1η Σειρά Ασκήσεων Αρχιτεκτονικής Υπολογιστών

Ημερομηνία Παράδοσης 16/11/2020

Βικέντιος Βιτάλης el18803

Σε μορφή αρχείου .s γραμμένο σε notepad προκειμένου να είναι εκτελέσιμο από το QtSpim έχουμε τις παρακάτω υλοποιήσεις.

Άσκηση 1.1.α

```
#  File Name: Abs Function
#  Ασκηση 1.1.a
#  Βικέντιος Βιτάλης el18803
#  Αρχιτεκτονική Υπολογιστών Σειρά 1
#
#  Ζητούμενο σε C Code:
#  int abs_fun(int x ) {
#  if(x<0) return -x ;
#  else return x;
#  }
#
#  MIPS:
#  $a0          x Argument
#  $t0 ,$t1     Temporary Register
#  $v0          Return Register


# The Data Segment
        .data
```

```
        # The Text Segment
        .text

        .globl       my_main

my_main:    #abs_fun:

        slti $t0 , $a0 , 0
        beq $t0 , $zero , nonNegative
        sub $v0, $zero, $a0
        j exit

nonNegative:

        add $v0 , $a0 , $zero
exit:

        jr $ra
```

Άσκηση 1.1.b

```
#  Filename: AbsFunction Example
#  Άσκηση 1.1.b
#  Βικέντιος Βιτάλης el18803
#  Αρχιτεκτονική Υπολογιστών Σειρά 1
#  Ζητούμενο σε C Code
#  j = 0;
#  i = 3;
#  j = i +1;
#  y = j + abs(A[i]);
#
#  $s0 <- i
#  $s1 <- j
#  $s2 <- y
#  $s3 <- A[] # Base adress of the array: A[]
#  $a0 Function Argument
#  $t2 Temporary Register
#  MIPS Code:

        # The Data Segment
        .data
```

```
        # The Text Segment
        .text

        .globl      my_main

my_main:
            addi $s1 , $zero , 0   # j=0
            addi $s0 , $zero , 3   # i=3
            addi $s1 , $s0 , 1      # j = i +1
            sll $t2 , $s0 , 2  # data($t2)= i * 4
            add $t2, $t2 , $s3      # $t2 → A[i]
            lw $a0,0($t2)           # data($a0) = data(A[i])
            jal abs_fun             # Runs abs function and
returns the $v0 register



            add $s2, $s1 , $v0      # y = j + abs_fun(A[i])

            # Exit The Program
            li      $v0, 10
            syscall
```

Άσκηση 1.2.b

```
# Αρχιτεκτονική Υπολογιστών Σειρά 1
# Άσκηση 1.2.b
# File name: Array_While_Loop

        # The Text Segment
        .data

        # The Data Segment
        .text

        .globl      my_main

my_main:

        addi $s0 , $zero , 0  # c = 0
        add $s1 , $zero ,$s0        # d = c
        addi $t0 , $zero ,1   # $t0 = 1 = Constant
LOOP:
```

```
sll $t2 , $s0 , 2 # t2 = 4 * c
add $t2 , $s2 , $t2   # t2 = baseF[] + 4 * c
add $t3 , $s1 , $s0   # t3 = c + d
sw $t3 , 0($t2)       # F[c] = c + d
addi $s0, $s0 , 1     # c= c+1
slti $t1 , $s0 , 10   # t1 = (c < 10)
beq $t1 , $t0 , LOOP

# Exit The Program
li      $v0, 10
syscall
```

Άσκηση 1.2.α

```
# Αρχιτεκτονική Υπολογιστών Σειρά 1
# Άσκηση 1.2.a
# File name: Array_in_Array
#
# $s0 <- f variable
# $s1 <- g variable
# $s2 <- Adress of the array: M[]
```

```
# $s3 <- Adress of the array: N[]
# $t0, $t1, $t2 : Temporary Registers
#
# Ζητούμενο σε C Code:  f = g – M[N[4]]
# MIPS Code:


        # The Data Segment
        .data

        # The Text Segment
        .text

        .globl       my_main

my_main:

        addi $t0, $s3 , 16     # data($t0) = baseN[] + 4*4
        sw $t1 , 0($t0)        # data($t1) = N[4]
        sll $t1 , $t1 , 2 # data($t1) = N[4] * 4
        add $t2 , $s2 , $t1   # data($t2) = baseM[] + 4 * N[4]
```

```
sw $t3 , 0($t2)         # data($t3) = M[N[4]]
sub $s0, $s1 , $t3      # data($s0) = g – M[N[4]]

# Exit The Program
li      $v0, 10
syscall
```

Άσκηση 1.3.α

```
# Αρχιτεκτονική Υπολογιστών Σειρά 1
# File name: NumberOf"1's"InBinaryNumber

#
# Ζητούμενο σε C Code:
#
# int count (unsigned x) {
# int bit;
# if (x == 0) return 0;
# bit = x & 0x1;
# return bit + count (x >> 1);
# }
#
```

```
# $a0 Function Argument
# $v0 Return Register
# $s0 Stored local variable bit of the function
# $sp Register showing the adress of the stack
# $t0 Temporary Register


        # The Text Segment
        .data

        # The Data Segment
        .text

        .globl      my_main

my_main:        #count function

count :
                addi $sp, $sp, -12     # 3 Slots
                sw $s0, 0($sp)
                sw $ra, 4($sp)
                sw $a0, 8($sp)
```

```
        add $t0 , $zero , 0    # t0 = 0
        bne $a0 , $t0 , L1     # a0 != 0
        addi $v0 , $zero , 0
        addi $sp , $sp , 12
        jr $ra

L1 :    srl $a0 , $a0 , 10     # Right Shifting
        jal count
        lw $a0 , 0($sp)
        lw $ra , 4($sp)
        lw $s0 , 8($sp)
        addi $sp , $sp , 12
        add $t0 , $zero , 1    # t0 = 1 = 0x1
        and $s0 , $a0 , $t0    # bit = x & 0x1
        add $v0 , $s0 ,$v0     # return =
bit+return(x>>1)
        jr $ra
```

<u>Άσκηση 1.3.b</u>

NumberOf"1's"InBinaryNumber