Εργαστήριο Μικροϋπολογιστών

1η Εργαστηριακή Αναφορά

Αμπατζή Ναυσικά (031 17 198)
Δήμος Δημήτρης (031 17 165)

7ο Εξάμηνο - Ροή Υ
Φθινόπωρο 2020

1 Δυαδικό Χρονόμετρο Δευτερολέπτων

Το ζητούμενο πρόγραμμα λειτουργεί ως μετρητής δευτερολέπτων από το 0 μέχρι ένα άνω όριο - που δίνεται από τα 4 LSB των dip switches - και αντίστροφα. Η αλλαγή του άνω ορίου γίνεται εφικτή μόνο όταν το χρονόμετρο είναι 0.

```
; main program
        LXI B,0500H     ; set delay to 1 sec
LOOP_:  MVI A,FFH
        STA 3000H       ; output logical 0
        LDA 2000H       ; read timer limit
        ANI 0FH
        CPI 00H
        JZ LOOP_        ; loop, while timer limit = 0
        MOV D,A         ; D <- timer limit
        ANI 00H         ; initialize counter
        CALL UP
        CALL DOWN
        JMP LOOP_
        HLT
```

```
; subroutine: waits till MSB = ON
SWITCH:  PUSH PSW
WAIT:    LDA 2000H
         RAL
         JC OK
         JMP WAIT
OK:      POP PSW
         RET

; subroutine: up counter
UP:      CALL SWITCH
         CALL SHOW
         CALL DELB        ; wait 1 sec
         INR A            ; count++
         CMP D            ; count = timer limit?
         JNC UP_END
         JMP UP
UP_END:  RET

; subroutine: down counter
DOWN:    CALL SWITCH
         CALL SHOW
         CALL DELB        ; wait 1 sec
         DCR A            ; count++
         CPI 00H          ; count = timer limit?
         JZ DN_END
         JMP DOWN
DN_END:  RET

; subroutine: outputs counter
SHOW:    CMA
         STA 3000H        ; show count
         CMA
         RET

END
```

2 Δεκαδική Απεικόνιση Δεκαεξαδικής Εισόδου

Το ζητούμενο πρόγραμμα δέχεται ως είσοδο από το πληκτρολόγιο δύο δεκαεξαδικά ψηφία (1ο και 2ο ψηφίο HEX αναπαράστασης) και στη συνέχεια απεικονίζει στα 7 segment display LEDs την δεκαδική αναπαράσταση της εισόδου.

```
; main program
MAIN:   IN 10H              ; remove memory protection
        LXI H,0A05H
        MVI B,03H           ; iterate over B
LP:     MVI M,10H           ; store "space" (= '10H')
        DCX H
        DCR B
        JNZ LP              ; repeat 3 times
        LXI H,0A00H
        CALL KIND           ; A <- x
        CALL MUL16          ; A <- 16x
        PUSH PSW            ; x -> stack
        CALL KIND           ; A <- y
        POP B               ; B <- 16x
        ADD B               ; A = 16x + y
        CALL DIV10          ; decical byte 0 -> B
        MOV M,B
        INX H
        CALL DIV10          ; dec byte 1 -> B, dec byte 2 -> A
        MOV M,B
        INX H
        MOV M,A
        CALL SHOW
        JMP MAIN
        HLT
```

```
; subroutine: multiplies A with 16
MUL16:  MVI C,10H
        MOV B,A ; B <- x
        ANI 00H ; initialize A
MUL:    ADD B
        DCR C
        JNZ MUL
        RET

; subroutine: divide A by 10 (A <- div, B <- mod)
DIV10:  MVI B,00H       ; initialize div
DIV:    CPI 0AH ; A - 10 > 0
        JC ENDV ; if yes -> stop division
        SUI 0AH
        CALL SWAP
        ADI 01H
        CALL SWAP
        JMP DIV
ENDV:   CALL SWAP
        RET

; subroutine: [0A00 - 0A05] -> 7-seg display
SHOW:   LXI D,0A00H
        CALL STDM
        CALL DCD
        RET

;subroutine: swaps A <-> B
SWAP:   PUSH B
        MOV B,A
        POP PSW
        RET

END
```

3 Εξομοίωση Αυτοματισμού Βαγονέτου

Το ζητούμενο πρόγραμμα εξομοιώνει τη λειτουργία ενός βαγονέτου που εκτελεί παλινδρομική κίνηση ανάμεσα σε δύο άκρα. Ανάλογα με τις αλλαγές - τις οποίες καθορίζει η εκφώνηση - των MSB και LSB dip switches η κίνηση του βαγονέτου υφίσταται συγκεκριμένες μεταβολές.

```
; main program
MAIN:   LXI B,014FH      ; time delay ~0.5 sec
        LDA 2000H        ; check input
        CALL MSB
        CALL LSB
        MVI A,01H        ; initialize wagon
        JMP LEFT

LEFT1:  RLC

LEFT:   CALL SHOW
        CALL DELB
        RLC
        MOV E,A ; save A in E
        CALL MSB
        CALL LSB_LATER1

        MOV A,D
        CPI 01H
        MOV A,E ; restore A
        JZ RIGHT1
        CPI 80H
        JZ RIGHT
        JMP LEFT

RIGHT1: RRC

RIGHT:  CALL SHOW
        CALL DELB
        RRC
        MOV E,A ; save A in E
        CALL MSB
        CALL LSB_LATER1
```

```
        MOV A,D
        CPI 01H
        MOV A,E ; restore A
        JZ LEFT1
        CPI 01H
        JZ LEFT
        JMP RIGHT

; checks MSB
MSB:    LDA 2000H
        ANI 80H
        CPI 80H
        JNZ MSB         ; wait until MSB is ON
        RET

; checks LSB (only for the start)
LSB:    LDA 2000H
        ANI 01H
        CPI 01H
        JNZ LSB         ; wait until LSB is ON
        RET

; checks LSB (main subroutine)
LSB_LATER1:     MVI D,00H
                LDA 2000H
                ANI 01H
                CPI 01H
                JZ OK1
                JMP LSB_LATER2
OK1:            RET

LSB_LATER2:     LDA 2000H
                ANI 01H
                CPI 01H
                JNZ LSB_LATER2
                MVI D,01H
                RET
```

```
; shows output
SHOW:    CMA
         STA 3000H
         CMA
         RET

END
```