Συστήματα Μικροϋπολογιστών

Αναφορά για 1ο εργαστήριο

Ιωάννης-Παναγιώτης Μπουντουρίδης 8872 - Κωνσταντίνος Χατζηαντωνίου 8941

9/12/2018

Τμήμα 1

I)

Για την αποθήκευση των AEM στη μνήμη του μικροελεγκτή χρησιμοποιήσαμε την εντολή *db*. Η εντολή *db* αποθηκεύει σταθερές binary ακολουθιών στην μνήμη προγραμματος. Οι εντολές *.db "8872"* και *.db "8941"* φέρουν το επιθυμητό αποτέλεσμα.

II) Για την σύγκριση των AEM πρόκειται να εξετάσουμε ένα ένα τα ψηφία για κάθε AEM αντίστοιχα. Η σύγκριση αυτή ξεκινάει απο το πρώτο ψηφίο αριστερά του κάθε AEM και συνεχίζει μέχρι το τέλος.Αρχικά φορτώνουμε την διευθυνση του AEM1 που αποθηκεύσαμε μέσω της .db στον καταχωρητή Z. Όμοια πράτουμε για την διεύθυνση του AEM2 την οποία εκχωρούμε στον καταχωρητή Y. Στην συνέχεια θέτουμε στον καταχωρητή temp1 την τιμή 5. Ο καταχωρητής temp1 θα είναι μια βοηθητική μεταβλητή του νοητού βρόχου που θα δημιουργήσουμε.Η temp1 στην συνέχεια θα μειώνεται κατα 1 κάθε φορά που γίνεται μια σύγκριση μεταξύ των ψηφίων των AEM. Στην περίπτωση που μηδενιστεί η temp1 γίνεται έξοδος απο τον νοητό βρόγχο που δημιουργήσαμε. Ο βρόγχος (loop) εκχωρεί το πρώτο ψηφίο του AEM1 στον digit_aem1 και στην συνέχεια αυξάνει την διεύθυνση του Z κατά 1. Όμοια εκχωρεί το πρώτο ψηφίο του AEM2 στον καταχωρητή digit_aem2 και αυξάνει την διευθυνση κατα 1.Συνολικά θα γίνουν το πολύ 4 συγκρίσεις, αν βρεθεί ενα ψηφίο που να είναι μικρότερο ή μεγαλύτερο απ το άλλο τότε η συγκριση λαμβάνει το τέλος της. Στην περιπτωσή μας AEM1 = 8872, AEM2 = 8941 το πρόγραμμα θα συγκρίνει το πρώτο αριστερό ψηφίο AEM1: 8 με το πρώτο αριστερό ψηφίο του AEM2: 8 λόγω ισότητας η σύγκριση θα συνεχιστεί για το δεύτερο ψηφίο του AEM1: 8 με το δεύτερο ψηφίο του AEM2: 9 και θα προκύψει:

$$AEM1 < AEM2$$

Ο καταχωρητής largeaem είναι ενα flag που δημιουργήσαμε ωστε να ξέρουμε ποιό απο τα δύο AEM είναι το μεγαλύτερο μετά την σύγκριση. Επειδή το AEM1 είναι μικρότερο του AEM2 καλούμαστε να ανάψουμε τα LED 0-7 ώστε τα LED 7-4 να αντιπροσωπεύουν το τρίτο ψηφιο του μικρότερου AEM και τα LED 3-0 να αντιπροσωπεύουν το τέταρτο ψηφίο του μικρότερου AEM αντίστοιχα. Το ζητούμενο ικανοποιήται στο κομμάτι του κώδικα *SUCCESS*. Αρχικά θέτουμε την διεύθυνση του AEM1 στον καταχωρητή Z και λαμβάνουμε το τρίτο ψηφίο του. Μέσω της lpm το εκχωρούμε στον καταχωρητή digit_aem1 και αυξάνουμε την διεύθυνση του Z κατά 1. Καθώς το digit_aem1 ειναι σε μορφή ASCII αφαιρούμε την τιμή που πήραμε με το 30. Τώρα ο καταχωρητής digit_aem1 περιέχει την επιθυμητή πληροφορία (στην περίπτωση μας τον αριθμό 7) σε μορφή binary. Όμοια εργαζόμαστε για να λάβουμε την τιμή του digit_aem2. Για να χωρέσουμε τα digit_aem1 και digit_aem2 σε μια έξοδο κάνουμε αριστερή ολίσθιση τον καταχωρητή digit_aem1 4 φορές και τους προσθέτουμε. Δίνουμε την έξοδο μέσω του καταχωρητή δεδομένων θύρας portb και στην συνέχεια καλούμε την υπορουτίνα PAUSE_LOOP ώστε να γίνει αντιληπτή η έξοδος στον χρήστη για 10 δευτερόλεπτα.

