ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΜΜΥ

ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ & ΥΛΙΚΟΥ

ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ ΓΙΑ ΤΟ ΜΑΘΗΜΑ:

ΗΡΥ 411 – ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ

ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2020

Καθ. Α. Δόλλας

Εργαστήριο 4

ΚΩΔΙΚΑΣ C ΚΑΙ ASSEMBLY ΣΤΟΝ ATMEL AVR

ΕΚΔΟΣΗ : 1.0

Προθεσμία: Πέμπτη 5 Νοεμβρίου 2020, έως τα μεσάνυχτα
Ηλεκτρονική υποβολή στο Webcourses

Όλα τα Εργαστήρια είναι ΑΤΟΜΙΚΑ και όχι κατά ομάδες

Σκοπός – Βήματα

Σκοπός του εργαστηρίου είναι να αρχίσουμε να μπαίνουμε στο πνεύμα του να γράφουμε κώδικα σε C αλλά έχοντας υπόψη την χρήση των πόρων του AVR, καθώς και την διεπαφή κωδίκων C με Assembly.

Στο εργαστήριο αυτό ο κώδικας δεν θα κάνει κάτι καινούργιο, οι λειτουργικές προδιαγραφές του εργαστηρίου είναι ίδιες με αυτές του Εργαστηρίου 3, αλλά με μία σημαντική αλλαγή: το κυρίως πρόγραμμα θα είναι σε C ενώ οι κώδικες εξυπηρέτησης των Interrupts (σειριακής και timer), και μόνο αυτοί, θα παραμείνουν σε Assembly.

Περιγραφή του Εργαστηρίου

Όπως προαναφέρθηκε, το εργαστήριο είναι λειτουργικά ισοδύναμο με το Εργαστήριο 3, εκτός του ότι οι αρχικοποιήσεις και το main() θα είναι πλέον σε C, ενώ οι Interrupt/Timer Service Routines παραμένουν σε Assembly. Για

να κρατήσουμε το πρόβλημα απλό, για τώρα δεν θα έχετε να λύσετε κάποια από τα προβλήματα που θα δούμε σε μεταγενέστερα εργαστήρια, όπως πέρασμα ορισμάτων σε υπορουτίνες, εκτεταμένη χρήση της στοίβας, κλπ. Θα πρέπει όμως να «καθαρίσετε» τον κώδικά σας όπου χρειάζεται, και να αρχίσετε να κάνετε «χάρτη» της μνήμης και της χρήσης των πόρων του AVR. Θα πρέπει επίσης στην τεκμηρίωση (δηλ. την αναφορά αλλά και τα αντίστοιχα κομμάτια του κώδικα) να υπάρχει καλή παρουσίαση της δομής, με Block Diagrams (και Block Comments), κλπ.

Εκτέλεση του Εργαστηρίου

Αλλαγές στον Κώδικα του Εργαστηρίου 3

Αφού μελετήσετε πως να ενσωματώσετε κώδικα Assembly με κώδικα C (το διαδικαστικό κομμάτι είναι πολύ εύκολο), προτείνουμε το εξής:

1. Δείτε καλά τον κώδικα του Εργαστηρίου 3 (είναι καλή ιδέα να τον εκτυπώσετε κιόλας). Φυσικά κρατήσετε πλήρες Backup του Εργαστηρίου 3, μαζί και με βοηθητικά αρχεία, για να μπορείτε να ανατρέξετε σε αυτό.
2. Κυκλώσετε (ή με κάποιο highlighter σημειώσετε) τα κομμάτια του κώδικα που εκτελούνται για τον TIMER (μαζί και την ανανέωση της οθόνης), καθώς και για την εξυπηρέτηση του (ή των) interrupts της σειριακής θύρας. Αυτά τα κομμάτια παραμένουν αυτούσια (προσοχή όμως – όχι οι αρχικοποιήσεις, μόνο η εξυπηρέτηση των Interrupts). Αν κάποιο πρόβλημα το λύνετε με polling (π.χ. τον έλεγχο ολοκλήρωσης αποστολής) αυτό θα συνεχίσει με τον ίδιο τρόπο, και θα γραφτεί ο αντίστοιχος κώδικας σε C. Σε περίπτωση που ο κώδικας δεν είναι επαρκώς «καθαρός» προτείνουμε να τον «συμμαζέψετε», για παράδειγμα αν μία ρουτίνα εξυπηρέτησης Interrupt βρίσκεται σε δύο διαφορετικά μέρη της μνήμης με κάποιο jump ενδιάμεσα καλό είναι να καθαρογραφτεί ο κώδικας ώστε να είναι ενιαίος όσον αφορά την μνήμη αποθήκευσής του, ακόμη και αν αποτελείται από ρουτίνες. Αν κάνετε τέτοιες αλλαγές προτείνουμε να τις δοκιμάσετε πρώτα στην δομή που έχετε ήδη, δηλαδή με το main() σε Assembly.
3. Σημειώσετε τι γίνεται στο κυρίως πρόγραμμα. Μία γενική προσέγγιση είναι να έχουμε κάτι (σε ψευδοκώδικα) όπως:

```
main(){
   init() ;
   while(1) {}
   }
```

