

## 2. Semestrální práce z předmětu KIV/ÚPA

Tomáš Maršálek, A10B0632P  
marsalet@students.zcu.cz

20. prosince 2012

## 1 Zadání

Dělení 16 bitů / 16 bitů = 16 bitů + 16 bitů (výsledek + zbytek) (bez použití instrukce dělení). Vstupy a výstupy hexadecimálně.

## 2 Řešení

Na dělení je použit klasický algoritmus dělení (long division). Ve vyšším jazyce odpovídá následujícímu kódu:

```
int div(int a, int b)
{
    int q, x;

    q = 0;
    x = b;

    while (x <= (a >> 1))
        x <<= 1;
    while (x >= b) {
        if (a >= x) {
            q |= 1;
            a -= x;
        }
        x >>= 1;
        q <<= 1;
    }
    q >>= 1;

    return q;
}
```

Vstupy a výstupy používají pomocný buffer, ze kterého jsou Hornerovým schématem rozkódovány, respektive zakódovány.

Oproti řešení z KIV/POT je zde dělení na 32 bitech namísto 16.

## 3 Uživatelská příručka

Uživatel je vyzván na zadání dvou hexadecimálních čísel, dělence a dělitele. Po zadání je uživateli představen výsledek. Výsledkem je bezznamínkový podíl a zbytek. Pozn. hexadecimální číslice musí být velkými písmeny (tzn. 12ABC3, ne 12abc3) a maximální délka může čísla může být 8 hexadecimálních znaků (32 bitů). Vstup není nijak kontrolován, je třeba zadat číslo přesně bez jakýkoliv přebytečných znaků.

## 4 Pseudoinstrukce

Pseudoinstrukce použité v programu jsou ve výsledku přeloženy do strojových instrukcí. Např.:

Pseudoinstrukce	Strojová instrukce
<code>move \$v0, \$0</code>	<code>addi \$v0, \$0, 0</code>
<code>b endif</code>	<code>beq \$0, \$0, endif</code>
<code>bltu \$s1, \$a1, L4</code>	<code>sltu \$1, \$s1, \$a1</code>
	<code>bne \$1, \$0, L4</code>

## 5 Zpožděné čtení z paměti

Instrukce čtení z paměti *la*, *lw* a *li* proběhnou zpožděně. Za poslední instrukcí čtení (tj. *li*) je vložena prázdná instrukce, aby nedošlo k datovému hazardu.

```
110 la $a0, buf
111 lw $a1, len
112 li $v0, 8
113 nop
```

## 6 Datový hazard

Jedná se o hazard typu RAW (Read after Write), protože registr `$t0` je použit v instrukci *addu*, která je závislá na předchozí *move* instrukci.

```
40 move $t0, $a2
41 addu $v0, $v0, $t0
```

## 7 Listing programu

---

```
1      .data
2
3  newline:      .ascii "\n"
4  delenec:      .ascii "delenec: "
5  delitel:      .ascii "delitel: "
6  podil:        .ascii "podil: "
7  zbytek:       .ascii "zbytek: "
8  buf:          .space 33
9  len:          .word 33
10
11      .text
12
```

```

13 # int _from_hex(void *buf as $a0)
14 _from_hex:
15         move    $v0, $0
16 cond:    lbu     $t0, 0($a0)
17         nop
18         beq     $t0, $0, endloop
19         nop
20         beq     $t0, 10, endloop
21         nop
22         bgeu    $t0, 'A', bigger
23         nop
24         subu    $t0, $t0, '0'
25         b      endif
26         nop
27 bigger:  subu    $t0, $t0, 'A'-10
28 endif:  sll     $v0, $v0, 4
29         addu    $v0, $v0, $t0
30         addu    $a0, $a0, 1
31         b      cond
32         nop
33 endloop: jr     $ra
34         nop
35
36
37 # void _to_hex(int n as $a0, void *buf as $a1, int len as $a2)
38 _to_hex:
39         move    $v0, $a1
40         move    $t0, $a2
41         addu    $v0, $v0, $t0
42         subu    $v0, $v0, 1
43         sb     $0, ($v0)
44 loop:    subu    $v0, $v0, 1
45         and     $t1, $a0, 15
46         bltu   $t1, 10, small
47         nop
48         addu    $t1, 'A'-10
49         b      store
50         nop
51 small:  addu    $t1, '0'
52 store: sb      $t1, ($v0)
53         srl     $a0, $a0, 4
54         bgtu    $a0, $0, loop
55         nop
56         jr     $ra
57         nop
58
59
60 _div:
61         subu    $sp, $sp, 8
62         sw     $s0, 0($sp)

