ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ

5η Εργαστηριακή Άσκηση

Ακαδημαϊκό έτος 2020-2021

Χριστίνα Προεστάκη | ΑΜ : 03118877

Νικόλαος Μπέλλος | ΑΜ : 03118183

1η Άσκηση

```
;======EXERCISE 1========

INCLUDE MACROS.ASM
DATA_SEG    SEGMENT
    TABLE DB 128 DUP(?)
    TWO DB DUP(2)
DATA_SEG ENDS

CODE_SEG SEGMENT
    ASSUME CS:CODE, DS:DATA

    MAIN PROC FAR
        MOV AX,DATA_SEG
        MOV DS,AX

        MOV DI,0
        MOV CX,128          ;array TABLE of 128 unsigned items

    FILL_ARRAY:
        MOV TABLE[DI],CL
        INC DI
        LOOP FILL_ARRAY

        MOV DH,0
        MOV AX,0
        MOV BX,0
        MOV DI,0
        MOV CX,128          ;repeat 128 times

    ODD:
        PUSH AX
        MOV  AH,0
```

```
    MOV  AL,TABLE[DI]
    DIV  TWO            ;div 2
    CMP  AH,0           ;check if even
    POP  AX
    JE   EVEN
    MOV  DL,TABLE[DI]
    ADD  AX,DX
    INC  BX

EVEN:
    INC  DI
    LOOP ODD
    MOV  DX,0
    DIV  BX

    PRINT_DEC:          ;print in decimal form

    PUSH    DX
    PUSH    BX
    MOV     AX,BX
    MOV     BL,10
    MOV     CX,1

 LOOP_1:
    DIV     BL
    MOV     DX,AX
    SAR     AX,8
    PUSH    AX
    MOV     DH,0
    MOV     AX,DX
    CMP     AX,0
    JE      PRINTDEC
    INC     CX
    JNE     LOOP_1


 PRINTDEC:
    POP     AX
    ADD     AX,48
    PRINT   AL
    LOOP    PRINTDEC

    POP     BX
    POP     DX
```

```
        PRINTLN
        MOV  AL,TABLE[0]            ;min
        MOV  BL,TABLE[127]          ;max
        MOV  DI,0
        MOV  CX,128


    MAX:                            ;if number > max then update max
        CMP  AL,TABLE[DI]           ;or check if min
        JC   NEW_MAX
        JMP  MIN
    MIN:                            ;same for min
        CMP  TABLE[DI],BL
        JC   NEW_MIN
        JMP  CONTINUE
    NEW_MAX:
        MOV  AL,TABLE[DI]           ;update max
        JMP  CONTINUE

    NEW_MIN:                        ;update min
        MOV  BL,TABLE[DI]
    CONTINUE:
        INC  DI
        LOOP MAX

        CALL PRINT_HEX_
        PRINTCH ' '
        MOV AL,BL
        CALL PRINT_HEX

        EXIT
MAIN ENDP



PRINT_HEX_ PROC NEAR
                MOV DL,AL
                AND DL,0F0H
                MOV CL,4
                ROR DL,CL
                CMP DL,0
                JE SKIPZERO
```

```
                CALL PRINT_HEX
            SKIPZERO:
                MOV DL,AL
                AND DL,0FH
                CALL PRINT_HEX
                RET
        PRINT_HEX_ ENDP

        PRINT_HEX PROC NEAR
                CMP DL,9
                JG LETTER
                ADD DL,48
                JMP SHOW
            LETTER:
                ADD DL,55
            SHOW:
                PRINTCH DL
                RET
        PRINT_HEX ENDP
 CODE_SEG ENDS
 END MAIN
```

