ΣΗΜΜΥ ΕΜΠ
Συστήματα Μικροϋπολογιστών / 3η ομάδα ασκήσεων
Νικόλαος Πηγαδάς / el18445
2020-2021

Άσκηση 1

```
MVI A,0DH            ; Μάσκα της διακοπής RST 6.5
SIM                  ; Αποθήκευση μάσκας διακοπής στον accumulator A
EI                   ; Ενεργοποίηση διακοπής τύπου RST 6.5

WAIT:
      JMP WAIT       ; Ατέρμονη επανάληψη μέχρι να έρθει διακοπή τύπου RST 6.5

INTR_ROUTINE:
      EI            ; Η ενεργοποίηση αυτή χρησιμοποιείται για πιθανές επόμενες διακοπές
      MVI A,10H     ; Σβήνω τα υπόλοιπα digits (θέσεων 1,2,5,6) της 7-segment display,
      STA 0B00H     ; μεταφέροντας την τιμή 10H στις αντίστοιχες θέσεις μνήμης
      STA 0B01H
      STA 0B04H
      STA 0B05H
      LXI B,03E8H   ; BC <- 1000 (δεκαδικό), ώστε να έχουμε καθυστέρηση 1000ms = 1s

INITIALISE_DEKADES:
      MVI A,06H     ; Αρχικοποιώ τις δεκάδες με 6, αλλά στην ετικέτα DECADES τις μειώνω
      STA 0B03H     ; για να έχω 50s. Μεταφορά στην θέση μνήμης του τέταρτου digit

INITIALISE_MONADES:
      MVI A,0AH     ; Αρχικοποιώ τις μονάδες με 10. Παρακάτω, στην ετικέτα MONADES
      STA 0B02H     ; γίνονται ίσες με 9, πριν τις εμφανίσουμε. Έχω 59s, δηλαδή, αρχικά.
                    ; Η θέση μνήμης των μονάδων αντιστοιχεί στο τρίτο digit
DEKADES:
      LDA 0B03H     ; Φέρνω από την μνήμη τις δεκάδες και τις βάζω στον A
      DCR A         ; Μειώνω τις δεκάδες όταν αρχικοποιήσω τις μονάδες
      STA 0B03H     ; Τις αποθηκεύω στην θέση των δεκάδων (τέταρτο digit)

MONADES:
      LDA 0B02H     ; Φέρνω από την μνήμη τις μονάδες και τις βάζω στον A
      DCR A         ; Μειώνω τις μονάδες
      STA 0B02H     ; Τις αποθηκεύω στην θέση των μονάδων (τρίτο digit)

      LXI D,0B00H   ; Φορτώνω στον D την θέση μνήμης του πρώτου digit, ώστε να τυπωθούν
      CALL STDM     ; το τρίτο και το τέταρτο στις μεσαίες θέσεις, όπου και θέλουμε
      CALL DCD      ; Εμφάνιση στην οθόνη
      CALL DELB     ; Καθυστέρηση 1s για κάθε μονάδα

LEDS_ON:            ; Άναμμα των LEDs
      MVI A,00H
      STA 3000H

DIGIT_DISPLAY:
      LDA 0B02H               ; Μονάδες
      CPI 00H                 ; Σύγκριση με το 0
      JNZ MONADES             ; Αν δεν είναι 0, συνεχίζει το countdown των μονάδων
      LDA 0B03H               ; Δεκάδες
      CPI 00H                 ; Σύγκριση με το 0
      JNZ INITIALISE_MONADES  ; Αν δεν είναι 0, αρχικοποιώ τον μετρητή μονάδων
                              ; και συνεχίζω το countdown των δεκάδων
LEDS_OFF:                     ; Αν φτάσεις εδώ, σβήσε τα LEDs
      MVI A,FFH
      STA 3000H

      JMP WAIT                ; και συνέχισε να περιμένεις μέχρι να έρθει νέα διακοπή
END
```

