

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

Εργαστήριο Μικροϋπολογιστών

Κριθαρίδης Κωνσταντίνος, el21045

Μπαλάτος Δημήτριος, el21170

15 Οκτωβρίου 2024

1 Ρυθμιζόμενες χρονικές καθυστερήσεις

Ορίζουμε εντολή `wait_approx_1_msec` που κάνει την πλειοψηφία των απαραίτητων κύκλων (15980 από 16000) για καθυστέρηση 1 msec μέσω απλού loop από το 3991 προς τα κάτω. Για να κάνουμε τους υπόλοιπους 20 κύκλους ανά ms, πρέπει να διακρίνουμε περιπτώσεις ανάλογα αν διανύουμε το τελευταίο δευτερόλεπτο, ώστε να αντιμετωπίσουμε τον επιπλέον απαραίτητο χρόνο της εξωτερικής `rcall`, της `ret` και του υπόλοιπου σταθερού overhead ($3 + 4 + 1 + 1 = 9$ κύκλοι). Για να το κάνουμε αυτό, αξιοποιούμε 9 `brne`, τα οποία εκτελούν 2 κύκλους σε κάθε επανάληψη, ενώ 1 στη τελευταία, οπότε μας επιτρέπουν να “αφαιρούμε” το σταθερό overhead μέσω της τελευταίας επανάληψης. Συνεπώς, σε κάθε επανάληψη γίνονται $2 + 9 * 2 = 20$ κύκλοι που προστίθενται στους 15980 για να αθροίσουν στους 16000, ενώ το σταθερό overhead λόγω των `brne` γίνεται $9 - 9 = 0$, επιτυγχάνοντας τέλεια ακρίβεια. Ο κώδικας δουλεύει για κάθε τιμή του `Delay_ms` ≥ 1 που χωράει σε 2 bytes, δηλαδή από 1 έως 65535.

Βλέπουμε χρόνο εκτέλεσης μέσω της προσομοίωσης για 1 msec και 1 sec αντίστοιχα:

| | |
|---------------|------------------|
| Cycle Counter | 16000 |
| Frequency | 16.000 MHz |
| Stop Watch | 1,000.00 μ s |

Σχήμα 1: Χρονομέτρηση καθυστέρησης 1 msec

| | |
|---------------|----------------------|
| Cycle Counter | 16000000 |
| Frequency | 16.000 MHz |
| Stop Watch | 1,000,000.00 μ s |

Σχήμα 2: Χρονομέτρηση καθυστέρησης 1 sec

```
1 ;  
2 ; Ex1.asm  
3 ;  
4 ; Created: 10/13/2024 2:48:02 PM  
5 ; Author : User  
6 ;  
7
```

```

8  .include "m328PBdef.inc"
9
10 .def msl=r24
11 .def msh=r25
12 .def il=r26
13 .def ih=r27
14
15 .equ Delay_ms=65535
16
17 ; Replace with your application code
18 init:
19     ldi r26, LOW(RAMEND)
20     out SPL, r26
21     ldi r26, HIGH(RAMEND)
22     out SPH, r26
23     rcall wait_x_msec
24     nop
25
26 wait_x_msec: ; Delay_ms*16000 +2-9+4 + 3 overhead = Delay_ms*16000
27     ldi msl, LOW(Delay_ms) ; 1 cycle
28     ldi msh, HIGH(Delay_ms) ; 1 cycle
29 delay1: ; Delay_ms*16000 - 9
30     rcall wait_approx_1_msec ; 15980 cycles = 15977 cycles
31                             ; + 3 cycles overhead
32     sbiw msl, 1 ; 2 cycles
33     brne label1 ; 2 or 1 cycles
34 label1:
35     brne label2 ; 2 or 1 cycles
36 label2:
37     brne label3 ; 2 or 1 cycles
38 label3:
39     brne label4 ; 2 or 1 cycles
40 label4:
41     brne label5 ; 2 or 1 cycles
42 label5:
43     brne label6 ; 2 or 1 cycles
44 label6:
45     brne label7 ; 2 or 1 cycles
46 label7:
47     brne label8 ; 2 or 1 cycles
48 label8:
49     brne delay1 ; 2 or 1 cycles
50 delay1_end:
51     ret ; 4 cycles
52
53 wait_approx_1_msec: ; 3991*4 - 1 + 14 cycles = 15977 cycles
54     push il ; 2 cycles
55     push ih ; 2 cycles
56     ldi il, LOW(3991) ; 1 cycle
57     ldi ih, HIGH(3991) ; 1 cycle
58 loop: ; x*4 - 1 cycles