δηλαδή μία αρχικοποίηση όποιων πόρων χρειάζεστε, και κατόπιν ένα infinite loop. Αυτό θα σας επιτρέψει να δείτε τι πρέπει να ξαναγράψετε σε C για το εργαστήριο.
4. Αφού κάνετε την παραπάνω διαδικασία, κάνετε (στο χαρτί αρχικά) ένα block diagram των κωδίκων και ένα «χάρτη» της χρήσης της μνήμης, αλλά και των πόρων. Για τώρα, επειδή δεν περνάμε ορίσματα σε υπορουτίνες, κλπ. η θέση και το μέγεθος της στοίβας δεν είναι πρόβλημα, αλλά πλέον πρέπει να ξέρετε ακριβώς που βρίσκονται. Αντίστοιχα ποιες θέσεις της RAM είναι για τον αριθμό που δείχνουμε, ποιες θέσεις της RAM ή της flash είναι για την αποκωδικοποίηση (εκτός αν το κάνετε με κώδικα), κλπ., αντίστοιχα και για καταχωρητές που χρησιμοποιείτε σαν global. Προτείνουμε να κάνετε καλή δουλειά γιατί θα προσθέτουμε συνέχεια πόρους στα εργαστήρια – αν έχετε καλή εποπτεία θα μπορείτε να κάνετε αλλαγές όπου χρειάζεται.

Νέος Κώδικας για το main()

Ο νέος κώδικας που θα γραφτεί αφορά το κυρίως πρόγραμμα. Προφανώς και πρέπει να φροντίσετε να του δώσετε οδηγίες για το φόρτωμα στην μνήμη (.org), και κατόπιν να κάνετε την αρχικοποίηση για TIMER, σειριακή θύρα, θύρες A και C σαν έξοδοι, κλπ. – αν το κάνετε με in-line κώδικα φροντίσετε να έχετε καλή δομή που να αλλάζει εύκολα, αλλιώς μπορεί η init() να καλεί ρουτίνες όπως init_ports_A_C(), init_timer_0, κλπ. Δεν υπάρχει κάποιος περιορισμός, αλλά μία καλή δομή θα σας επιτρέπει να προσθαφαιρείτε κώδικα στο μέλλον με σπονδυλωτό τρόπο.

Ανάλογα με το πως λύνετε προβλήματα, για το Εργ. 4 είναι αποδεκτό να έχετε κάποιους καταχωρητές σαν global_variables, αρκεί να φροντίσετε να έχετε έλεγχο ώστε να είναι οι ίδιοι που χρησιμοποιείτε στα interrupt service routines, και να είναι καλά τεκμηριωμένοι ώστε να μπορείτε να τους αλλάξετε εφόσον χρειαστεί στο μέλλον.

Αφού τρέξετε και προσομοιώσετε τον κώδικα, βγάλετε Assembly ώστε να δείτε πως υλοποιείται από τον compiler ο κώδικας σε C που γράψατε. Μοιάζει με τον δικό σας κώδικα Assembly;

Παρουσίαση / Εξέταση / Αναφορά

Ισχύει ότι και για τα προηγούμενα εργαστήρια. Επειδή κάποιοι/κάποιες μπορεί να εξεταστούν είτε μέχρι το Εργ. 3 στο STK500 είτε μέχρι το Εργ. 4, το αυτονόητο είναι πως από

όποια εργαστηριακή άσκηση και μετά δεν έχει εξεταστεί η ύλη στο εργαστήριο, αυτό θα γίνει στο τέλος του εξαμήνου, σε μία εξέταση στον προσομοιωτή, μέχρι και του τελευταίου εργαστηρίου.

Για την αναφορά ισχύουν όσα έχω γράψει και στις ανακοινώσεις του Webcourses, επί πλέον, από το παρόν εργαστήριο και κατόπιν ισχύει ότι θα πρέπει να υπάρχει σε κάθε εργαστήριο μία σύνοψη των βασικών δομικών στοιχείων του κώδικα (ουσιαστικά block diagrams, με ορίσματα όπου χρειάζεται), χάρτης της μνήμης, και τυχόν χάρτης δεσμευμένων πόρων.

ΠΡΟΣΟΧΗ (τα ξέρετε, αλλά τα ξαναθυμίζουμε)!

1) Η προεργασία να είναι σε ηλεκτρονική μορφή και μαζί με αρχεία με κώδικες που να μπορούμε να εκτελέσουμε. Το αρχείο πρέπει να το υποβάλλετε στο Webcourses.

2) Η έλλειψη προετοιμασίας ή επαρκούς τεκμηρίωσης οδηγεί σε απόρριψη.

3)Η διαπίστωση αντιγραφής σε οποιοδήποτε σκέλος της άσκησης οδηγεί στην απόρριψη όλων των εμπλεκομένων από το σύνολο των εργαστηριακών ασκήσεων, άρα και του μαθήματος. Αυτό γίνεται οποιαδήποτε στιγμή στη διάρκεια του εξαμήνου. Ως αντιγραφή νοείται και μέρος της αναφοράς, π.χ. σχήματα.

ΚΑΛΗ ΕΠΙΤΥΧΙΑ! ☺