Άσκηση 2

```
MVI D,55H          ; Κατώφλι 1, K1 = 255/3 = 85 = 55H
MVI E,AAH          ; Κατώφλι 2, K2 = 2(255/3) = 170 = AAH

MVI A,10H          ; Σβήσιμο digits, εκτός των δύο πρώτων
STA 0B02H
STA 0B03H
STA 0B04H
STA 0B05H

MVI A,0DH          ; Μάσκα για RST 6.5
SIM                ; Αποθήκευση μάσκας διακοπής στον accumulator A
EI                 ; Ενεργοποίηση διακοπής τύπου 6.5 RST

WAIT:
       JMP WAIT    ; Ατέρμονη επανάληψη μέχρι να έρθει διακοπή τύπου RST 6.5

INTR_ROUTINE:      ; Ρουτίνα διακοπής
       CALL KIND   ; Είσοδος πληκτρολογίου (μονάδες) και αποθήκευσή της στον acc A
       STA 0B00H   ; Αποθήκευση εισόδου στο πρώτο digit
       MOV B,A     ; Αποθήκευση μονάδων στην βοηθητική μεταβλητή B ως μονάδες hex αριθμού
       CALL KIND   ; Είσοδος πληκτρολογίου ξανά (δεκαεξάδες) και αποθήκευσή της στον acc
A
       STA 0B01H   ; Αποθήκευση εισόδου στο δεύτερο digit
       RLC         ; 4 rotations για πολλαπλασιασμό με 16 και εύρεση δεκαεξάδων
       RLC
       RLC
       RLC
       ADD B       ; Πρόσθεση μονάδων και δεκαεξάδων για τελική διαμόρφωση του
                   ; hex αριθμού εισόδου
       PUSH PSW    ; Αποθήκευση A στην στοίβα, καθώς περιέχει χρήσιμα για μετά δεδομένα
       PUSH D      ; Το ίδιο και για τον διπλό καταχωρητή DE (περιέχουν τα κατώφλια)
       LXI D,0B00H ; Δίνουμε στον D την διεύθυνση του πρώτου digit
       CALL STDM   ; Μετακινεί το μήνυμα στην θέση που περιμένει να το βρει η ρουτίνα DCD
       CALL DCD    ; Ενημερώνει το display
       POP D       ; Δίνει στο D την προηγούμενη τιμή του (κατώφλι 2), από την στοίβα
       POP PSW     ; Το ίδιο και στον A που περιείχε την τιμή της εισόδου σε hex μορφή

       MOV B,D     ; Χρησιμοποιούμε βοηθητικά τον B γιατί αναφέρεται ρητά ότι τα D,E
                   ; περιέχουν τις τιμές των κατωφλιών
       INR B       ; Αύξηση του B για την απαραίτητη σύγκριση
       CMP B       ; Σύγκριση αν A =< 55H
       JC LT_56    ; Αν ναι πήγαινε στην ετικέτα για άναμμα του πρώτου LED
       MOV B,E     ; Αν όχι, ακολουθούμε την ίδια διαδικασία για το κατώφλι 2
       INR B
       CMP B         ; Σύγκριση αν A =< ABH
       JC LT_AB      ; Αν ναι, πήγαινε στην ετικέτα για άναμμα του δεύτερου LED
       MVI A,04H     ; Αν όχι, ετοίμασε το τρίτο LED
       JMP LED_IT_GO ; και πήγαινε να το ανάψεις

LT_56:
       MVI A,01H     ; Πρώτο LED
       JMP LED_IT_GO

LT_AB:
       MVI A,02H     ; Δεύτερο LED. Δεν χρειάζεται JMP καθώς το άναμμά του γίνεται
                     ; ακριβώς παρακάτω

LED_IT_GO:
       CMA           ; Συμπλήρωμα ως προς 1 του A, ώστε μέσω αντίστροφης λογικής,
                     ; να ανάψει το σωστό LED
       STA 3000H     ; Άναμμα LED
       EI            ; Ενεργοποίηση για πιθανή επόμενη διακοπή τύπου 6.5 RST
       JMP WAIT      ; Επιστροφή στο να περιμένουμε διακοπή

END
```

Άσκηση 3

A)

```
SWAP Nible MACRO Q
      PUSH PSW ; Σώζω το περιεχόμενο του A στην στοίβα. Τον χρησιμοποιώ ως
      MOV A,Q  ; βοηθητική μεταβλητή που φυλάει το περιεχόμενο της παραμέτρου Q
      RLC      ; Για να εναλλάξουμε τα MSB με τα LSB ψηφία του hex
      RLC      ; αριθμού, κάνουμε 4 ολισθήσεις προς τα αριστερά, γιατί
      RLC      ; ένας hex με 2 ψηφία είναι ένας binary με 8. Έτσι, με
      RLC      ; αυτές τις ολισθήσεις εναλλάσσουμε τις δεκαεξάδες με τις
               ; μονάδες του αριθμού αυτού, όπως και ζητείται
      MOV Q,A  ; Κατόπιν, αποθηκεύουμε τον αριθμό που προκύπτει στην
               ; παράμετρο-καταχωρητή Q

      MOV A,M  ; Μέσω της M παίρνουμε την διεύθυνση που 'δείχνει' ο HL
      RLC      ; Ομοίως, κάνουμε τις 4 ολισθήσεις προς τα αριστερά για
      RLC      ; τον ίδιο λόγο με πριν
      RLC
      RLC
      MOV M,A  ; και αποθηκεύουμε την τελική τιμή πάλι στον HL
      POP PSW  ; Επαναφέρω το περιεχόμενο του βοηθητικού καταχ./acc A
ENDM
```

B)

```
FILL MACRO RP,X,K
      PUSH PSW     ; Φυλάω το περιεχόμενο του A στην στοίβα
      PUSH H       ; και του HL
      MOV H,RP     ; Φορτώνω την ζητούμενη θέση στον HL
      MVI A,00H    ; Ο A μετράει τις επαναλήψεις

      LOOP:
      MVI M,K  ; Αποθηκεύουμε το K στη θέση που δείχνει ο HL
      INR M    ; Επόμενη θέση μνήμης
      INR A    ; Αυξάνουμε τον μετρητή επαναλήψεων
      CPI X    ; Σύγκριση με τον αριθμό X, το μέγεθος του τμήματος μνήμης
      JNZ LOOP ; Αν A != X συνέχισε, αν A = X τέλος

      POP H
      POP PSW    ; Επαναφορά καταχωρητών που χρησιμοποιήθηκαν
ENDM
```

Σημείωση: Επειδή ο μετρητής A ξεκινά από το 00H, στην περίπτωση που X = 256, το A θα πάρει τις τιμές 01H-FFH -λόγω της INR A- και τελικά την τιμή 00H. Άρα, θα γίνουν 256 επαναλήψεις.

Γ)

```
RHLR MACRO n
      PUSH PSW
      PUSH B

      MVI A,n
      CPI 00H     ; Αν n = 0, δεν θα εκτελεστεί το LOOP
      JZ END
      MVI B,n     ; B ως μετρητής επαναλήψεων

      LOOP:
      MOV A,L     ; (A) <- (L), ώστε να περιστρέψουμε τον L
      RAL         ; Περιστρέφουμε τον A προς τα αριστερά και
      MOV L,A     ; τον αποθηκεύουμε στον L. Πλέον CY = MSB(L)
      MOV A,H     ; (A) <- (H), ώστε να περιστρέψουμε τον H
      RAL         ; Το παλιό MSB του L έγινε LSB του H, ενώ
      MOV H,A     ; το MSB του H, βρίσκεται στο CY
      DCR B
      JNZ LOOP    ; Επανάλαβε, μέχρι B = 0

      END:
      POP B       ; Επαναφορά καταχωρητών που χρησιμοποιήθηκαν
      POP PSW
ENDM
```

Άσκηση 4

Η διακοπή RST 7.5 είναι διακοπή hardware και έχει διεύθυνση 003CH. Για να εφαρμοστεί αυτή η μάσκα πρέπει να έχουμε ενεργοποιήσει τις διακοπές και να λάβουμε την κατάλληλη μάσκα.

