Μικροϋπολογιστές

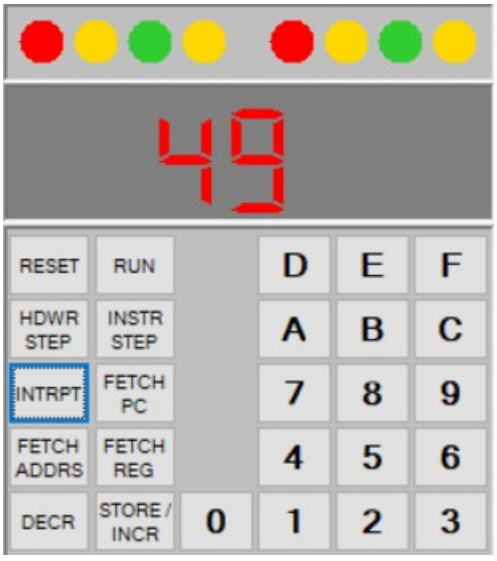
3η Ομάδα Ασκήσεων

Παπαρρηγόπουλος Θοδωρής el18040 paparrigopoulosthodoris@gmail.com
Φιλιππόπουλος Ορφέας el18082 orfeasfil2000@gmail.com

Άσκηση 1η

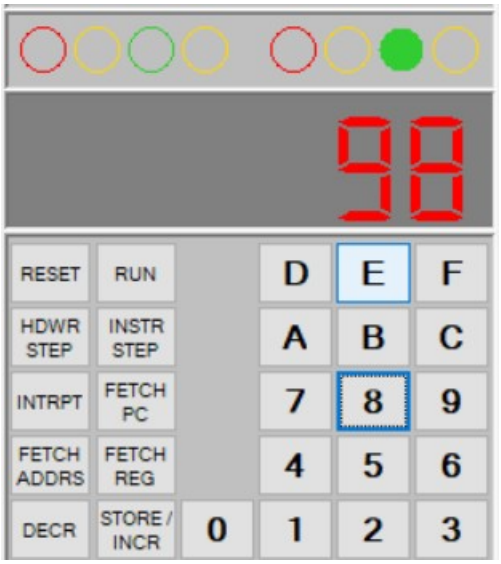
Αρχικά με κατάλληλη μάσκα και με τις εντολές SIM και EI επιτρέψαμε την διακοπή RST 6.5. Αρχικοποιούμε τα LED να είναι σβηστά ενώ γράφουμε το κενό στις 1,2,5,6,7 θέσεις των 7-segment. Στη συνέχεια απλά περιμένουμε μέχρι να μας έρθει η συγκεκριμένη διακοπή γίνουν τα ζητούμενα. Όταν έρθει κάποια διακοπή (τη δημιουργούμε εμείς πατώντας το κουμπί INTRPT) ανάβουν τα 3,4 των 7-segment αρχικοποιημένα στη τιμή 60 (αφού θέλουμε να είναι αναμμένα για 1 λεπτό) και ξεκινάει η αντίστροφη μέτρηση. Προκειμένου να μην αναβοσβήνουν τα LED δεν τα ανάβουμε στιγμιαία και στη να συνέχεια περιμένουμε να περάσει η διάρκεια του ενός δευτερολέπτου, τα ανάβουμε για 100ms 10 φορές αξιοποιώντας έναν counter (μέχρι το 10) και την εντολή DELB αφού έχουμε περάσει την τιμή 100 στον καταχωρητή B (προκειμένου το ανθρώπινο μάτι να έχει την εντύπωση ότι παραμένουν ανοιχτά καθ όλη τη διάρκεια του δευτερολέπτου). Αφού γίνουν 10 τέτοιες επαναλήψεις (δηλαδή πέρασε 1s) τότε μειώνουμε τον counter των δευτερολέπτων κατά ένα, τον φέρνουμε σε BCD μορφή και στη συνέχεια τον γράφουμε στα 3,4 των 7-segment. Εάν ο μετρητής των δευτερολέπτων φτάσει στο 0 τότε αρχικοποιούμε τα 3,4 των 7-segment στην τιμή 0 και περιμένουμε για τον επόμενο RST 6.5 interrupt. Τέλος έχουμε ενεργοποιήσει με την εντολή EI το interrupt μέσα στην υπορουτίνα εξυπηρέτησης του προκειμένου εάν ξαναπατηθεί το κουμπί INTRPT (δηλαδή ξαναέρθει interrupt) να αρχικοποιηθούν τα 3,4 των 7-segment στην τιμή 60.

