ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ

(2020-2021)

6η ΟΜΆΔΑ ΑΣΚΉΣΕΩΝ

Ονοματεπώνυμο:

- Χρήστος Τσούφης

Αριθμός Μητρώου:

- 03117176

Ομάδα Εργαστηρίου:

- Β19

Στοιχεία Επικοινωνίας:

- el17176@mail.ntua.gr

1η Άσκηση

Ο πηγαίος κώδικας:

Κώδικας σε **assembly** (με τα απαραίτητα σχόλια δίπλα από κάθε γραμμή):

```
.globl main #

.equ N, 10

.data

A: .word 0, 1, 2, 7, -8, 4, 5, -12, 11, -2
B: .word 0, 1, 2, 7, -8, 4, 5, 12, -11, -2

.bss
C: .space 4*N

.text
main:
la t0, A                // load address of A in reg t0
la t1, B                // load address of B in reg t1
addi t1, t1, 4*(N-1)    // the value 4*(N-1) is added to the contents
                        // of t1 and the result is placed in t1

la t2, C                // load address of C in reg t2
li t3, N                // load the immediate value N and copy it in t3
                        // this will be the counter of the loop

loop:
lw t4, 0(t0)            // load word A[i] in reg t4
lw t5, 0(t1)            // load word B[N-i-1] in reg t5
add t4, t4, t5          // the contents of t5 is added to the contents of t4
                        // and the result is stored in t4
bge t4, zero, absolute  // the contents of t4 is compared to zero
                        // if t4 is greater than or equal to zero
                        // control jumps to absolute
not t4, t4              // the contents of t4 is fetched and each of the bits
                        // is flipped and the result is copied into t4
addi t4, t4, 0x00000001 // the value 0x00000001 is added to the contents of t4
                        // and the result is placed in t4
                        // and then continue

absolute:
sw t4, 0(t2)            // 0 is copied from reg t4 to memory. The memory address
                        // is formed by adding the offset to the contents of t2
addi t0, t0, 4          // the value 4 is added to the contents of t0
                        // and the result is placed in t0
                        // in order to get the next value of A
```

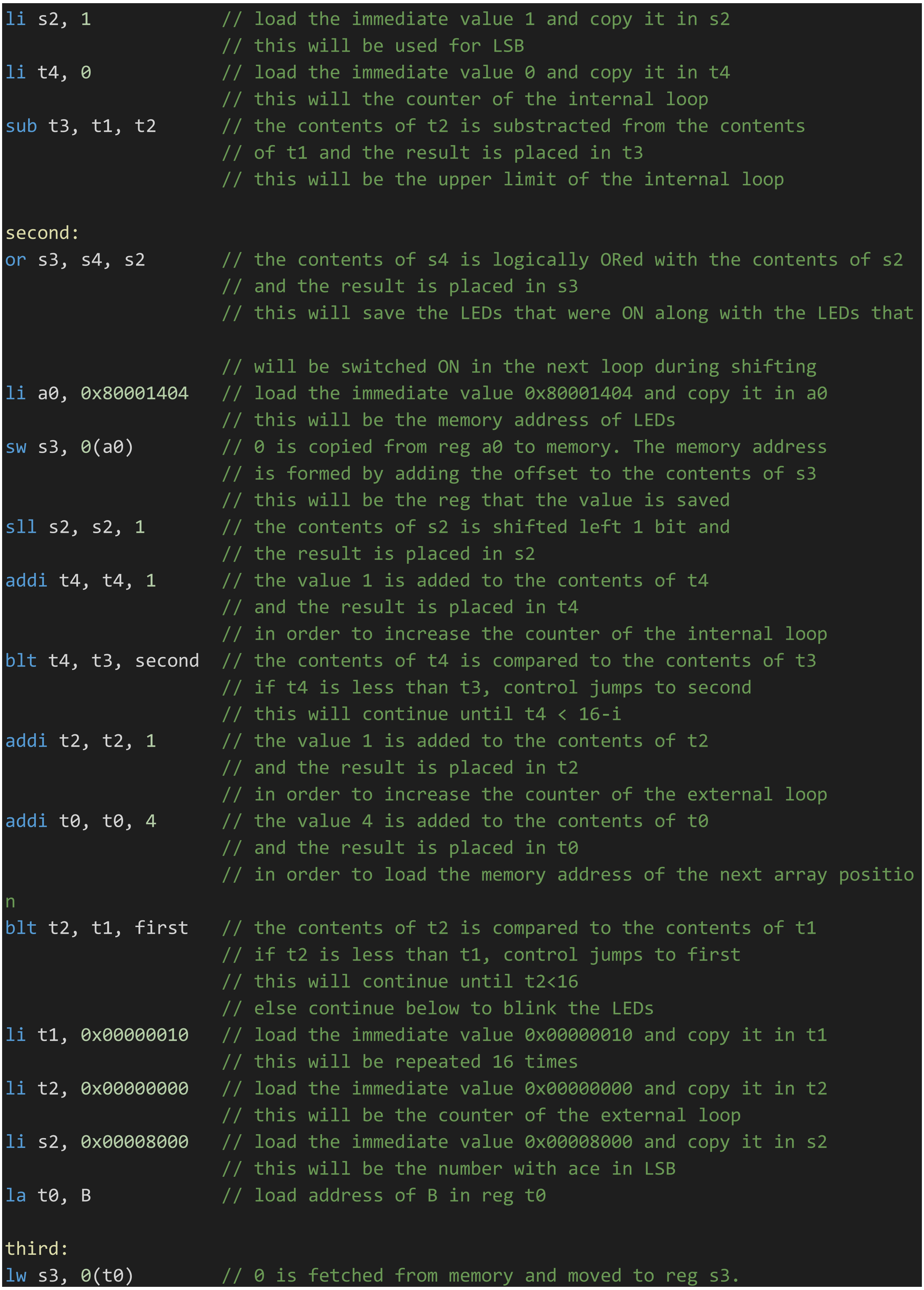
```
addi t1, t1, -4         // the value -4 is added to the contents of t1
                        // and the result is placed in t1
                        // in order to get the next value of B
addi t2, t2, 4          // the value 4 is added to the contents of t2
                        // and the result is placed in t2
                        // in order to get the next value of C
addi t3, t3, -1         // the value -1 is added to the contents of t3
                        // and the result is placed in t3
                        // in order to decrease the counter
bne t3, zero, loop      // the contents of t3 is compared to the contents of zero
                        // if not equal, control jumps to loop

.end
```

