ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΜΜΥ

ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ & ΥΛΙΚΟΥ

ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ ΓΙΑ ΤΟ ΜΑΘΗΜΑ:

ΗΡΥ 411 – ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ

ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2020

Καθ. Α. Δόλλας

Εργαστήρια 7 και 8

ΥΠΟΡΟΥΤΙΝΕΣ, ΚΑΙ ΕΝΑΣ ΠΟΛΥ ΑΠΛΟΣ ΔΡΟΜΟΛΟΓΗΤΗΣ

ΕΚΔΟΣΗ : 1.0

Προθεσμία: Τετάρτη 2 Δεκεμβρίου 2020 έως τα μεσάνυχτα Εργ. 7,
Τετάρτη 9 Δεκεμβρίου έως τα μεσάνυχτα για το Εργ. 8

Ηλεκτρονική υποβολή στο Webcourses
Όλα τα Εργαστήρια είναι ΑΤΟΜΙΚΑ και όχι κατά ομάδες

Σκοπός – Βήματα

Σκοπός των εργαστηρίων 7 και 8 είναι να δημιουργήσουμε έναν πάρα πολύ απλό δρομολογητή, κρατώντας την πολυπλοκότητα στο απολύτως ελάχιστο,και «σπάζοντας» την δουλειά σε δύο απλά εργαστήρια για να είναι λογικός ο φόρτος.

Περιγραφή των Εργαστηρίων 7 και 8 (συνολικά)

Στα εργαστήρια αυτά, αντί για το main() να έχει αρχικοποιήσεις και ένα κενό βρόχο (idle loop, infinite loop), θα έχει αρχικοποιήσεις και ένα βρόχο, ο οποίος στην βάση της τιμής μίας μεταβλητής θα τρέχει μία από τρεις διαφορετικές υποροutίνες. Ορίζουμε το timeslice σαν 100msec (για να κάνουμε context switch). Με αυτόν τον τρόπο θα λειτουργεί σαν ένας απλός δρομολογητής. Για να γίνουν λίγο πιο ενδιαφέροντα τα εργαστήρια, το ποιες υποροutίνες από τις τρεις είναι ενεργές και ποιες όχι θα μας έρχεται από την σειριακή θύρα. Τα εργαστήρια αυτά είναι αρκετά απλά γιατί σημαντικά κομμάτια του κώδικα τα έχουμε έτοιμα, και πρέπει κυρίως να ασχοληθούμε με την υλοποίηση του απλού μας δρομολογητή.

Εκτέλεση του Εργαστηρίου (όλα μαζί, 7 και 8)

Υλοποίηση των τριών απλών υποροutινών: κάθε μία από τις τρεις υποροutίνες (που θα ονομάσω Process_1, Process_2, Process_3 αλλά εσείς μπορείτε να δώσετε άλλα ονόματα) μπορεί να είναι π.χ. ένας μετρητής BCD που ανεβαίνει, ένας μετρητής BCD που κατεβαίνει, μία οθόνη που αναβοσβήνει, ένας μετρητής δακτυλίου δεξιόστροφος σε κάποια θύρα (αλλά όχι εκεί που είναι η UART), ένας μετρητής δακτυλίου αριστερόστροφος στην ίδια θύρα, άναμμα/σβήσιμο LED κάποιων ακροδεκτών (π.χ. εναλλαγή 01010101 με 10101010), κλπ. Ήδη εδώ δώσαμε **επτά** παραδείγματα απλών διεργασιών ενώ για το εργαστήριο χρειάζεστε τρεις, και δεν είναι απαραίτητο να είναι από την παραπάνω λίστα. Επίσης, επειδή η έννοια του χρόνου επιδέχεται πολλές ερμηνείες, το κατά πόσο π.χ. ένας μετρητής δακτυλίου θα ανανεώνεται σε κάθε κύκλο, που σημαίνει ότι δεν θα είναι ορατός από το ανθρώπινο μάτι, ή θα βάλετε κώδικα καθυστέρησης ώστε η ανανέωση να είναι πιο αργή, είναι στην δική σας ευχέρεια. Είναι όμως σημαντικό, ότι επιλέξετε να μπορείτε να το προσομοιώσετε. Χωρίς να είναι απαραίτητο, μπορείτε να κρατήσετε τον display driver για αριθμούς BCD που έχετε από προηγούμενα εργαστήρια, μαζί με τον χρονιστή του, ή να βγάλετε τον κώδικα αυτόν και να βάλετε κάτι άλλο. Η μόνη προδιαγραφή για τις τρεις υποροutίνες όσον αφορά την λειτουργικότητα είναι να «κάνουν κάτι» σε κάποια πόρτα (ή πόρτες) του AVR, και οι τρεις όμως στην ίδια πόρτα/πόρτες (να επικαλύπτονται έστω), ώστε να μπορεί να γίνει η δοκιμή του κώδικα. Αρχικά καμία υποροutίνα δεν είναι ενεργή, και επομένως ο δρομολογητής μας λειτουργεί σαν κενός βρόχος (idle loop), όπως και πριν δηλαδή.