2η Άσκηση

Κώδικας ASSEMBLY για 8086

```
;========== EXERCISE 2 ==========

INCLUDE MACROS.ASM

DATA_SEG    SEGMENT
    NEWLINE DB 0AH,0DH,'$'
DATA_SEG    ENDS

CODE_SEG    SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA_SEG

;========== PART 1 ==========
MAIN PROC FAR
    MOV AX,DATA_SEG
    MOV DS,AX
START:
    MOV BX,0
    ; routine for 1st decimal number
    CALL DEC_KEYB       ; read 1st digit
    MOV BL,10           ; multiply 1st digit x 10
```

```
    MUL BL
    MOV BL,AL          ; store 1st number in BL
    CALL DEC_KEYB      ; read 2nd digit
    ADD AL,BL          ; add two numbers
    MOV BL,AL          ; store sum in BL

    ; routine for 2nd decimal number
    CALL DEC_KEYB      ; read 1st digit
    MOV CL,10          ; multiply 1st digit x 10
    MUL CL
    MOV CL,AL          ; store 1st number in CL
    CALL DEC_KEYB      ; read 2nd digit
    ADD AL,CL          ; add two numbers
    MOV CL,AL          ; store sum in CL

    PRINT 'Z'          ; printing routine
    PRINT '='
    MOV DL,BL
    CALL PRINT_DEC
    PRINT ' '
    PRINT 'W'
    PRINT '='
    MOV DL,CL
    CALL PRINT_DEC

    PRINT_STR NEWLINE

;========== PART 2 ==========
    ; calculate sum
    MOV AX,0
    MOV AL,BL
    ADD AX,CX
    PRINT 'Z'
    PRINT '+'
    PRINT 'W'
    PRINT '='
    MOV DX,AX
    CALL PRINT_HEX
    PRINT 'H'

    ; calculate diff
    PRINT ' '
    PRINT 'Z'
    PRINT '-'
    PRINT 'W'
    PRINT '='
```

```
    MOV AX,0
    CMP BL,CL
    JAE POSITIVE_DIF
    SUB CX,BX
    PRINT '-'
    MOV DL,CL
    JMP PRINT_DIF
POSITIVE_DIF:
    SUB BX,CX
    MOV DL,BL
PRINT_DIF:
    CALL PRINT_HEX
    PRINT 'H'

    JMP START
MAIN ENDP


;========== SUPLEMENTARY ROUTINES ==========

;========== READ DECIMAL NUMBER ==========
DEC_KEYB PROC NEAR
    PUSH DX
IGNORE:
    READ
    CMP AL,'Q'
    JE  ADDR2
    CMP AL,30H
    JL  IGNORE
    CMP AL,39H
    JG  IGNORE
    PUSH AX
    POP AX
    SUB AL,30H
ADDR2:
    POP DX
    RET
DEC_KEYB ENDP

;========== PRINT DECIMAL NUMBER (DL) ==========
PRINT_DEC PROC NEAR

    PUSH AX ; save registers
    PUSH CX
    PUSH DX

    MOV CX,1    ; initialize digit counter
```

```
    MOV AL,DL
    MOV DL,10

LD:
    MOV AH,0    ; divide number by 10
    DIV DL
    PUSH AX     ; save
    CMP AL,0    ; if quot = 0, start printing
    JE PRNT_10
    INC CX      ; increase counter (aka digits number)
    JMP LD      ; repeat dividing quotients by 10
PRNT_10:
    POP AX      ; get digit
    MOV AL,AH
    MOV AH,0
    ADD AX,'0'  ; ASCII coded
    PRINT AL    ; print
    LOOP PRNT_10 ; repeat till no more digits

    POP DX
    POP CX ; restore registers
    POP AX
    RET

PRINT_DEC ENDP

;========== PRINT DECIMAL NUMBER (DX) ==========
PRINT_DEC_2 PROC NEAR

 PUSH AX ; save registers
 PUSH BX
 PUSH CX
 PUSH DX

 MOV CX,1 ; initialize digit counter

 MOV AX,DX
 MOV BX,10

LD_2:
    MOV DX,0
    DIV BX
    PUSH DX ; save
    CMP AX,0 ; if quot = 0, start printing
    JE PRNT_10_2
    INC CX ; increase counter (aka digits number)
```

```
    JMP LD_2 ; repeat dividing quotients by 10
PRNT_10_2:
    POP DX ; get digit
    MOV AL,DL
    MOV AH,0
    ADD AX,'0' ; ASCII coded
    PRINT AL ; print
    LOOP PRNT_10_2 ; repeat till no more digits

    POP DX
    POP CX ; restore registers
    POP BX
    POP AX
    RET

PRINT_DEC_2 ENDP


;========== PRINT HEXIMAL NUMBER (DL) ==========

PRINT_HEX PROC NEAR

    PUSH AX
    MOV AL,DL
    SAR AL,4
    AND AL,0FH ; isolate 4 MSB
    ADD AL,30H ; ASCII code it
    CMP AL,39H
    JLE NEX
    ADD AL,07H ; if it's a letter, fix ASCII
NEX:
    CMP AL,'0'
    JE DONT_PRINT_IT
    PRINT AL ; print the first hex digit
DONT_PRINT_IT:
    MOV AL,DL
    AND AL,0FH ; isolate 4 LSB
    ADD AL,30H ; ASCII code it
    CMP AL,39H
    JLE OK
    ADD AL,07H ; if it's a letter, fix ASCII
OK:
    PRINT AL ; print the second hex digit

    POP AX
    RET
```

```
PRINT_HEX ENDP


CODE_SEG    ENDS
END MAIN
```

3η Άσκηση

```
;=====EXERCISE 3======

INCLUDE     MACROS.ASM

DATA_SEG    SEGMENT
DATA_SEG    ENDS

CODE_SEG    SEGMENT
    ASSUME  CS:CODE_SEG, DS_DATA_SEG

MAIN PROC FAR
    START:
        CALL    HEX_KEYB        ;third digit
        CMP     AL,'T'          ;if T is given then STOP
        JE      STOP
        MOV     BH,AL

        CALL    HEX_KEYB        ;second digit
        CMP     AL,'T'
        JE      STOP
        ROL     AL,4
        MOV     BL,AL


        CALL    HEX_KEYB        ;first digit
        CMP     AL,'T'
        JE      STOP
        OR      BL,AL

        PRINTCH '='
        CALL    PRINT_DEC       ;decimal
        PRINTCH '='             ;oct
        CALL    PRINT_OCT       ;binary
        PRINTCH '='
```

```
        CALL    PRINT_BIN

        PRINTLN
        JMP     START

     STOP:
        EXIT
MAIN ENDP



HEX_KEYB PROC NEAR

    PUSH DX
IGNORE:
    READ
    CMP AL,'T'
    JE QUIT
    CMP AL,30H
    JL IGNORE
    CMP AL,39H
    JG LETTER
    PUSH AX
    PRINT AL
    POP AX
    SUB AL,48
    JMP QUIT

LETTER:
    CMP AL,'A'
    JL IGNORE
    CMP AL,'F'
    JG IGNORE
    PUSH AX
    PRINT AL
    POP AX
    SUB AL,37H

QUIT:
    POP DX
    RET

HEX_KEYB ENDP


PRINT_DEC PROC NEAR
```

```
        PUSH    DX              ;save registers
        PUSH    BX
        MOV     AX,BX
        MOV     BL,10
        MOV     CX,1            ;initialize digit counter

    LOOP_1:
        DIV     BL              ;divide number by 10
        MOV     DX,AX
        SAR     AX,8            ;shift in order to fit next number
        PUSH    AX              ;save
        MOV     DH,0
        MOV     AX,DX
        CMP     AX,0
        JE      PRINTDEC
        INC     CX              ;increase counter(aka digits number)
        JNE     LOOP_1          ;repeat dividing


    PRINTDEC:
        POP     AX              ;get digit
        ADD     AX,48           ;ASCII coded
        PRINTCH   AL            ;print
        LOOP    PRINTDEC        ;repeat till no more digits

        POP     BX
        POP     DX
        RET

PRINT_DEC ENDP



PRINT_OCT PROC NEAR

        PUSH    DX
        PUSH    BX
        MOV     AX,BX
        MOV     BL,8
        MOV     CX,1

    LOOP_2:
        DIV     BL
        MOV     DX,AX
```

```
        SAR     AX,8
        PUSH    AX
        MOV     DH,0
        MOV     AX,DX
        CMP     AX,0
        JE      PRINTOCT
        INC     CX
        JNE     LOOP_2


    PRINTOCT:
        POP     AX
        ADD     AX,48
        PRINTCH   AL
        LOOP    PRINTOCT

        POP     BX
        POP     DX
        RET

PRINT_OCT ENDP


PRINT_BIN PROC NEAR

        PUSH    BX
        PUSH    DX
        MOV     AX,BX
        MOV     BL,2
        MOV     CX,1

    LOOP_3:
        DIV     BL
        MOV     DX,AX
        SAR     AX,8
        PUSH    AX
        MOV     DH,0
        MOV     AX,DX
        CMP     AX,0
        JE      PRINTOCT
        INC     CX
        JNE     LOOP_3

    PRINTBIN:
        POP     AX
        ADD     AX,48
```

```
        PRINTCH   AL
        LOOP    PRINTBIN

        POP     BX
        POP     DX
        RET

PRINT_BIN ENDP




CODE_SEG ENDS
END MAIN
```