2η Άσκηση

Ο <u>πηγαίος κώδικας</u>:

Κώδικας σε **assembly** (με τα απαραίτητα σχόλια δίπλα από κάθε γραμμή):

```
.globl main #

#define GPIO_LEDs 0x80001404
#define GPIO_INOUT 0x80001408

.data

A: .word 0x0000, 0x8000, 0xc000, 0xe000, 0xf000, 0xf800, 0xfc00, 0xfe00, 0xff00, 0xff80, 0xffc0, 0xffe0, 0xfff0, 0xfff8, 0xfffc, 0xfffe

B: .word 0xffff, 0xfffe, 0xfffc, 0xfff8, 0xfff0, 0xffe0, 0xffc0, 0xff80, 0xff00, 0xfe00, 0xfc00, 0xf800, 0xf000, 0xe000, 0xc000, 0x8000

.text

main:
la t0, A            // load address of A in reg t0
addi t1, zero, 16   // the value 16 is added to the contents
                    // of zero and the result is placed in t1
                    // this will be the number of external loops
li t2, 0            // load the immediate value 0 and copy it in t2
                    // this will be the counter of the external loop
li s4, 0            // load the immediate value 0 and copy it in s4
                    // this will be the reg that will have the value from the array

first:
lw s4, 0(t0)        // 0 is fetched from memory and moved to reg s4.
                    // The memory address is formed by adding the offset
                    // to the contents of t0 in order to load the needed value
```



```
li s2, 1            // load the immediate value 1 and copy it in s2
                    // this will be used for LSB
li t4, 0            // load the immediate value 0 and copy it in t4
                    // this will the counter of the internal loop
sub t3, t1, t2      // the contents of t2 is substracted from the contents
                    // of t1 and the result is placed in t3
                    // this will be the upper limit of the internal loop

second:
or s3, s4, s2       // the contents of s4 is logically ORed with the contents of s2
                    // and the result is placed in s3
                    // this will save the LEDs that were ON along with the LEDs that

                    // will be switched ON in the next loop during shifting
li a0, 0x80001404   // load the immediate value 0x80001404 and copy it in a0
                    // this will be the memory address of LEDs
sw s3, 0(a0)        // 0 is copied from reg a0 to memory. The memory address
                    // is formed by adding the offset to the contents of s3
                    // this will be the reg that the value is saved
sll s2, s2, 1       // the contents of s2 is shifted left 1 bit and
                    // the result is placed in s2
addi t4, t4, 1      // the value 1 is added to the contents of t4
                    // and the result is placed in t4
                    // in order to increase the counter of the internal loop
blt t4, t3, second  // the contents of t4 is compared to the contents of t3
                    // if t4 is less than t3, control jumps to second
                    // this will continue until t4 < 16-i
addi t2, t2, 1      // the value 1 is added to the contents of t2
                    // and the result is placed in t2
                    // in order to increase the counter of the external loop
addi t0, t0, 4      // the value 4 is added to the contents of t0
                    // and the result is placed in t0
                    // in order to load the memory address of the next array positio
n
blt t2, t1, first   // the contents of t2 is compared to the contents of t1
                    // if t2 is less than t1, control jumps to first
                    // this will continue until t2<16
