J
Finale:
```

```
        #### Ελευθερώνω πραγματίδια απ' τη στοίβα ####
        lw $ra, 24($sp)
        lw $s0, 20($sp)
        lw $s1, 16($sp)
        lw $s2, 12($sp)
        lw $s3, 8($sp)
        lw $s4, 4($sp)
        lw $s5, 0($sp)
        addi $sp, $sp, 28
        #### Επιστρέφω main ####
        jr $ra
#### Θεωρώ στην κλήση της Swap: a0 = *x, a1 = *y ####
```

Swap:

```
        lw $t0, 0($a0)          # t0 = *x
        lw $t1, 0($a1)          # t1 = *y
        sw $t1, 0($a0)          # Αποθηκεύω στην διεύθυνση του a0 (x) το περιεχόμενο του t1 (*y)
        sw $t0, 0($a1)          # Αποθηκεύω στην διεύθυνση του a1 (y) το περιεχόμενο του t0 (*x)
        jr $ra
############# Τέλος των Bubble_Sort & Swap #############
```

```
void cocktailSort(int *A, int N) {
      bool swapped = true;
      int start = 0;
      int end = N-1;

      while (swapped) {
            swapped = false;
            for(int i = start; i < end; i++) {
                  if (A[i] > A[i+1]) {
                        swap(&A[i], &A[i+1]);
                        swapped = true;
                  }
            }
            if (!swapped) break;

            swapped = false;
            end--;
            for(int i = end - 1; i >= start; i--) {
                  if(A[i] > A[i+1] {
                        swap(&A[i], &A[i+1]);
                        swapped = true;
                  }
            }
            start++;
      }
}
```

```
void swap (int *x, int * y) {
      int temp = *x;

      *x = *y;
      *y = temp;
}
```

############ Αρχή των Cocktail_Sort & Swap ############

 # s0 = &A[0] # t0 = &A[i]

 # s1 = swapped (0=F) # t1, slt purposes

 # s2 = start # t2 = A[i]

 # s3 = end # t3 = A[i+1]

 # s4 = i

Θεωρώ πως στην κλήση της συνάρτησης Cocktail_Sort: a0 = &A[0], a1 = N

Cocktail_Sort:

```
#### Αποθηκεύω πραγματίδια στην στοίβα ####
addi $sp, $sp, -24
sw $ra, 20($sp)
sw $s0, 16($sp)
sw $s1, 12($sp)
sw $s2, 8($sp)
sw $s3, 4($sp)
sw $s4, 0($sp)
#### Αρχικοποίηση πραγματιδίων ####
add $s0, $a0, $zero    # s0 = &A[0]
```

```
        addi $s1, $zero, 1      # s1 = swapped = 1 (T)
        add $s2, $zero, $zero   # s2 = start = 0
        addi $s3, $a1, -1       # s3 = end = N-1
        j While
While:
        #### Συνθήκη While ####
        beq $s1, $zero, Finale  # if(swapped == F) break;
        #### Ισχύει swapped == T ####
        add $s1, $zero, $zero   # s1 = swapped = 0 (F)
        add $s4, $s2, $zero     # s4 = i = start
        j First_For
First_For:
        #### Συνθήκη First_For ####
        slt $t1, $s4, $s3       # if(i < end) t1 == 1;
                                # else t1 == 0;
        beq $t1, $zero, Mid     # if(t1 == 0) break;
        #### Ισχύει i<end ####
        sll $t0, $s4, 2
        add $t0, $t0, $s0       # t0 = &A[i]
        lw $t2, 0($t0)          # t2 = A[i]
        lw $t3, 4($t0)          # t3 = A[i+1]
        addi $s4, $s4, 1        # i++
        slt $t1, $t3, $t2       # if(t3<t2) <=> if(A[i+1] < A[i]) t1=1;
                                # else t1=0;
        beq $t1, $zero, First_For # if(t1==0) <=> if(if A[i+1] >= A[i]) First_For();
        #### Καλώ την Swap(*x, *y) ####
        add $a0, $t0, $zero     # a0 = t0 = &A[i]
        addi $a1, $t0, 4        # a1 = t0+4 = &A[i+1]
        jal Swap
        addi $s1, $zero, 1      # s1 = swapped = 1 (T)
        j First_For
Mid:
```