III) Τέλος καλούμαστε να ενεργοποιήσουμε τα LED 0-1 σύμφωνα με τον πίνακα που μας δίνεται στην εκφώνηση. Το παραπάνω ζητούμενο ικανοποιήται απο το κομμάτι του κώδικα *FUN_EXC2*. Φορτώνουμε την διεύθυνση του AEM1 στον καταχωρητή Y και την διεύθυνση του AEM2 στον καταχωρητή Z. Μέσω της εντολής adiw αυξάνουμε την διεύθυνση του καταχωρητή Y και Z αντίστοιχα ωστε η τιμή που θα κανουν point να ειναι το τέταρτο ψηφίο. Μέσω της lpm εκχωρούμε στον καταχωρητή digit_aem2 την τιμή της διεύθυνσης του Z και αφαιρούμε απο αυτή

την τιμή 30 γιατί είναι μορφής ASCII. Στην συνέχεια μεσω της movw φορτώνουμε την διεύθυνση του AEM1 στον καταχωρητή Z. Όμοια μέσω της lpm εκχωρούμε στον καταχωρητή digit_aem2 την τιμή της διεύθυνσης του Z και αφαιρούμε απο αυτή την τιμή 30 γιατί είναι μορφής ASCII. Αφού έχουμε πλέον στην διαθεσή μας τις επιθυμητές τιμές στους καταχωρητές digit_aem1 και digit_aem2 μπορούμε να καλέσουμε την υπορουτίνα *FUN_ODDEVEN*. Η λειτουργία της υπορουτίνας *FUN_ODDEVEN* είναι αναδρομική και βρίσκει αν το ψηφίο του κάθε digit_aem είναι αρτιο ή περριτό με τον εξής τρόπο: έχει ως όρισμα τον καταχωρητή arg και ελέγχει αν αυτός ισούται με 0 ή 1. Αν όχι αφαιρεί απο αυτόν το 2 και καλεί τον εαυτό της. Καλούμε την παραπάνω υπορουτίνα δύο φορές με όρισμα το digit_aem2 και digit_aem1 αντίστοιχα, κάνουμε δεξιά ολίσθηση στον καταχωρητή digit_aem2, τον προσθέτουμε με τον digit_aem1 και δίνουμε την επιθυμητή έξοδο μέσω του καταχωρητή δεδομένων θύρας portb.

Τμήμα 2

Για την υλοποίηση του τμήματος δύο χρησιμοποιήσαμε την εντολή sbis ωστε να ελέγχουμε έαν τα κουμπιά sw0,sw1,sw2,sw3,sw7 έχουν πατηθεί. Με την εντολή sbis βλέπουμε αν ενα συγκεκριμένο bit στο PIND είναι 1. Στην περίπτωση που είναι κανουμε skip την επόμενη γραμμή κώδικα. Αυτή η κατάσταση μας δείχνει οτι το κουμπί δεν έχει πατηθεί ακόμη. Αντίστοιχα η εντολή sbic PIND,0x00 παραβλέπει την επόμενη εντολή κώδικα αν το κουμπί sw0 έχει πατηθεί. Βλέπουμε λοιπόν πως οι εντολές sbis και sbic είναι χρήσιμα εργαλεία ωστε να ελέγχουμε πότε ο χρήστης πατάει και πότε απελευθερώνει το κουμπιά sw 0-7.

Αν για παράδειγμα ο χρήστης πατήσει το κουμπί sw0 τότε λόγω της sbis PIND,0x00 η επόμενη εντολή, rjmp RELEASE_LOOP_0 θα εκτελεστεί και θα ξεκινήσει η λειτουργία του sw0 μόλις ο χρήστης αφήσει το κουμπί. Η DO_BTN_0 φορτώνει τις διευθύνσεις των AEM1 και AEM2 στους καταχωρητές Z και Y. και θέτει στον καταχωρηη temp1 την τιμή 1. Αν το AEM1 είναι μεγαλύτερο αφήνει τον AEM1 στον Z και παίρνει τις τιμές απο τα δύο τελευτάια ψηφία του. Στην συνέχεια εμφανίζει τα δυο τελευταία ψηφία του AEM1 στα LED 7-0 με παρόμοιο τρόπο που έγινε στην υλοποίηση του τμήματος 1. Τα κουμπιά sw1,sw2,sw3 λειτουργούν με παρόμοιο τρόπο οπως περιγράφτηκε παραπάνω για την λειτουργία του sw0.

Τέλος μέσω πάλι της sbis γίνεται έλεγχος αν έχουμε πατήσει το κουμπί sw7. Μόλις πατηθεί το sw7 κάνουμε jump στην REALEASE_LOOP_7 η οποία ελέγχει πότε ο χρήστης απελευθερώνει το sw7. Όταν το sw7 απελευθερωθεί τότε εμφανίζεται μόνιμα το αποτέλεσμα του τμήματος 1 απο το μέρος 3 στα LED 0-1.

Δυσκολίες και περαιτέρω βελτίσωση

Γενικά δεν αντιμετωπίσαμε κάποια μεγάλη δυσκολία στην οποία να ξοδέψαμε περισσότερο χρόνο από ότι περιμέναμε. Ίσως το πιο δύσκολο και ενδιαφέρον ήταν να βρούμε τρόπο να χρησιμοποιούμε διαφορετίκο AEM (ανάλογα ποιο είναι μεγαλύτερο) χωρίς branch αλλά με skip.
Ο κωδίκας δουλεύει ακόμα και αν το AEM1 είναι μεγαλύτερο, οπότε με βάση αυτά που ζητήθηκαν δεν είναι απαραίτητη κάποια βελτίωση. Ίσως θα μπορούσαμε να βάλουμε έναν έλεγχο, η τιμή κάθε byte που είναι αποθηκευμένα τα ψηφία να είναι ανάμεσα στο 30-39 (ASCII) ή και να ελέγχουμε αν μας έδωσε τα αεμ σε ascii ή κατευθείαν τον αριθμό. Όμως στην άσκηση εμείς ελέγχαμε την είσοδο.