Υπόθεση:
Στη θέση μνήμης (SP)-1 αποθηκεύεται ο PCH και στη θέση μνήμης (SP)-2, αποθηκεύεται ο PCL.

- **Αρχικά** έχουμε ότι

(PC) = 0800H
(SP) = 3000H

Αρχή ρουτίνας εξυπηρέτησης της CALL

- Με την εκτέλεση της εντολής CALL 0880H αποθηκεύεται η τρέχουσα τιμή του μετρητή προγράμματος στον σωρό, ενώ ο δείκτης σωρού μειώνεται κατά 2 και άρα αυτός ανεβαίνει 2 θέσεις πάνω. Το μέγεθος του σωρού αυξήθηκε κατά δύο θέσεις. Ο μετρητής προγράμματος πλέον περιέχει τη διεύθυνση 0880H.
  Έχουμε: ((SP)-1) <- 08H και ((SP)-2) <- 00H και (SP) <- (SP)-2, δηλαδή

Σωρός
(2FFFH) <- 08H
(2FFEH) <- 00H
(SP) <- 2FFEH
Μετρητής προγράμματος
(PC) <- 0880H

Συνεχίζεται η εκτέλεση του προγράμματος.

Αρχή διακοπής

- Μόλις εντοπιστεί η διακοπή, κατόπιν της εντολής CALL 0880H, πρέπει να αποθηκεύσουμε εκ νέου τον τρέχοντα μετρητή προγράμματος στον σωρό. Άρα, σώζουμε στον σωρό την τιμή 0880H και ο μετρητής προγράμματος δείχνει στην διεύθυνση που υποδεικνύει η παρούσα διακοπή, δηλαδή την 003CH. Ο δείκτης σωρού μειώνεται κατά 2 και άρα αυτός ανεβαίνει ξανά 2 θέσεις πάνω. Το μέγεθος του σωρού αυξήθηκε κατά άλλες δύο θέσεις.
  Έχουμε: ((SP)-1) <- 08H και ((SP)-2) <- 80H και (SP) <- (SP)-2, δηλαδή

Σωρός
(2FFDH) <- 08H
(2FFCH) <- 80H
(SP) <- 2FFCH
Μετρητής προγράμματος
(PC) <- 003CH

Ολοκλήρωση διακοπής

- Μόλις ολοκληρωθεί η διακοπή, στον μετρητή προγράμματος επαναφέρεται από τον σωρό η τιμή που είχε ακριβώς πριν φύγει από την ρουτίνα εξυπηρέτησης της CALL, την τιμή 0880H. Γίνονται POP οι δύο θέσεις του σωρού που περιείχαν την θέση αυτή και ο δείκτης του σωρού πάλι δείχνει την διεύθυνση που μας επιστρέφει στο κυρίως πρόγραμμα, 0080H.
  Έχουμε: (SP) <- (SP)-2, δηλαδή

Σωρός
(SP) <- 2FFEH
όπου
(2FFFH) <- 08H
(2FFEH) <- 00H
Μετρητής προγράμματος
(PC) <- 0880H

