Σχολή Ηλεκτρολόγων Μηχανικών
&
Μηχανικών Υπολογιστών

Εργαστήριο Μικροϋπολογιστών

Δεύτερη Εργαστηριακή Άσκηση

Λαμπράκος Χρήστος, Α.Μ: 03112062
Μανδηλαράς Νικηφόρος, Α.Μ: 03112012
Σπαθαράκης Δημήτριος, Α.Μ: 03113523
Έβδομο Εξάμηνο

Παραδοτέα: 25/10/2015

Άσκηση 3i

Για τον καθορισμό της εξόδου της κάθε πύλης, βασιστήκαμε στο "κυρίαρχο" στοιχείο της εκάστοτε περίπτωσης. Ελέγξαμε, λόγου χάριν, αν τουλάχιστον μία από τις 2 εισόδους μιας πύλης AND είναι μηδέν--τότε θέταμε την έξοδο σε μηδέν και δεν μπαίναμε στον κόπο να ελέγχουμε το δεύτερο bit. Ομοίως για τις OR και τους άσσους.

Ακριβώς επειδή η παραπάνω τακτική δεν εξασφαλίζει πως σε κάθε έλεγχο θα γίνουν δύο ολισθήσεις της εισόδου, ο κώδικας περιέχει ετικέτες του τύπου **FIX**Y, όπου Y αριθμός. Κάθε φορά που γίνονται και οι 2 απαραίτητες ολισθήσεις για τον καθορισμό του αποτελέσματος, προκειμένου να αποφευχθεί μια τρίτη η οποία θα κατέστρεφε την έξοδο, οι ετικέτες που αναφέραμε φροντίζουν να προηγηθεί μια ολίσθηση προς την αντίθετη κατεύθυνση, ώστε να διατηρηθεί η ευθυγράμμιση.

Για τον ίδιο λόγο χρησιμοποιήθηκε και ο καταχωρητής **L** ως σημαία.

Τέλος, όταν καταλήξουμε στην έξοδο της πρώτης AND, την αποθηκεύουμε στον καταχωρητή **B**, τον οποίο και αργότερα συγκρίνουμε με την έξοδο της δεύτερης AND. Τότε, και μόνο τότε, αποφασίζουμε αν το πρώτο bit της εξόδου (δηλαδή η XOR) θα τεθεί.

Ακολουθεί ο πλήρης κώδικας, συνοδευόμενος από σχόλια. Το εκτελέσιμο αρχείο της άσκησης επισυνάπτεται στην παραδοτέα εργασία.

threeone

```
BEGIN: IN 10H

LOOPA: MVI H,00H     ;H will have the final result
       MVI L,00H     ;just a flag
       MVI B,00H     ;...and another
       LDA 2000H
       RRC
       JNC G10       ;zero determines AND output
       RRC
       JC G11        ;same here
       JMP FIX1
G11:   INR B         ;remember the first result
FIX1:  RLC
G10:   RRC           ;else let it be. check second gate
       RRC
       JNC G20       ;same way as before
       RRC
       JC G21
       JMP FIX2
G21:   MOV D,A       ;whatever G1 was, set second gate
       MOV A,B
       CPI 01H       ;was the first gate an ace?
       JZ CONT       ;if yes, do nothing
       INR H         ;if not, set the XOR
CONT:  MOV A,H       ;we have to add 2 to H
       ADI 02H
       MOV H,A       ;update H
       MOV A,D       ;retrieve input
FIX2:  RLC
G20:   MOV D,A
       MOV A,B
       CPI 00H       ;we check what the XOR will be
       JZ DONT       ;in this case, same as before
       INR H
DONT:  MOV A,D
       RRC
       RRC
       JC G31        ;G3 is OR, so an ace determines it
       RRC
```

```
       JNC FIX3
       INR L
G31:   MOV D,A      ;we just set G3 by adding 4 to the result
       MOV A,H
       ADI 04H
       MOV H,A
       MOV A,L      ;check the flag
       CPI 01H
       JZ FIX4
       MOV A,D
G30:   RRC
       RRC
       JC G41
       RRC
       JNC LOAD
G41:   MOV A,H
       ADI 08H
SHOOT: CMA          ;inverse logic!
       STA 3000H
       JMP LOOPA

FIX4:  MOV A,D
FIX3:  RLC
       JMP G30

LOAD:  MOV A,H
       JMP SHOOT

       END
```

Άσκηση 4ii

Αυθαίρετα, και με βάση το μέγεθος του κώδικα στη μνήμη, επιλέξαμε τις έξι θέσεις που αρχίζουν από την **0840H** για την κλήση της **STDM**. Αρχικά, για αρχικοποίηση της οθόνης, φορτώνουμε και στις έξι αυτές θέσεις τον χαρακτήρα του κενού και τις αποτυπώνουμε.

Κατόπιν, η **KIND** περιμένει την όποια είσοδο. Φροντίζουμε πριν την κλήση της--δηλαδή στο τέλος της λούπας **GOON**--ο διπλός καταχωρητής **DE** να δείχνει στο δεύτερο αριστερότερο ψηφίο. Όταν πατηθεί κάποιο πλήκτρο, ο διψήφιος κωδικός του αποθηκεύεται στον συσσωρευτή. Απομονώνουμε μέσω δύο **ANI** τα ψηφία, και σώζουμε το καθένα τους στην αντίστοιχη θέση μνήμης. Το δεξί μισό πάει στο δεύτερο αριστερότερο ψηφίο, και το αριστερό μισό στο πρώτο. Έτσι εκμεταλλευόμαστε την ταύτιση της αριθμητικής τιμής του όποιου ψηφίου θέλουμε να απεικονίσουμε με τον αντίστοιχο κωδικό του.

Τέλος, επιστρέφουμε στην αρχή της λούπας ώστε το πρόγραμμα να είναι συνεχούς λειτουργίας.

Ακολουθεί ο κώδικας.

```
                          fourtwo
BEGIN: IN 10H
       MVI A,10H          ;blank char code
       MVI B,06H          ;loop counter
       LXI D,0840H        ;display start address

LOOPA: STAX D             ;load blank char to ALL disp digits
       INX D
       DCR B
       MOV C,A
       MOV A,B
```

```
       CPI 00H
       JNZ TRV
       JMP LIGHT           ;initialize display (kill all)

GOON:  CALL KIND
       MOV C,A             ;save input
       ANI 0FH             ;keep right half of code
       STAX D              ;save it in fifth digit
       MOV A,C
       ANI F0H             ;likewise for left half
       RRC
       RRC
       RRC
       RRC
       INX D
       STAX D
LIGHT: LXI D,0840H         ;so that STDM works right
       CALL STDM
       CALL DCD            ;light em up
       LXI D,0844H         ;remember to point again
       JMP GOON            ;at the fifth digit!

TRV:   MOV A,C             ;here we just retrieve the blank
       JMP LOOPA           ;in case we lost it
                           ;during the loop counter check
       END
```

Άσκηση 4iv

Παρόμοια με προηγουμένως, φορτώνουμε τον κενό χαρακτήρα, στα δύο αριστερότερα ψηφία αυτή τη φορά.

Για το τρίτο από αριστερά ψηφίο, ελέγχουμε το MSB της εισόδου. Αν αυτό είναι άσσος, ο αριθμός είναι αρνητικός σύμφωνα με την αριθμητική του συμπληρώματος ως προς δύο, και συνεπώς φορτώνουμε τον χαρακτήρα του μείον. Επίσης παίρνουμε το συμπλήρωμα ως προς 2 του αριθμού, ώστε να προκύψει το μέτρο του. Στην περίπτωση που το MSB ήταν μηδέν, και συνεπώς ο αριθμός θετικός, η τιμή της εισόδου ταυτίζεται με το μέτρο του, και δεν χρειαζόμαστε κάποια περαιτέρω επεξεργασία.

Από εδώ και κάτω, το όλο ζήτημα ήταν η μετατροπή του μέτρου του αριθμού σε μορφή BCD, ώστε να απεικονιστεί κάθε ένα από τα ψηφία του. Για το σκοπό αυτό έγινε χρήση του διαγράμματος ροής που συνοδεύει την εκφώνηση, σύμφωνα με το οποίο αφαιρούμε διαρκώς δέκα από τον αριθμό, προσθέτοντας δεκάδες σε ένα μετρητή, μέχρις ότου ο αριθμός να γίνει μικρότερος του 10--οπότε και έχουμε φτάσει στις μονάδες. Τα σχόλια του κώδικα δείχνουν σε ποιους καταχωρητές αποθηκεύσαμε το εκάστοτε μέγεθος.

Το μόνο "πρόβλημα", λόγω του εύρους που επιτρέπουν τα 8 bits της εισόδου, ήταν ο καθορισμός των εκατοντάδων. Φυσικά, αυτές μπορούσαν να είναι είτε μηδέν, είτε ένα. Αποφασίσαμε στο τέλος του κώδικα που αντιστοιχεί στο διάγραμμα ροής που αναφέραμε, να ελέγξουμε την τιμή των δεκάδων. Αν αυτή ήταν μεγαλύτερη ή ίση του 10, αφαιρούσαμε δέκα από τις δεκάδες και προσθέταμε ένα στις εκατοντάδες. Σε αντίθετη περίπτωση, οι εκατοντάδες παρέμεναν 0.

Ακολουθεί ο κώδικας.

```
                    fourfour
BEGIN: IN 10H

       MVI A,10H     ;load blank space code
       STA 0B04H
```

```
STA 0B05H

LOOPA: LDA 2000H
       MOV B,A
       RLC           ;check sign (MSB)
       JC MIN        ;if it's an ace, display minus
PLS:   MVI A,10H     ;else kill third digit as well
       STA 0B03H
       JMP POS       ;positive numbers don't need transform
MIN:   MVI A,1CH     ;minus symbol
       STA 0B03H
       MOV A,B
       CMA
       INR A         ;two's complement
       JMP KAT3

POS:   MOV A,B
KAT3:  MVI H,00H     ;decades here
       MVI L,00H     ;units here
       MVI C,00H     ;...and finally the hundreds
LOOPB: CPI 0AH
       JNC KAT
KAT2:  MOV L,A       ;number was less than 10, so this
       JMP OUTB      ;is the units' value
KAT:   INR H         ;number was greater than 10, keep
       SUI 0AH       ;another decade and uppdate
       JMP LOOPB
OUTB:  MOV A,H
       CPI 0AH       ;if there were more than 10 decades...
       JC GOON
       MVI C,01H     ;...hundreds go to one,
       MOV A,H
       SUI 0AH       ;and subtract 10 from the decades
       MOV H,A
GOON:  MOV A,C       ;this just save the BCD values...
       STA 0B02H
       MOV A,H       ;...in the right...
       STA 0B01H
       MOV A,L       ;...place.
       STA 0B00H
       LXI D,0B00H   ;light em upp
       CALL STDM
       CALL DCD
       JMP LOOPA

       END
```

ΤΕΛΟΣ ΕΡΓΑΣΙΑΣ