$6^{η}$ Εργαστηριακή Άσκηση στο Εργαστήριο Μικροεπεξεργαστών

Ομάδα Β 3

Αλέξανδρος Κυριακάκης (03112163),
Ιωάννης Αλεξόπουλος (03117001)

Ιανουάριος 2021

$1^{η}$ Άσκηση

```
1  .globl main
2
3  .equ N, 10
4
5  .data
6  A: .word 0,1,2,7,-8,4,5,-12,11,-2
7  B: .word 0,1,2,7,-8,4,5,12,-11,-2
8
9  .bss
10
11 C: .space 4*N
12
13 .text
14
15 main:
16     la t0, A     # beginning address of A array
17     la t1, B     # beginning address of B array
18     la t4, C     # beginning address of C array
19     add t1, t1, 4*(N-1) # t1 -> address of last element of B array
20                         # (word is 4 bytes, byte addressed memory)
21     li t2, 0     # t2 is our counter (i)
22     li t5, N     # t5 holds N
23 looper:
24     lw t3, 0(t0) # load ith element of A array
25     lw a0, 0(t1) # load (N-1-i)th element of B array
26     add t3, t3, a0 # t3 = A[i] + B[N-1-i]
27     sw t3, 0(t4)   # C[i] = t3
28     addi t0, t0, 4  # increase by 4 to get address of next word of A
29     addi t1, t1, -4 # decrease by 4 to get address of previous word of B
```

```
30      addi t2, t2, 1  # increase counter by one (i++)
31      addi t4, t4, 4  # increase by 4 to get address of next word of C
32      bgt t5, t2, looper  # branch to looper if i < N (i = 0,1,2...9)
33
34  .end
```

2^{η} Άσκηση

```
1   #define GPIO_SWs    0x80001400
2   #define GPIO_LEDs   0x80001404
3   #define GPIO_INOUT  0x80001408
4
5   .globl main
6
7   .text
8
9   main:
10  # Einai synexomenhs leitourgias. Molis teleiwsei ksekina apo thn arxh.
11  li x28, 0xFFFF
12  li x29, GPIO_INOUT
13  sw x28, 0(x29)                  # Write the Enable Register
14
15  # Initializations
16  # s2 : Moving Led
17  li s2, 1
18  # s3 : current led value, without moving Led
19  li s3, 0
20  # s6 : current led value, with moving Led
21  li s6, 0
22  # s4 : counter
23  li s4, 0
24  # s5 : 15 is the end
25  li s5, 15
26  # s7 : Flag for reverse movement, (1 -> reverse)
27  li s7, 0
28  # s8 : 1
29  li s8, 1
30
31  # Initialize leds to zero
32  li  a0, GPIO_LEDs
33  li  s6, 1
34  jal x1, WriteToLed
35  # Routine to move leds
36  method:
37     # Na Valeis break edw gia na deis to kinoumeno led!
38     beq   s4, s5, LineEnd # Reached the end
39     addi  s4, s4, 1           # Increase Counter
```

```
40      slli  s2, s2, 1          # Shift left the mooving led
41      or    s6, s2, s3       # Write the new value to leds
42      jal   x1, WriteToLed
43      j     method
44  LineEnd:
45      # Na Valeis break kai edw gia na deis to telos ths grammhs!
46      or    s3, s3, s2       # Save to neo current led value ston s3
47      beq   s5, zero, TheEnd # An anapsane/svisane ola go to  TheEnd
48      addi  s5, s5, -1          # Meiwse to telos kata ena
49      li    s4, 0           # Initalize counter
50      li    s2, 1           # Initialize moving led
51      or    s6, s2, s3      # Light up LSB
52      jal   x1, WriteToLed
53      j     method
54
55  TheEnd:
56      # Na Valeis break kai edw gia na deis to telos ths grammhs!
57      li    t0, 0
58      beq   s7, s8, main
59      # Initializations
60      # s3 : current led value, without moving Led
61      li s3, 0
62      # s6 : current led value, with moving Led
63      li s6, 0
64      # s4 : counter
65      li s4, 0
66      # s5 : 15 is the end
67      li s5, 15
68      # s7 : Flag for reverse movement, (1 -> reverse)
69      li s7, 1
70      # s8 : 1
71      li s8, 1
72      # s2 : Moving Led
73      li s2, 1
74      add s6, s2, zero
75      jal x1, WriteToLed
76
77      j  method
78
79  # Routine to write s6 to LEDs
80  WriteToLed:
81      add   x2, x1, zero            # save return address
82      beq   s7, s8, Reverse         # If flag == 1 then reverse
83      sw    s6, 0(a0)               # Write the LEDs 0000
84      jal   x1, delay
85      add   x1, x2, zero            # write return address
86      ret
87  Reverse:
```

```
88      jal   x1, rotateReverse
89      not   s6, s6                  # Not
90      sw    s6, 0(a0)               # Write the LEDs 0000
91      jal   x1, delay
92      add   x1, x2, zero            # write return address
93      ret
94
95  rotateReverse:
96
97  #          7 6 5 4 3 2 1 0
98  #           X   X   X   X
99  #          6 7 4 5 2 3 0 1
100 #           \ X /   \ X /
101 #            X X     X X
102 #           / X \   / X \
103 #          4 5 6 7 0 1 2 3
104 #           \ \ \ X / / /
105 #            \ \ X X / /
106 #             \ X X X /
107 #              X X X X
108 #             / X X X \
109 #            / / X X \ \
110 #           / / / X \ \ \
111 #          0 1 2 3 4 5 6 7
112
113     # x = ((x & 0x55555555) << 1) | ((x & 0xAAAAAAAA) >> 1); // Swap _<>_
114     li    s9, 0x55555555
115     and  x30, s6, s9
116     slli  x30, x30, 1
117     li    s9, 0xAAAAAAAA
118     and  x31, s6, s9
119     srli  x31, x31, 1
120     or    s6, x30, x31
121     # x = ((x & 0x33333333) << 2) | ((x & 0xCCCCCCCC) >> 2); // Swap __<>__
122     li    s9, 0x33333333
123     and   x30, s6, s9
124     slli  x30, x30, 2
125     li    s9, 0xCCCCCCCC
126     and   x31, s6, s9
127     srli  x31, x31, 2
128     or    s6, x30, x31
129     # x = ((x & 0x0F0F0F0F) << 4) | ((x & 0xF0F0F0F0) >> 4); // Swap ____<>____
130     li    s9, 0x0F0F0F0F
131     and  x30, s6, s9
132     slli  x30, x30, 4
133     li    s9, 0xF0F0F0F0
134     and  x31, s6, s9
135     srli  x31, x31, 4
```

```
136     or    s6, x30, x31
137     # x = ((x & 0x00FF00FF) << 8) | ((x & 0xFF00FF00) >> 8); // Swap ...
138     li    s9, 0x00FF00FF
139     and   x30, s6, s9
140     slli  x30, x30, 8
141     li    s9,  0xFF00FF00
142     and   x31, s6, s9
143     srli  x31, x31, 8
144     or    s6, x30, x31
145     ret
146
147 delay:
148     ret
149
150 .end
```