Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εθνικό Μετσόβιο Πολυτεχνείο

Εργαστήριο Μικροϋπολογιστών

Έβδομη εργαστηριακή άσκηση

Ομάδα Δ07
Παπαλεξανδράκης Εμμανουήλ (Α.Μ.: 03114203)
Παπασκαρλάτος Αλέξανδρος (Α.Μ.: 03111097)

Ημερομηνία Υποβολής Αναφοράς: 16 Δεκεμβρίου 2018

Θα παρουσιάσουμε μια σύντομη και ουσιαστική ανάλυση των προγραμμάτων που κατασκευάσαμε.
Τα προγράμματα αυτά καθ' αυτά είναι σε ξεχωριστά αρχεία.
Κάποιες λεπτομέρειες θα παραλειφθούν, αλλά εάν ο αναγνώστης ενδιαφέρεται, μπορεί να ανατρέξει στον κώδικα και στα σχόλια αυτού, όπου έχουμε μια πιο αναλυτική παρουσίαση.

Σημειώνουμε δε πως οι ρουτίνες που δίνονται στο pdf της εκφώνησης θεωρούνται γνωστές και δε θα τις αναλύσουμε περαιτέρω.

Αυτή η σειρά ασκήσεων αφορά τη μονάδα USART του avr και την ασύγχρονη επικοινωνία μέσω αυτής.
Επίσης, ασχολούμαστε και με το μετατροπέα από αναλογική σε ψηφιακή μορφή και τη μέτρηση τάσης μέσω αυτού.

Άσκηση 7.1i

Κατασκευάζουμε το πρόγραμμα αποστολής συμβολοσειράς που ζητείται σε assembly.

Καταρχάς, δημιουργούμε ένα string στην αρχή του code segment μέσω της ψευδοεντολής .db η οποία ορίζει ένα byte για τον κάθε χαρακτήρα του string μας. Ως τελευταίο χαρακτήρα του string θέτουμε το '\0'.
Έπειτα, κάνουμε τις κατάλληλες αρχικοποιήσεις.

Στη συνέχεια, στη main έχουμε ένα loop.
Σε κάθε iteration, παίρνουμε ένα χαρακτήρα από το string μέσω του ζεύγους καταχωρητών Z και της εντολής lpm. Αυξάνουμε την τιμή του Z, ώστε να δείχνει πια στον επόμενο χαρακτήρα για το επόμενο iteration.

Εξετάζουμε το χαρακτήρα που πήραμε.
Εάν ο χαρακτήρας είναι το '\0' δηλαδή έχει ascii τιμή 0x00, τότε τερματίζουμε το loop.

Διαφορετικά, αν έχουμε οποιοδήποτε άλλο χαρακτήρα, καλούμε την usart_transmit, ώστε να μεταδώσουμε αυτό το byte. Σημειώνουμε πως για δική μας ευκολία, προβάλλουμε το χαρακτήρα επιπλέον και στην οθόνη lcd του avr, μέσω των γνωστών εντολών. Λοιπόν, αφού στείλουμε το χαρακτήρα, περνάμε στο επόμενο iteration, για τον επόμενο χαρακτήρα του string.

Όταν θα τερματίσει το loop, στέλνουμε το χαρακτήρα αλλαγής γραμμής '\n' και ξαναξεκινάμε τη διαδικασία από την αρχή.

Άσκηση 7.1ii

Κατασκευάζουμε το πρόγραμμα λήψης byte που ζητείται σε C.

Καταρχάς, δημιουργούμε δύο πίνακες χαρακτήρων (ουσιαστικά δύο string), το “Read “ και το “Invalid number”.

Δημιουργούμε ένα αέναο loop.
Σε κάθε iteration, περιμένουμε να λάβουμε ένα byte μέσω της usart_receive, την οποία υλοποιήσαμε στη C με λογική αντίστοιχη της assembly.
Άπαξ και λάβουμε ένα byte, ελέγχουμε αν πρόκειται για χαρακτήρα x με ‘0’ ≤ x ≤ ‘8’.

Εάν έχουμε μη έγκυρο χαρακτήρα, αποστέλλουμε το “Invalid number” χρησιμοποιώντας τη συνάρτηση usart_transmit (η οποία στέλνει ένα byte τη φορά) και ένα for loop (για να στείλουμε όλους τους χαρακτήρες με τη σειρά). Επιπλέον, για δική μας ευκολία, το προβάλλουμε και στην οθόνη lcd του avr.

Εάν ο χαρακτήρας είναι πράγματι ψηφίο από το ‘0’ εώς το ‘8’, τότε καταρχάς στέλνουμε το string “Read “ με τον ίδιο τρόπο με πριν. Στέλνουμε και το χαρακτήρα που λάβαμε. Επιπλέον, για δική μας ευκολία, το προβάλλουμε και στην οθόνη lcd του avr.
Έπειτα, προκειμένου να ανάψουμε το x-οστο led της θύρας C, θέτουμε την C ως έξοδο και βγάζουμε στην PortC τον αριθμό (0b1 00000000 >> (9-x)). Δηλαδή, ο αριθμός θα έχει άσσο μόνο στο x-οστό bit (θεωρώντας πως το bit 0 είναι το πρώτο).

