ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

# Συστήματα Μικρουπολογιστών

$5^{η}$ σειρά ασκήσεων

Αναστάσιος Λαγός - el13531
Κωνσταντίνος Βασιλάκης - el16504

# Ασκήσεις Προσομοίωσης

Στις παρακάτω 4 ασκήσεις προσομοίωσης χρησιμοποιήθηκαν επτά μακροεντολές με χρήση του `INCLUDE macros.asm` , οι οποίες παρουσιάζονται στο τέλος αυτής της αναφοράς.

## Άσκηση 1

```
INCLUDE macros.asm

DATA SEGMENT
    TABLE DB 128 DUP(?) ;directive to create array of 128 unassigned items
    NUM DB 2            ;variable NUM = 2
DATA ENDS

CODE SEGMENT

MAIN PROC FAR
    ASSUME CS:CODE, DS:DATA  ;CS points to code seg, DS points to data seg
    MOV AX,DATA
    MOV DS,AX
    MOV DI,0    ;index register = 0
    MOV CX,128  ;counter register = 128
FILL_TABLE:
    MOV TABLE[DI],CL    ;table[0] = 128
    INC DI              ;DI++
    LOOP FILL_TABLE     ;LOOP while CX-- != 0

    MOV DH,0    ;DX gets the odd value of table
    MOV AX,0    ;AX keeps the cumulative sum
    MOV BX,0    ;odd counter
    MOV DI,0    ;index register to iterate TABLE
    MOV CX,128
CUMSUMODD:
    PUSH AX     ;save current sum
    MOV AH,0    ;AH might not be zero since it holds the current sum therefore
    ↪ we need to make it zero
    MOV AL,TABLE[DI]  ;get TABLE[DI] value on AL
    DIV NUM           ;we basically check for a remainder with division by 2
    CMP AH,0          ;AH is the remainder of the division above
    POP AX            ;get back the current sum
    JE SKIP           ;remainder = 0 then ZF = 1 and jump else ZF = 0 and dont
    ↪ jump
    MOV DL,TABLE[DI]  ;get the odd value
    ADD AX,DX         ;and add to current sum
    INC BX            ;increment the odd counter so we can do a division for the
    ↪ average
SKIP:                 ;if remainder != 0 then we just get to the next value
    INC DI
```

```
    LOOP CUMSUMODD

    MOV DX,0          ;when operand is a word then DIV X ::= AX = (DX AX) / X,
    ↪  DX = remainder
    DIV BX
    CALL PRINT_NUM8_HEX
    PRINTLN ;print new line

    MOV AL,TABLE[0]   ;current max
    MOV BL,TABLE[127] ;current min
    MOV DI,0
    MOV CX,128
FIND_MAX_MIN:
    CMP AL,TABLE[DI]
    JC MAX            ;if AL < TABLE[DI] then CF is set and we jump to MAX
    JMP CHECK_MIN
MAX:
    MOV AL,TABLE[DI]  ;TABLE[DI] is the new current max
    JMP NEXT          ;get the next number
CHECK_MIN:
    CMP TABLE[DI],BL
    JC MIN            ;if TABLE[DI] < BL then CF is set and we jump to MIN
    JMP NEXT
MIN:
    MOV BL,TABLE[DI]  ;TABLE[DI] is the new current MIN
NEXT:
    INC DI
    LOOP FIND_MAX_MIN
    ;the following procedures are copied from the lectures
    CALL PRINT_NUM8_HEX
    PRINT_CHAR "/"
    MOV AL,BL
    CALL PRINT_NUM8_HEX
    EXIT

MAIN ENDP

PRINT_NUM8_HEX PROC NEAR
    MOV DL,AL
    AND DL,0F0H
    MOV CL,4
    ROR DL,CL
    CMP DL,0
    JE SKIPFIRST
    CALL PRINT_HEX
SKIPFIRST:
    MOV DL,AL
    AND DL,0FH
```

```
    CALL PRINT_HEX
    RET
PRINT_NUM8_HEX ENDP

PRINT_HEX PROC NEAR
    CMP DL,9
    JG ADDR1
    ADD DL,48
    JMP ADDR2
ADDR1:
    ADD DL,55
ADDR2:
    PRINT_CHAR DL
    RET
PRINT_HEX ENDP

CODE ENDS
END MAIN
```