```
        beq $s1, $zero, Finale# if(!swapped) break;
        add $s1, $zero, $zero # swapped = 0 = F
        addi $s3, $s3, -1       # end--
        addi $s4, $s3, -1       # i = end - 1
        j Second_For
Second_For:
        slt $t1, $s4, $s2       # if(i<start) t1=1;
                                # else t1=0;
        bne $t1, $zero, Mid2  # if(t1==1) break;
        sll $t0, $s4, 2
        add $t0, $t0, $s0       # t0 = &A[i]
        lw $t2, 0($t0)          # t2 = A[i]
        lw $t3, 4($t0)          # t3 = A[i+1]
        addi $s4, $s4, -1       # i--
        slt $t1, $t3, $t2       # if(A[i+1] < A[i]) t1 == 1;
                                # else t1 == 0;
        beq $t1, $zero, Second_For
        #### Καλώ την Swap(*x, *y) ####
        add $a0, $t0, $zero     # a0 = t0 = &A[j]
        addi $a1, $t0, 4        # a1 = t0+4 = &A[j+1]
        jal Swap
        addi $s1, $zero, 1      # s1 = swapped = 1 (T)
        j Second_For
Mid2:
        addi $s2, $s2, 1        # s2 == start++
        j While
Finale:
        #### Ελευθερώνω πραγματίδια απ' τη στοίβα ####
        lw $ra, 20($sp)
        lw $s0, 16($sp)
        lw $s1, 12($sp)
        lw $s2, 8($sp)
```

```
        lw $s3, 4($sp)
        lw $s4, 0($sp)
        addi $sp, $sp, 24
        #### Επιστρέφω main ####
        jr $ra
#### Θεωρώ στην κλήση της Swap: a0 = *x, a1 = *y ####
Swap:
        lw $t0, 0($a0)          # t0 = *x
        lw $t1, 0($a1)          # t1 = *y
        sw $t1, 0($a0)          # Αποθηκεύω στην διεύθυνση του a0 (x) το περιεχόμενο του t1 (*y)
        sw $t0, 0($a1)          # Αποθηκεύω στην διεύθυνση του a1 (y) το περιεχόμενο του t0 (*x)
        jr $ra
############# Τέλος των Cocktail_Sort & Swap #############
```

```
void insertionSort(int *A, int N) {
      int key, j;

      for(int i = 1; i < N; i++) {
            key = A[i];
            j = i - 1;

            while (j >=0 && A[j] > key) {
                  A[j+1] = A[j];
                  j--;
            }
            A[j+1] = key;
      }
}
```

############ Αρχή της Ins_Sort ############

 # s0 = &A[0] *# t0 = &A[i]*

 # s1 = N *# t1, slt purposes*

 # s2 = key *# t2 = A[i]*

 # s3 = i *# t3 = &A[j]*

 # s4 = j *# t4 = A[j]*

Θεωρώ πως στην κλήση της συνάρτησης Ins_Sort: a0 = &A[0], a1 = N

Ins_Sort:

Αποθηκεύω πραγματίδια στην στοίβα

addi $sp, $sp, -24

sw $ra, 20($sp)

sw $s0, 16($sp)

sw $s1, 12($sp)

sw $s2, 8($sp)

sw $s3, 4($sp)

sw $s4, 0($sp)

Αρχικοποίηση πραγματιδίων

add $s0, $a0, $zero *# s0 = &A[0]*

add $s1, $a1, $zero *# s1 = N*

addi $s3, $zero, 1 *# i = 1*

j ***For***

For:

beq $s3, $s1, ***Finale*** *# if(s3==s1) <=> if(i==N) Finale();*

```
        sll $t0, $s3, 2          # t0 = s3*4 = i*4
        add $t0, $t0, $s0        # t0 += s0  <=>  t0 = &A[i]
        lw $s2, 0($t0)           # s2 = key = A[i]
        addi $s4, $s3, -1        # s4 = s3 - 1  <=>  j = i-1
        j While
While:
        #### 1η Συνθήκη While ####
        slt $t1, $s4, $zero      # if(s4<0) <=> if(j<0) t1=1;
                                 # else t1=0;
        bne $t1, $zero, Mid      # if(t1==0) <=> if(j<0) Mid();
        #### 2η Συνθήκη While ####
        sll $t3, $s4, 2          # t3 = s4*4 <=> t3 = j*4
        add $t3, $t3, $s0        # t3 += s0 <=> t3 = &A[j]
        lw $t4, 0($t3)           # t4 = A[j]
        slt $t1, $s2, $t4        # if(s2<t4) <=> if(key<A[j]) t1=1;
                                 # else t1=0;
        beq $t1, $zero, Mid      # if(t1==0) <=> if(key>=A[j]) Mid();
        #### Ισχύουν και οι 2 συνθήκες του While ####
        sw $t4, 4($t3)           # A[j+1] = A[j]
        addi $s4, $s4, -1        # s4-- <=> j--
        j While
Mid:
        sll $t3, $s4, 2          # t3 = s4*4 <=> t3 = j*4
        add $t3, $t3, $s0        # t3 += s0 <=> t3 = &A[j]
        sw $s2, 4($t3)           # A[j+1] = key
        addi $s3, $s3, 1         # s3++ <=> i++
        j For
Finale:
        #### Ελευθερώνω πραγματίδια απ'τη στοίβα ####
        lw $ra, 20($sp)
        lw $s0, 16($sp)
        lw $s1, 12($sp)
```

