**1η Σειρά Ασκήσεων**

Ειρήνη Στρουμπάκου 03121183

Χαράλαμπος Παπαδόπουλος 03120199

1. Το πρόγραμμα σε γλώσσα assembly είναι το εξής:

|  |
| --- |
| START:  MVI B,01H  LDA 2000H  CPI 00H  JZ FIRST  THIRD:  RAR  JC SECOND  INR B  JNZ THIRD    SECOND:  MOV A,B  FIRST:  CMA  STA 3000H  RST 1  END |

Και εάν κάνουμε την μετάφρασή του παίρνουμε πράγματι τον ακόλουθο κώδικα:

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Παρατηρούμε ότι το πρόγραμμα επιτελεί την ακόλουθη λειτουργία:

Αρχικά μέσω της διεύθυνσης 2000Η φορτώνεται στην είσοδο ένας δυαδικός αριθμός. Έπειτα κάνουμε right shift(RAR) τον δυαδικό αριθμό μέχρι το MSB του νέου αριθμού που προκύπτει από τα shifts να γίνει 1. Κατά την διάρκεια αυτών των shifts, αυξάνουμε έναν counter B για να δούμε πόσες μετακινήσεις χρειάστηκαν. Τέλος, φορτώνουμε στην έξοδο το συμπλήρωμα(CMA) του counter B, οπότε λόγω της αρνητικής λογικής των LEDS, τελικά στην έξοδο παίρνουμε τον αριθμό των shifts που χρειάστηκαν.

Διάγραμμα Ροής:

Α=00Η?

(A)<-(Συμπλήρωμα του A)

Μ(3000H)<-(A)

(A)<-(B)

(B)=0?

(B)<-(B)+1

CY=1?

Περιστροφή δεξιά του Α κατά 1 bit

(Β)<-01Η

(Α)<-Μ(2000Η)

**ΝΑΙ**

**ΟΧΙ**

**ΟΧΙ ΟΧΙ**

**ΝΑΙ**

**ΝΑΙ**

Για να αποκτήσει συνεχή λειτουργία το πρόγραμμα, αφαιρούμε την εντολή RST 1 και στη θέση της βάζουμε την εντολή JMP START, όπου με START συμβολίζουμε πλέον την αρχή του κώδικα. Προκύπτει, επομένως ο ακόλουθος κώδικας:

|  |
| --- |
| START:  MVI B,01H  LDA 2000H  CPI 00H  JZ FIRST  THIRD:  RAR  JC SECOND  INR B  JNZ THIRD    SECOND:  MOV A,B  FIRST:  CMA  STA 3000H  JMP START  END |

α να αποκτήσει συνεχή λειτουργία το πρόγραμμα, αφαιρούμε την εντολή RST 1 και

2)TART συμβολίζουμε πλέον τ

|  |
| --- |
| START:  IN 10H  LXI H,0001H ;Rightmost led  MOV A,L  CMA  STA 3000H  MVI A,01H ;Turning on rightmost led  LOOP1:  LXI B,FFFFH ;Delay  MOV E,A ;Saving the position of the last lit led  CALL ENABLE ;Checking if the second last LSB is on  LDA 2000H ;Input  CPI 01H ;Comparing input to 00000001B  JZ DIP\_ON  JNZ DIP\_OFF  ENABLE:  LDA 2000H ;Constantly checking input to see whether enable changed  ANI 02H ;Compare input to enable-bit  JNZ ENABLE  RET ;Return to main program  DIP\_ON:  MOV A,D ;D is our loop counter  ;For 8<= D <15 the LEDs move right.  CPI 0EH  JNC RESET\_COUNTER ;For D = 15 we reset the counter  CPI 07H  JC LEFT ;For 0<= D <7 the LEDs move left.  JNC RIGHT ;For 8<= D <15 the LEDs move right.  DIP\_OFF:  MOV A,E  MVI L,07H  CALL CHECK\_BIT ;With this subroutine we are aiming to get the  ;position of the LED so in case the dip-switch  ;switched back on we can resume our counter from that point.  MOV D,L  JMP LEFT  CHECK\_BIT:  RLC  JC FOUND\_BIT  DCR L  JNZ CHECK\_BIT  FOUND\_BIT:  RET  RESET\_COUNTER:  MVI D,00H  LEFT:  MOV A,E  RLC ;Rotate the LED to the left  MOV E,A  JMP DELAY  RIGHT:  MOV A,E ;Rotate the LED to the right  RRC  MOV E,A  JMP DELAY  DELAY:  DCX B  MOV A,B  ORA C  JNZ DELAY  END:  MOV A,E  INR D ;Increment the Counter  CMA  STA 3000H  CMA  JMP LOOP1  END |

