Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα **Συνεργάτες**

Νικόλαος Αναγνώστου Νικόλαος Λάππας

03121818 03121098

**Ζήτημα 1.1**

Σε αυτή την άσκηση ζητήθηκε να παρέχουμε σε avr assembly μια συνάρτηση wait\_x\_msec η οποία ανάλογα με το τι αριθμός αναπαριστά το ζεύγος καταχωρητών r25 – r24 να παράγει την αντίστοιχη καθυστέρηση σε msec.

Από την αρχή φάνηκε ότι κάτι τέτοιο δεν μπορεί να επιτευχθεί με απόλυτη ακρίβεια λόγω της εντολής rcall και της ret που καταναλώνουν μαζικά 7 κύκλους. Αυτό δεν μπορεί να ξεπεραστεί διότι αν επιχειρήσουμε να ενσωματώσουμε στους καταναλισκόμενους κύκλους του κάθε msec αυτούς τους 7 , μπορεί για καθυστέρηση ενός και μόνο 1 msec να έχουμε την απόλυτη ακρίβεια , αλλά για περισσότερα του 1 msec μέσα στην συνεχόμενη λούπα τα υπόλοιπα msec πέραν του πρώτου μοιραία θα αναγκαστούν να καταναλώσουν λιγότερους από τους απαιτούμενους κύκλους για να έχουν ουσιαστική διάρκεια του 1 msec…

Προχωρούμε στην εξήγηση. Αφού ο μικροεπεξεργαστής τρέχει με συχνότητα 16 Mhz πρέπει για κάθε msec η ζητούμενη συνάρτηση wait\_x\_msec να φροντίζει εσωτερικά αυτής να καταναλώνονται 16000 κύκλοι αθροιστικά!

Βλέποντας τον κώδικα πιο κάτω παρατηρούμε πως καταναλώνονται συνολικά για κάθε msec εκτός του τελευταίου:

1+1+(3998\*2+3997\*2+1)”helper for each msec” +3+2+2 = 16000 κύκλοι.

Για το τελευταίο msec έχουμε:

1+1+(3998\*2+3997\*2+1)”helper for each msec” +3+2+1 = 15999 κύκλους, κάτι προβληματικό εκτός και αν αξιοποιήσουμε εδώ τους 7 αναγκαίους πλεονάζοντες κύκλους…αν αξιοποιήσουμε έναν από αυτόν τότε εύκολα έχουμε 16000 κύκλους και για το τελευταίο msec.

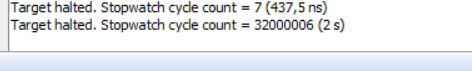
**Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα**Συνολικά λοιπόν για όποιο x αριθμό msec έχουμε συνολικά ακρίβεια τόσο μεγάλη όπου η τυχόν απόκλιση υπέρβαση είναι 6/16000 = 0,000375 ***msec***

Ενδεικτικά παραδείγματα

Εικόνα που περιέχει κείμενο, γραμματοσειρά, λευκό, άλγεβρα

Περιγραφή που δημιουργήθηκε αυτόματαx = 1msec

****x = 2000msec or 2 sec

**Ζήτημα 1.2**

Στην συγκεκριμένη άσκηση καλούμαστε να υπολογίσουμε τις λογικές συναρτήσεις

σε avr assembly για τον μικροελεγκτή ATmega328.

Στον κώδικά μας, αφού φορτώσουμε στους καταχωρητές τις αρχικές τιμές των A, B, C και D, υπολογίσουμε μεμονωμένα τις λογικές τιμές σιγά σιγά με την βοήθεια προσωρινών καταχωρητών και τον εντολών OR (λογικό Ή) και AND (λογικό ΚΑΙ). Τα αποτελέσματα τα τυπώνουμε στο PORTD του μικροελεγκή, αφού πρόκειται για λογικές συναρτήσεις με 8-bit η καθεμία (και το PORTD έχει 8 leds).