```
        lw $s2, 8($sp)
        lw $s3, 4($sp)
        lw $s4, 0($sp)
        addi $sp, $sp, 24
        #### Επιστρέφω στην main ####
        jr $ra
############## Τέλος της Ins_Sort #############
```

ΜΕΡΟΣ Γ

Υλοποιήστε ξανά τον αλγόριθμο ταξινόμησης insertion sort χρησιμοποιώντας αυτή τη φορά αναδρομή.

```
void InsRec_Sort(int A[], int N){
    if (N <= 1) return;

    InsRec_Sort(A, N-1);

    int last = A[N-1];
    int j = N-2;

    while (j >= 0 && A[j] > last){
        A[j+1] = A[j];
        j--;
    }
    A[j+1] = last;
}
```

```
############ Αρχή της InsRec_Sort ############
# t1, slt purposes        # t4 = j
# t2 = &A[N-1]            # t5 = &A[j]
# t3 = last               # t6 = A[j]
#### Θεωρώ πως στην κλήση της συνάρτησης InsRec_Sort: a0 = &A[0], a1 = N ####
InsRec_Sort:
    addi $t1, $a1, 2        # if(a1<2) <=> if(N<2) t1=1;
                            # else t1=0;
    bne $t1, $zero, Return
    #### Αποθηκεύω πραγματίδια στην στοίβα ####
    add $sp, $sp, -8
    sw $ra, 4($sp)
    sw $a1, 0($sp)
    addi $a1, $a1, -1       # a1 = N-1
    jal InsRec_Sort         # InsRec_Sort(A, N-1)
    #### Ελευθερώνω πραγματίδια απ'τη στοίβα ####
    lw $ra, 4($sp)
    lw $a1, 0($sp)
    add $sp, $sp, 8
    #### Δημιουργία last, j ####
```

```
        addi $t2, $a1, -1       # t2 = a1-1 = N-1
        sll $t2, $t2, 2         # t2 *= 4
        add $t2, $t2, $a0       # t2 += a0 = &A[N-1]
        lw $t3, 0($t2)          # t3 = last = A[N-1]
        addi $t4, $a1, -2       # t4 = j = a1-2 = N-2
        j While
Return:
        jr $ra                  # Los πούλος Hermanos *Breaking Bad ref*
While:
        #### 1η Συνθήκη While ####
        slt $t1, $t4, $zero     # if(t4<0) <=> if(j<0) t1=1;
                                # else t1=0;
        bne $t1, $zero, Mid     # if(t1==1) <=> if(j<0) Mid();
        #### 2η Συνθήκη While ####
        sll $t5, $t4, 2         # t5 = t4*4 = j*4
        add $t5, $t5, $a0       # t5 += a0 <=> t5 = &A[j]
        lw $t6, 0($t5)          # t6 = A[j]
        slt $t1, $t3, $t6       # if(t3<t6) <=> if(key<A[j]) t1=1;
                                # else t1=0;
        beq $t1, $zero, Mid     # if(t1==0) <=> if(key>=A[j]) Mid();
        #### #### Ισχύουν και οι 2 συνθήκες του While ####
        sw $t6, 4($t5)          # A[j+1] = A[j]
        addi $t4, $t4, -1       # s2-- <=> j--
        j While
Mid:
        sll $t5, $t4, 2         # t5 = t4*4 = j*4
        add $t5, $t5, $a0       # t5 += a0 <=> t5 = &A[j]
        sw $t3, 4($t5)          # A[j+1] = last
        jr $ra
############# Τέλος της InsRec_Sort #############
```