ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ

(2020-2021)

1η Σειρά Ασκήσεων

Ονοματεπώνυμο, Α.Μ., στοιχεία επικοινωνίας:

- Χρήστος Τσούφης
    - 03117176
    - el17176@mail.ntua.gr

Μέρος Α

Δίνεται το πρόγραμμα σε **C**:

```
int dotProduct(int v[], int u[], int n){
    int res = 0;
    for(int i=0; i<n; i++)
        res += v[i]*u[i];

    return res;
}
```

Το ίδιο πρόγραμμα σε **assembly MIPS**:

```
      addi  $sp, $sp, -8          #create 8 positions in stack
      sw    $s1, 4($sp)           #initialization, (v)
      sw    $s2, 0($sp)           #initialization, (u)
      add   $t1, $zero, $zero     #initialization of t1 with 0, (res)
      add   $t2, $zero, $zero     #initialization of t2 with 0, (i)
FOR:  lw    $s1, 0($a0)           #load to s1 the value of a0 that has the
                                  #address of the first element of array v
      lw    $s2, 0($a1)           #same for s2
      mul   $s1, $s1, $s2         #v[i]*u[i]
      add   $t1, $t1, $s1         #res goes to t1 for each i
      addi  $a0, $a0, 4           #save to a0 the next value of array v
      addi  $a1, $a1, 4           #same for u
      addi  $t2, $t2, 1           #i++
      slt   $t3, $t2, $a2         #n = a2
                                  #cheack if t2<a2, which means i<n
      bne   $t3, $zero, FOR       #if i<n then go to FOR
      add   $v0, $t1, $zero       #v0 is the returning value
      lw    $s2, 0($sp)           #load register for main
      lw    $s1, 4($sp)           #same here
      addi  $sp, $sp, 8           #dismiss stack memory
      jr    ra                    #jump to return address
```

Μέρος Β

Δίνεται το πρόγραμμα σε **C**:

```
int isPalindrome(char *s){
    if(s==nullptr) return 0;     //null string
    char *ptr=s;
    while(*(ptr+1)!='\0')        //ptr shows at the end of string
    ptr++;
    while(*s==*ptr && s<ptr){    //s shows right & ptr shows left
        s++; ptr--;
    }
    if(s>=ptr) return 1;
    else return 0;
}
```

Το ίδιο πρόγραμμα σε **assembly MIPS**:

```
      lbu   $t3, 0($a0)                #dereference s
      beq   $t3, $zero, END            #if null string, return 0
      addi  $t0, $a0, 0                #char *ptr=s
      addi  $t4, $zero, 1              #t4 = 1

FIRST_LOOP:
      lbu   $t2, 1($t0)                #t2 = *(ptr+1)
      beq   $t2, $zero, SECOND_LOOP    #if *(ptr+1)==’\0’, jump to 2nd loop
      addi  $t0, $t0, 1                #ptr++
      j     FIRST_LOOP                 #jump to 1st loop

SECOND_LOOP:
      lbu   $t3, 0($a0)                #dereference s, t3 = *s
      lbu   $t1, 0($t0)                #dereference ptr, t1 = *ptr
      bne   $t3, $t1, END              #if *ptr != *s, jump to end
      addi  $a0, $a0, 1                #s++
      addi  $t0, $t0, -1               #ptr--
      slt   $t5, $a0, $t0              #s < ptr
      beq   $t5, $t4, SECOND_LOOP      #if s >= ptr, jump to 2nd loop
      addi  $v0, $zero, 1              #v0 = 1
      jr    ra                         #return to ra

END:
      add   $v0, $zero, $zero          #there is no palindrome, v0 = 0
      jr    ra                         #jump to return address
```

Μέρος Γ

Δίνεται το πρόγραμμα σε **assembly MIPS**:

```
     li   $t5, ‘$’             #load operator $
     li   $t6, ‘/’             #load operator /
     li   $t7, ‘*’             #load operator *
     li   $t8, ‘-’             #load operator –
     li   $t9, ‘+’             #load operator +
LOOP:
     lw   $t0, 0($a0)          #put a0 to t0
     beq  $t0, $t5, END        #if equal, jump to end
     addi $a0, $a0, 1          #the current symbol calc. with t0
     beq  $t0, $t6, DIVIDE     #if division, jump to DIVIDE
     beq  $t0, $t7, MULTIPLY   #if multiplication, jump to MULTIPLY
     beq  $t0, $t8, SUBTRACT   #if subtraction, jump to SUBTRACT
     beq  $t0, $t9, ADD        #if addition, jump to ADD
     addi $t0, $t0, -48        #transform with ASCII to int
     addi $sp, $sp, -4         #4 positions in stack
     sw   $t0, 0($sp)          #save t0 in stack
     j    LOOP                 #jump to LOOP
DIVIDE:
     lw   $t1, 0($sp)          #save to t1 the 1st number
     lw   $t2, 4($sp)          #save to t2 the 1st number
     addi $sp, $sp, 4          #use the first 4 positions for result
     div  $t2, $t2, $t1        #division
     sw   $t2, 0($sp)          #save result to position 0
     j    LOOP                 #jump to LOOP
MULTIPLY:
     lw   $t1, 0($sp)          #save to t1 the 1st number
     lw   $t2, 4($sp)          #save to t2 the 1st number
     addi $sp, $sp, 4          #use the first 4 positions for result
     mul  $t2, $t2, $t1        #multiplication
     mflo $t2, 0($sp)          #save lo since it begins with one-digit
     sw   $t2, 0($sp)          #save result to position 0
     j    LOOP                 #jump to LOOP
SUBTRACT:
     lw   $t1, 0($sp)          #save to t1 the 1st number
     lw   $t2, 4($sp)          #save to t2 the 1st number
     addi $sp, $sp, 4          #use the first 4 positions for result
     sub  $t2, $t2, $t1        #subtraction
     sw   $t2, 0($sp)          #save result to position 0
     j    LOOP                 #jump to LOOP
ADD:
     lw   $t1, 0($sp)          #save to t1 the 1st number
     lw   $t2, 4($sp)          #save to t2 the 1st number
     addi $sp, $sp, 4          #use the first 4 positions for result
     add  $t2, $t2, $t1        #addition
     sw   $t2, 0($sp)          #save result to position 0
     j    LOOP                 #jump to LOOP
END:
     lw   $t1, 0($sp)          #save result to t1
     addi $sp, $sp, 4          #dismiss stack memory
     add  $v0, $t1, $zero      #save result to v0
     jr   ra                   #jump to return address
```