Τέλος, στέλνουμε το χαρακτήρα αλλαγής γραμμής ‘\n’ και εκκινούμε τη διαδικασία από την αρχή.

Άσκηση 7.2i

Κατασκευάζουμε το πρόγραμμα μέτρησης τάσης μέσω διακοπής που ζητείται σε assembly.

Καταρχάς, κάνουμε τις κατάλληλες αρχικοποιήσεις. Επιπλέον, ενεργοποιούμε τη διακοπή για το μετατροπέα ADC καθώς και, προφανώς, τις καθολικές διακοπές.

Η main μας αποτελείται από ένα αέναο loop.

Σε κάθε iteration, αυξάνουμε ένα μετρητή, προβάλλουμε στα led και περιμένουμε 100 msec μέσω της γνωστής ρουτίνας χρονοκαθυστέρησης.

Όμως, πιο σημαντικά, στην αρχή του κάθε iteration ξεκινάμε μία μέτρηση τάσης, θέτοντας 1 στο bit ADSC του καταχωρητή ADCSRA.

Όταν ολοκληρώνεται η μέτρηση, αιτείται διακοπή και μεταφερόμαστε στη ρουτίνα εξυπηρέτησης της διακοπής.

Εδώ, παίρνουμε την τιμή ADC. Προκειμένου να τη μετατρέψουμε στη σωστή τιμή τάσης, εφαρμόζουμε τη σχέση V_{in} = ADC * V_{ref} / 1024.
Όμως, καθώς επιθυμούμε να κρατήσουμε και το πρώτο κλασματικό ψηφίο, πολλαπλασιάζουμε την τιμή επί 10. Τότε, το τελευταίο dec ψηφίο είναι στην πραγματικότητα το κλασματικό μέρος, και το προτελευταίο είναι το ακέραιο.

Έτσι, με V_{ref} = 5, καταρχάς πολλαπλασιάζουμε το ADC επί 50.
Καθώς δεν υπάρχει απλή μέθοδος για να πολλαπλασιάσουμε ζεύγος καταχωρητών, αντ' αυτού το προσθέτουμε 50 φορές στον εαυτό του μέσω της add και της adc.

Στη συνέχεια, για να κάνουμε διαίρεση με το 1024, το κάνουμε shift 10 φορές δεξιά ως εξής:
Κάνουμε shift το low. Εξετάζουμε το lsb του high. Avv είναι 1, θέτουμε 1 το msb του low. Κάνουμε shift το high.

Έπειτα, μετατρέπουμε σε bcd μορφή με τη γνωστή διαδικασία, δηλαδή αφαιρώντας διαδοχικά εκατοντάδες κι έπειτα δεκάδες, προκειμένου να υπολογίσουμε το πλήθος των εκατοντάδων, των δεκάδων και των μονάδων. Σημειώνουμε πως οι εκατοντάδες είναι προφανώς 0 εδώ.

Το πλήθος των δεκάδων είναι το ακέραιο μέρος της τάσης, οπότε και το στέλνουμε στη USART. Έπειτα, στέλνουμε το ‘,’. Τέλος, στέλνουμε το πλήθος των μονάδων το οποίο αντιστοιχεί στο κλασματικό μέρος της τάσης.
Επιπλέον, για δική μας ευκολία, τα προβάλλουμε και στην οθόνη lcd του avr.

Άσκηση 7.2i

Κατασκευάζουμε το πρόγραμμα μέτρησης τάσης μέσω polling που ζητείται σε C.

Καταρχάς, έχουμε τις κατάλληλες αρχικοποιήσεις. Δεν ενεργοποιούμε τη διακοπή για ADC, ούτε τις καθολικές διακοπές.

Εκκινούμε μια μέτρηση τάσης γράφοντας 1 στο bit ADSC του ADCSRA.

Έπειτα, έχουμε ένα αέναο loop.
Διαβάζουμε το bit ADSC του ADCSRA. Όσο είναι 1, περιμένουμε ξαναδιαβάζοντας συνεχώς.

Όταν θα γίνει 0, αυτό σημαίνει πως ολοκληρώθηκε η μέτρηση.
Συνεπώς, λαμβάνουμε την τιμή ADC, πολλαπλασιάζουμε επί 50 και διαιρούμε με το 1024.
Στέλνουμε το προτελευταίο dec ψηφίο, στέλνουμε το ‘,’ και τέλος το τελευταίο ψηφίο.
Επιπλέον, για δική μας ευκολία, τα προβάλλουμε και στην οθόνη lcd του avr.
Ο λόγος που ακολουθούμε αυτή τη διαδικασία αναλύθηκε στο προηγούμενο ερώτημα.
Φυσικά, εδώ η πράξεις γίνονται απλούστερα καθώς η C είναι υψηλότερου επιπέδου γλώσσα και έχει εύκολες εντολές για πολλαπλασιασμό, διαίρεση και modulo (και με 16-bit αριθμούς).

Τέλος, στέλνουμε το χαρακτήρα αλλαγής γραμμής ‘n’, γράφουμε 1 στο bit ADSC του ADCSRA ώστε να ξεκινήσουμε νέα μέτρηση τάσης και περνάμε στο επόμενο iteration.