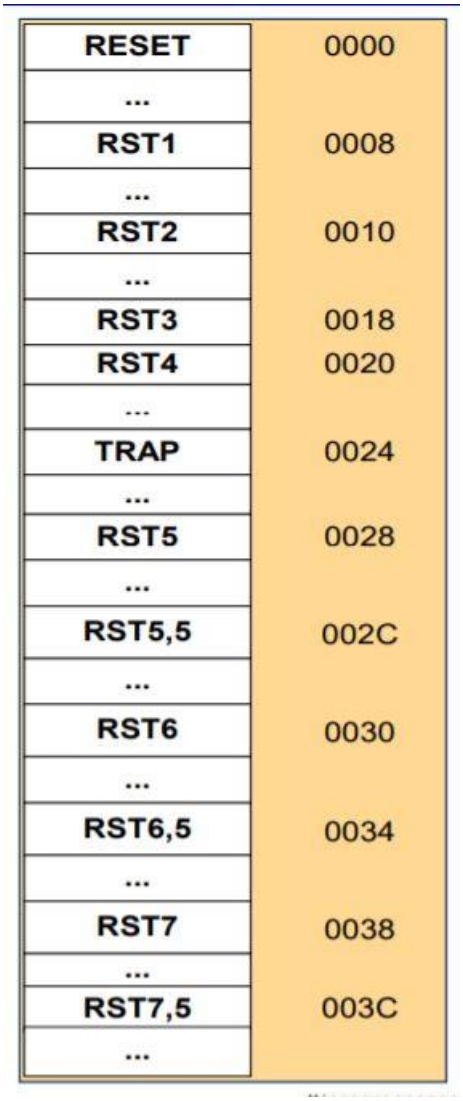
Ολοκλήρωση ρουτίνας εξυπηρέτησης της CALL και επιστροφή στο κύριο πρόγραμμα

- Μόλις ολοκληρωθεί η ρουτίνα εξυπηρέτησης της CALL, στον μετρητή προγράμματος επαναφέρεται από τον σωρό η τιμή που είχε ακριβώς πριν φύγει από την κύρια ροή του προγράμματος, την τιμή 0080H, μέσω της εντολής RET. Γίνονται POP οι δύο θέσεις του σωρού που περιείχαν την θέση αυτή και ο δείκτης του σωρού δείχνει ξανά την τιμή που έδειχνε αρχικά, 3000H.
  Έχουμε: (SP) <- (SP)-2, δηλαδή

Σωρός
(SP) <- 3000H
Μετρητής προγράμματος
(PC) <- 0800H

Εδώ αξίζει να σημειωθεί ότι όταν έρχεται διακοπή, δεν πηγαίνουμε απευθείας στις θέσεις μνήμης της ρουτίνας διακοπής. Γενικά, ομαδοποιούνται οι διευθύνσεις των διακοπών σε κοντινές θέσεις μνήμης για πρακτικούς/μνημονικούς λόγους. Εκεί δεν αποθηκεύουμε τα δεδομένα προγράμματος της ρουτίνας της διακοπής, καθώς αυτά τα δεδομένα μεταβάλλονται συχνά, ανάλογα με τον ορισμό της ρουτίνας που δημιουργούμε ως προγραμματιστές. Για τους δύο παραπάνω λόγους, στις ομαδοποιημένες θέσεις μνήμης των διακοπών, οι μόνες εντολές που φυλάσσονται είναι JMP για κάθε διακοπή, το οποίο μας οδηγεί στις θέσεις μνήμης που έχουν δεσμευτεί για τον ορισμό της εκάστοτε ρουτίνας διακοπής. Οπότε, ο PC μεταπηδά σε μακρινές θέσεις μνήμης και ο σωρός φυλάει τις προηγούμενες τιμές του PC, ώστε να μπορούμε να επιστρέφουμε κάθε φορά. Στα πλαίσια της άσκησης θεώρησα ότι αν έχω διακοπή πηγαίνω κατευθείαν στις θέσεις μνήμης που περιέχουν τον ορισμό της ρουτίνας διακοπής, χωρίς να παρεμβάλλεται JMP ενδιάμεσα.

Σχηματική απεικόνιση της ομαδοποίησης των διακοπών σε κοντινές θέσεις μνήμης. Στις θέσεις μνήμης που υπάρχουν αποσιωπητικά, συνήθως συναντάμε JMP στις θέσεις μνήμης με τους ορισμούς της εκάστοτε ρουτίνας διακοπής.



