Συστήματα Μικροϋπολογιστών
3η σειρά ασκήσεων

Νικόλαος Παγώνας, el18175
Αναστάσιος Παπαζαφειρόπουλος, el18079

1η άσκηση

Η 1η άσκηση, μαζί με τα κατάλληλα σχόλια, βρίσκεται στο αρχείο `ex1.8085`. Για λόγους πληρότητας παρατίθεται και εδώ.

```
START:
    IN 10H              ; Remove memory protection
    MVI A,10H           ; 10H --> empty LCD
    STA 0A00H           ; LCD: _ _ _ _ _ x
    STA 0A01H           ; LCD: _ _ _ _ x _
    STA 0A04H           ; LCD: _ x _ _ _ _
    STA 0A05H           ; LCD: x _ _ _ _ _

    MVI A,0DH           ; 00001101 -->5.5 no, 6.5 yes, 7.5 no
    SIM                 ; Set interrupt mask
    EI                  ; Enable interrupt

WAIT:
    JMP WAIT            ; Wait for interrupt

INTR_ROUTINE:
    POP H               ; Pops PC from stack
    MVI A,00H           ; Inverse LED logic
    STA 3000H           ; Light up LEDs
    MVI E,3CH           ; 60 sec
    EI                  ; For time reset

COUNTDOWN:              ;
    CALL DECA           ; for LCD DEC printing
    DCR E               ; E is the counter
    JNZ COUNTDOWN       ; if E != 0, repeat

    MVI A,FFH           ; After 60sec,
    STA 3000H           ; turn off leds
```

```
    JMP WAIT

DECA:                   ; Prints E in DEC to LCD
    PUSH PSW
    PUSH B
    PUSH D
    PUSH H
    MVI C,00H           ; Counts tens
    MOV A,E             ; Counts ones
TENS:
    INR C               ; C++
    SUI 0AH             ; Subtract 10
    JNC TENS            ; If not negative, repeat
ONES:
    ADI 0AH             ; Correct negative
    DCR C               ; Correct tens' counter

    LXI H,0A02H         ; Memory place for ones-lcd
    MOV M,A             ; Store ones
    LXI H,0A03H         ; Memory place for tens-lcd
    MOV M,C             ; Store tens

    LXI D,0A00H         ; Starting position for lcd
    CALL STDM           ; Store Display Message

    LXI B,0064H         ; 100ms delay
    MVI A,0AH           ; A --> cnt, 10x100ms=1sec
ONESEC:
    CALL DCD            ; Print to lcd
    CALL DELB           ; 100ms delay
    DCR A               ; A--
    JNZ ONESEC          ; If A != 0, repeat

    POP H
    POP D
    POP B
    POP PSW
    RET

END:
    RST 1
    END
```

2η άσκηση

Η 2η άσκηση, μαζί με τα κατάλληλα σχόλια, βρίσκεται στο αρχείο `ex2.8085`. Για λόγους πληρότητας παρατίθεται και εδώ.

```
    IN 10H
    LXI H,0A00H   ;----------------
    MVI M,10H
    INX H
    MVI M,10H
    INX H
    MVI M,10H     ;LED screen output blank
    INX H
    MVI M,10H
    INX H
    MVI M,10H
    INX H
    MVI M,10H
    MVI A,0DH     ; --------------------
    SIM           ;Enable 6.5 Interrupts
    EI            ;----------------------

LOOP_A:           ;infinite loop
    JMP LOOP_A    ;wait for INTRPT

INTR_ROUTINE:

    MVI D,64H     ; D = 100
    MVI E,C8H     ; E = 200
    INR E         ; increase C,D
    INR D         ;
    CALL KIND     ; keyboard input
    LXI H,0A01H   ; load address for output of first digit given
    MOV M,A       ; first digit given = MSB
    RLC           ; rotate 4 times left to send it to MSBs and save temporarily at B
    RLC           ;
    RLC           ;
    RLC           ;
    MOV B,A
    LXI H,0A00H   ; load address for output of second digit
    CALL KIND     ; read again from keyboard
    MOV M,A       ; second digit given = LSB
    ADD B         ; add B to A
    CMP D         ;
    JC LED_1      ; if [0..K1] turn LED 0 on
    CMP E         ;
    JC LED_2      ; if (K1..K2] turn LED 1 on
    MVI A,04H     ; else if (K2..255] LED 2 on
    CMA           ; complement (negative logic LEDs)
    STA 3000H     ;
    JMP PRINT     ;
LED_1:
    MVI A,01H     ;-------------------------------------
    CMA           ; labels to turn on proper LED
    STA 3000H
```

```
    JMP PRINT
LED_2:
    MVI A,02H
    CMA
    STA 3000H
    JMP PRINT

PRINT:              ; -------------------------------
    LXI D,0A00H     ; load to D address to show
    CALL STDM
    CALL DCD
    EI              ; enable interrupts again
    JMP PRINT       ; print until interrupt happens
END
```

3η άσκηση

Η 3η άσκηση, μαζί με τα κατάλληλα σχόλια, βρίσκεται στο αρχείο `ex3.8085`. Για λόγους πληρότητας παρατίθεται και εδώ.