                    // else continue below to blink the LEDs
li t1, 0x00000010   // load the immediate value 0x00000010 and copy it in t1
                    // this will be repeated 16 times
li t2, 0x00000000   // load the immediate value 0x00000000 and copy it in t2
                    // this will be the counter of the external loop
li s2, 0x00008000   // load the immediate value 0x00008000 and copy it in s2
                    // this will be the number with ace in LSB
la t0, B            // load address of B in reg t0

third:
lw s3, 0(t0)        // 0 is fetched from memory and moved to reg s3.
```

```
                    // The memory address is formed by adding the offset
                    // to the contents of t0
li s2, 0x00008000   // load the immediate value 0x00008000 and copy it in s2
li t4, 0            // load the immediate value 0 and copy it in t4
                    // this will be the counter of the internal loop
sub t3, t1, t2      // the contents of t2 is substracted from the contents
                    // of t1 and the result is placed in t3
                    // this will be the upper limit of the internal loop

fourth:
xor s4, s3, s2      // the contents of s3 is logically XORed with the
                    // contents of s2 and the result is placed in s4
                    // this will be the LEDs that were ON previously or OFF during s
hifting
li a0, 0x80001404   // load the immediate value 0x80001404 and copy it in a0
                    // this will be the memory address of LEDs
sw s4, 0(a0)        // 0 is copied from reg a0 to memory. The memory address
                    // is formed by adding the offset to the contents of s4
                    // this will be the reg that the value is saved
srli s2, s2, 1      // the contents of s2 is shifted right 1 bit and
                    // the result is placed in s2 until it reaches LSB
addi t4, t4, 1      // the value 1 is added to the contents of t4
                    // and the result is placed in t4
                    // in order to increase the external counter
blt t4, t3, fourth  // the contents of t4 is compared to the contents of t3
                    // if t4 is less than t3, control jumps to fourth
addi t2, t2, 1      // the value 1 is added to the contents of t2
                    // and the result is placed in t2
                    // in order to increase the internal counter
addi t0, t0, 4      // the value 4 is added to the contents of t0
                    // and the result is placed in t0
                    // in order to load the memory address of the previous array pos
ition
blt t2, t1, third   // the contents of t2 is compared to the contents of t1
                    // if t2 is less than t1, control jumps to third

.end
```

Παρατήρηση – Επεξήγηση:

Κατά το άναμμα των LEDs, η διαδικασία ξεκινάει από το LSB και ολισθαίνει προς τα αριστερά (δηλ. σβήνει το LSB LED και ανάβει το 2^{o} LSB LED) έως ότου φτάσει στο MSB. Κρατώντας αναμμένο το MSB LED, επαναλαμβάνει την ίδια διαδικασία έως ότου η ένδειξη φτάσει στο 2^{o} MSB LED. Τέλος, κρατώντας αναμμένα τα 2 MSB LEDs, επαναλαμβάνει την ίδια διαδικασία. Αφού ανάψουν όλα τα LEDs, αρχίζει το σβήσιμό τους με την εξής διαδικασία. Σε κάθε επανάληψη σβήνεται το MSB LED και γίνεται ολίσθηση προς τα δεξιά του σβησμένου LED έως ότου φτάσει στο LSB. Ύστερα, κρατώντας σβησμένο το LSB αρχίζει πάλι σβήνοντας το MSB και γίνεται ολίσθηση προς τα δεξιά του σβησμένου LED έως ότου φτάσει στο 2^{o} LSB. Το πρόγραμμα τελειώνει όταν σβήσουν όλα τα LEDs.