4η Άσκηση

Κώδικας ASSEMBLY για 8086

```
;========== EXERCISE 4 ==========

INCLUDE MACROS.ASM

DATA_SEG    SEGMENT
    ARRAY DB 20 DUP(?)
    NEWLINE DB 0AH,0DH,'$'
    TERMINATE_MSG DB 0AH,0DH,'Exiting...$'
DATA_SEG    ENDS

CODE_SEG    SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA_SEG

;========== PART 1 ==========
MAIN PROC FAR
    MOV AX,DATA_SEG
    MOV DS,AX
START:
    MOV CX,20       ; set counter to 20
    MOV DI,0
INPUT_LOOP:
    READ
    CMP AL,3DH          ; check if input is '='
```

```
    JE EXIT
    CMP AL,30H          ; compare with 0
    JL  INPUT_LOOP      ; if smaller repeat
    CMP AL,39H          ; compare with 9
    JLE CONTINUE_LOOP   ; if smaller then number is valid
    CMP AL,'a'          ; compare with 'a'
    JL  INPUT_LOOP      ; if smaller repeat
    CMP AL,'z'          ; compare with 'z'
    JG  INPUT_LOOP      ; if greater repeat (else valid)
CONTINUE_LOOP:
    MOV [ARRAY + DI],AL
    INC DI
    LOOP INPUT_LOOP

    MOV CX,20
    MOV DI,0
PRINT_LINE1:
    MOV AL,[ARRAY + DI]
    PRINT AL
    INC DI
    LOOP PRINT_LINE1       ; loops 20 times (CX)
    PRINT_STR NEWLINE

;========== PART 2 ==========
    MOV CX,20
    MOV DI,0
PRINT_LINE2_LET:
    MOV AL,[ARRAY + DI]
    CMP AL,39H
    JLE  LINE2_LET_CONTINUE   ; if it is a number continue to next
    SUB AL,20H                ; convert letters to upper case
    PRINT AL
LINE2_LET_CONTINUE:
    INC DI
    LOOP PRINT_LINE2_LET
    PRINT '-'

    MOV CX,20
    MOV DI,0
PRINT_LINE2_NUM:
    MOV AL,[ARRAY + DI]
    CMP AL,39H
    JG  LINE2_NUM_CONTINUE
    PRINT AL
LINE2_NUM_CONTINUE:
    INC DI
    LOOP PRINT_LINE2_NUM
```

```
    PRINT_STR NEWLINE
    JMP START

EXIT:
    PRINT_STR TERMINATE_MSG
    EXIT
MAIN ENDP


CODE_SEG    ENDS
END MAIN
```