## Άσκηση 2

```
INCLUDE macros.asm

data segment
    msg1 db 0AH,0DH,"Z=$"
    msg2 db " W=$"
    msg3 db 0AH,0DH,"Z+W=$"
    msg4 db " Z-W=$"
ends

stack segment
    dw   128  dup(0)
ends

code segment
START:
    ; set segment registers:
    MOV AX, data
    MOV DS, AX
    MOV ES, AX

    ; BL holds Z+W
    MOV BL,0

    ; BH holds the sub
    MOV BH,0

```

```
    ; Print Z and read the value
    PRINT_STR msg1
    ; We read the first digit of Z
    CALL DEC_KEYB
    MOV BL,AL
    MOV AL,0x0A
    MUL BL
    MOV BL,AL
    ; We read the second digit of Z
    CALL DEC_KEYB
    ADD BL,AL

    ; Print W and read the value
    PRINT_STR msg2
    ; We read the first digit of W
    CALL DEC_KEYB
    MOV BH,AL
    MOV AL,0x0A
    MUL BH
    MOV BH,AL
    ; We read the second digit of W
    CALL DEC_KEYB
    ADD BH,AL


    ; Print Z+W
    PRINT_STR msg3
    MOV AL,BL
    ADD AL,BH
    CALL PRINT_RESULT

    ; Print Z-W
    PRINT_STR msg4
    MOV AL,BL
    CMP AL, BH
    ; If AL is less than BH then
    ; do W-Z
    JL display_result_negative
    ; calculate and print Z-W
display_result_positive:
    SUB AL,BH
    CALL PRINT_RESULT
    JMP end_of_program
; calculate and print W-Z
display_result_negative:
    PRINT_CHAR '-'
    MOV AL,BH
    SUB AL,BL
```

```
75      CALL PRINT_RESULT
76
77  end_of_program:
78      JMP START
79  ends
80
81  DEC_KEYB PROC NEAR
82      PUSH DX
83      IGNORE:
84      INPUT_CHAR
85      CMP AL, 'Q'
86      JE ADDR2
87      CMP AL, 30H
88      JL IGNORE
89      CMP AL, 39H
90      JG IGNORE
91      PUSH AX
92      PRINT_CHAR AL
93      POP AX
94      SUB AL, 30H
95      ADDR2: POP DX
96      RET
97  DEC_KEYB ENDP
98
99  PRINT_RESULT PROC NEAR
100     MOV CX,2
101 print_numbers:
102     ; Get first the 4 MSB and
103     ; then the 4 LSB of the byte
104     ; and print the result
105     ROL AL,4
106     MOV DL,AL
107     AND DL,0xF
108     CMP DL,0
109     JNE continue
110     CMP CX,2
111     JE loop_end
112 continue:
113     ; Add 30H to display the
114     ; character correctly
115     ADD DL, 30H
116     CMP DL,'9'
117     JBE below_9
118     ; If it is 0x0A-0x0F
119     ; add 7 to display the
120     ; character correctly
121     add DL,7
122     below_9:
```

```
    PUSH AX
    MOV AH,2
    INT 21H
    POP AX
loop_end:
    LOOP print_numbers
    RET
PRINT_RESULT ENDP

END:
end start
```

### Άσκηση 3

```
INCLUDE macros.asm

PRINT_NUMBER MACRO CHAR
    PUSH AX
    PUSH DX
    MOV DL,CHAR
    ADD DL,30H
    MOV AH,2
    INT 21H
    POP DX
    POP AX
ENDM

MAIN PROC FAR
start:
    ; Get first digit
    CALL HEX_KEYB
    MOV BH,AL

    ; Get second digit
    CALL HEX_KEYB
    MOV BL,AL
    SHL BL,4

    ; Get third digit
    CALL HEX_KEYB
    ADD BL,AL

    PRINT_CHAR '='

    CALL PRINT_DEC
    PRINT_CHAR '='

```

```
34
35      CALL PRINT_OCT
36      PRINT_CHAR '='
37
38      CALL PRINT_BIN
39      PRINTLN
40
41      JMP start
42  MAIN ENDP
43
44  HEX_KEYB PROC NEAR
45      PUSH DX
46      IGNORE:
47      INPUT_CHAR
48      CMP AL, 'Q'
49      JE ADDR2
50      CMP AL,30H
51      JL IGNORE
52      CMP AL,39H
53      JG ADDR1
54      PUSH AX
55      PRINT_CHAR AL
56      POP AX
57      SUB AL,30H
58      JMP ADDR2
59      ADDR1: CMP AL,'A'
60      JL IGNORE
61      CMP AL,'F'
62      JG IGNORE
63      PUSH AX
64      PRINT_CHAR AL
65      POP AX
66      SUB AL,37H
67      ADDR2: POP DX
68      RET
69  HEX_KEYB ENDP
70
71
72  PRINT_DEC PROC NEAR
73      PUSH BX
74      MOV AX,BX
75      MOV BL,100
76      DIV BL
77      MOV AH,0
78      MOV BL,10
79      DIV BL
80
81      PRINT_NUMBER AL
```

```
82      ; Subtract the thousands calculated
83      ; We subtract because the remainder that
84      ; is returned in AH can overflow. For example
85      ; 3999/1000 has remainder 999 which is more
86      ; than one byte and cannot fit in AH to get
87      ; it directly from there
88      ;MOV CL,AL
89
90      MOV DX,1000
91      MOV AH,0
92      MUL DX
93      POP BX
94      PUSH BX
95      SUB BX,AX
96
97      ; If the thousands are not 0 display
98      ; the number
99      ;CMP CL,0
100     ;JE three_decimal_digits
101
102
103 three_decimal_digits:
104     MOV AX,BX
105     MOV BL,100
106     DIV BL
107
108     ; Here the remainder will always fit in
109     ; the AH so we can get it directly from
110     ; there
111     MOV BL,AH
112     MOV BH,0
113     CMP AL,0
114     ;JE two_decimal_digits
115     PRINT_NUMBER AL
116
117 two_decimal_digits:
118     MOV AX,BX
119     MOV BL,10
120     DIV BL
121
122     ; Get the last two digits from the
123     ; remainder and quotidient respectively
124     PRINT_NUMBER AL
125     PRINT_NUMBER AH
126
127     POP BX
128     RET
129 PRINT_DEC ENDP
```

```
130
131 PRINT_OCT PROC NEAR
132     ; Here three bits are directly
133     ; one character in the octal system
134     ; so we divide them in triads
135
136     ; First 3 bits
137     MOV AL,BH
138     SHR AL,1
139     AND AL,0x07
140     PRINT_NUMBER AL
141
142     ; Second 3 bits
143     MOV AL,BH
144     AND AL,0x01
145     SHL AL,2
146     MOV AH,BL
147     SHR AH,6
148     AND AH,0x07
149     OR AL,AH
150     PRINT_NUMBER AL
151
152     ; Third 3 bits
153     MOV AL,BL
154     SHR AL,3
155     AND AL,0x07
156     PRINT_NUMBER AL
157
158     ; Fourth 3 bits
159     MOV AL,BL
160     AND AL,0x07
161     PRINT_NUMBER AL
162
163     RET
164 PRINT_OCT ENDP
165
166 PRINT_BIN PROC NEAR
167     ; In this function we loop
168     ; through all the bits one
169     ; by one
170
171     ; First 4 bits are in BH
172     ROL BH,4
173     MOV CX,4
174 first_4_bits:
175     ROL BH,1
176     MOV AL,BH
177     AND AL,01H
```

```
    ADD AL,30H
    PRINT_CHAR AL
    LOOP first_4_bits

    ; First 8 bits are in BL
    MOV AL,BL
    MOV CX,8
last_8_bits:
    ROL BL,1
    MOV AL,BL
    AND AL,01H
    ADD AL,30H
    PRINT_CHAR AL
    LOOP last_8_bits
    RET
PRINT_BIN ENDP
```

## Άσκηση 4

```
INCLUDE macros.asm

DATA SEGMENT
    CHAR_ARR DB 20 DUP(?);array of 20 unassigned variables (chars)
    NUM_ARR DB 20 DUP(?);array of 20 unsassigned variables (numbers)
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA

    MAIN PROC FAR
            MOV AX,DATA
            MOV DS,AX


        START:
            MOV DI,0    ;index register = 0
            MOV BX,0    ;second index register = 0
        NEXT_CHAR:
            INPUT_CHAR  ;get character from standard input
            CMP AL,61   ; AL == "="?, then exit
            JE FINISH
            CMP AL,13   ; AL == "\n"? then print result
            JE PRINT_CAPITALS
            CMP AL,48   ; AL < "0"? then skip
            JB NEXT_CHAR
            CMP AL,122  ; AL > "z"? then skip
            JA NEXT_CHAR
```

```
            CMP AL,57   ; AL <= "9"? then save num
            JBE KEEP_NUM
            CMP AL,97   ; AL < "a"? then save char
            JB NEXT_CHAR
        KEEP_CHAR:
            PRINT_CHAR AL
            MOV CHAR_ARR[BX],AL
            INC BX
            JMP CHECK
        KEEP_NUM:
            PRINT_CHAR AL
            MOV NUM_ARR[DI],AL
            INC DI
        CHECK:          ;adding indexes to check if 20 is reached
            MOV AX,DI
            ADD AX,BX
            CMP AX,20
            JB NEXT_CHAR
        PRINT_CAPITALS:
            PRINTLN
            MOV CX,BX
            MOV DI,0
        BLOCK1:         ;loop for all CHAR_ARR and print
            MOV AL,CHAR_ARR[DI]
            SUB AL,32
            PRINT_CHAR AL
            INC DI
            LOOP BLOCK1
        PRINT_NUM:
            PRINT_CHAR "-"
            MOV CX,DI
            MOV DI,0
        BLOCK2:         ;loop for all NUM_ARR and print
            MOV AL,NUM_ARR[DI]
            PRINT_CHAR AL
            INC DI
            LOOP BLOCK2
            PRINTLN
            PRINTLN
            JMP START

        FINISH:
            EXIT
    MAIN ENDP
CODE ENDS
END MAIN
```

## Μακροεντολές

```
PRINT_CHAR MACRO CHAR
    PUSH AX
    PUSH DX
    MOV DL,CHAR
    MOV AH,2
    INT 21H
    POP DX
    POP AX
ENDM

PRINT_STR MACRO STR
    PUSH AX
    PUSH DX
    MOV DX,OFFSET STR
    MOV AH,9
    INT 21H
    POP DX
    POP AX
ENDM

;PRINT NEW LINE
PRINTLN MACRO
    PUSH AX
    PUSH DX
    MOV DL,13
    MOV AH,2
    INT 21H
    MOV DL,10
    MOV AH,2
    INT 21H
    POP DX
    POP AX
ENDM

PRINT_TAB MACRO
    PUSH AX
    PUSH DX
    MOV DL,9
    MOV AH,2
    INT 21H
    POP DX
    POP AX
ENDM

INPUT_CHAR MACRO
    MOV AH,8
```

```
47      INT 21H
48  ENDM
49
50  INPUT_PRINT_CHAR MACRO
51      MOV AH,1
52      INT 21H
53  ENDM
54
55  EXIT MACRO
56      MOV AX,4C00H
57      INT 21H
58  ENDM
```