Άσκηση 5

A)

```
LXI H,0000H ; Μηδενισμός του HL
MVI C,40H   ; Ο μετρητής C αρχικοποιείται με 64, όσα τα βήματα που
            ; χρειάζονται για την μεταφορά των δεδομένων
MVI A,0DH   ; Μάσκα της διακοπής RST 6.5
SIM         ; Εφαρμογή μάσκας
EI          ; Ενεργοποίηση διακοπών

WAIT:
     JMP WAIT    ; Επαναλήψεις μέχρι να έρθει διακοπή

AVERAGE:       ; Έχει γίνει η μεταφορά δεδομένων,
     DI          ; απενεργοποίηση των διακοπών και υπολογισμός MO
     DAD H       ; 2^5=32 αριθμοί, με 32 πρέπει να διαιρέσω το
     DAD H       ; αποτέλεσμα για να βρω τον MO ή στο δυαδικό να κάνω 5
     DAD H       ; ολισθήσεις δεξιά. Ισοδύναμα κάνουμε 3 αριστερά και
                 ; αποθηκεύουμε το αποτέλεσμα στον καταχωρητή H.
     HLT         ; τέλος προγράμματος

INTR_ROUTINE:  ; RST 6.5
     MOV A,C     ; Ο A αποκτά την τιμή του μετρητή
     ANI 01H     ; Απομόνωση του LSB του C
     CPI 00H     ; Σύγκριση με το 0
     JNZ MSB     ; Αν δεν είναι ίσο με 0, υπολόγισε MSBs, αλλιώς LSBs(~)

LSB:
     IN 20H      ; Διάβασμα των LSBs της θύρας μας, που αν έχουμε φτάσει
                 ; εδώ, αντιπροσωπεύουν τα LSBs του αριθμού μας
     ANI 0FH     ; Μάσκα για τα LSBs
     MVI D,00H   ; ο DE περιέχει τα LSB του αριθμού μας και στην θέση
     MOV E,A     ; των MSBs μηδενικά (μορφή 0000W'X'Y'Z')
     DCR C       ; Μείωση του μετρητή βημάτων Ξέρουμε ότι δεν θα
                 ; μηδενίσει τον μετρητή μας και δεν κάνουμε έλεγχο
     EI          ; Ενεργοποίηση επόμενων διακοπών
     JMP WAIT

MSB:
     IN 20H      ; Διάβασμα των LSBs της θύρας μας, που αν έχουμε φτάσει
                 ; εδώ, αντιπροσωπεύουν τα MSBs του αριθμού μας
     ANI 0FH     ; Μάσκα για τα MSBs, ο A τα φυλάει στη μορφή (0000WXYZ)
     RLC         ; 4 αριστερές ολισθήσεις για να βρεθούν στην σωστή θέση
     RLC         ; και έχουμε WXYZ0000
     RLC         ;
     RLC         ;
     ORA E       ; Ο A περιέχει τον αριθμό μας (WXYZW'X'Y'Z')
     MVI D,00H
     MOV E,A     ; Πλέον ο DE περιέχει ολόκληρο τον ζητούμενο αριθμό
     DAD D       ; και τον προσθέτουμε στον HL (00000000WXYZW'X'Y'Z')
     DCR C       ; Μείωση του μετρητή βημάτων για μεταφορά δεδομένων
     MOV A,C     ; Ο έλεγχος για τον μετρητή γίνεται μόνο για τα MSBs,
                 ; αφού ξέρουμε ότι αυτά θα έρθουν τελευταία για τον
     CPI 00H     ; τελευταίο αριθμό. Αν τελειώσουν τα βήματα για τα
     JZ AVERAGE  ; δεδομένα πήγαινε στην ετικέτα AVERAGE να υπολογίσεις
                 ; τον MO και τέλος.
     EI          ; Ενεργοποίηση επόμενων διακοπών
     JMP WAIT
```

(~) Ο μετρητής ξεκινά πρώτα από άρτιο αριθμό (64) και λαμβάνει πρώτα τα LSB του πρώτου αριθμού και κατόπιν, όταν ο μετρητής μειωθεί (63) τα MSB του. Επαγωγικά έχουμε ότι μετράμε LSBs όταν ο μετρητής C είναι άρτιος και MSBs όταν είναι περιττός.

