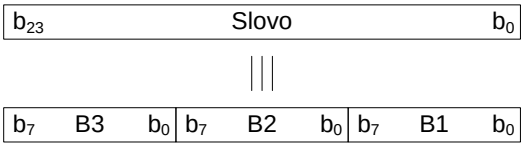
Vícebytová celočíselná aritmetika

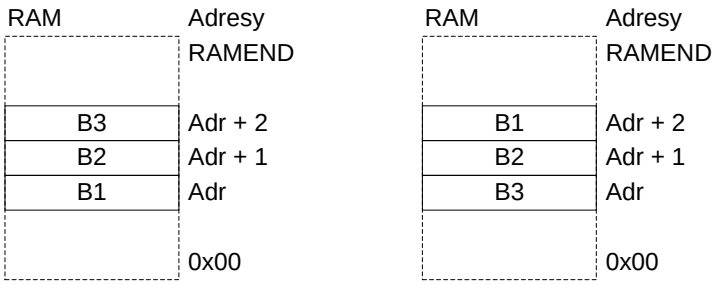
- = bitová šířka zpracovávaných dat > než šířka slova CPU
- např.: 8 b CPU zpracovává 32 b data
- dále teoretické příklady: 3 b CPU zpracovává 6 b slova

Uložení dat v paměti

- data nutno rozdělit do několika PM (Př.: 24 b slovo na 8 b CPU)



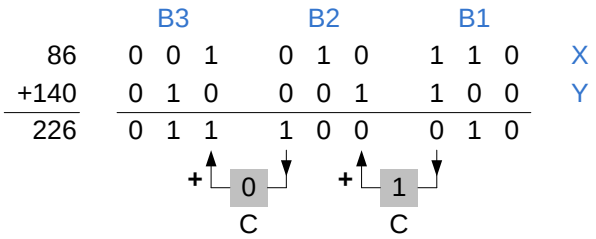
- dle zarovnání:
  - Little-Endian → na nižší adrese nejméně významný byte
  - Big-Endian → na nižší adrese nejvýznamnější byte



LE BE

Sčítání

- Princip: příznak C = mezi byty běžný přenos
- příklad: 9 b slova na 3 b CPU



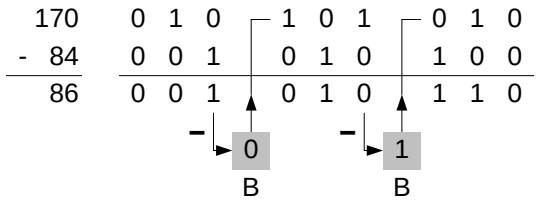
- kód v JSA AVR: využijeme `adc Rd, Rr` (Add with Carry, `Rd = Rd + Rr + C`)

```
add    B1X, B1Y
adc    B2X, B2Y
adc    B3X, B3Y
// výsledek je v B3X B2X B1X
```

Odčítání

- Princip: příznak C = mezi byty běžná výpůjčka

- příklad: 9 b slova na 3 b CPU

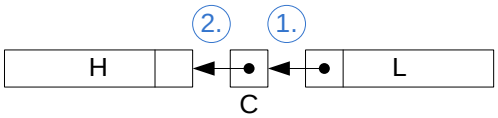


- kód v JSA AVR: využijeme `sbc Rd, Rr` (Substract with Carry, `Rd = Rd - Rr - C`)

```
sub    B1X, B1Y
sbc    B2X, B2Y
sbc    B3X, B3Y
// výsledek je v B3X B2X B1X
```

Násobení

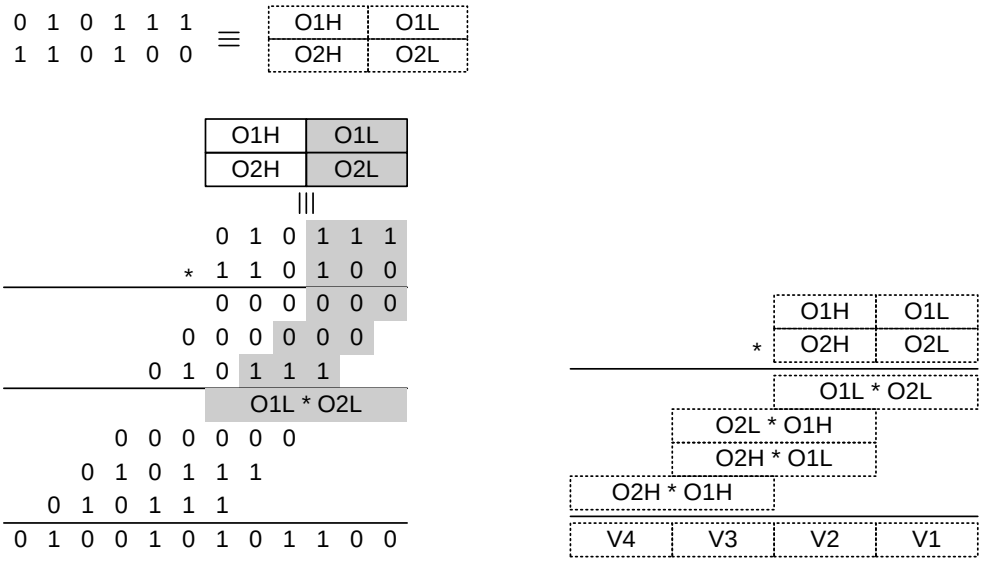
- násobení *2 – využijeme vícebitový posuv
  - princip: postupná rotace skrz C



  - kód v JSA AVR:

```
clc
rol    L
rol    H
```

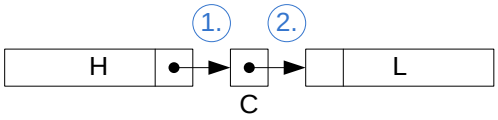
- Obecné násobení (6 b násobení na 3 b CPU):



- lze využít násobičku procesoru a vícebitové posuny

Dělení

- Dělení /2 – ekvivalentní *2 (obrácené pořadí operací):




```
lsl    H      // signed: asr
rol    L
```

- Obecné dělení
  - případnou děličku nelze použít
  - nutno rozšířit (vícebitové odčítání a posuvy) dříve prezentovaný algoritmus pro CPU bez děličky

Celočíselná vícebytová aritmetika

- použijeme bezznaménkové postupy

BCD kód

Princip

- (binary code digit)
- pro reprezentaci desítkových čísel v paměti procesoru
- velké množství BCD kódů – nejčastěji BCD8421 (dále jen BCD)
  - 8421 = váhy jednotlivých bitů (normální vyjádření čísel od 0do 9)
  - každá dekadická číslice = 4 b
- kódovací tabulka

```
0      0000
1      0001
2      0010
         |
8      1000
9      1001
```

- zakázané kombinace: 10 až 15
- Desítkové číslo dvojkově zapsané zabírá větší počet míst, než jeho dvojkový ekvivalent

  $123_{10} = 0111\ 1011_2 = 0001\ 0010\ 0011_{BCD8421}$

- pakovaný zápis (packed number) –v jednom bytu uloženy 2 desítkové číslice.
  - Př.: 16b číslo (0÷65536) v packed BCD kódu 3 byty.

    $12345_{10} = 0000\ 0001\ |\ 0010\ 0011\ |\ 0100\ 0101_{BCD8421}$

Aritmetika

Sčítání

- problém: CPU nemají BCD sčítačku, ale binární:

- binární sčítačka: na 4 b operace modulo 16.

```
 0  9     0000 1001
 0  5     0000 0101
 0 14     0000 1110
```

BCD – zakázaná kombinace

  - BCD součet: na 4b nutno operaci modulo 10

```
 0 9     0000 1001
 0 5     0000 0101
 1 4     0001 0100
```

- Náprava: nutno provést tzv. dekadickou korekci (DA – decadic addition):
  - je-li na nižších 4 bitech po součtu číslo větší než 9, nebo byl překročen modul sčítačky (součet byl větší než 15), je k výsledku přičtena hodnota 0x06 (rozdíl mezi součtem modulo 10 a modulo 16)
  - je-li na vyšších 4 bitech po součtu číslo větší než 9, nebo překročen modul sčítačky (součet byl větší než 255), je k výsledku přičtena hodnota 0x60.
- Příklad: DA v nižším řádu:

```
    23   0010 0011
   +48   0100 1000
 6(11)   0110 1011
```

Aplikace DA (+0x06):

```
 6(11)   0110 1011
    +6   0000 0110
    71   0111 0001
```

Odčítání

- převedením na součet v desítkovém doplňku
- Příklad:
  168–44; $L_{(10)} = 3$
  - –44 v desítkovém doplňku
    $Z = z^n = 10^3 = 1000$
    $D(A) = Z + A = -44 + 1000 = 956$
  - BCD součet

```
      168
 (-44)+956
      ~~1~~124
```

    - přenos ignorujeme

Převody BIN → BCD

- Metody řešení:
  - Postupné dělení binárního čísla základem
  - Postupné odečítání základu
  - Převod přes Hornerovo schéma

Postupné dělení binárního čísla základem

- Každé celé dekadické číslo A lze vyjádřit jako:

$$A = \sum_{i=0}^{m} a_i \cdot 10^i$$

- princip = postupné dělení převáděného čísla a zbytků po předcházejícím dělení mocninami čísla 10^i

- Př.:
  1234 = 1*1000 + 2*100 + 3*10 + 4
  1234/1000 = 1, zbytek 234
  234/100 = 2, zbytek 34
  34/10 = 3, zbytek 4

- Výhody:
  - U jednobytových čísel jasný, jednoduchý algoritmus.
- Nevýhody:
  - nutnost dělení
  - čísla delší než šířka slova ALU – komplikovaná realizace (vícebytové dělení)

Postupné odečítání základu

- dělení nahrazeno odečítáním mocnin 10^i.
- Po každém kroku se zkoumání, je-li výsledek záporný
  - ne → inkrementován čítač na dané pozici *i*.
  - ano = daný řád byl eliminován; čítač obsahuje počet základů pro daný řád (tedy hledanou číslici v daném řádu)
- Příklad:
  tisíce = 0
  1234 – 1000 = 234 < 0 – ne, tisíce = 1
  234 – 1000 = –766 < 0 – ano, hledané tisíce = 1, 234 je základ pro eliminaci v řádu 100vek

  stovky = 0
  234 – 100 = 134 < 0 – ne, stovky = 1
  134 – 100 = 34 < 0 – ne, stovky = 2
  34 – 100 = –76 < 0 – ano, hledané stovky = 2, 34 je základ pro eliminaci v řádu 10tek

  atd…

- Výhody:
  - poměrně jednoduchá realizace i pro delší čísla (vícebytové odčítání je jednoduché).
- Nevýhody:
  - nekonstantní doba převodu (závislá na převáděném čísle)

Převod přes Hornerovo schéma

- Každé celé binární číslo *A* šířky *n* bitů lze vyjádřit pomocí vztahu

$$A_{(2)} = \sum_{i=0}^{n-1} a_i \cdot 2^i = a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \ldots + a_1 2 + a_0$$

- lze pomocí postupného vytýkání základu ve vyšších řádech snadno upravit na tzv. Hornerovo schéma

$$A_{(2)} = (\ldots(a_{n-1} \cdot 2 + a_{n-2}) \cdot 2 + a_{n-3}) \cdot 2 \ldots + a_1) \cdot 2 + a_0$$

  - Pozn.: platí pro jakýkoli základ
- Příklad: 4-bitové číslo

$$A_{(2)} = a_3 2^3 + a_2 2^2 + a_1 2 + a_0 =$$
$$= (a_3 2^2 + a_2 2 + a_1) \cdot 2 + a_0 =$$
$$= ((a_3 2 + a_2) \cdot 2 + a_1) \cdot 2 + a_0$$

- Princip: hledané dekadické číslo *A* lze vytvářet od nejvyššího bitu postupným násobením mezivýsledků dvěma a přičítáním dalšího následujícího nižšího bitu. Budeme-li toto číslo od začátku chápat jako dekadické (v BCD kódu), bude dekadickým i po každé operaci (násobení 2 a součet), a pokud ne (viz. BCD sčítání), lze na něj aplikovat dekadickou korekci, po které opět získáme dekadické číslo.

- Př.: $52_{(10)} = 0011\ 0100_{(2)}$

legenda

BCD	BIN	operace

1. krok

00	0000 000**0**	0*2
00	0000 000**0**	0*2+a7=a7
00	0000 000**0**	DA

2. krok

00	0000 00**00**	a7*2
00	0000 00**00**	a7*2+a6
00	0000 00**00**	DA

3. krok

00	0000 0**000**	(a7*2+a6)*2
01	0000 0**001**	(a7*2+a6)*2+a5
01	0000 0**001**	DA

4. krok

02	0000 **0010**	((a7*2+a6)*2+a5)*2
03	0000 **0011**	((a7*2+a6)*2+a5)*2+a4
03	0000 **0011**	DA

5. krok

06	000**0 0110**	(((a7*2+a6)*2+a5)*2+a4)*2
06	000**0 0110**	(((a7*2+a6)*2+a5)*2+a4)*2+a3
06	000**0 0110**	DA

6. krok

0(12)	00**00 1100**	((((a7*2+a6)*2+a5)*2+a4)*2+a3)*2
0(13)	00**00 1101**	((((a7*2+a6)*2+a5)*2+a4)*2+a3)*2+a2
13	00**01 0011**	DA

7. krok

26	0**010 0110**	(((((a7*2+a6)*2+a5)*2+a4)*2+a3)*2+a2)*2
26	0**010 0110**	(((((a7*2+a6)*2+a5)*2+a4)*2+a3)*2+a2)*2+a1
26	0**010 0110**	DA

8. krok

4(12)	**0100 1100**	((((((a7*2+a6)*2+a5)*2+a4)*2+a3)*2+a2)*2+a1)*2
4(12)	**0100 1100**	((((((a7*2+a6)*2+a5)*2+a4)*2+a3)*2+a2)*2+a1)*2+a0
52	**0101 0010**	DA

  - Pozn.: Výsledek je v komprimovaném formátu.
- Výhody:
  - lze realizovat ve smyčkách pro jakkoli dlouhé číslo.
- Nevýhody:
  - nejsou

Převody BCD → BIN

- nejvýhodnější je rozklad na Hornerovo schéma
- Př: čtyřmístné BCD číslo A = abcd

$$A_{(2)} = 10(10(10a + b) + c) + d$$

  - násobení x10 lze nahradit posuvy

$$10a = 8a + 2a = (a \ll 3) + (a \ll 1)$$