```

```

63      sw      $s1, 4($sp)
64      # -----
65
66      # a = $a0
67      # b = $a1
68      # q = $s0
69      # x = $s1
70
71      li      $s0, 0          # q = 0
72      move    $s1, $a1        # x = b
73
74      srl     $t0, $a0, 1      # a >> 1
75  L1:      bgtu  $s1, $t0, L2   # if (x > (a >> 1)) goto L2
76      nop
77      sll     $s1, $s1, 1      # x <<= 1
78      b       L1              # goto L1
79
80  L2:      bltu  $s1, $a1, L4   # if (x < b) goto L4
81      nop
82      bltu    $a0, $s1, L3     # if (a < x) goto L3
83      nop
84      or      $s0, $s0, 1      # q |= 1
85      subu    $a0, $a0, $s1     # a -= x
86  L3:      srl  $s1, $s1, 1      # x >>= 1
87      sll     $s0, $s0, 1      # q <<= 1
88      b       L2              # goto L2
89      nop
90  L4:      srl  $s0, $s0, 1      # q >>= 1
91      move    $v0, $s0         # quotient = q
92      move    $v1, $a0         # remainder = a
93
94
95      # -----
96      lw      $s0, 0($sp)
97      lw      $s1, 4($sp)
98      addu    $sp, $sp, 8
99
100     jr      $ra
101     nop
102
103
104  main:
105     # get dividend
106     la      $a0, delenec
107     li      $v0, 4
108     nop
109     syscall
110     la      $a0, buf
111     lw      $a1, len
112     li      $v0, 8

```

```

113      nop
114      syscall
115      jal      _from_hex
116      nop
117      move     $s0, $v0
118
119
120      # get divisor
121      la       $a0, delitel
122      li       $v0, 4
123      nop
124      syscall
125      la       $a0, buf
126      lw       $a1, len
127      li       $v0, 8
128      nop
129      syscall
130      jal      _from_hex
131      nop
132      move     $s1, $v0
133
134      # call divide
135      move     $a0, $s0
136      move     $a1, $s1
137      jal      _div
138      nop
139      move     $s0, $v0      # quotient
140      move     $s1, $v1      # remainder
141
142
143      # print hex quotient
144      la       $a0, podil
145      li       $v0, 4
146      nop
147      syscall
148      move     $a0, $s0
149      la       $a1, buf
150      lw       $a2, len
151      jal      _to_hex
152      nop
153      move     $a0, $v0
154      li       $v0, 4
155      nop
156      syscall
157      la       $a0, newline
158      li       $v0, 4
159      nop
160      syscall
161
162      # print hex remainder

```

```

163      la      $a0, zbytek
164      li      $v0, 4
165      nop
166      syscall
167      move    $a0, $s1
168      la      $a1, buf
169      lw      $a2, len
170      jal     _to_hex
171      nop
172      move    $a0, $v0
173      li      $v0, 4
174      nop
175      syscall
176      la      $a0, newline
177      li      $v0, 4
178      nop
179      syscall
180
181      # EXIT
182      li      $v0, 10
183      nop
184      syscall

```

---

## 8 Závěr

Program byl vyvíjen na platformě GNU/Linux a testován pomocí simulátoru QtSpim.