Ένα στιγμιότυπο του κώδικά μας είναι:



Άσκηση 2

Αρχικά με κατάλληλη μάσκα και με τις εντολές SIM και EI επιτρέψαμε την διακοπή RST 6.5.
Στη συνέχεια με την εντολή KIND διαβάζουμε 2 φορές το πληκτρολόγιο και αποθηκεύουμε τους αριθμούς που διαβάσαμε στις σωστές θέσεις και με χρήση των DCD και STDM γράφουμε στα 2 δεξιότερα των 7-segment τους αριθμούς που διαβάσαμε. Ακολούθως, στο πρώτο διάβασμα κάνουμε 4 RLC's γιατί είναι τα 4 MSB's και προσθέτουμε τους 2 (τον ολισθημένο και την 2η είσοδο από το πληκτρολόγιο) αυτούς αριθμούς εξάγοντας έτσι τον αριθμό. Μετά συγκρίνουμε τον αριθμό που προέκυψε με τα κατώφλια K1 και K2 (K1 = 32H και K2 = A8H) και ανάλογα με την περιοχή όπου βρισκόμαστε ανάβουμε και το αντίστοιχο LED.
Ένα παράδειγμα του προγράμματος μας με τα παραπάνω K1,K2 είναι:



Άσκηση 3

a)
SWAP Nibble MACRO Q

```
PUSH PSW ;store A and F
MOV A,Q ;move input register
RRC
RRC
RRC
RRC ;4 right rotations with RRC
MOV Q,A ;now A has the number with swapped hex digits
MOV A,L
MOV L,H
```

```
        MOV H,A ;swap H with L using A
        POP PSW ;restore A and F
ENDM
```

β)

```
FILL MACRO RP,X,K
        PUSH PSW ;store A and F
        PUSH H ;store H,L
        MOV H,R
        MOV L,P ;HL has the first memory position
        MVI A,X ;A has the size of the section
        CPI 00H
        JNZ SKIP ;if size is not 0 then go to skip
        MOV M,K ;else store K in the first memory position
        INX H
        MVI A,FFH ;and continue to do the same for the next 255 positions (256 total).

        SKIP:

        MOV M,K ;store number K
        INX H ;next memory position
        DCR A ;decrease size counter
        CPI 00H
        JNZ SKIP ;if counter is not 0 then repeat
        POP H ;restore H,L
        POP PSW ;restore A,F
ENDM
```

c) RHLR MACRO n

```
        PUSH B ;store B
        MVI B,n
        MOV A,B ;rotation counter

        LOOP:

        CPI 00H
        JZ FINISH ;if rotation counter is 0 then finish
        MOV A,H
        RAR ;rotate H right so now CY has the LSB of H and H has previous CY as the MSB
        MOV H,A ;store rotated H
        MOV A,L
        RAR ;rotate L right so now CY has the LSB of L and L has as MSB the LSB of H.
```

```
        MOV L,A ;store rotated L
        DCR B ;decrease rotation counter
        MOV A,B
        JMP LOOP ;repeat
        FINISH:

        POP B ;restore B
ENDM
```

Δεν χρησιμοποιήσαμε PUSH PSW γιατί εάν το κάναμε θα αλλοιωνόταν το αποτέλεσμα καθώς το CY είναι μέρος του αποτελέσματος, αλλά είναι και μέρος του flag register που αποθηκεύεται με το PUSH PSW.

Άσκηση 4

- Αρχικά έχουμε ότι PC = 0800H και SP = 3000H και καλείται η CALL, από την οποία υπορουτίνα πρέπει να επιστρέψουμε και άρα αποθηκεύεται στον SP το PC+3=0803H, που είναι η διεύθυνση της επόμενης εντολής, καθώς η CALL είναι 3 bytes και άρα θα έχουμε και SP = SP-2 = 2FFEH (για την ακρίβεια SP[2FFFH] = 08H και SP[2FFEH] = 03H).
- Στη συνέχεια το PC γίνεται 0880H
- Τώρα συμβαίνει η διακοπή RST 7.5 και άρα πρέπει να αποθηκεύσουμε το PC(=0880H) στον SP. Οπότε έχουμε SP = SP-2 = 2FFCH και SP[2FFDH] = 08H και SP[2FFCH] = 80H. Το PC παίρνει τη τιμή της διεύθυνσης της ρουτίνας εξυπηρέτησης της διακοπής RST 7.5, δηλαδή, με βάση της διαφάνειες, γίνεται PC = 003CH.
- Τέλος, αφού τελειώσει η ρουτίνα εξυπηρέτησης της συγκεκριμένης διακοπής το PC θα επανέλθει στη τιμή 0880 καθώς θα πρέπει να συνεχιστεί εκτέλεση του προγράμματος. Επιπροσθέτως η στοίβα θα μειώσει κατά 2 το μέγεθός της (SP = SP+2 = 2FFEH).

Άσκηση 5

```
RST6.5:


JMP SERV


START:


MVI D,40H ;step counter = 64 decimal
MVI H,00H
MVI L,00H ;HL is used for sum of numbers
MVI B,00H
MVI A,0DH ;mask to allow interrupt RST6.5
SIM
```

```
RETURN:


MOV A,D
CPI 00H ;if counter has reached 0 then finish
JZ FINISH
EI ;else allow interrupt
WAIT:
JMP WAIT ;and wait for interrupt


SERV:


INX SP
INX SP ;reduce stack by 2(we don't need PC so we save space)
DCR D ;decrease step counter
MOV A,D
ANI 01H ;keep the LSB
CPI 00H ;if LSB = 0 then it means we are about to read the 4 MSBs
JZ MSBS
IN 20H ;else read input, which is the 4 LSBs of the number
ANI 0FH ;keep X0 – X3
MOV C,A ;store in C
JMP RETURN ;repeat


MSBS:


IN 20H ;read input
ANI 0FH ;keep X0 - X3

RLC  ;move the bits to the 4 MSBs
RLC
RLC
RLC

ADD C ;and add C so now A has the correct number
MOV C,A
DAD B ;add BC to HL (B is 0 and C has the input number)
JMP RETURN


FINISH:


DI ;the whole proccess is completed so we have to calculate the mean before the whole proccess be
    ;repeated again
```

```
MVI B,05H ;5 right “rotations” of HL


LOOP1:


MOV A,H ;mean
RAR
MOV H,A
MOV A,L
RAR
MOV L,A
DCR B
CPI 00H
JNZ LOOP1

;JMP START ;uncomment if we want to repeat the whole proccess perpetualy (we will wait for the
                ;next interrupt which => repeat proccess)

END
```

Στο LOOP1 υπολογίζουμε το μέσο όρο διαιρώντας με το 32 (5 δεξιές ολισθήσεις). Μία σημαντική παρατήρηση είναι ότι αφού ο μεγαλύτερος αριθμός που μπορεί να παραχθεί από το παραπάνω άθροισμα είναι ο 255*32 = 0001 1111 1110 0000 (με κόκκινο είναι ο L και με πράσινο ο H, αφού το άθροισμα το αποθηκεύουμε στον HL) παρατηρούμε ότι διαιρώντας το με το 32 πρακτικά ο H θα είναι το 00000000 και ο L το 11111111. Άρα παρατηρούμε ότι αφού στη περίπτωση του μεγαλύτερου μέσου όρου όλα τα bis που τον αναπαριστούν (στρογγυλοποιημένο) είναι “μαζεμένα” στον L τότε σε κάθε περίπτωση θα είναι “μαζεμένα” στον L και άρα χρειαζόμαστε μόνο αυτόν για το τελικό αποτέλεσμα. Επιπροσθέτως για την ολίσθηση (δεξιά) του διπλού καταχωτηρή HL χρησιμοποιούμε παρόμοια λογική με την άσκηση 3.α).

β)

Στο ερώτημα αυτό κάνουμε σχεδόν την ίδια διαδικασία με πριν μόνο που τώρα διαβάζουμε την είσοδο από τη θύρα, κοιτάμε το X7 και εάν λάβουμε θετικό μέτωπο τότε διαβάζουμε την είσοδο. Στη συνέχεια περιμένουμε να μας έρθει 0 (WAIT label) και όταν μας έρθει κάνουμε JUMP στην WAIT_FOR_ONE ανιχνεύοντας πάντα έτσι θετικά μέτωπο και όχι απλά άσσους.

```
START:


MVI D,40H ;step counter = 64 decimal
MVI H,00H
MVI L,00H ;HL is used for sum of numbers
MVI B,00H
```

```
WAIT_FOR_ONE:


MOV A,D
CPI 00H ;if counter has reached 0 then finish
JZ FINISH
IN 20H ;else read input port
ANI 80H ;keep the MSB(x7)
CPI 80H
JZ WAIT_FOR_ZERO ;if MSB = 1 then start reading input
JMP WAIT_FOR_ONE


WAIT_FOR_ZERO:


DCR D ;decrease step counter
MOV A,D
ANI 01H ;keep the LSB
CPI 00H ;if LSB = 0 then it means we are about to read the 4 MSBs
JZ MSBS
IN 20H ;else read input, which is the 4 LSBs of the number
ANI 0FH ;keep X0 – X3
MOV C,A ;store in C


WAIT:


IN 20H ;wait for x7 to be 0 in order to continue the proccess
CPI 00H
JZ WAIT_FOR_ZERO
JMP WAIT


MSBS:


IN 20H ;read input
ANI 0FH ;keep X0 - X3

RLC  ;move the bits to the 4 MSBs
RLC
RLC
RLC

ADD C ;and add C so now A has the correct number
MOV C,A
```

```
DAD B ;add BC to HL (B is 0 and C has the input number)
JMP WAIT ;wait for x7 to be 0 in order to continue the proccess


FINISH:



MVI B,05H ;5 right “rotations” of HL


LOOP1:


MOV A,H ;mean
RAR
MOV H,A
MOV A,L
RAR
MOV L,A
DCR B
CPI 00H
JNZ LOOP1

;JMP START ;uncomment if we want to repeat the whole proccess perpetualy (we will wait for the
            ;next interrupt which => repeat proccess)

END
```