ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΑΚΑΔ. ΕΤΟΣ 2024-2025

ΑΘΗΝΑ 4 Οκτωβρίου 2024

1η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ
ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Εργαστήριο Μικροϋπολογιστών"

Ανάπτυξη κώδικα για το μικροελεγκτή ATmega328 και προσομοίωση της εκτέλεσης του στο αναπτυξιακό περιβάλλον MPLAB X

ΟΜΑΔΑ 23

Συνεργάτες

Νικόλαος Αναγνώστου
03121818

Νικόλαος Λάππας
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.

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 για όλα τα ζητούμενα msec πέραν του τελευταίου όπου για να πετύχουμε την απόλυτη ακρίβεια κάναμε κάτι διαφορετικό!

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

1+1+(3997*2+3996*2+1)$^{\text{"helper for each msec"}}$ +2+1+10+2 = 16000 κύκλοι.

Για το τελευταίο msec έχουμε:

1+1+(3997*2+3996*2+1)$^{\text{"helper for each msec"}}$ +2+2+4+**3+4** = 16000 κύκλους, συμπεριλαμβάνοντας και τους κύκλους της rcall εντολής (3) και της ret (4).

```
wait_x_msec:
ldi r26,LOW(15984);1 cycle
ldi r27,HIGH(15984);1 cycle
helper:
sbiw r26,4 ;2 cycles
brne helper ;2 cycles or 1 cycle for the last iteration
;15984 -> helper consumes 15983 cycles
;so after helper we consume totally 15985 cycles

sbiw r24,1 ;2 cycle
breq last_msec ;1 cycle but if last msec 2 cycles

;for all msec except from the last -> 15985 + 2 + 1 = 15988 cycles

nop
nop
nop
nop
nop
nop
nop
nop
nop
nop ;10 cycles

;extra 10 cycles -> 15998

brne wait_x_msec ;2 cycles total 16000 cycles with this operation

last_msec:
;in the last iteration (last msec) we have 15989 cycles
nop
nop
nop
```

```
nop

;extra 4 cycles -> 15993 cycles
ret ;4 cycles

;with ret and rcall we calculated exactly 16000 cycles again
;so in both cases we end up having 16000 cycles -> 1 msec * (desired time)
```

Ενδεικτικά παραδείγματα

```
Target halted. Stopwatch cycle count = 7 (437,5 ns)
Target halted. Stopwatch cycle count = 16000 (1 ms)
```

x = 1msec

x = 2000msec or 2 sec

```
Target halted. Stopwatch cycle count = 7 (437,5 ns)
Target halted. Stopwatch cycle count = 32000000 (2 s)
```

Ζήτημα 1.2

Στην συγκεκριμένη άσκηση καλούμαστε να υπολογίσουμε τις λογικές συναρτήσεις

$$F0 = (A \cdot B' + B' \cdot D)'$$

$$F1 = (A + C') \cdot (B + D')$$

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) – Δεν ζητείται η εκτύπωση, απλά για οπτικοποίηση.

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	0xEF	0xDF
0x52	0x43	0x24	0x05	0xEB	0xDB
0x53	0x45	0x27	0x09	0xE5	0xD3
0x54	0x47	0x2A	0x0D	0xE7	0xD5
0x55	0x49	0x2D	0x11	0xEB	0xC7
0x56	0x4B	0x30	0x15	0xEB	0xCB

```
main:
  ;initialize the registers with their values
  ;minus the number we add in each iteration
  ldi A, 0x50
  ldi B, 0x3F
  ldi C, 0x1E
  ldi D, 0xFD
  ldi counter, 0x06

  ser r24
  out DDRD,r24; Init PORTD as output

main_loop:
  subi A, -1
  subi B, -2
  subi C, -3
  subi D, -4

  mov complement_B, B
  mov complement_C, C
  mov complement_D, D
  mov temp_F0, A
  mov temp_F1, A
  mov temp, D

  com complement_B
  com complement_C
  com complement_D

  ;calculate F0
  and temp_F0, complement_B
  and temp, complement_B
  or temp_F0, temp
  com temp_F0 ;value F0

  out PORTD, temp_F0

  ;calculate F1
  or temp_F1, complement_C
  or complement_D, B
  and temp_F1, complement_D ;value F1

  out PORTD, temp_F1

  dec counter
  brne main_loop ;loop 6 times
```

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 0xFF 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 (εντολή clt) και μετά με lsr για την κίνηση προς τα δεξιά με ανάλογες καθυστερήσεις, όπου θέταμε το T flag του sreg στο λογικό 1 (εντολή set). Αυτές οι δύο εντολές επενεργούν στο 6^{ο} μόνο bit του καταχωρητή.

Τέλος, να 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.

```
ldi train, 0x01 ;start from right -> left : code 0
                            ;left -> right : code 1
  ;ldi r28,0xFF
  out       PORTD,train
transportation:

  clt ;right -> left : code 0 in 6th bit of register SREG

  rcall wait_x_msec
  lsl train
  out PORTD,train
  rcall wait_x_msec
  lsl train
  out PORTD,train
  rcall wait_x_msec
  lsl train
  out PORTD,train
  rcall wait_x_msec
  lsl train
  out PORTD,train
  rcall wait_x_msec
  lsl train
  out PORTD,train
  rcall wait_x_msec
  lsl train
  out PORTD,train
  rcall wait_x_msec
  lsl train
  out PORTD,train
  rcall wait_x_msec
  lsl train
  out PORTD,train
```

```
rcall wait_x_msec
rcall wait_x_msec ;reached the edge

set ;left -> right : code 1 in 6th bit of register SREG

lsr train
out PORTD,train
rcall wait_x_msec
lsr train
out PORTD,train
rcall wait_x_msec
lsr train
out PORTD,train
rcall wait_x_msec
lsr train
out PORTD,train
rcall wait_x_msec
lsr train
out PORTD,train
rcall wait_x_msec
lsr train
out PORTD,train
rcall wait_x_msec
lsr train
out PORTD,train
rcall wait_x_msec ;reached the edge
rjmp transportation ;loop again
```