**A screenshot of a computer program

Description automatically generatedA screen shot of a computer code

Description automatically generated**Το loop υπολογισμού των λογικών συναρτήσεων εκτελέστηκε 6 φορές, και σε κάθε loop αυξάναμε τους καταχωρητές τόσο όσο καθορίζει η εκφώνηση της άσκησης. Για να το πετύχουμε αυτό, επειδή αυξάνουμε απευθείας τους καταχωρητές όσο μας λέει η εκφώνηση, τους αρχικοποιήσαμε εκτός του loop με τιμή ανάλογη (μικρότερη τόσες φορές όσες έπρεπε). Τα αποτελέσματα φαίνονται στον επόμενο πίνακα:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **F0** | **F1** |
| 0x51 | 0x41 | 0x21 | 0x01 | 0x40 | 0x50 |
| 0x52 | 0x43 | 0x24 | 0x05 | 0x43 | 0x52 |
| 0x53 | 0x45 | 0x27 | 0x09 | 0x44 | 0x54 |
| 0x54 | 0x47 | 0x2A | 0x0D | 0x43 | 0x56 |
| 0x55 | 0x49 | 0x2D | 0x11 | 0x48 | 0x58 |
| 0x56 | 0x4B | 0x30 | 0x15 | 0x4B | 0x4E |

Να πούμε συμπληρωματικά ότι η τελευταία εντολή () είναι για την συνεχή λειτουργία του προγράμματος και μπορεί να παραληφθεί.

**Ζήτημα 1.3**

Σε αυτή την άσκηση ζητήθηκε να προσομοιώσουμε ένα τρενάκι που ξεκινάει από δεξιά και πάει αριστερά και σε κάθε βήμα κάνει στάση 1 sec και όταν φτάσει στο MSB της PORTD τότε κάνει 2 sec στάση και ξεκινάει κίνηση προς τα δεξιά όπου πάλι κάθε βήμα πρέπει να καθυστερεί 1 sec μέχρι να φτάσει στο LSB όπου κάνει στάση πάλι 2 sec και έπειτα πάει αριστερά ….και αυτό έπρεπε να εκτελείται ατέρμονα.

Αυτό υλοποιήθηκε με την σύνθετη συνάρτηση της άσκησης 1.1 wait\_x\_msec.

Πέρα από αυτή τη συνάρτηση η λογική της άσκησης είναι απλή , θέσαμε την ddrd σε 0ΧFF ώστε η portd να είναι η έξοδος και έπειτα φορτώσαμε την τιμή 0x01 στην portd μιας και το τρένο κλήθηκε να ξεκινά από δεξιά. Κατόπιν ακολουθούν πολλαπλά lsl με ανάλογες καθυστερήσεις για κίνηση αριστερά ,όπου θέταμε το T flag του sreg στο λογικό 0 και μετά με lsr για την κίνηση προς τα δεξιά με ανάλογες καθυστερήσεις ,όπου θέταμε το T flag του sreg στο λογικό 1.

Τέλος ας σημειωθεί πως τροποποιήσαμε την wait\_x\_msec ώστε στο τέλος αυτής τα r24 και r25 να αρχικοποιούνται ξανά στα 1000msec. Αυτό επέφερε μια ακόμη πιο μικρή απόκλιση από τους 6 κύκλους που εξηγήσαμε στην 1.1 όμως σε πραγματικά νούμερα αυτή η απόκλιση δεν έπαψε να είναι παρά ελάχιστη και ο χρόνος μας είχε ως αποτέλεσμα ακρίβεια τουλάχιστον 3 δεκαδικών αν όχι και παραπάνω.

Ένας ενδεικτικός χρόνος για αυτή την άσκηση και για την ακρίβεια του χρόνου μας είναι ο εξής:



\*17 sec είναι ο χρόνος από την εκκίνηση του τρένου από τα δεξιά μέχρι να βρεθεί στην ίδια θέση και αφού έχουν περιέλθει τα δύο sec και το τρένο είναι έτοιμο να προχωρήσει.