5η Άσκηση

Κώδικας ASSEMBLY για 8086

```
;========== EXERCISE 5 ==========

INCLUDE MACROS.ASM

DATA_SEG    SEGMENT
    ARRAY DB 20 DUP(?)
    BOOT_MSG      DB 0AH,0DH,'START (Y, N):$'
    NEWLINE       DB 0AH,0DH,'$'
    TERMINATE_MSG DB 0AH,0DH,'Exiting...$'
    ERROR_MSG     DB 0AH,0DH,'ERROR$'
    NUM_1     DB ?
    NUM_2     DB ?
    NUM_3     DB ?
DATA_SEG    ENDS

CODE_SEG    SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA_SEG

;========== MAIN ==========
MAIN PROC FAR
    MOV AX,DATA_SEG
    MOV DS,AX
START:
    PRINT_STR BOOT_MSG
BOOT_LOOP:
    READ
    CMP AL,'Y'
    JE  PROGRAM_START
```

```
    CMP AL,'N'
    JE  EXIT
    JMP BOOT_LOOP
PROGRAM_START:
    PRINT 'Y'
    PRINT_STR NEWLINE
    CALL HEX_KEYB
    MOV  NUM_1,AL
    CALL HEX_KEYB
    MOV  NUM_2,AL
    CALL HEX_KEYB
    MOV  NUM_3,AL

    CALL INPUT_SUM
    MOV  CX,AX

    ; calculate ADC output
    ; (Input / Volts = 4095 / 4) => (Input / Volts = 20475 / 20)
    ; (Volts(AX) = (INPUT * 20 / 20475))
    MOV BX,20
    MUL BX
    MOV BX,20475
    DIV BX
    CMP AX,3                ; check if temperature is over 999,9C
    JL  VALID_TEMP
    PRINT_STR ERROR_MSG
    JMP PROGRAM_START
VALID_TEMP:
    CMP CX,2047
    JLE FIRST_REG
    JMP SECOND_REG

FIRST_REG:
    ; (A/D / Temp = 2047 / 400) => (Temp = A/D(CX) * 400 / 2047)
    MOV AX,CX
    MOV BX,4000           ; multiply 400 by 10 to preserve decimal digit
    MUL BX
    MOV BX,2047
    DIV BX
    CALL PRINT_TEMP
    JMP PROGRAM_START

SECOND_REG:
    ; (A/D / Temp = (3071 - 2047) / (1200 - 400)) => (Temp = A/D(CX) * 800
/ 1024)
    MOV AX,CX
    MOV BX,8000           ; multiply 800 by 10 to preserve decimal digit
```

```
    MUL BX
    MOV BX,1024
    DIV BX
    CALL PRINT_TEMP
    JMP PROGRAM_START

EXIT:
    PRINT_STR TERMINATE_MSG
    EXIT
MAIN ENDP

;========== SUPPLEMENTARY ROUTINES ==========

;========== READ HEXADECIMAL NUMBER ==========
HEX_KEYB PROC NEAR
IGNORE:
    READ
    CMP AL,30H        ; if input < 30H ('0') then ignore it
    JL IGNORE
    CMP AL,39H        ; if input > 39H ('9') then it may be a hex letter
    JG CHECK_LETTER
    SUB AL,30H        ; otherwise make it a hex number
    JMP INPUT_OK

CHECK_LETTER:
    CMP AL,'N'        ; if input = N then exit program
    JE  EXIT
    CMP AL,'A'        ; if input < 'A' then ignore it
    JL IGNORE
    CMP AL,'F'        ; if input > 'F' then ignore it
    JG IGNORE
    SUB AL,37H        ; otherwise make it a hex number

INPUT_OK:
    RET
HEX_KEYB ENDP

;========== SUMS 3 HEX INPUT AND PUTS IT TO AX ==========
INPUT_SUM PROC NEAR
    PUSH BX
    MOV AH,NUM_1        ; AH = 0000XXXX

    MOV AL,NUM_2        ; AL = 0000YYYY
    SAL AL,4
    AND AL,0F0H         ; AL = YYYY0000

    MOV BL,NUM_3
```

```
    AND BL,0FH          ; BL = 0000ZZZZ
    OR  AL,BL           ; AL = YYYYZZZZ
                        ; AX = 0000XXXX YYYYZZZZ (FULL NUMBER)
    POP BX
    RET
INPUT_SUM ENDP

;========== PRINTS TEMPERATURE ON SCREEN (stored in AX) ==========
PRINT_TEMP PROC NEAR
    MOV CX,0        ; initialize counter
SPLIT:
    MOV DX,0
    MOV BX,10
    DIV BX          ; take the last decimal digit
    PUSH DX         ; save it
    INC CX
    CMP AX,0
    JNE SPLIT       ; continue, till we split the whole number

    DEC CX
    CMP CX,0
    JNE PRINT_
    PRINT '0'
    JMP ONLY_DECIMAL

PRINT_:
    POP DX          ; print the digits we saved in reverse
    CALL PRINT_DEC
    LOOP PRINT_

ONLY_DECIMAL:
    PRINT '.'       ; the last digit is the decimal
    POP DX
    CALL PRINT_DEC

    PRINT ' '
    PRINT 0F8H
    PRINT 'C'
    PRINT_STR NEWLINE

    RET
PRINT_TEMP ENDP

;========== PRINT DECIMAL NUMBER (DL) ==========
PRINT_DEC PROC NEAR

    PUSH AX ; save registers
```

```
    PUSH CX
    PUSH DX

    MOV CX,1    ; initialize digit counter

    MOV AL,DL
    MOV DL,10

LD:
    MOV AH,0    ; divide number by 10
    DIV DL
    PUSH AX     ; save
    CMP AL,0    ; if quot = 0, start printing
    JE PRNT_10
    INC CX      ; increase counter (aka digits number)
    JMP LD      ; repeat dividing quotients by 10
PRNT_10:
    POP AX      ; get digit
    MOV AL,AH
    MOV AH,0
    ADD AX,'0'  ; ASCII coded
    PRINT AL    ; print
    LOOP PRNT_10 ; repeat till no more digits

    POP DX
    POP CX ; restore registers
    POP AX
    RET

PRINT_DEC ENDP


CODE_SEG    ENDS
END MAIN
```