B)

```
LXI H,0000H
MVI C,40H

INP: IN DATA      ; Διάβασμα του X7
     ANI 80H      ; μάσκα για απομόνωση του MSB X7
     RLC          ; Το φέρνω στην θέση του LSB για να το ελέγξω
     CPI 00H      ; Σύγκριση με το 0
     JZ INP       ; Επανάληψη αν X7==0
     JMP ROUTINE  ; Αλλιώς θα ακολουθήσει έγκυρο δεδομένο και συνεχίζουμε

AVERAGE:
     DAD H        ; 2^5=32 αριθμοί, με 32 πρέπει να διαιρέσω το
     DAD H        ; αποτέλεσμα για να βρω τον MO ή στο δυαδικό να κάνω 5
     DAD H        ; ολισθήσεις δεξιά. Ισοδύναμα κάνουμε 3 αριστερά και
                  ; αποθηκεύουμε το αποτέλεσμα στον καταχωρητή H.
     HLT          ; τέλος προγράμματος

ROUTINE:
     MOV A,C      ; Ο A αποκτά την τιμή του μετρητή
     ANI 01H      ; Απομόνωση του LSB του C
     CPI 00H      ; Σύγκριση με το 0
     JNZ MSB      ; Αν δεν είναι ίσο με 0, υπολόγισε MSBs, αλλιώς LSBs(~)

LSB: IN 20H       ; Διάβασμα των LSBs της θύρας μας, που αν έχουμε φτάσει
                  ; εδώ, αντιπροσωπεύουν τα LSBs του αριθμού μας
     ANI 0FH      ; Μάσκα για τα LSBs
     MVI D,00H    ; ο DE περιέχει τα LSB του αριθμού μας και στην θέση
     MOV E,A      ; των MSBs μηδενικά (μορφή 0000W'X'Y'Z')
     DCR C        ; Μείωση του μετρητή βημάτων Ξέρουμε ότι δεν θα
                  ; μηδενίσει τον μετρητή μας και δεν κάνουμε έλεγχο
     JMP INP

MSB: IN 20H       ; Διάβασμα των LSBs της θύρας μας, που αν έχουμε φτάσει
                  ; εδώ, αντιπροσωπεύουν τα MSBs του αριθμού μας
     ANI 0FH      ; Μάσκα για τα MSBs, ο A τα φυλάει στη μορφή (0000WXYZ)
     RLC          ; 4 αριστερές ολισθήσεις για να βρεθούν στην σωστή θέση
     RLC          ; και έχουμε WXYZ0000
     RLC          ;
     RLC          ;
     ORA E        ; Ο A περιέχει τον αριθμό μας (WXYZW'X'Y'Z')
     MVI D,00H
     MOV E,A      ; Πλέον ο DE περιέχει ολόκληρο τον ζητούμενο αριθμό
     DAD D        ; και τον προσθέτουμε στον HL (00000000WXYZW'X'Y'Z')
     DCR C        ; Μείωση του μετρητή βημάτων για μεταφορά δεδομένων
     MOV A,C      ; Ο έλεγχος για τον μετρητή γίνεται μόνο για τα MSBs,
                  ; αφού ξέρουμε ότι αυτά θα έρθουν τελευταία για τον
     CPI 00H      ; τελευταίο αριθμό. Αν τελειώσουν τα βήματα για τα
     JZ AVERAGE   ; δεδομένα πήγαινε στην ετικέτα AVERAGE να υπολογίσεις
                  ; τον MO και τέλος.
     JMP INP
```