$1^{η}$ Άσκηση στην Αρχιτεκτονική Υπολογιστών

Στεφανάκης Γεώργιος (el18436)

Ακ. έτος 2020-2021, 5ο Εξάμηνο, Σχολή ΗΜΜΥ

Μέρος Α – Εσωτερικό Γινόμενο

```
            addi  $sp, $sp, -8
            sw    $s2, 4($sp)
            sw    $s1, 0($sp)        # save previous values of $s1, $s2
            add   $t1, $zero, $zero  # $t1 = result
            add   $t2, $zero, $zero  # $t2 = i
FOR:        lw    $s1, 0($a0)
            lw    $s2, 0($a1)        # $s1 = v[i], $s2 = u[i]
            mul   $s1, $s1, $s2
            add   $t1, $t1, $s1      # add last product to result
            addi  $a0, $a0, 4
            addi  $a1, $a1, 4        # move v, u pointers one position to the right
            addi  $t2, $t2, 1        # i = i + 1
            slt   $t3, $t2, $a2      # $t3 = flag(is i < n)
            bne   $t3, $zero, FOR    # if i < n goto FOR
            add   $v0, $t1, $zero
            lw    $s1, 0($sp)
            lw    $s2, 4($sp)        # restore previous values of $s1, $s2
            addi  $sp, $sp, 8
            jr    $ra
```

Μέρος Β – Παλινδρομικές Συμβολοσειρές

```
# $a0 contains *s, that is the address of the first element of s[]
# function returns $v0 = 1 if s is palindrome, $v0 = 0 if it isn't
# using $t1 as a pointer to the byte on the left s[0], $t2 as a pointer to the right of
# the last character of s and checking for symmetry sequentially

isPalindrome:
            addi  $sp, $sp, -8
            sw    $s1, 0($sp)
            sw    $s2, 4($sp)

            addi  $v0, $zero, 1  # $v0 = 1, useful for later comparisons
            addi  $t1, $a0, 0
            addi  $t2, $a0, 0

WHILE:      lb    $t0, 0($t2)
            addi  $t2, $t2, 1    # since chars are 1 byte long, this moves $t2 one
                                 # position to the right
            bne   $t0, $0, WHILE

            addi  $t2, $t2, -1   # after while loop, $t2 points two positions to the
                                 # of the last element of s
            addi  $t1, $t1, -1

LOOP:       addi  $t1, $t1, 1
            addi  $t2, $t2, -1
            beq   $t1, $t2, END
            slt   $t3, $t2, $t1
            beq   $t3, $v0, END  # exit loop if $t2 < $t1, $t2 goes to the left of $t1
            lb    $s1, 0($t1)
            lb    $s2, 0($t2)
            beq   $s1, $s2, LOOP

END:        slt   $t3, $t1, $t2  # $t1 < $t2
            nor   $t3, $t3, $t3  # equivalent to nor($t3, $t3) = not($t3) = $t1 >= $t2
            addi  $v0, $t3, 0

            lw    $s1, 0($sp)
            lw    $s2, 4($sp)
            addi  $sp, $sp, 8
            jr    $ra
```

Μέρος Γ – Επιθεματική Αριθμητική

```
postfixArithmetic:
            addi  $sp, $sp, -8
            sw    $ra, 0($sp)
            sw    $a0, 4($sp)

WHILE:      lb    $t0, 0($a0)
            beq   $t0, '$', EXIT
            beq   $t0, '+', CALCULATE
            beq   $t0, '-', CALCULATE
            beq   $t0, '*', CALCULATE
            beq   $t0, '/', CALCULATE    # checking if current character is operator or
                                         # exit character
            addi  $sp, $sp, -4
            addi  $t0, $t0, -48          # ascii code for ‘1’ is 49, for ‘2’ is 50 etc.
                                         # -48 gives the real value of the symbol
            add   $t1, $zero, $t0
            sw    $t1, 0($sp)            # push new operand to the stack
            addi  $a0, $a0, 1
            j     WHILE

CALCULATE:                               # procedure that calculates needed operations
            lw    $t3, 0($sp)            # input arguments: $t2, $t3
            lw    $t2, 4($sp)            # output register (result): $t4
            beq   $t0, '+', ADD
            beq   $t0, '-', SUB
            beq   $t0, '*', MUL
            beq   $t0, '/', DIV

ADD:        add   $t4, $t2, $t3
            addi  $sp, $sp, 4
            sw    $t4, 0($sp)
            addi  $a0, $a0, 1
            j     WHILE

SUB:        sub   $t4, $t2, $t3
            addi  $sp, $sp, 4
            sw    $t4, 0($sp)
            addi  $a0, $a0, 1
            j     WHILE

MUL:        mult  $t2, $t3
            mflo  $t4
            addi  $sp, $sp, 4
            sw    $t4, 0($sp)
            addi  $a0, $a0, 1
            j     WHILE

DIV:        div    $t2, $t3
            mflo   $t4
            addi   $sp, $sp, 4
            sw     $t4, 0($sp)
            addi   $a0, $a0, 1
            j      WHILE
```

```
EXIT:                                       # exit method when ‘$’ character is loaded
            lw      $v0, 0($sp)             # storing result on $v0
            lw      $ra, 4($sp)             # restoring $ra, $a0
            lw      $a0, 8($sp)
            addi    $sp, $sp, 12
            jr      $ra
```