Ενεργοποίηση/απενεργοποίηση υποροutινών: Αυτή γίνεται με την σειριακή θύρα, προσθέτοντας δύο πολύ απλές εντολές, δηλαδή Sx<CR><LF> (x=1,2,3) για ενεργοποίηση της διεργασίας 1,2, και 3 αντίστοιχα, και Qx<CR><LF> (x=1,2,3) για απενεργοποίηση (για Start και Quit αντίστοιχα, όχι πως έχει σημασία). Κάθε τέτοια εντολή ακολουθείται από απάντηση του AVR όπως και στα προηγούμενα εργαστήρια (δηλ.OK<CR><LF>). Για να γίνει πιο απλή η ζωή, αν πάρουμε Q χωρίς να έχει γίνει S για κάποια διεργασία, δεν κοιτάμε για λάθος, απλά (ξανά)κάνουμε 0 το σχετικό σημαιάκι. Θεωρούμε εν γένει ότι η είσοδος είναι σωστή και δεν έχει λάθη.

Χρονομετρητής των 100msec: Αυτός μπορεί να υλοποιηθεί είτε με κάποια μεταβλητή που αλλάζει από τον μετρητή που ήδη έχετε, ή με διαφορετικό TIMER. Τα 100msec που ορίζουμε σαν timeslice είναι προσεγγιστικά, ας πούμε ότι απόκλιση μέχρι και 10% στην μέτρηση του χρόνου είναι απολύτως ανεκτή.

Υλοποίηση του Δρομολογητή: Από πλευράς του πως απεικονίζουμε το πρόβλημα σε πόρους υπάρχουν προσεγγίσεις που επιτρέπονται και προσεγγίσεις που απαγορεύονται. **Τι επιτρέπεται:** Επιτρέπεται αν π.χ. έχουμε μία υποροutίνα που υλοποιεί μετρητή, στο παράθυρο των 100msec να κάνει μία μόνο αλλαγή στην τιμή του μετρητή και μετά να μπαίνει σε κάποιο idle loop, ή να κάνει πολλές/διαρκείς αλλαγές, κλπ. Επιτρέπεται (κατά κάποιο τρόπο επιβάλλεται, βλ. παρακάτω) η υποροutίνα να ολοκληρώνεται, το πρόγραμμα να επανέρχεται στον δρομολογητή, και επειδή το σημαιάκι είναι ενεργό για την ροutίνα αυτή να ξανατρέχει μέχρι να έρθει το interrupt του timeslice. Επιτρέπεται επίσης, αλλά θέλει προσοχή – οπότε δεν προτείνεται – να υλοποιηθούν τα Process με JMP και όχι με CALL. Αυτό το τελευταίο έχει να κάνει με το τι γίνεται στον τερματισμό/context switch. Το CALL έχει αποθηκευμένη μία διεύθυνση επιστροφής ενώ το JMP όχι και άρα αν διακόψουμε CALL σημαίνει ότι μας ξεφεύγει ο stack Pointer για την υποροutίνα. **Τι απαγορεύεται:** Απαγορεύεται να έχετε CALL χωρίς RETURN γιατί αυτό σημαίνει ότι γεμίζετε το stack με διευθύνσεις επιστροφής και μετά από κάποιο διάστημα θα κάψετε όλη σας την μνήμη (ή θα πανωγράψετε την μνήμη του προγράμματος). **Τι επιβάλλεται:** Κάθε υποροutίνα πρέπει να έχει ένα μικρό δικό της χώρο στην μνήμη, ώστε τυχόν πανωγράψιμο καταχωρητών να μην οδηγεί σε απώλεια πληροφορίας. Αν π.χ. έχουμε ένα μετρητή 1,2,3,4,...,9,0,1,.. τότε η πληροφορία για το ότι είμαστε π.χ. στο 5 εκτός από τον καταχωρητή εξόδου πρέπει να γραφτεί και στην μνήμη για να συνεχίσουμε με το 6 αν μεσολαβήσει context switch, καθότι ο καταχωρητής εξόδου μπορεί να πανωγραφεί από άλλη διεργασία. Επίσης, επιβάλλεται όσο είναι ενεργή μία διεργασία να παίρνει και να ξαναπαίρνει timeslice όποτε έρχεται η σειρά της – δεν την τρέχουμε μία φορά και τέλος, περιμένοντας την σειριακή θύρα να την ενεργοποιήσει. Από αυτή την άποψη έχουμε μέχρι τρεις διεργασίες, με οποιοδήποτε υποσύνολο να είναι ενεργό, και οι οποίες μοιράζονται τον χρόνο του AVR, ενώ παράλληλα, εφόσον έχουμε ενεργό τον display driver αυτός «κλέβει» κάποιους κύκλους χωρίς να επηρεάζει κατά τα άλλα τις διεργασίες μας.