(a)

```
SWAP Nible MACRO Q
    PUSH PSW
    MOV A,Q          ; Q <-- A = 12345678, numbers mean bits, from MSB to LSB
    RLC              ; A = 23456781
    RLC              ; A = 34567812
    RLC              ; A = 45678123
    RLC              ; A = 56781234
    MOV Q,A          ; Q <-- A

    MOV A,M          ; same as above
    RLC
    RLC
    RLC
    RLC
    MOV M,A
    POP PSW
ENDM
```

(b)

```
FILL MACRO RP,X,K
    PUSH PSW
    PUSH H

    LXI H,0000H
    DAD RP           ; Add RP to H-L
```

```
    MVI A,X         ; A is the counter

COUNTDOWN:
    MVI M,K         ; Place K in the memory position pointed by H-L
    INX H           ; Point to next memory address
    DCR A           ; Decrement counter
    JNZ COUNTDOWN   ; If counter is not zero, continue

    POP H
    POP PSW
ENDM
```

(c)

```
RHLR MACRO n

    PUSH PSW
    PUSH D

    MVI A,n
    CPI 00H
    JZ DONE     ; If n is zero from the start, then we are done
    MVI E,n     ; E is the counter

LOOP1:
    MOV A,H
    RAR
    MOV H,A     ; H has shifted one to the right, and the CY is ready to go to L
    MOV A,L
    RAR
    MOV L,A     ; L has shifted one to the right, and CY has LSB of L
    DCR E       ; Decrement counter
    JNZ LOOP1   ; If not finished, continue loop
DONE:
    POP D
    POP PSW
ENDM
```

4η άσκηση

Η διακοπή συμβαίνει στο μέσο της εντολής CALL 0880H, οπότε θα ολοκληρωθεί η εκτέλεση της τρέχουσας εντολής. Η τρέχουσα τιμή του Program Counter (0800H) αποθηκεύεται στη στοίβα, ο δείκτης στοίβας ανεβαίνει δύο θέσεις πάνω και στον μετρητή προγράμματος καταχωρείται η διεύθυνση 0880H. Στη συνέχεια, σώζεται η τιμή του Program Counter και η κατάσταση του 8085, ενώ εκτελείται η ρουτίνα εξυπηρέτησης της διακοπής RST 7.5. Δηλαδή, η τιμή του PC αποθηκεύεται ξανά στη στοίβα, ο SP ανεβαίνει άλλες δύο θέσεις πάνω και στον PC καταχωρείται η διεύθυνση της διακοπής για να εκτελεστεί η σχετική ρουτίνα. Όταν ολοκληρωθεί

η εκτέλεση της ρουτίνας εξυπηρέτησης της διακοπής, η διεύθυνση που βρίσκεται στην κορυφή της στοίβας (0880H) επανέρχεται στον PC, ο SP κατεβαίνει δύο θέσεις κάτω και εκτελείται η ρουτίνα που αρχίζει από τη διεύθυνση 0880H, σύμφωνα με την εντολή CALL 0880H. Αφού ολοκληρωθεί η εκτέλεση και της τελευταίας ρουτίνας, η διεύθυνση στην κορυφή της στοίβας (0800H) επαναφέρεται στον PC, ο SP κατεβαίνει 2 θέσεις κάτω και συνεχίζεται η εκτέλεση του προγράμματος από τη διεύθυνση 0801H. Η όλη διαδικασία αποτυπώνεται καλύτερα παρακάτω, όπου εμφανίζονται σχηματικά τα περιεχόμενα PC και στοίβας.

- αρχικά:

PC	0800H
SP	00H
SP+1	30H

- μετά την εκτέλεση της εντολής CALL 0880H:

PC	0800H
SP	00H
SP+1	08H
SP+2	00H
SP+3	30H

- μετά την πραγματοποίηση της διακοπής RST 7.5:

PC	(RST 7.5)
SP	80H
SP+1	08H
SP+2	00H
SP+3	08H
SP+4	00H
SP+5	30H

- μετά την εκτέλεση της ρουτίνας εξυπηρέτησης της διακοπής:

PC	0800H
SP	00H
SP+1	08H
SP+2	00H
SP+3	30H

- μετά την εκτέλεση της ρουτίνας που καλεί την εντολή CALL 0880H:

PC	0800H
SP	00H
SP+1	30H

5η άσκηση

Η 5η άσκηση, μαζί με τα κατάλληλα σχόλια, βρίσκεται στα αρχεία `ex5a.8085` και `ex5b.8085`, για υλοποίηση με και χωρίς διακοπές αντίστοιχα. Για λόγους πληρότητας παρατίθεται και εδώ.