```

```

59     sbiw il, 1 ; 2 cycles
60     brne loop ; 2 or 1 cycle
61     pop ih ; 2 cycles
62     pop il ; 2 cycles
63     ret ; 4 cycles

```

2 Υπολογισμός λογικών συναρτήσεων

Η υλοποίηση είναι straight-forward μετάφραση των λογικών συναρτήσεων σε κώδικα. Αρχικά υπολογίζουμε τα συμπληρωματικά των μεταβλητών, μετά υπολογίζουμε τις συναρτήσεις. Για τα ενδιαμέσως αποτελέσματα σε κάθε περίπτωση αξιοποιούμε τον καταχωρητή του αποτελέσματος και έναν βοηθητικό καταχωρητή `tmp`.

Πίνακας:

| A | B | C | D | FO | 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 |

```

1 ;
2 ; Ex2.asm
3 ;
4 ; Created: 10/13/2024 2:48:57 PM
5 ; Author : User
6 ;
7
8 .include "m328PBdef.inc"
9
10 .def a=r16
11 .def b=r17
12 .def c=r18
13 .def d=r19
14 .def nota=r20
15 .def notb=r21
16 .def notc=r22
17 .def notd=r23
18 .def f0=r24
19 .def f1=r25
20 .def i=r26
21 .def temp=r27
22
23 ; Replace with your application code

```

```

24 start:
25     ldi a, 0x51
26     ldi b, 0x41
27     ldi c, 0x21
28     ldi d, 0x01
29     ldi i, 6
30 loop:
31     mov nota, a
32     com nota
33     mov notb, b
34     com notb
35     mov notc, c
36     com notc
37     mov notd, d
38     com notd
39
40     mov f0, a
41     and f0, notb
42     mov temp, notb
43     and temp, d
44     or f0, temp
45     com f0
46
47     mov f1, a
48     or f1, notc
49     mov temp, b
50     or temp, notd
51     and f1, temp
52
53     inc a
54     subi b, -2 ; since there is no addi, we subtract its opposite
55     subi c, -3 ; similarly
56     subi d, -4 ; similarly
57
58     dec i
59     brne loop

```




























3 Αυτοματισμός βαγονέτου

Για την καθυστέρηση αξιοποιήθηκε η μέθοδος που ορίστηκε στην Άσκηση 1. Μετά, εναλλάξ τρέχουμε τις υπορουτίνες `RIGHT` – `LEFT` , που επιλέγονται βάσει της σημαίας `T` .




























Οι `RIGHT` – `LEFT` με την σειρά τους καλούν το `delay` , ενημερώνουν τον καταχωρητή `TRAIN` και, στο label `OUTPUT` , τον εκτυπώνουν στα LEDs του PORTD πριν επιστρέψουν στην `start` για την επόμενη επανάληψη. Υπεύθυνη για την επιλογή μεταξύ των 2 είναι η `start` .

Για την ενημέρωση της `T` , ελέγχουμε αν το 1 hi-bit του καταχωρητή `TRAIN` αφαιρέθηκε και πήγε

στο Carry λόγω των shifts που γίνονται. Όταν αυτό γίνεται, επαναφέρουμε τον καταχωρητή στην προηγούμενη τιμή (κάνουμε έτσι 2 καθυστερήσεις σε κάθε άκρο) και ενημερώνουμε το T πριν πάμε στην OUTPUT .

| Name | Address | Value | Bits |
|---|---------|-------|---|
|  PIND | 0x29 | 0x80 |         |
|  DDRD | 0x2A | 0xFF |         |
|  PORTD | 0x2B | 0x80 |         |

Σχήμα 3: Αρχική κατάσταση

| Name | Address | Value | Bits |
|---|---------|-------|---|
|  PIND | 0x29 | 0x40 |         |
|  DDRD | 0x2A | 0xFF |         |
|  PORTD | 0x2B | 0x40 |         |

Σχήμα 4: Κατάσταση μετά από 1 επανάληψη

Output:

out PORTD, TRAIN

start:

brtc RIGHT

Σχήμα 5: Μέσω αυτού του breakpoint μπορούμε να πάρουμε τα screenshots

```

1 ;
2 ; Ex3.asm
3 ;
4 ; Created: 10/13/2024 2:49:41 PM
5 ; Author : User
6 ;
7
8 .include "m328PBdef.inc"
9
10 .def msl=r24
11 .def msh=r25
12 .def il=r26
13 .def ih=r27
14
15 .def TRAIN=r19
16
17 .equ Delay_ms=1000
18
19 ; Replace with your application code
20 init:
21     ldi r26, LOW(RAMEND)
22     out SPL, r26
23     ldi r26, HIGH(RAMEND)
24     out SPH, r26
25
26     ;Init PORTD as output
27     ser r16

```

```

28         out DDRD, r16
29
30         set ; LEFT (shifting right)
31         ldi TRAIN, 0x80
32
33 Output:
34         out PORTD, TRAIN
35 start:
36         brtc RIGHT
37 LEFT: ; T = 1
38         rcall wait_x_msec ; wait 1 sec
39         clc
40         ror TRAIN
41         brcc OUTPUT
42         rol TRAIN ; reset train
43         clt
44         rjmp OUTPUT
45 RIGHT: ; T = 0
46         rcall wait_x_msec ; wait 1 sec
47         clc
48         rol TRAIN
49         brcc OUTPUT
50         ror TRAIN ; reset train
51         set
52         rjmp OUTPUT
53
54 wait_x_msec: ; Delay_ms*16000 +2-9+4 + 3 overhead = Delay_ms*16000
55         ldi msl, LOW(Delay_ms) ; 1 cycle
56         ldi msh, HIGH(Delay_ms) ; 1 cycle
57 delay1: ; Delay_ms*16000 - 9
58         rcall wait_approx_1_msec ; 15980 cycles = 15977 cycles + 3 cycles overhead
59         sbiw msl, 1 ; 2 cycles
60         brne label1 ; 2 or 1 cycles
61 label1:
62         brne label2 ; 2 or 1 cycles
63 label2:
64         brne label3 ; 2 or 1 cycles
65 label3:
66         brne label4 ; 2 or 1 cycles
67 label4:
68         brne label5 ; 2 or 1 cycles
69 label5:
70         brne label6 ; 2 or 1 cycles
71 label6:
72         brne label7 ; 2 or 1 cycles
73 label7:
74         brne label8 ; 2 or 1 cycles
75 label8:
76         brne delay1 ; 2 or 1 cycles
77 delay1_end:
78         ret ; 4 cycles

```

```

79
80 wait_approx_1_msec: ; 3991*4 - 1 + 14 cycles = 15977 cycles
81     push il ; 2 cycles
82     push ih ; 2 cycles
83     ldi il, LOW(3991) ; 1 cycle
84     ldi ih, HIGH(3991) ; 1 cycle
85 loop: ; x*4 - 1 cycles
86     sbiw il, 1 ; 2 cycles
87     brne loop ; 2 or 1 cycle
88     pop ih ; 2 cycles
89     pop il ; 2 cycles
90     ret ; 4 cycles

```