**Συστήματα Μικροϋπολογιστών**

**3η Σειρά Ασκήσεων**

**Τσιλιμιγκουνάκης Μιχάλης (el19001)**

**Στεφανής Παναγιώτης (el19096)**

**Ασκήσεις Προσομοίωσης:**

**1η Άσκηση**

Ζητούμενο:

Το πρόγραμμα, μέσω διακοπής RST6.5 ελέγχει τα φώτα ενός χώρου. Όταν λάβει διακοπή αναβοσβήνει τα LED για 45 δευτερόλεπτα με διάρκεια άναψε-σβήσε 0.5 δευτερόλεπτα. Αν λάβει πάλι διακοπή όσο γίνεται αυτό κάνει reset τον timer των 45 δευτερολέπτων. Σε όλη την διάρκεια που αναβοσβήνει τα LED, εμφανίζει στα 2 LSB των 7-segment πόσα δευτερόλεπτα μένουν ακόμα.

Η ροή του προγράμματος έχει ως εξής:

Στην αρχή του προγράμματος κάνουμε τις απαραίτητες αρχικοποιήσεις, δηλαδή βγάζουμε την προστασία της μνήμης, αρχικοποιούμε την οθόνη με κενούς χαρακτήρες και ενεργοποιούμε τις διακοπές. Μετά υπάρχει ένας ατέρμονος βρόγχος που απλά λουπάρει στο εαυτό του (περιμένει δηλαδή εκεί να έρθει διακοπή). Όταν έρθει διακοπή, πάμε στην ρουτίνα εξυπηρέτησης της, η οποία αρχικοποιεί το delay και το πόσα δευτερόλεπτα θα αναβοσβήσουν τα LEDs και ξαναενεργοποιεί τις διακοπές ώστε να μπορεί να έρθει πάλι διακοπή και θα ξαναξεκινήσει η όλη διαδικασία από την αρχή. Στην συνέχεια ανάβει και σβήνει τα LED, προσθέτοντας και μισό δευτερόλεπτο καθυστέρηση ενδιάμεσα και καλεί την συνάρτηση “COUNTDONW” η οποία φορτώνει στην μνήμη τον μετρητή πόσων δευτερολέπτων απομένουν και κάνει τις κατάλληλες μετατροπές ώστε να εμφανιστεί στα 7 segment displays. Στην συνέχεια καλείται η ρουτίνα “SHOW” η οποία εμφανίζει με τις κατάλληλες κλήσεις τα εναπομείναντα δευτερόλεπτα στην οθόνη. Επαναλαμβάνουμε την όλη παραπάνω διαδικασία 45 φορές.

Ο κώδικας (επόμενη σελίδα)

**Text

Description automatically generated**A screenshot of a computer

Description automatically generated with medium confidence

**2η Άσκηση**

Ζητούμενο:

Το πρόγραμμα, όταν προκαλείται διακοπή RST 6.5, διαβάζει 2 δεκαεξαδικά ψηφία από το πληκτρολόγιο και τα απεικονίζει στα 2 μεσαία 7 segment display. Στην συνέχεια συγκρίνει τον αριθμό αυτό με 3 κατώφλια και ανάλογα το διάστημα στο οποίο ανήκει ο αριθμός ανάβει το πρώτο, δεύτερο, τρίτο ή τέταρτο LED (από δεξιά, δηλαδή τα LSB).

Η ροή του προγράμματος έχει ως εξής:

Ξεκινάμε με τις αρχικοποιήσεις (μνήμη, κατώφλια, αρχικοποίηση της οθόνης με κενούς χαρακτήρες). Στην συνέχεια μπαίνουμε σε μια ατέρμονη λούπα, η οποία αποτελεί το κύριο πρόγραμμα (label “WAIT”). Εκεί εκτελείται κώδικας στο μικροεπεξεργαστή περιμένοντας να έρθει κάποιο interrupt. Η ρουτίνα εξυπηρέτησης του interrupt αρχικά διαβάζει από το πληκτρολόγιο και τους 2 χαρακτήρες και τους αποθηκεύει προσωρινά στον Α. Στην συνέχεια κάνει συγκρίσεις για να δει σε ποιο διάστημα ανήκει ο αριθμός και ανάβει το αντίστοιχο LED. Στην συνέχεια επιστρέφει από το την ρουτίνα διαχείρισης της διακοπής στο κύριο πρόγραμμα.