1. Ο κώδικας για την 3η άσκηση μαζί με σχόλια για την λειτουργία του είναι ο εξής:

|  |
| --- |
| START:  LDA 2000H  MVI B,FFH ;Store -1 to B  HUND:  CPI C7H ;Check if greater than 199  JNC LEDS ;If yes go to loop LEDS  CPI 64H ;Check if greater than 100  JC DECA ;If not go to DECA  SUI 64H ;If yes subtract 100  DECA:  INR B ;Increase the counter of tens  SUI 0AH ;Subtract 10  JNC DECA ;If positive again loop  ADI 0AH ;If negative add 10  MOV C,A ;Store the ones in C  SUM:  MOV A,B ;Store the tens in A  RLC  RLC  RLC  RLC ;Create 2^4  ADD C ;Make of the final number with tens and ones  CMA  STA 3000H  JMP RESTART  LEDS:  MVI A,00H ;Turn on all LEDs  STA 3000H  JMP RESTART  RESTART:  RST 1  JMP START  END |

1. Για την άσκηση αυτή σχεδιάζουμε τα διαγράμματα για τις ακόλουθες συναρτήσεις:

f1(x) = 15000 + 20x

f2(x) = 7000 + 60x

f3(x) = 47000 + 4x

f4(x) = 61000 + 2x

με x τα τεμάχια

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα

Για να βρω τις περιοχές χαμηλότερου κόστους για κάθε τεχνολογία αρχικά βρίσκω τα σημεία τομής των τεχνολογιών:

1. f1(x)=f2(x) => x=200
2. f1(x)=f3(x) => x=2000
3. f1(x)=f4(x) => x=2555
4. f2(x)=f3(x) => x=714
5. f2(x)=f4(x) => x=931
6. f3(x)=f4(x) => x=7000

Οπότε με βάση τις τιμές αυτές και τα διαγράμματα παρατηρούμε ότι:

Αρχικά η 2 έχει χαμηλότερο κόστος μέχρι να συναντήσει την 1 στο σημείο x=200. Έπειτα, έχει η 1 μέχρι να συναντήσει την 3 στο σημείο x=2000. Έπειτα, έχει η 3 μέχρι να συναντήσει την 4 στο σημείο x=7000. Τέλος, μετά από το σημείο x=7000 χαμηλότερο κόστος έχει η 4.

Τελικά:

1η: 200<x<2000

2η: 0<x<200

3η: 2000<x<7000

4η: x>7000

Για να εξαφανιστεί η 1η τεχνολογία πρέπει f2’(x)<f1(x) =>

7000+(λ+10)x<15000+20x => λ<

Αυτό όμως πρέπει να ισχύει για κάθε x, οπότε και για x->άπειρο. Για

x->άπειρο προκύπτει λ< 10( Βάζω ισότητα, γιατί δεν γίνεται όντως να πάρω άπειρα κομμάτια, οπότε παίρνω όριο στο άπειρο. Επομένως, για οποιαδήποτε τιμή του νέου κόστους μέχρι 10€ η 1η τεχνολογία εξαφανίζεται.

2)@ακόλουθος κώδικας: