Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχ. Και Μηχ. Υπολογιστών
Εργαστήριο Μικροϋπολογιστών , 7ο εξάμηνο - Ροή Υ
Ακαδημαϊκή Περίοδος : 2011-2012

3η ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ

Ομάδα: C16
Γερακάρης Βασίλης Α.Μ.: 03108092
Διβόλης Αλέξανδρος Α.Μ.: 03107238
Ίεντσεκ Στέργιος Α.Μ.: 03108690

Ειδικό Θέμα 2

Στην άσκηση αυτή υλοποιούμε μια αριθμομηχανή που μπορεί να εκτελέσει πρόσθεση και αφαίρεση δυο διψήφιων δεκαδικών και εμφανίζει την απόλυτη τιμή του αποτελέσματος ως έναν διψήφιο δεκαεξαδικό. Τα δεδομένα, καθώς και τα αποτελέσματα, πρέπει να εμφανίζονται στην οθόνη. Για την υλοποίηση έχουμε χρησιμοποιήσει μια ρουτίνα η οποία διαβάζει τα δυο πρώτα δεκαδικά ψηφία από το πληκτρολόγιο. Κάθε φορά που διαβάζεται ένα έγκυρο ψηφίο το αποθηκεύουμε στη μνήμη και το εμφανίζουμε στην οθόνη. Η χρήση της ρουτίνας **STDM** πειράζει όλους τους καταχωρητές και για αυτό κάνουμε κατάλληλα **PUSH** και **POP**, ώστε να εξασφαλίσουμε τη σωστή λειτουργία του προγράμματος.

Έπειτα, διαβάζουμε το πλήκτρο που υποδηλώνει ποια πράξη πρέπει να γίνει. Αν είναι **DECR** (πλην) βάζουμε στον καταχωρητή (C) τον αριθμό 1, αλλιώς αφήνουμε στον (C) το 0. Έπειτα διαβάζουμε και τον επόμενο διψήφιο δεκαδικό, τον αποθηκεύουμε και τον εμφανίζουμε όπως πριν.

Αφού διαβάσουμε τα δεδομένα, τα μετατρέπουμε σε δεκαεξαδικούς αριθμούς και τα αποθηκεύουμε στους καταχωρητές (D)-(E). Για τη μετατροπή αυτή χρησιμοποιούμε τη ρουτίνα **CONVERT**, η οποία παίρνει τον αριθμό των δεκάδων και για κάθε μια προσθέτει 10 και έπειτα προσθέτει τον αριθμό των μονάδων φτιάχνοντας έτσι το σωστό αποτέλεσμα. Είναι απαραίτητο να αναφέρουμε ότι στην αρχή της ρουτίνας αυξάνουμε κατά μια τις δεκάδες, έτσι ώστε στην περίπτωση που οι μονάδες είναι μηδέν να δουλεύει η λογική **DCR (δεκάδες)-IF_NOT_ZERO ADD 10**. Στο τέλος της επαναληπτικής διαδικασίας αφαιρούμε 10 ώστε να εξουδετερώσουμε τη μια δεκάδα που προσθέσαμε αρχικά. Σε αυτό το σημείο ελέγχουμε αν ο (C) είναι 0 ή 1. Αν είναι 0 τότε κάνουμε μια πρόσθεση μεταξύ των (D)-(E) για την οποία είμαστε σίγουροι ότι δεν θα υπάρχει υπερχείλιση, μιας και το μέγιστο αποτέλεσμα είναι **99 + 99 = 198 < 255.** Αν ο (C) είναι 1 τότε ελέγχουμε ποιος από τους δυο αριθμούς είναι μεγαλύτερος ώστε να κάνουμε την κατάλληλη αφαίρεση. Αφού βρούμε το αποτέλεσμα, χωρίζουμε τα δυο δεκαεξαδική ψηφία, ώστε να τα εμφανίσουμε κατάλληλα. Όταν εμφανίσουμε το αποτέλεσμα, καλούμε τη ρουτίνα συστήματος **BEEP** προκειμένου να ενημερώσουμε το χρήστη ότι μπορεί να κάνει μια επόμενη πράξη, αρχικοποιούμε τους καταχωρητές καλώντας τη ρουτίνα **RST_ALL** και ξεκινούμε από την αρχή, περιμένοντας την επόμενη είσοδο. Ο κώδικας παρατίθεται παρακάτω:

```
        IN 10H
        CALL RST_ALL ;Καλούμε ρουτίνα αρχικοποίησης
        PUSH PSW     ;πριν από την κλήση της ρουτίνας
        PUSH B
        PUSH D
        PUSH H       ;Φυλάσσουμε όλους τους καταχωρητές γιατί η STDM τους πειράζει
        CALL STDM    ;Σβήνουμε την οθόνη
        POP H
        POP D
        POP B
        POP PSW
START:

        LXI H,0A01H  ;Φορτώνουμε τη θέση μνήμης όπου θέλουμε να δουλέψει
        CALL READ_IN ;η ρουτίνα ανάγνωσης και την καλούμε για τον πρώτο αριθμό εισόδου
READ_OP:
        CALL KIND    ;Διαβάζουμε το πρόσημο
        CPI 81H      ;Ελέγχουμε αν πατήθηκε το DECR
        JZ MINUS     ;Αν είναι πλην βάζουμε στον (C) το 1
        CPI 85H      ;Ελέγχουμε αν πατήθηκε το FETCH PC
        JZ CONT
        JMP READ_OP  ;Απορρίπτουμε οποιοδήποτε άλλο κουμπί
CONT:
        LXI H,0A05H
        CALL READ_IN ;Διαβάζουμε τον δεύτερο αριθμό εισόδου
        LXI H,0A05H  ;Φορτώνουμε τη θέση μνήμης όπου θέλουμε να δουλέψει
        CALL CONVERT ;η ρουτίνα μετατροπής του δεκαδικού σε δεκαεξαδικό
        MOV D,A      ;Κρατάμε τη μια είσοδο στον (D)
        LXI H,0A01H
        CALL CONVERT ;Μετατρέπουμε και τον δεύτερο
        MOV E,A      ;Τον κρατάμε στον (E)
        MOV A,C      ;Ελέγχουμε ποια πράξη έχει πραγματοποιηθεί
        RAR
        JNC SUM      ;Αν είναι αφαίρεση
        MOV A,D      ;Ελέγχουμε ποιος είναι μεγαλύτερος
        CMP E
        JC SWAP      ;Αν (E)>(D) κάνουμε (D)-(E)
        SUB E        ;Αλλιώς (E)-(D)
        MOV B,A      ;Κρατάμε το αποτέλεσμα στον (B)
        JMP RESULT
SWAP:
        MOV A,E
        SUB D
        MOV B,A      ;Κρατάμε το αποτέλεσμα στον (B)
        JMP RESULT
SUM:                 ;Αν είναι πρόσθεση απλά προσθέτουμε
        MOV A,D
        ADD E
        MOV B,A      ;Κρατάμε το αποτέλεσμα στον (B)
RESULT:
        RAR
        RAR
        RAR
        RAR          ;Κάνουμε 4 ολισθήσεις και περνάμε μέσα από μια μάσκα
        ANI 0FH      ;το αποτέλεσμα ώστε να κρατήσουμε τα 4 MSB
        LXI H,0A03H
        MOV M,A      ;Τα βάζουμε στη θέση μνήμης που πρέπει ώστε να τα εμφανίσουμε
        MOV A,B
        ANI 0FH      ;Κρατάμε τα 4  LSB
        DCX H
        MOV M,A      ;Και τα βάζουμε στην αντίστοιχη θέση μνήμης
        LXI D,0A00H  ;Βάζουμε στον (D)-(E) τη θέση μνήμης όπου ξεκινούν τα αποτελέσματα
        CALL STDM    ;και τα εμφανίζουμε
        CALL BEEP    ;Ειδοποιούμε το χρήστη ότι μπορεί να βάλει νέα δεδομένα
        CALL RST_ALL ;Αρχικοποιούμε όλους τους καταχωρητές σε κάθε εκτέλεση
        JMP START    ;Δημιουργούμε συνεχή λειτουργία

MINUS:
        MVI C,01H
        JMP CONT
```

```
READ_IN:            ;Ρουτίνα ανάγνωσης διψήφιου δεκαδικού
       CALL KIND    ;Διαβάζουμε από το πληκτρολόγιο
       CPI 0AH      ;Κοιτάμε αν είναι δεκαδικό ψηφίο
       JNC READ_IN  ;Περιμένουμε μέχρι να έρθει ένα έγκυρο ψηφίο
       MOV M,A      ;Το κρατάμε στη θέση μνήμης όπου πρέπει να έχει καθοριστεί
       PUSH PSW     ;πριν από την κλήση της ρουτίνας
       PUSH B
       PUSH D
       PUSH H       ;Φυλάσσουμε όλους τους καταχωρητές γιατί η SRDM του πειράζει
       CALL STDM    ;Εμφανίζουμε το αποτέλεσμα
       POP H
       POP D
       POP B
       POP PSW
       CALL DCD
READ_SCND:          ;Κάνουμε το ίδιο και για το δεύτερο ψηφίο
       CALL KIND
       CPI 0AH
       JNC READ_SCND
       DCX H        ;Το δεύτερο ψηφίο αποθηκεύεται στην προηγούμενη θέση μνήμης
       MOV M,A
       PUSH PSW
       PUSH B
       PUSH D
       PUSH H
       CALL STDM
       POP H
       POP D
       POP B
       POP PSW
       CALL DCD
       RET

CONVERT:            ;Ρουτίνα μετατροπής δεκαδικού σε δεκαεξαδικό
       MVI A,00H    ;Μηδενίζουμε τον (A)
       MOV B,M      ;Φέρνουμε τις δεκάδες από τη μνήμη που έχει οριστεί πριν την κλήση της ρουτίνας
       INR B        ;Αυξάνουμε τον αριθμό των δεκάδων κατά 1 ώστε να δουλέψει η
LOOP1:              ;επανάληψη σε περίπτωση που έχουμε 0 δεκάδες
       ADI 0AH      ;Για κάθε δεκάδα προσθέτουμε 10 στον (A)
       DCR B
       JNZ LOOP1
       SBI 0AH      ;Αφαιρούμε 10 λογω της πλεονάζουσας δεκάδας που προσθέσαμε
       DCX H        ;Παίρνουμε από την προηγούμενη θέση μνήμης τις μονάδες
       MOV B,M
       ADD B        ;Και τις προσθέτουμε
       RET

RST_ALL:            ;Ρουτίνα αρχικοποίησης καταχωρητων
       MVI A,00H
       LXI B,0000H
       LXI D,0A00H
       LXI H,0A00H  ;Βάζουμε σε 6 διαδοχικές θέσεις μνήμης το 10H ώστε να κρατήσουμε
       MVI M,10H    ;σβηστό το display
       INX H
       MVI M,10H
       INX H
       MVI M,10H
       INX H
       MVI M,10H
       INX H
       MVI M,10H
       INX H
       MVI M,10H
       RET
END
```