Μια αποδεκτή προτεινόμενη λύση για να λειτουργεί το εργαστήριο και να είναι και απλός ο κώδικας: Η υλοποίηση του δρομολογητή, στην γενική περίπτωση αλλά όχι απαραίτητα στο εργαστήριό μας, **διακόπτει** κάποια διεργασία για να βάλει την επόμενη, και κάθε διεργασία είναι ένα αυτόνομο πρόγραμμα με exit() όταν ολοκληρωθεί (δηλ. όχι υποροutίνα που επαναλαμβάνεται). Αυτό όμως, και πάλι στην γενική περίπτωση, περιλαμβάνει διαχείριση (από το λειτουργικό σύστημα – που

εμείς δεν έχουμε) των δεδομένων κάθε διεργασίας, κλπ. Εδώ, επιτρέπεται στο εργαστήριό μας να έχουμε διεργασίες με μορφή υπορουτίνας, που έχει λίγες εντολές και κατόπιν επιστρέφει στο main() και ξαναεκτελείται όσο κρατάει το timeslice της. Ακόμη και αν θέλουμε π.χ. σε ένα timeslice να αλλάξουμε μία μόνο φορά την τιμή ενός μετρητή μπορούμε να έχουμε μία μεταβλητή την οποία ο δρομολογητής θέτει σε «1» και όταν μπαίνουμε στην υπορουτίνα, αν δούμε «1» αλλάζουμε τον μετρητή και θέτουμε την μεταβλητή στο «0», και με αυτόν τον τρόπο κάθε timeslice επιφέρει μόνο μία αλλαγή.

Και το «δια ταύτα» – η υλοποίηση του ίδιου του δρομολογητή: Για να κρατήσουμε το εργαστήριο απλό, μπορούμε να δεσμεύσουμε λίγες θέσεις στην μνήμη για την πληροφορία του αν κάποια διεργασία είναι ενεργή/ανενεργή και για το ποια διεργασία τρέχει στο τωρινό timeslice. Με αυτόν τον τρόπο όταν έρχεται το Interrupt του TIMER για τον δρομολογητή, κάνουμε ένα απλό έλεγχο για το ποια είναι η επόμενη ενεργή διεργασία , και βάζουμε την τιμή της (αρχικά 0 για καμία διεργασία ενεργή, και 1,2, ή 3, για όποια διεργασία είναι η επόμενη ενεργή διεργασία). Οι ρουτίνες τελειώνουν μετά από λίγες εντολές, και ο βρόχος επαναλαμβάνεται με την επόμενη διεργασία στην λίστα. Με αυτό τον τρόπο παρακάμπτουμε το να έχουμε CALL χωρίς RETURN, και εφόσον οι διεργασίες είναι απλές έχουμε φραγμένο και τον χρόνο που θα χρειαστεί στην χειρότερη περίπτωση, έναντι ενός πολύπλοκου προγράμματος που πραγματικά διακόπτει μία διεργασία και σώζει τα δεδομένα της σε μνήμη για να την συνεχίσει. Προφανώς οι απλουστεύσεις μας αυτές δεν καλύπτουν την γενική περίπτωση, αλλά είναι σαφώς ένας τρόπος να τρέχουμε διεργασίες που ενεργοποιούνται/ απενεργοποιούνται σε έναν επεξεργαστή με μικρή υπολογιστική ισχύ και ο οποίος δεν έχει λειτουργικό σύστημα.

Προθεσμίες και Παραδοτέα για τα εργαστήρια 7 και 8

Το Εργαστήριο 7 ουσιαστικά είναι η μελέτη των διεπαφών και χρήσης της μνήμης (στο «χαρτί» για τώρα), και η πλήρης υλοποίηση και αποσφαλμάτωση των τριών ρουτινών. Αυτές μπορούν να τρέχουν διαδοχικά, και οι τρεις (χωρίς ενεργοποίηση/απενεργοποίηση – για το Εργαστήριο 7 θεωρούνται και οι τρεις ενεργές), σε έναν ατέρμονα βρόχο. Επίσης, πρέπει να υλοποιηθεί το πρωτόκολλο της σειριακής θύρας UART με τις αντίστοιχες ενέργειες στην μνήμη που έχει δεσμευθεί για αυτό.

Το Εργαστήριο 8 είναι ολόκληρο το σύστημα με τον απλό δρομολογητή, και πρέπει να υποβληθεί στο Webcourses σαν ένα ολοκληρωμένο Εργαστήριο, όπου **τόσο οι κώδικες όσο και η αναφορά θα είναι απολύτως ολοκληρωμένα (επαναλαμβάνοντας**

αυτούσιο ότι χρειάζεται από το εργαστήριο 7 αλλά χωρίς αναφορά του τι άλλαξε από το εργαστήριο 7, παρά μόνο πως λειτουργεί το ολοκληρωμένο σύστημα). Προφανώς για ένα τέτοιο σύστημα η αναφορά δεν θα είναι των δύο σελίδων, εσείς θα δείτε τι χρειάζεται για επαρκή τεκμηρίωση.

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! ☺