Ο κώδικας: (επόμενη σελίδα)

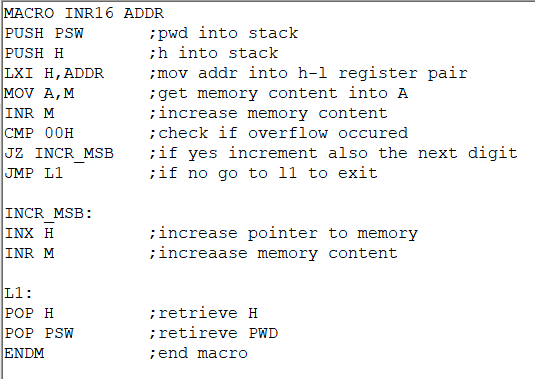
Graphical user interface, text

Description automatically generatedText

Description automatically generated

**Θεωρητικές Ασκήσεις:**

**3η Άσκηση**

1. ****Φτιάχνουμε μια μακροεντολή που να αυξάνει κατά 1, ένα δεκαεξάμπιτο αριθμό αποθηκευμένο σε 2 διαδοχικές θέσεις στην μνήμη.  
     
   Το μάκρο αυξάνει κατά 1 την τιμή της πρώτης θέσης στην μνήμη και ελέγχει αν συνέβη overflow. Αν συνέβη, αυξάνει και την τιμή στην επόμενη θέση μνήμης κατά 1.
2. Text

   Description automatically generated Φτιάχνουμε μια μακροεντολή η οποία γεμίζει ένα τμήμα της μνήμης (μεγέθους Κ) με τους αριθμούς Κ, Κ-1, Κ-2, …, 1. Το μέγεθος είναι μεταξύ 1 και 256.  
     
     
   Το μάκρο αποθηκεύει προσωρινά στον Β την τιμή Κ και τρέχει έναν βρόχο που αποθηκεύει την τιμή του Κ όπου δείχνει ο H-L, αυξάνει κατά 1 τον pointer στην μνήμη, μειώνει κατά 1 το Β και επαναλαμβάνει όσο το Β δεν είναι 0.
3. Graphical user interface, text

   Description automatically generatedΦτιάχνουμε μια μακροεντολή η οποία περιστρέφει κατά μία θέση αριστερά τα περιεχόμενα ενός «εικονικού» καταχωρτητή 17 bit που αποτελείται από τους καταχωρητές CY-Q-R.

Εδώ φορτώνουμε τον R στο Α και τον ολισθαίνουμε αριστερά 1 θέση. Άρα έχει πάρει το CY για LSB και το MSB του το έχει δώσει στο CY. Γυρνάμε τον Α πίσω στον R. Κάνουμε την ίδια διαδικασία για το Q και τώρα έχει πάρει για LSB του το CY που είχε δώσει ο R και έχει δώσει το δικό του MSB στο CY.

**4η Άσκηση**

Η RST6.5 είναι hardware διακοπή και θα πάρει προτεραιότητα έναντι της εντολής JMP 2200H, εφόσον οι διακοπές είναι ενεργοποιημένες και η μάσκα διακοπών επιτρέπει την συγκεκριμένη διακοπή.

* Αρχικά το PC είναι ίσο με 2000Η και το SP ίσο με 3000Η
* Πραγματοποιείται η εντολή JMP 2200H άρα το PC γίνεται ίσο με 2200H ενώ το stack pointer δεν αλλάζει αφού έχουμε ένα απλό jump και όχι κάποια κλήση για να μας χρειάζεται η διεύθυνση επιστροφής
* Έρχεται η διακοπή RST6.5 και η τρέχουσα τιμή του PC αποθηκεύεται στην στοίβα. Πιο συγκεκριμένα ο stack pointer γίνεται 2FFEH, δηλαδή αυξάνεται κατά δύο θέσεις, στις οποίες μπαίνει η high και low τιμή του PC. Άρα (2FFF) = 22H και (2FFE) = 00H.
* Ο PC παίρνει την διεύθυνση εξυπηρέτησης αυτής της διακοπής, δηλαδή την τιμή 0034H.
* Μόλις τελειώσει η ρουτίνα εξυπηρέτησης ο PC θα ανακτήσει την προηγούμενη τιμή του από το stack, δηλαδή PC = 2200H
* Ο SP θα αυξηθεί κατά 2 ώστε να αποδεσμευτεί ο χώρος που είχε δεσμευτεί για να αποθηκευτεί προσωρινά το PC
* Η εκτέλεση θα συνεχίσει από την θέση 2200Η

**5η Άσκηση**

1. Στην άσκηση αυτή θα γράψουμε ένα πρόγραμμα το οποίο θα λαμβάνει 16 δεδομένα των 8 μπιτ από μια συσκευή, από την θύρα 20Η. Θα τα λαμβάνει σε κομμάτια των 4 μπιτ και θα ενημερώνεται ότι αυτά είναι έτοιμα να τα παραλάβει μέσω μια διακοπής RST5.5. Αφού λάβει όλα τα δεδομένα, πρέπει να υπολογίσουμε τον μέσο όρο τους.

Η ροή του προγράμματος:

Κάνουμε τις αρχικοποιήσεις (αριθμό μνημάτων που περιμένουμε να λάβουμε, ενεργοποίηση διακοπών) και μετά μπαίνουμε σε ένα κυρίως πρόγραμμα που απλά τσεκάρουμε ατέρμονα αν έχουμε λάβει και τα 32 κομμάτια πληροφορίας. Όταν έρθει διακοπή πάμε ασύγχρονα στην ρουτίνα εξυπηρέτησης της, μέσα στην οποία τσεκάρουμε αν περιμένουμε τα 4 LSB ή τα 4 MSB ενός από τους 16 8μπιτους αριθμούς. Ανάλογα την περίπτωση διαχειριζόμαστε τα 4 ληφθέντα μπιτ διαφορετικά. Αν είναι τα MSB (θεωρούμε πως πρώτα έρχονται τα LSB) πάμε και αθροίζουμε τον ολόκληρο πια 8μπιτο αριθμό στο ζεύγος καταχωρητών H-L. Η ρουτίνα εξυπηρέτησης την διακοπής μόλις λάβει τα δεδομένα και τα αποθηκεύσει επιστρέφει στην κύρια συνάρτηση που ελέγχει ατέρμονα αν έχουμε λάβει όλα τα δεδομένα. Μόλις τα λάβουμε υπολογίζει το 4\*(Η-L) το οποίο ισοδυναμεί με 4 shift αριστερά. Αυτό μας επιτρέπει να πάρουμε στον H τα 4 MSB του L και τα 4 LSB του Η. Άρα στον Η, αυτή την στιγμή, έχουμε το αρχικό άθροισμα shifted 4 θέσεις δεξιά, διαιρεμένο δηλαδή με το 16. Άρα έχουμε τον μέσο όρο των 16 αυτών αριθμών.

Text

Description automatically generatedΟ κώδικας:

1. Παραποιούμε τον προηγούμενα κώδικα ώστε ο μικροεπεξεργαστής να ενημερώνεται για έτοιμα δεδομένα με το LSB της θύρας 20Η (ενεργό χαμηλά).

Η ροή του προγράμματος:

Η αλλαγή στον κώδικα είναι ότι ο κορμός του προγράμματος ελέγχει πότε θα πέσει στο 0 το x0 και όταν γίνει αυτό πάει και τρέχει το ίδιο κομμάτι κώδικα που έτρεχε και στο προηγούμενο ερώτημα η εξυπηρέτηση της διακοπής. Απλά τώρα δεν είναι ρουτίνα αλλά ένα απλό branch. Στο τέλος του διαβάσματος των 4 ληφθέντων μπιτ ελέγχουμε αν έχουμε λάβει και τους 16 αριθμούς. Αν ναι πάμε να υπολογίσουμε τον μέσο όρο, αλλιώς πάμε να περιμένουμε τον επόμενο αριθμό. Αφού επιστρέψουμε στην κεντρική ροή, ελέγχουμε ατέρμονα αν το x0 επανήλθε από 0 στο 1. Όταν συμβεί αυτό πάμε και περιμένουμε να πέσει πάλι στο 0 (πότε δηλαδή θα έρθει το επόμενο πακέτο δεδομένων). Αυτό προϋποθέτει ότι ο μικροεπεξεργαστής διαβάζει και διαχειρίζεται το πακέτο δεδομένων και επιστρέφει στο κεντρικό πρόγραμμα γρηγορότερα από τον ρυθμό που στέλνει δεδομένα η εξωτερική συσκευή. Ο υπολογισμός του μέσο όρου γίνεται με τον ίδιο τρόπο με το α) ερώτημα.

Graphical user interface, text

Description automatically generated Ο κώδικας: