Συστήματα Μικροϋπολογιστών – 5η Σειρά Ασκήσεων

Κυριακόπουλος Γιώργος – el18153
Τζελέπης Σεραφείμ – el18849

Σημαντική σημείωση:

Η αναφορά για την 5η εργαστηριακή άσκηση υποβλήθηκε κανονικά από τον Τζελέπη Σεραφείμ – el18849 στο mycourses. Εγώ, Κυριακόπουλος Γιώργος – el18153, αντιμετώπιζα ένα θέμα με την υποβολή της, καθώς εμφανιζόταν στο mycourses πως είχα υποβάλει την εργασία στις 25/6 (ημέρα του μαθήματος), ενώ δεν είχα κάνει κάτι τέτοιο, συνειδητά τουλάχιστον. Απευθύνθηκα με email στον κ.Πεκμεστζή και στον κ.Μάραντο, ωστόσο δεν λύθηκε το πρόβλημα μου, το οποίο θεωρώ πως ίσως οφείλεται σε κάποια υποβολή θέματος της εξέτασης μου σε λάθος σημείο, ίσως στο τελευταίο θέμα κατά την διάρκεια της παράτασης. Έχω υποβάλει την άσκηση αυτή μέσω email, ωστόσο θα ήθελα εφόσον είναι δυνατόν να έρθετε σε επικοινωνία μαζί μου, ώστε να γνωρίζω εάν υπάρχει κάποιο πρόβλημα με την υποβολή των θεμάτων μου, καθώς, όπως θυμάμαι, τα υπέβαλα όλα και εμπρόθεσμα. Ευχαριστώ πολύ.

macros.asm:

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

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

```
READ MACRO
    MOV AH,8
    INT 21H
ENDM

EXIT MACRO
    PUSH AX
    MOV AX,4C00H
    INT 21H
    POP AX
ENDM

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

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

1η Άσκηση:

```
INCLUDE macros.asm

DATA SEGMENT
    TABLE DB 128 DUP(?)
    TWO DB DUP(2)
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA

MAIN  PROC FAR
        MOV AX,DATA
        MOV DS,AX
        MOV DI,0
        MOV CX,128
    STORE:
        MOV TABLE[DI],CL
        INC DI
        LOOP STORE
        MOV DI,1
        MOV CX,128
        MOV AX,0

    SUM:
        MOV AL,[TABLE + DI]
        ADD DX,AX
        ADD DI,2
        CMP DI,129
        JL SUM
        MOV AX,DX
        MOV BH,0
        MOV BL,64
        DIV BL

        MOV AH,0
        CALL PRINT_DEC
        PRINTLN
        MOV AL,TABLE[0]
        MOV BL,TABLE[127]
        MOV DI,0
        MOV CX,128
```

```
    ISMAX:
        CMP AL,TABLE[DI]
        JC NEWMAX
        JMP ISMIN
    NEWMAX:
        MOV AL,TABLE[DI]
        JMP NEXT
    ISMIN:
        CMP TABLE[DI],BL
        JC NEWMIN
        JMP NEXT
    NEWMIN:
        MOV BL,TABLE[DI]
    NEXT:
        INC DI
        LOOP ISMAX
        CALL PRINT_HEX8
        PRINTLN
        MOV AL,BL
        CALL PRINT_HEX8

        EXIT
MAIN ENDP


PRINT_HEX8 PROC NEAR
    MOV DL,AL
    AND DL,0F0H
    MOV CL,4
    RCR DL,CL
    CALL PRINT_HEX;
    MOV DL,AL
    AND DL,0FH
    CALL PRINT_HEX
    RET
PRINT_HEX8 ENDP

PRINT_HEX PROC NEAR
    CMP DL,9
    JG ADDR1
    ADD DL,30H
    JMP ADDR2
ADDR1:
```

```
    ADD DL,37H
ADDR2:
    PRINT DL
    RET
PRINT_HEX ENDP

PRINT_DEC proc NEAR
    MOV BL,10
    MOV CX,1                ;decades counter
LOOP_10:
    DIV BL              ;divide number with 10
    PUSH AX                     ;save units
    CMP AL,0                ;if quotient zero I have splitted
    JE PRINT_DIGITS_10              ;the whole number into dec digits

    INC CX              ;increase number of decades
    MOV AH,0
    JMP LOOP_10             ;if quotient is not zero I have to divide
 again
PRINT_DIGITS_10:
    POP DX              ;pop dec digit to be printed
    MOV DL,DH
    MOV DH,0                ;DX = 00000000xxxxxxxx (ASCII of number t
o be printed)
    ADD DX,30H              ;make ASCII code
    MOV AH,2
    INT 21H             ;print
    LOOP PRINT_DIGITS_10
    RET
ENDP PRINT_DEC

CODE ENDS
END MAIN
```

2η Άσκηση:

```
INCLUDE macros.asm

DATA SEGMENT
    MSG1 DB "Z=$"
    MSG2 DB "W=$"
    MSG3 DB "Z+W=$"
    MSG4 DB "Z-W=$"
    MSG5 DB "Z-W=-$"
    Z DB 0
    W DB 0
    SPACE DB " $"
    TEN DB DUP(10)
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA

    MAIN PROC FAR
        MOV AX,DATA
        MOV DS,AX

    START:
        PRINT_STR MSG1 ;print Z=
        CALL READ_DEC_DIGIT    ; read the first digit(tens) of Z
        MUL TEN                ; multiply it so its on ten base
        LEA DI,Z
        MOV [DI],AL            ; store AL(decadal digit of Z) for  in
 [DI] = Z
        CALL READ_DEC_DIGIT    ; read the second digit(ones) of Z
        ADD [DI],AL            ; store it in [DI] = Z
        PRINT ' '
        PRINT_STR MSG2         ; same procedure with Z for W
        CALL READ_DEC_DIGIT
        MUL TEN
        LEA DI,W
        MOV [DI],AL
        CALL READ_DEC_DIGIT
        ADD [DI],AL
        PRINTLN                ; print new line

        MOV AL,[DI]            ; AL = W
```

```
        LEA DI,Z                ; DI points to Z
        ADD AL,[DI]             ; ADD Z + W
        PRINT_STR MSG3          ; print 'Z+W='
        CALL PRINT_HEX8         ; print the sum in hex
        PRINT ' '

        MOV AL,[DI]             ; AL = Z
        LEA DI,W
        MOV BL,[DI]             ; BL = W
        CMP AL,BL               ; compare Z with W
        JB MINUS                ; jump if Z < W
        SUB AL,BL               ; else Z - W
        PRINT_STR MSG4          ; print "Z-
W=" because the sub is positive
        JMP SHOWSUB             ; jump to address for printing the sub
    MINUS:
        SUB BL,AL               ; W - Z because Z<W
        MOV AL,BL               ; keep the positive value of the sum i
n AL
        PRINT_STR MSG5          ; print "Z-W=-
" because Z<W and the sub and AL has the absolute value of it
SHOWSUB:
        CALL PRINT_HEX8         ; print sub in hex
        PRINTLN
        PRINTLN
        JMP START               ; jump to start for continuous running
MAIN ENDP

INPUT_TO_HEX PROC NEAR
     PUSHF
     SUB DH,30H
     SUB DL,30H
     MOV BL,10
     MOV AL,DH
     MUL BL
     ADD AL,DL
     MOV DL,AL
     POPF
     RET
ENDP INPUT_TO_HEX
READ_DEC_DIGIT PROC NEAR
    READ1:
        READ
```

```
        CMP AL,48
        JB READ1
        CMP AL,57
        JA READ1
        PRINT AL
        SUB AL,48
        RET
READ_DEC_DIGIT ENDP


PRINT_HEX8 PROC NEAR
    MOV DL,AL
    AND DL,0F0H
    MOV CL,4
    RCR DL,CL
    CALL PRINT_HEX;
    MOV DL,AL
    AND DL,0FH
    CALL PRINT_HEX
    RET
PRINT_HEX8 ENDP

PRINT_HEX PROC NEAR
    CMP DL,9
    JG ADDR1
    ADD DL,30H
    JMP ADDR2
ADDR1:
    ADD DL,37H
ADDR2:
    PRINT DL
    RET
PRINT_HEX ENDP

CODE ENDS
END MAIN
```

3η Άσκηση:

```
INCLUDE macros.asm

CODE SEGMENT
    ASSUME CS:CODE

    MAIN PROC FAR
        START:
            CALL HEX_KEYB       ; read first hex
            CMP AL,'T'
            MOV BH, AL          ; store it in BH
            CALL HEX_KEYB       ; read second hex number
            CMP AL,'T'          ; check for T to terminate
            JE FINISH
            MOV BL,AL           ; store second hex in BL
            ROL BL,4            ; rotate it to 4 msb of BL
            CALL HEX_KEYB       ; read third hex number
            CMP AL,'T'          ; check for T to terminate
            JE FINISH
            OR BL,AL            ; store AL in 4 lsb of BL


            PRINT '='
            CALL PRINT__DEC
            PRINT '='
            CALL PRINT_OCT
            PRINT '='
            CALL PRINT_BIN

            PRINTLN
            JMP START

        FINISH:
            EXIT
    MAIN ENDP


    HEX_KEYB PROC NEAR
        READ1:
            READ
            CMP AL,'T'          ; check for T
            JE RETURN
```

```
            CMP AL,48           ; check if <0
            JL READ1
            CMP AL,57           ; check if >9
            JG LETTER
            PRINT AL
            SUB AL,48           ; turn it to binary number
            JMP RETURN
        LETTER:
            CMP AL,'A'          ; check if <A
            JL READ1
            CMP AL,'F'          ; check if >F
            JG READ1
            PRINT AL
            SUB AL,55           ; turn hex ASCII to number
        RETURN:
            RET
    HEX_KEYB ENDP

    PRINT__DEC PROC NEAR
            PUSH BX

            MOV AX,BX           ; store BX in AX
            MOV BX,10           ; BX = 10 for divisions
            MOV CX,0            ; digits counter

        GETDEC:
            MOV DX,0            ; clear previous reminder
            DIV BX              ; divide by 10
            PUSH DX             ; push reminder to stack
            INC CX              ; increment digit counter
            CMP AX,0            ; check if we finished with div being
0
            JNE GETDEC
        PRINTDEC:
            POP DX              ; pop from stack
            ADD DX,48           ; add 48 for the ASCII code
            PRINT DL            ; print ASCII
            LOOP PRINTDEC       ; print CX times

            POP BX
            RET
    PRINT__DEC ENDP
```

```
    PRINT_OCT PROC NEAR         ; same with dec but we divide with 8
now
            PUSH BX

            MOV AX,BX
            MOV BX,8
            MOV CX,0

        GETOCT:
            MOV DX,0
            DIV BX
            PUSH DX
            INC CX
            CMP AX,0
            JNE GETOCT
        PRINTOCT:
            POP DX
            ADD DX,48
            PRINT DL
            LOOP PRINTOCT

            POP BX
            RET
    PRINT_OCT ENDP

    PRINT_BIN PROC NEAR         ; same with dec but we divide with 2
now
            PUSH BX

            MOV AX,BX
            MOV BX,2
            MOV CX,0

        GETBIN:
            MOV DX,0
            DIV BX
            PUSH DX
            INC CX
            CMP AX,0
            JNE GETBIN
        PRINTBIN:
            POP DX
            ADD DX,48
```

```
            PRINT DL
            LOOP PRINTBIN

            POP BX
            RET

            RET
    PRINT_BIN ENDP
CODE ENDS
END MAIN
```

4η Άσκηση:

```
INCLUDE macros.asm

DATA SEGMENT
CHARS DB 20 DUP(?)
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA

MAIN PROC FAR
        MOV AX,DATA
        MOV DS,AX
        MOV CL,0            ; digits counter
    START:
        MOV DI,0            ; array pointer
    NEXTCHAR:
        READ
        CMP AL,61           ; check for "="
        JE FINISH           ; if input is "=" then terminate the prog
ramm
        CMP AL,13           ; if input is ENTER
        JE OUTPUT           ; stop reading and go to the output
        CMP AL,48           ; check if input<0
        JB NEXTCHAR
        CMP AL,122          ; check if input>z
        JA NEXTCHAR
        CMP AL,57           ; check if input<=9
        JBE SAVECHAR        ; then in addition with the previous chec
ks our character is acceptable
        CMP AL,97           ; check if input<a
        JB NEXTCHAR
    SAVECHAR:
        PRINT AL            ; print the accepted input
        MOV CHARS[DI],AL    ; save it on the array chars
        INC DI
        INC CL
        CMP CL,20
        JB NEXTCHAR
    OUTPUT:
        PRINTLN             ; print new line
        CMP CL,0
        JE NEXTCHAR         ; check for empty area
```

```
        MOV CX,20           ; array size
        MOV DI,0            ; index to start of the array
    LETTERS:
        MOV AL,CHARS[DI]
        CMP AL,97           ; check if it is lower case letter
        JB SKIP
        CMP AL,122          ; check if it is lower case letter
        JA SKIP
        SUB AL,32           ; lower case to upper case
        PRINT AL            ; print the upper case letter
    SKIP:
        INC DI              ; procceed to the next element of the arr
ay
        LOOP LETTERS        ; loop through the whole array printing o
nly letters
        PRINT "-"           ; print "-"
        MOV CX,20           ; counter of array size
        MOV DI,0            ; index to start of the array
    NUMBERS:
        MOV AL,CHARS[DI]
        CMP AL,48           ; check if it is a number
        JB SKIP2
        CMP AL,57           ; check if it is a number
        JA SKIP2
        PRINT AL            ; if its is number print it
    SKIP2:
        INC DI              ; go the next element
        LOOP NUMBERS        ; loop through the whole array printing o
nly numbers
        PRINTLN             ; print new line
        JMP START           ; jump to start
    FINISH:
        EXIT
MAIN ENDP
CODE ENDS
END MAIN
```

5η Άσκηση:

(input bits) = 4095/4000 mv => mv = 4000/4095 (input bits)

Για τις θερμοκρασίες:

1: mv = 2000t/400 => **t = 0.2mv**

2: mv = 1000t/800 + β

Για t=400:
2000 = 1000*400/800 + β => β = 1500

Άρα: mv = 1000t/800 + 1500 => **t = 0.8mv - 1200**

Για τις περιοχές:

1: 0 < mv < 2000 => **0 < (input bits) < 2047.5**

2: 2000 < mv < 3000 => **2047.5 < (input bits) < 3071.25**

Τελικά:

1: t = 4000/(5*4095) * (input bits) = **800/4095 * (input bits)**

2: t = (8*4000)/(10*4095) * (input bits) - 1200 = **(3200/4095) * (input bits) - 1200**

```
INCLUDE macros.asm

DATA SEGMENT
    STARTPROMPT DB "START(Y,N):$"; Starting message
    ERRORMSG DB "ERROR$"         ; Error message
ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA

    MAIN PROC FAR
            MOV AX,DATA
            MOV DS,AX

            PRINT_STR STARTPROMPT
        START:                   ; Read Y or N
            READ
            CMP AL,'N'           ; = N ?
```

```
        JE FINISH           ; End program
        CMP AL,'Y'          ; = Y ?
        JE CONT             ; Continue program
        JMP START
    CONT:
        PRINT AL            ; Print starting character
        PRINTLN
        PRINTLN
    NEWTEMP:
        MOV DX,0
        MOV CX,3            ; 3 HEX digits
    READTEMP:               ; Read input
        CALL HEX_KEYB       ; Read first bit
        CMP AL,'N'          ; Check if it is N
        JE FINISH
                            ; Get digits to DX
        PUSH CX
        DEC CL              ; For the shift
        ROL CL,2
        MOV AH,0
        ROL AX,CL           ; Shift left 8, 4, 0 digits
        OR DX,AX            ; Add digit to the number
        POP CX
        LOOP READTEMP

        PRINTTAB
        MOV AX,DX
        CMP AX,2047         ; V <= 2 ?
        JBE BRANCH1
        CMP AX,3071         ; V <= 3 ?
        JBE BRANCH2
        PRINT_STR ERRORMSG  ; V > 3
        PRINTLN
        JMP NEWTEMP

    BRANCH1:                ; 1st branch: V <= 2, T = (800 * V) d
iv 4095
        MOV BX,800
        MUL BX
        MOV BX,4095
        DIV BX
        JMP SHOWTEMP
```

```
        BRANCH2:                ; 2nd branch: 2 < V <= 3, T = ((3200
* V) div 4095) - 1200
            MOV BX,3200
            MUL BX
            MOV BX,4095
            DIV BX
            SUB AX,1200
        SHOWTEMP:
            CALL PRINT_DEC16    ; Print integer part stored on AX
                                ; Decimal part = (remainder * 10) div
 4095
            MOV AX,DX
            MOV BX,10
            MUL BX
            MOV BX,4095
            DIV BX

            PRINT '.'           ; Dot for decimal part
            ADD AL,48           ; ASCII code
            PRINT AL            ; Print decimal part
            PRINTLN
            JMP NEWTEMP

        FINISH:
            PRINT AL
            EXIT
    MAIN ENDP
                                ; Insert HEX bit into AL
    HEX_KEYB PROC NEAR          ; see: mP11_80x86_programs.pdf pg. 20
-21
        READ:
            READ
            CMP AL,'N'          ; = N ?
            JE RETURN
            CMP AL,48           ; < 0 ?
            JL READ
            CMP AL,57           ; > 9 ?
            JG LETTER
            PRINT AL
            SUB AL,48           ; ASCII code
            JMP RETURN
        LETTER:                 ; A ... F
            CMP AL,'A'          ; < A ?
```

```
            JL READ
            CMP AL,'F'          ; > F ?
            JG READ
            PRINT AL
            SUB AL,55           ; ASCII code
        RETURN:

            RET
    HEX_KEYB ENDP
                                ; Print 16-bit decimal number from AX
    PRINT_DEC16 PROC NEAR       ; see: mP11_80x86_programs.pdf pg. 26
-27
            PUSH DX

            MOV BX,10           ; Decimal -> divide by 10
            MOV CX,0            ; Digits counter
        GETDEC:                 ; Get digits
            MOV DX,0            ; Number mod 10 (remainder)
            DIV BX              ; Divide by 10
            PUSH DX             ; Store it temporarily
            INC CL
            CMP AX,0            ; Number div 10 = 0 ? (quotient)
            JNE GETDEC
        PRINTDEC:               ; Print digits
            POP DX
            ADD DL,48           ; ASCII code
            PRINT DL
            LOOP PRINTDEC

            POP DX
            RET
    PRINT_DEC16 ENDP
CODE ENDS
END MAIN
```