Υλοποίηση με διακοπές:

```
BEGIN:
    MVI A,0DH       ; Interrupt mask: 00001101 (Disable 7.5, Enable 6.5, Disable 5.5)
    SIM             ; Set Interrupt Mask
    LXI H,0000H     ; H-L is the accumulator of the whole sum
    MVI C,40H       ; Counter. 40 hex = 64 dec
    EI              ; Enable interrupt
WAITING:            ; Wait for all data
    MOV A,C
    CPI 00H         ; Check to see if finished
    JNZ WAITING     ; If not finished, continue waiting for data
    DI              ; Disable interrupt
    DAD H           ; Normally, we would shift 5 to the right (division by 32)
    DAD H           ; Instead, we move the 3 MSB of the result to H (integer part)
    DAD H           ; So we keep the digits after the comma in the 5 MSB of L
                    ; Like this:
                    ; H --> _ _ _ _ _ x x x (result of div)
                    ; L --> x x x x x _ _ _ (result of mod)
    HLT

RST6.5:
    PUSH PSW

    MOV A,C
    ANI 01H         ; 00000001 in order to extract LSB
    JPO READ4MSB    ; If LSB is 1, go read 4 MSB

    IN 20H          ; Else, read 4 LSB
    ANI 0FH         ; 00001111 --> x3x2x1x0 Mask
    MOV B,A         ; Store temporarily until 4 MSB come
    DCR C           ; Decrement counter
    JMP 4LSBDONE    ; Return to main program until we read 4 MSB

READ4MSB:
    IN 20H          ; Read 4 MSB
    ANI 0FH         ; 00001111 --> x3x2x1x0 Mask
    RLC             ; Shift 4 times to the left --> MSB in right place
    RLC
    RLC
    RLC
    ORA B           ; Combine MSB with LSB

    MVI D,00H       ; D-E pair is used to temporarily keep the whole result
    MOV E,A         ; D = 0 and E = A (A has 4 MSB and 4 LSB we have already read)
    DAD D           ; Add D-E to H-L (H-L is the accumulator)
    DCR C           ; Decrement counter

4LSBDONE:
```

```
    POP PSW
    EI
    RET
END
```

Υλοποίηση χωρίς διακοπές:

```
BEGIN:
    LXI H,0000H             ; H-L is the accumulator of the whole sum
    MVI C,40H               ; Counter. 40 hex = 64 dec
WAITING:                    ; Wait for all data
    MOV A,C
    CPI 00H                 ; Check to see if finished
    JZ FINISHED

    IN 20H
    ANI 80H                 ; 10000000 --> x_7 Mask
    CPI 80H
    JZ DATA_READY           ; If x_7 is 1, start reading
    JMP WAITING             ; Else continue waiting for data
    DAD H                   ; Normally, we would shift 5 to the right (division by 32)
    DAD H                   ; Instead, we move the 3 MSB of the result to H (integer part)
    DAD H                   ; So we keep the digits after the comma in the 5 MSB of L
                            ; Like this:
                            ; H --> _ _ _ _ _ x x x (result of div)
                            ; L --> x x x x x _ _ _ (result of mod)
    HLT

DATA_READY:
    PUSH PSW

    MOV A,C
    ANI 01H                 ; 00000001 in order to extract LSB
    JPO READ4MSB            ; If LSB is 1, go read 4 MSB

    IN 20H                  ; Else, read 4 LSB
    ANI 0FH                 ; 00001111 --> x3x2x1x0 Mask
    MOV B,A                 ; Store temporarily until 4 MSB come
    DCR C                   ; Decrement counter
    JMP 4LSBDONE            ; Return to main program until we read 4 MSB

READ4MSB:
    IN 20H                  ; Read 4 MSB
    ANI 0FH                 ; 00001111 --> x3x2x1x0 Mask
    RLC                     ; Shift 4 times to the left --> MSB in right place
    RLC
    RLC
    RLC
    ORA B                   ; Combine MSB with LSB
```

```
    MVI D,00H               ; D-E pair is used to temporarily keep the whole result
    MOV E,A                 ; D = 0 and E = A (A has 4 MSB and 4 LSB we have already read)
    DAD D                   ; Add D-E to H-L (H-L is the accumulator)
    DCR C                   ; Decrement counter

4LSBDONE:
    POP PSW
WAIT_FOR_X7_ZERO:
    IN 20H
    ANI 80H                 ; 10000000 --> x7 Mask
    CPI 00H
    JNZ WAIT_FOR_X7_ZERO    ; If x7 hasn't returned to 0, keep waiting
    JMP WAITING             ; Else wait for new data
END
```