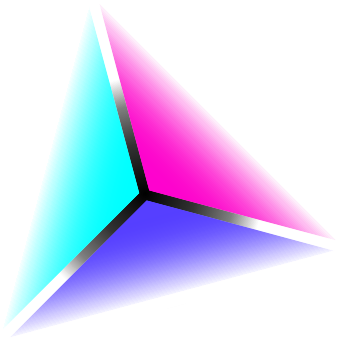
Struktura a architektura počítačů

Katedra číslicového návrhu
Fakulta informačních technologií
České vysoké učení technické

Aritmetické operace, posuvy, pohyblivá řádová čárka

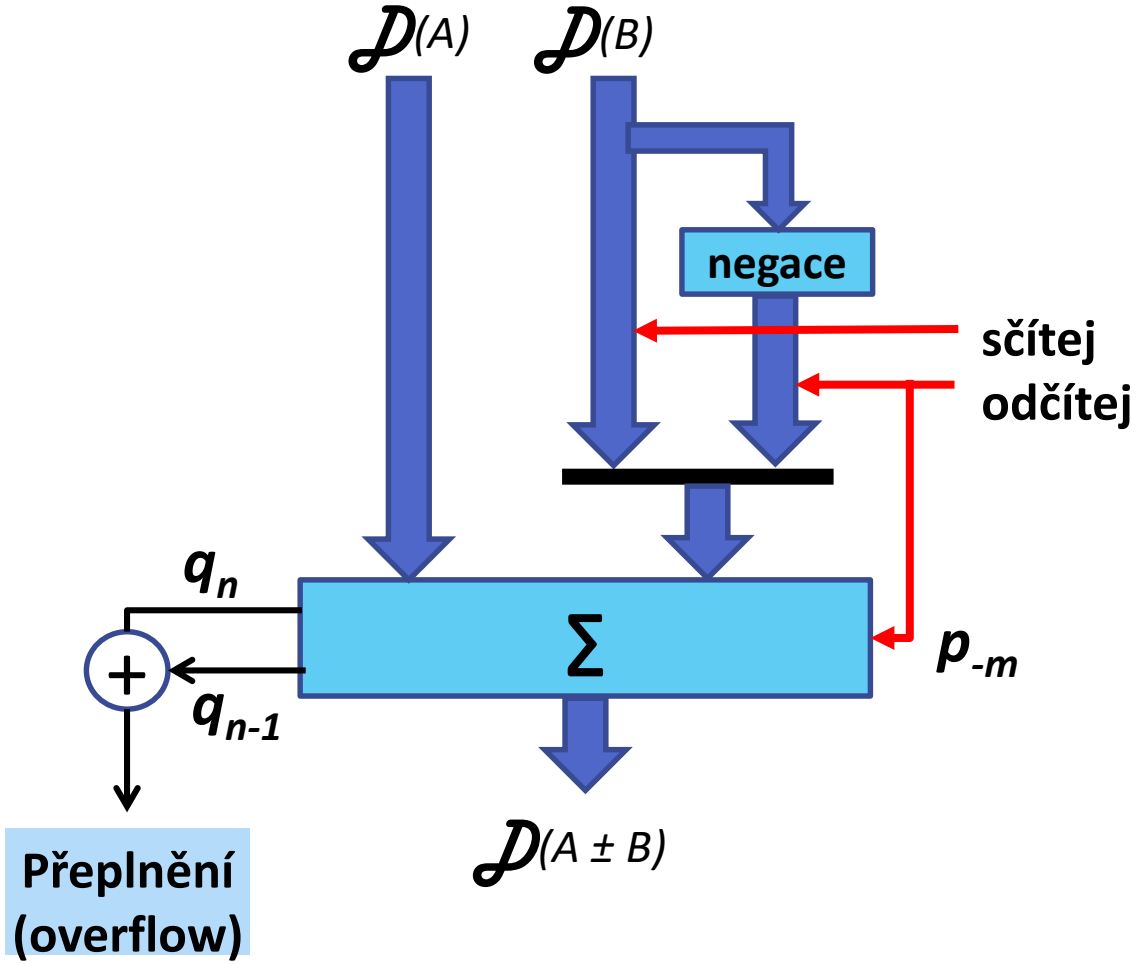
BI-SAP, březen 2021



Obsah

- Aritmetické obvody:
  - Paralelní sčítačka/odčítačka,
  - Obvody pro posuvy, násobička, dělička.
  - Sčítačka v kódu BCD
- Pohyblivá řádová čárka
  - zobrazení čísel
  - standard IEEE 754
  - příklady

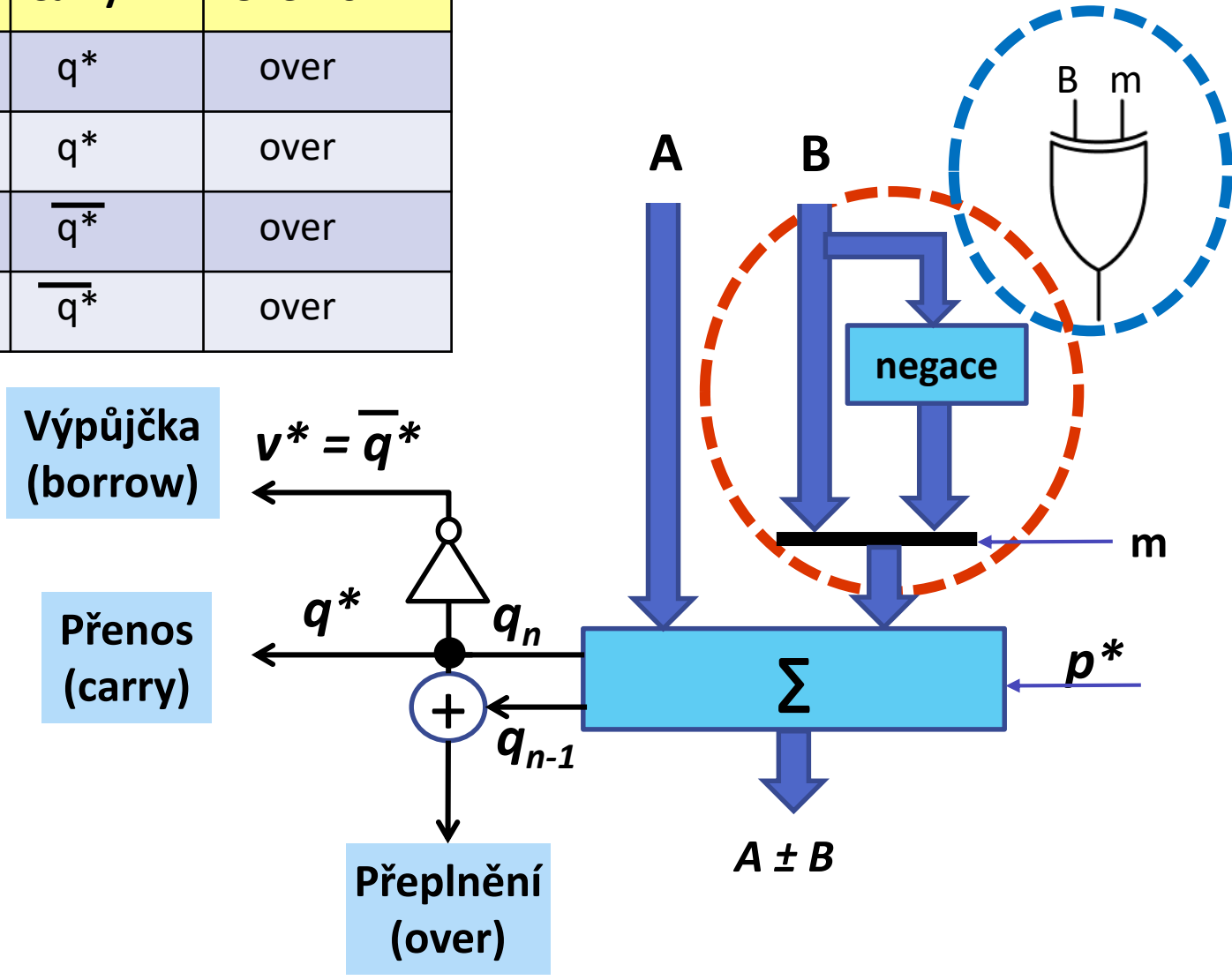
Sčítačka-odčítačka v doplňkovém kódu



DEMO:
http://www.ecs.umass.edu/ece/koren/arith/simulator/

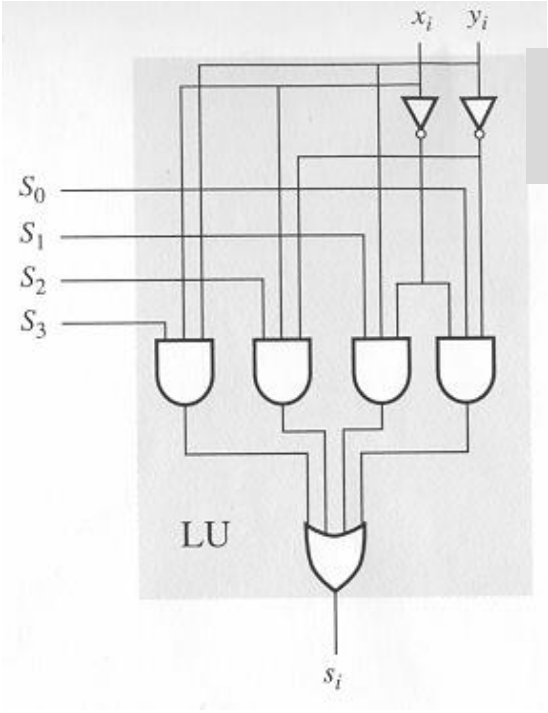
Sčítačka-odčítačka v procesoru

	m	p*	Carry	Overflow
ADD	0	0	q*	over
ADC	0	Carry	q*	over
SUB	1	1	$\overline{q^*}$	over
SBB	1	$\overline{Carry}$	$\overline{q^*}$	over



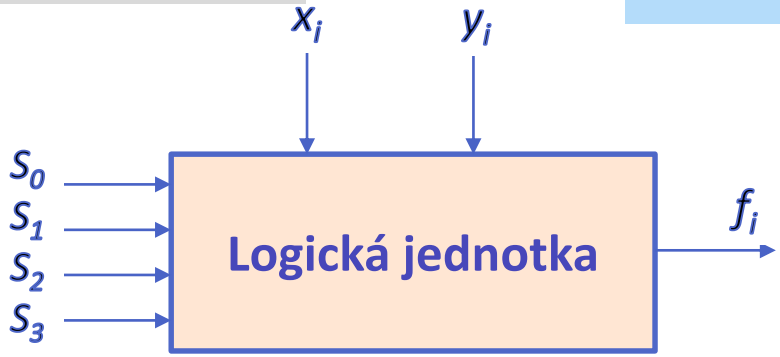
(A)LU - logická jednotka

	x_i	y_i	f0	f1	f2	F3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15
S_0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
S_1	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
S_2	1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
S_3	1	1	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1



Bloky bez přenosů mezi sebou

XOR ... $f_6 = \bar{S}_0 S_1 S_2 \bar{S}_3$



Obvykle realizuje jen 4 funkce: AND, OR, XOR, NOT

Rozšíření řádové mřížky

- záleží na kódu pro zobrazení čísel
- hodnota čísla musí zůstat stejná
  - čísla bez znaménka: doplňují se nuly
  - přímý kód: znaménko se kopíruje a doplňují se nuly
  - doplňkový kód: znaménko je organickou součástí obrazu (doplňují se nuly nebo jedničky) – *viz dále*

Rozšíření řádové mřížky v doplňkovém kódu

$$D(X) = \begin{cases} X \\ X + M \end{cases} \qquad D'(X) = \begin{cases} X \\ X + N \end{cases}$$

M a N ... moduly řádových mřížek

$$D'(X) = D(X) + \begin{cases} 0 & \text{pro } X \geq 0 \\ N - M & \text{pro } X < 0 \end{cases}$$

$M = 2^m$, $N = 2^n \rightarrow N - M = 2^n - 2^m$

$N - M = (2^{n-m} - 1) \cdot 2^m$

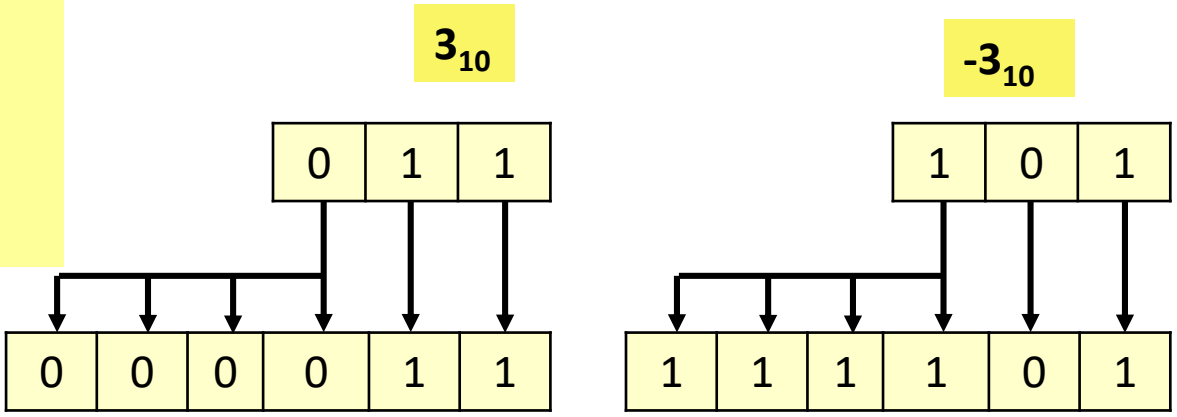
znaménkové rozšíření
sign extension

Příklad

$M = 1000_2$ ... 3bitová čísla

$N = 1000000_2$ ... 6bitová čísla

$N - M = 111000_2$



Posuvy a jejich realizace (*shifters*)

Problémy:

- *vypadávají některé bity (číslice)*
- *co uložit na uvolněná místa*

- logický posuv ... ignoruje se/nuly
- cyklický posuv ... vypadávající číslice se ve stejném pořadí nasunou na uvolněná místa
- aritmetický posuv ... detekce ***přeplnění*** (*over*)nebo ***ztráty přesnosti*** *(ZP)*.... výsledek má odpovídat přísl. násobení nebo dělení:
- ***záleží na kódu,*** *viz dále*

$A \times 2^k = A << k$

$A : 2^k = A >> k$

Aritmetické posuvy

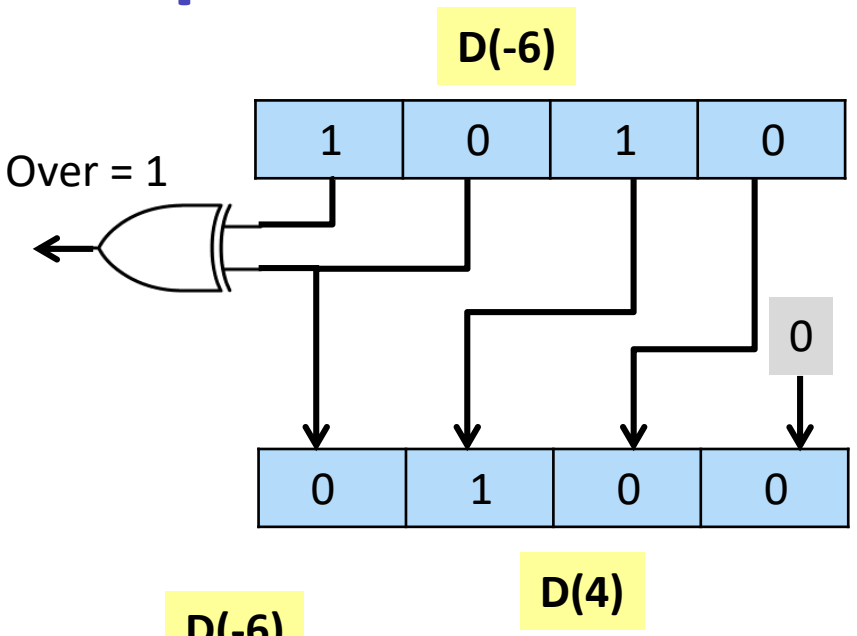
- Posuvy v doplňkovém kódu
- Posuvy v přímém kódu
- Kombinované posuvy
- Barrel shifter

} o jedno či více míst

Aritmetický posuv **doplňkovém** kódu

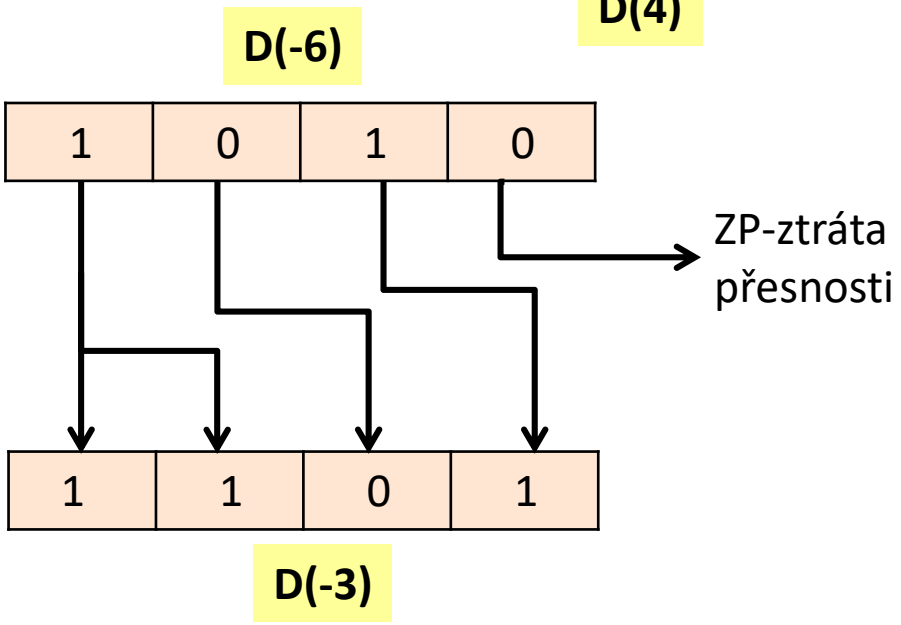
posuv vlevo ... násobení 2, 4, ...

A << = A x 2 = A + A



posuv vpravo ... dělení 2, 4, ...

A >> = A : 2



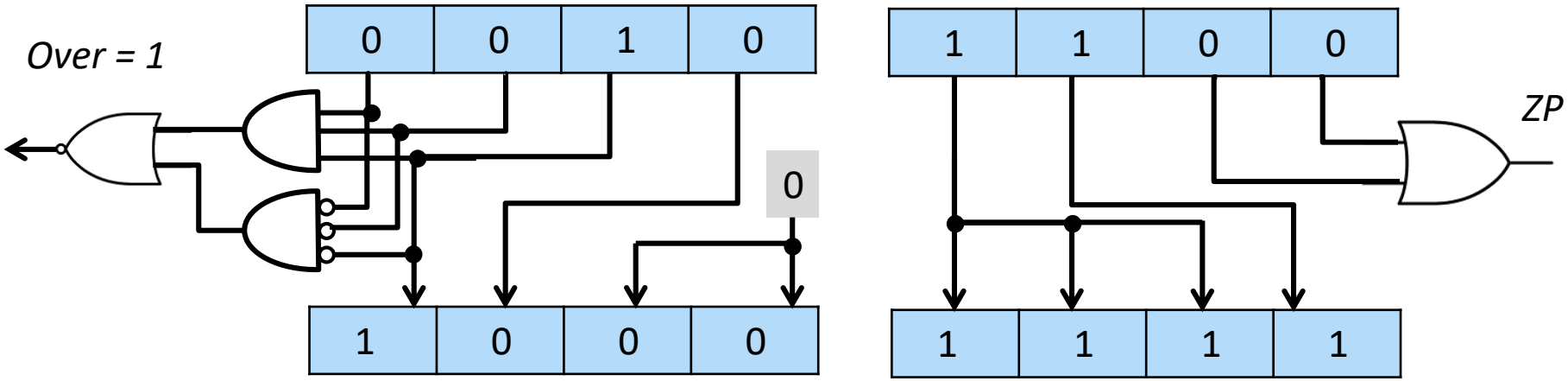
001,01 x 100 = 101,00
101,00 : 100 = 001,01

Obvody pro posuv o dvě místa

DOPLŇKOVÝ KÓD

aritmetický posuv o dvě místa vlevo
s detekcí přeplnění - *overflow*

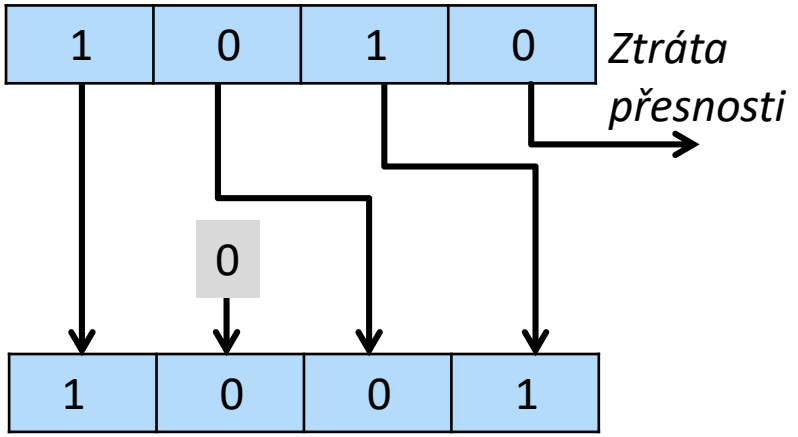
aritmetický posuv o dvě místa vpravo
s detekcí ztráty přesnosti ZP
(je stejná jako u přímého kódu)



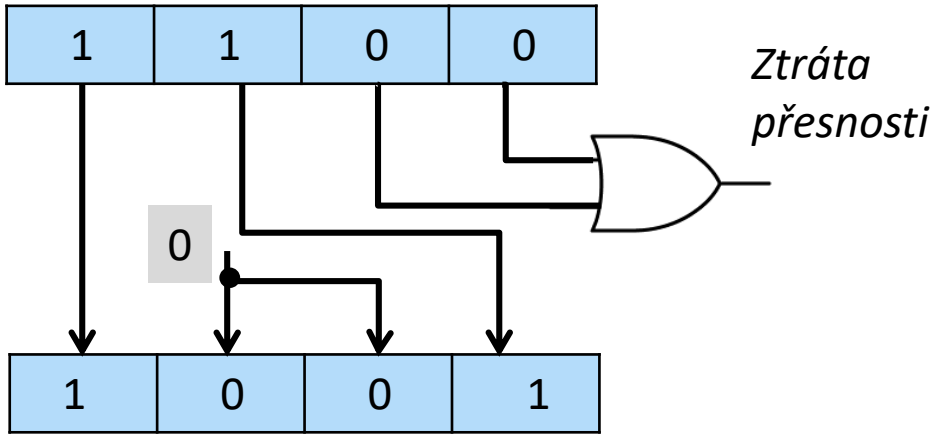
Příklad: 0010_2 (= 2_{10}) ... o 2 vlevo, tj. krát 4 .... 1000_2 (zde - 8_{10})

Přímý kód

- Vpravo o jedno místo:

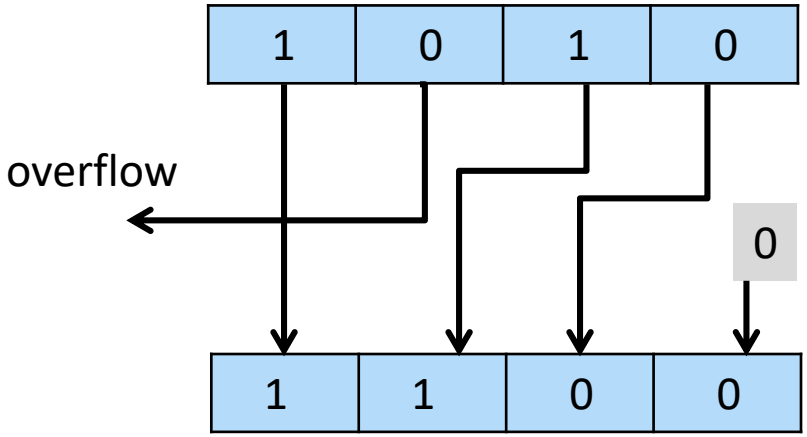


- Vpravo o dvě místa:

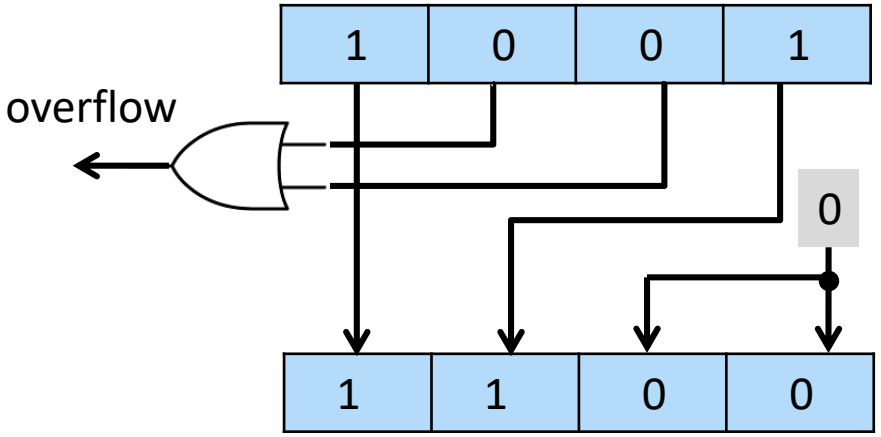


Přímý kód

- Vlevo o jedno místo:



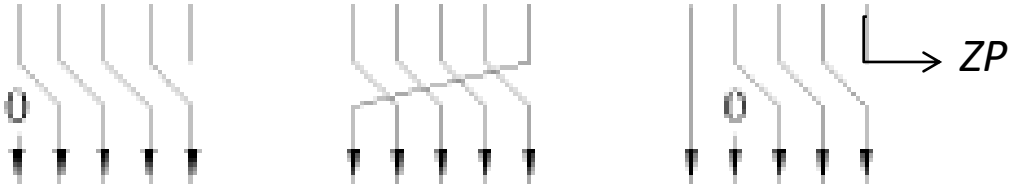
- Vlevo o dvě místa:



Kombinace posuvů pro přímý kód

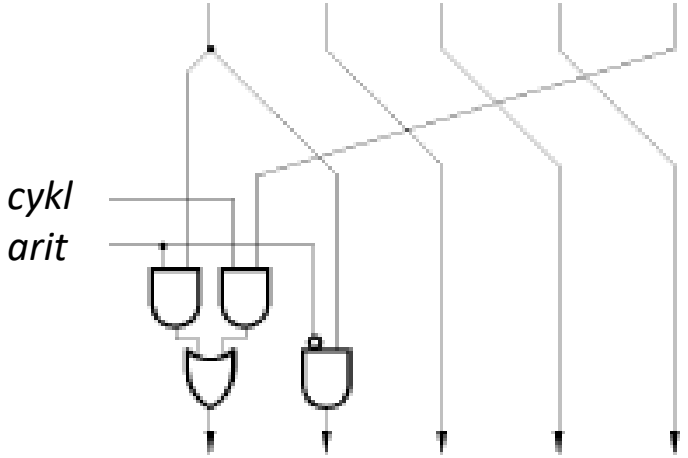
- princip posuvů o **jedno místo vpravo**:

PŘÍMÝ KÓD



logický cyklický aritmetický

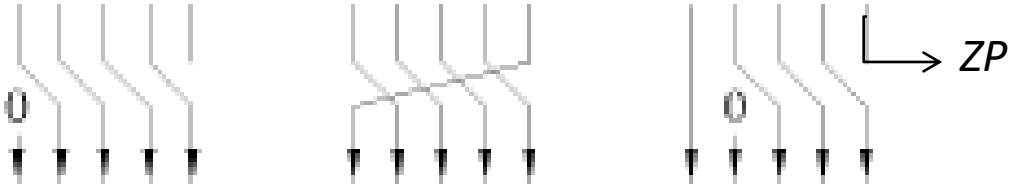
arit	cykl	Typ posuvu
0	0	logický
0	1	cyklický
1	0	aritmetický
1	1	



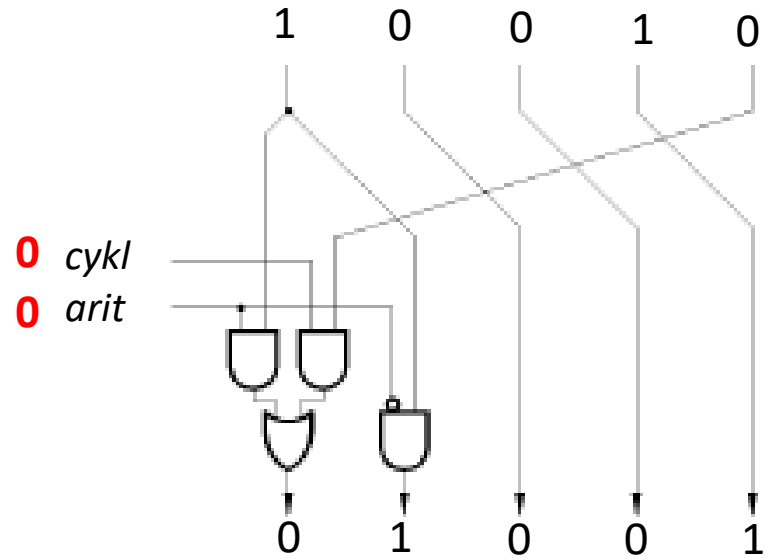
Animace: logický posuv

- princip posuvů o **jedno místo vpravo**:

PŘÍMÝ KÓD



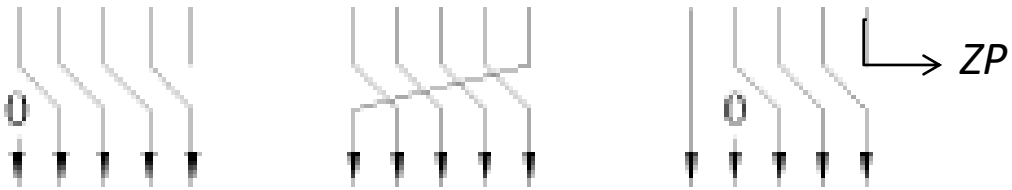
arit	cykl	Typ posuvu
0	0	logický
0	1	cyklický
1	0	aritmetický
1	1	



Animace – cyklický posuv

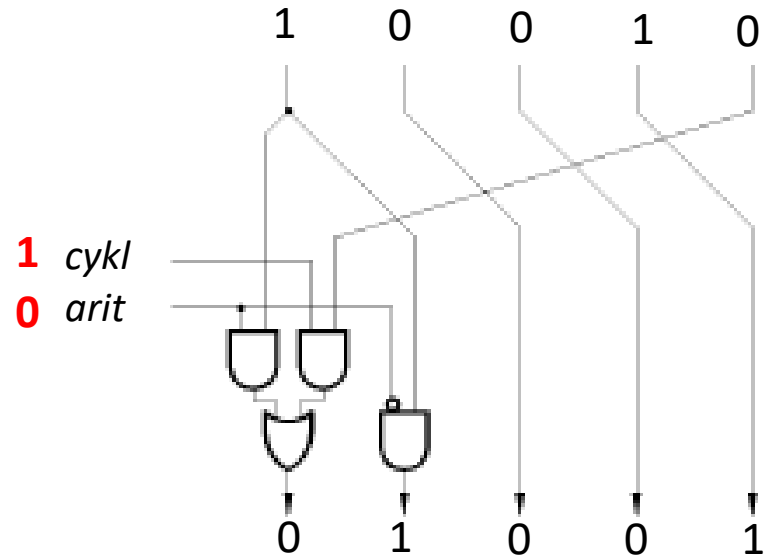
- princip posuvů o **jedno místo vpravo**:

PŘÍMÝ KÓD



logický cyklický aritmetický

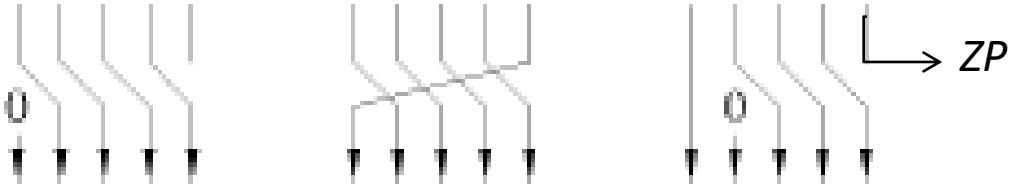
arit	cykl	Typ posuvu
0	0	logický
0	1	cyklický
1	0	aritmetický
1	1	



Animace: aritmetický posuv

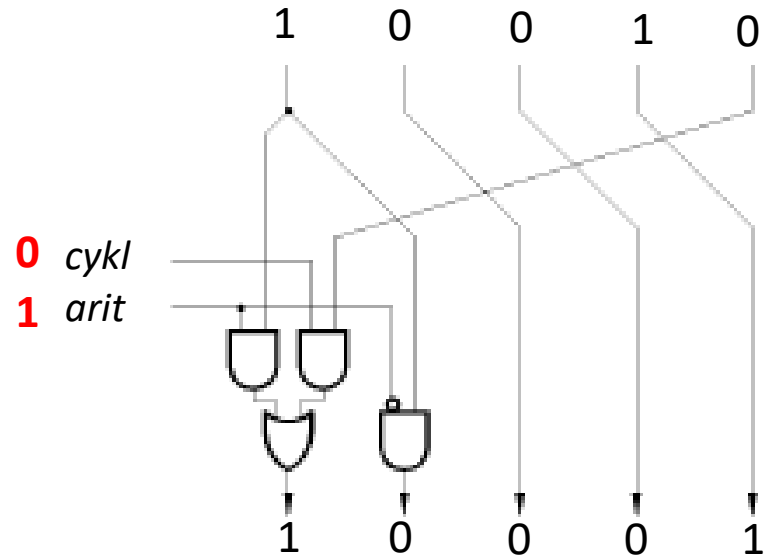
- princip posuvů o **jedno místo vpravo**:

PŘÍMÝ KÓD



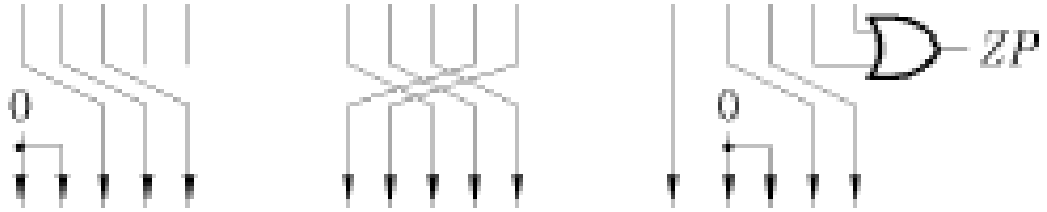
logický cyklický aritmetický

arit	cykl	Typ posuvu
0	0	logický
0	1	cyklický
1	0	aritmetický
1	1	



Posuvy o dvě místa

- posuvy o dvě místa **vpravo** s detekcí ztráty přesnosti pro aritmetický posuv

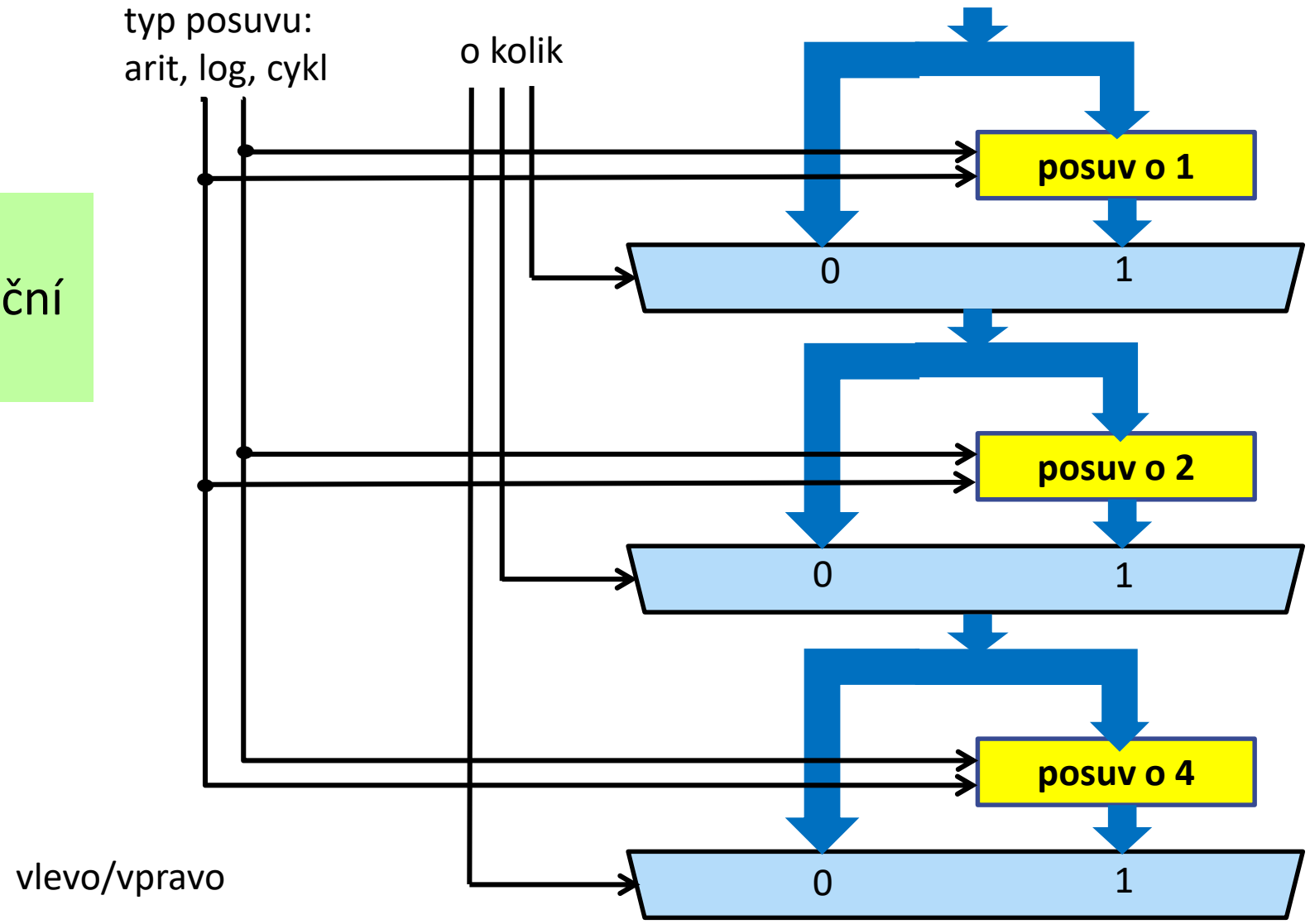


logický cyklický aritmetický

PŘÍMÝ KÓD

Barrel shifter

Je to kombinační obvod!!!



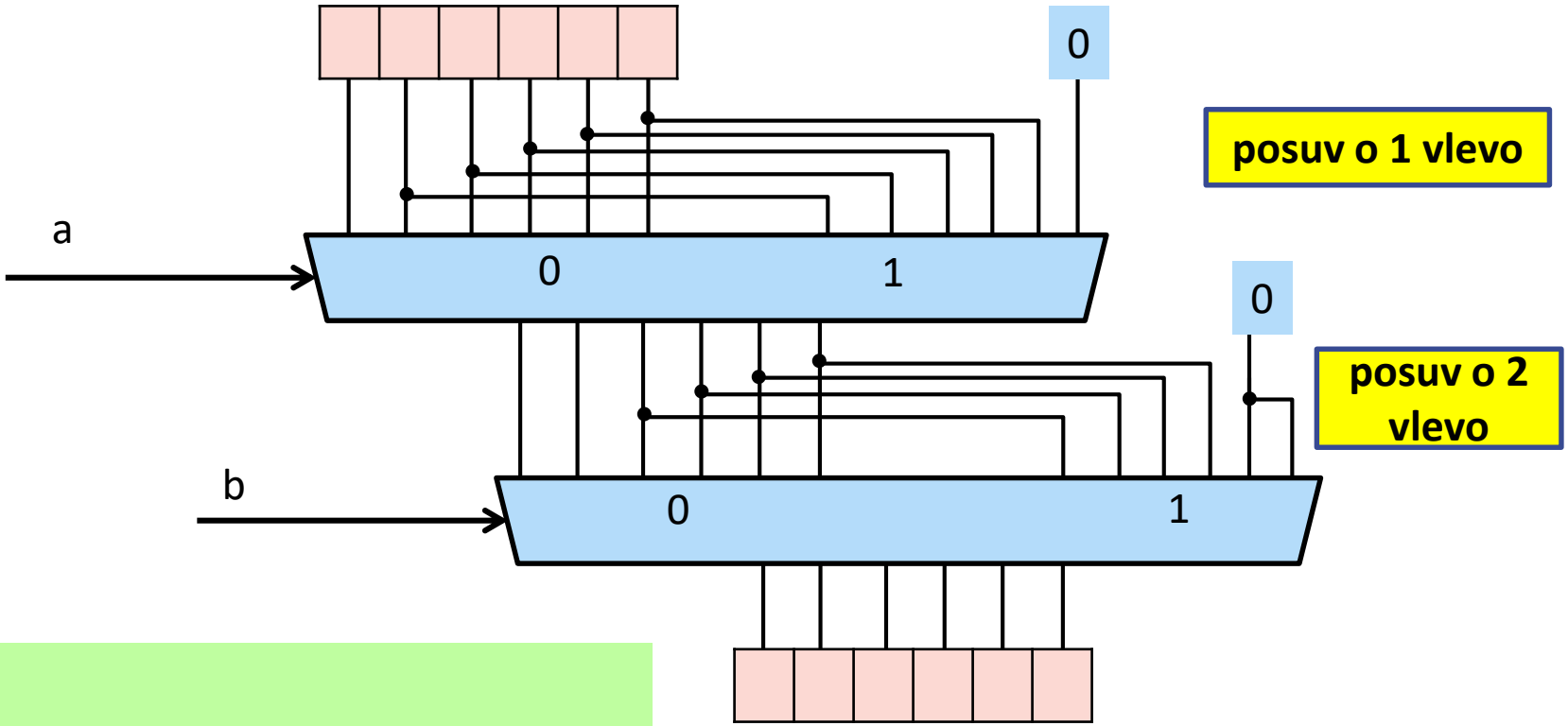
Příklad

Logický posuv šestibitového čísla o 0 až 3 bity vlevo podle dvou řídících vstupů a, b:

o_ kolik	0	1	2	3
ba	00	01	10	11

... příklad

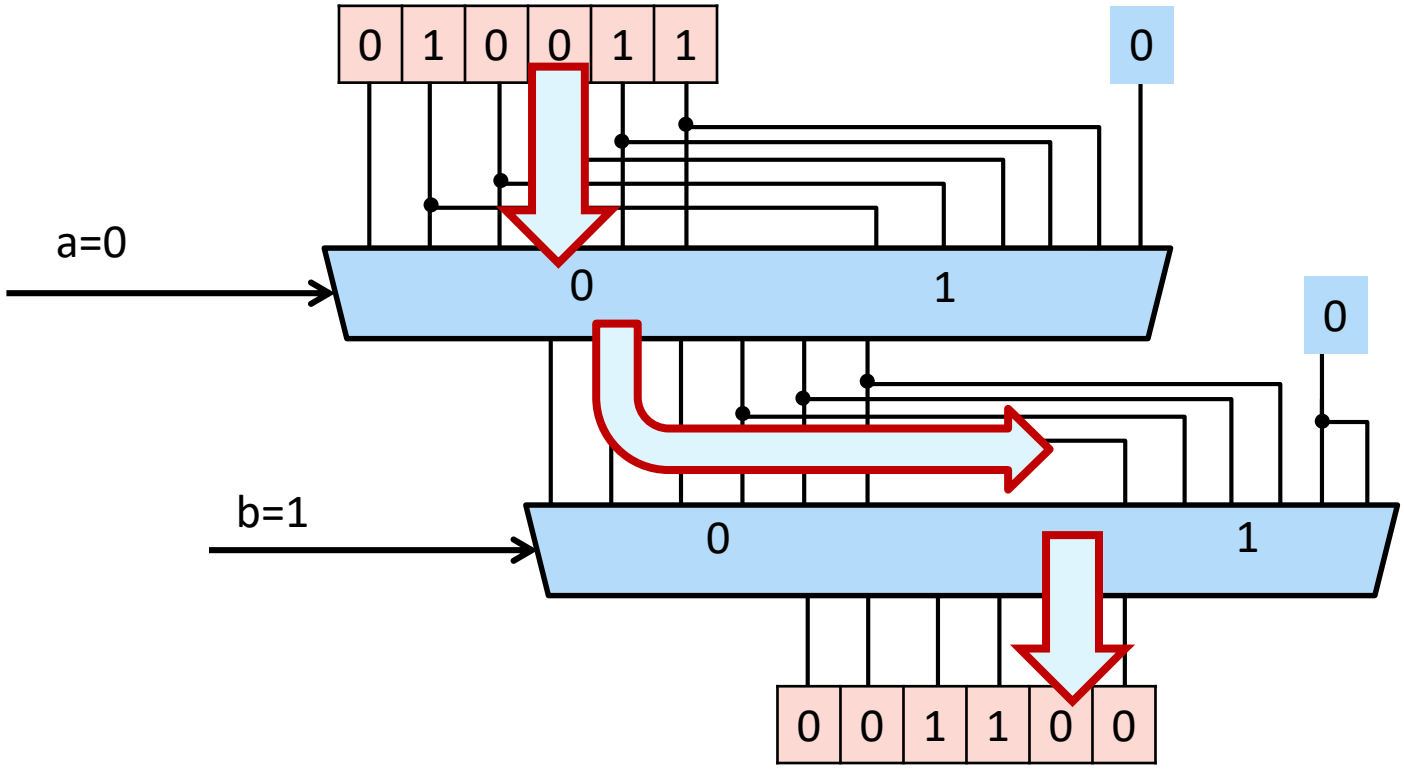
Řešení:



Řešení:
2 x MUX s 2 krát 6 vstupy) + vodiče

... příklad

Posuv dat 010011 o dvě místa vlevo (a=0 a b=1) ... 001100



Násobení

A	x	B	=	C

111 X 101 = 100 *011*

000 ↓↓↓ C1 C0

111 ←

0111

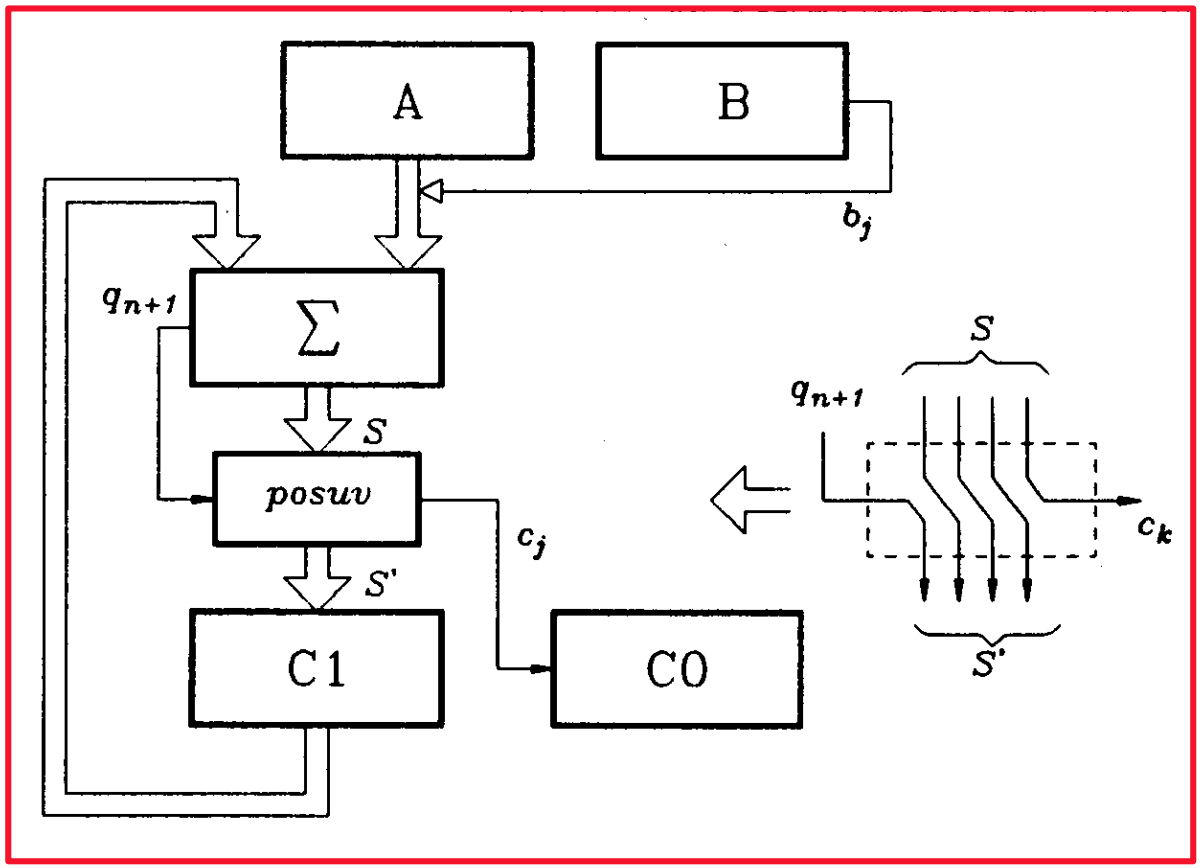
000 ←

0011

111 ←

1000

Namísto dvou registrů (B a C0) lze použít jen jeden: zprava vypadává signál pro přičtení obsahu A nebo „0“, zleva se nasouvají bity výsledku



Dělení

101 : 110 = 0,110

podíl

– 110

– 1 → 0,

návrat

+ 110

1010

– 110

1000 → 1

– 110

100 → 1

– 110

–10 → 0

návrat

+ 110

100

zbytek

návrat přes nulu ... restaurace nezáporného zbytku

pomocná operace obnovující původní dílčí zbytek pro následující odčítání

Dělení bez návratu přes nulu

- bez restaurace nezáporného zbytku

101 : 110 = 0,110 — podíl

– 110			
– 10	→	0,	
+ 110			
1000	→	1	
– 110			
100	→	1	
– 110			
–10	→	0	
+ 110	návrat		
100	zbytek		

Úkol: vydělte si na více řádů a ověřte správnost a přesnost výsledku

návrat jen v posledním kroku, je-li zbytek záporný a jen v případě, když má být správný

Srovnání

101 : 110 = 0,110			podíl
– 110			
– 1	→	0,	
+ 110			návrat
101*0*			
– 110			
100*0*	→	1	
– 110			
10*0*	→	1	
– 110			
–10	→	0	
+ 110			návrat
100			zbytek

			podíl
101 : 110 = 0,110			
– 110			
– 1*0*	→	0,	
+ 110			
100*0*	→	1	
– 110			
10*0*	→	1	
– 110			
–10	→	0	
+ 110			návrat
100			zbytek

Desítkové kódy

- zobrazení desítkových číslic
- k-bitové kódy ... $2^k >= 10$ ... $k>=4$

	BCD 8421	+3	2 4 2 1	8,4,-2,-1
0	0000	0011	0000	0000
1	0001	0100	0001	0111
2	0010	0101	0010	0110
3	0011	0110	0011	0101
4	0100	0111	0100	0100
5	0101	1000	1011	1011
6	0110	1001	1100	1010
7	0111	1010	1101	1001
8	1000	1011	1110	1000
9	1001	1100	1111	1111

Sčítačka v kódu BCD (jednomístná desítková)

a,b .. číslice sčítanců

s ... číslice součtu

p .. přenos z nižšího řádu

q .. přenos do vyššího řádu

označme ***y=a+b+p*** *... pak pro* ***y>=10*** *má být* ***q=1***

hledáme vztah mezi ***y*** a ***s*** ... korekce **-10** pro ***q=1***

-10 = -16 + 6

Př. **3+2+0 ... 05 ... 0 0101**

5+6+1 ...12 ... 0 1100+0110 ... 1 0010

8+9+0 ...17 ...1 0001+0110 ... 1 0111

10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Sčítačka v kódu BCD (pokračování)

.... *pro* **y>=10** *má být* **q=1:**

Jak poznáme, že je **y>=10** (čísla 10 až 15, ale ne 8 a 9?)

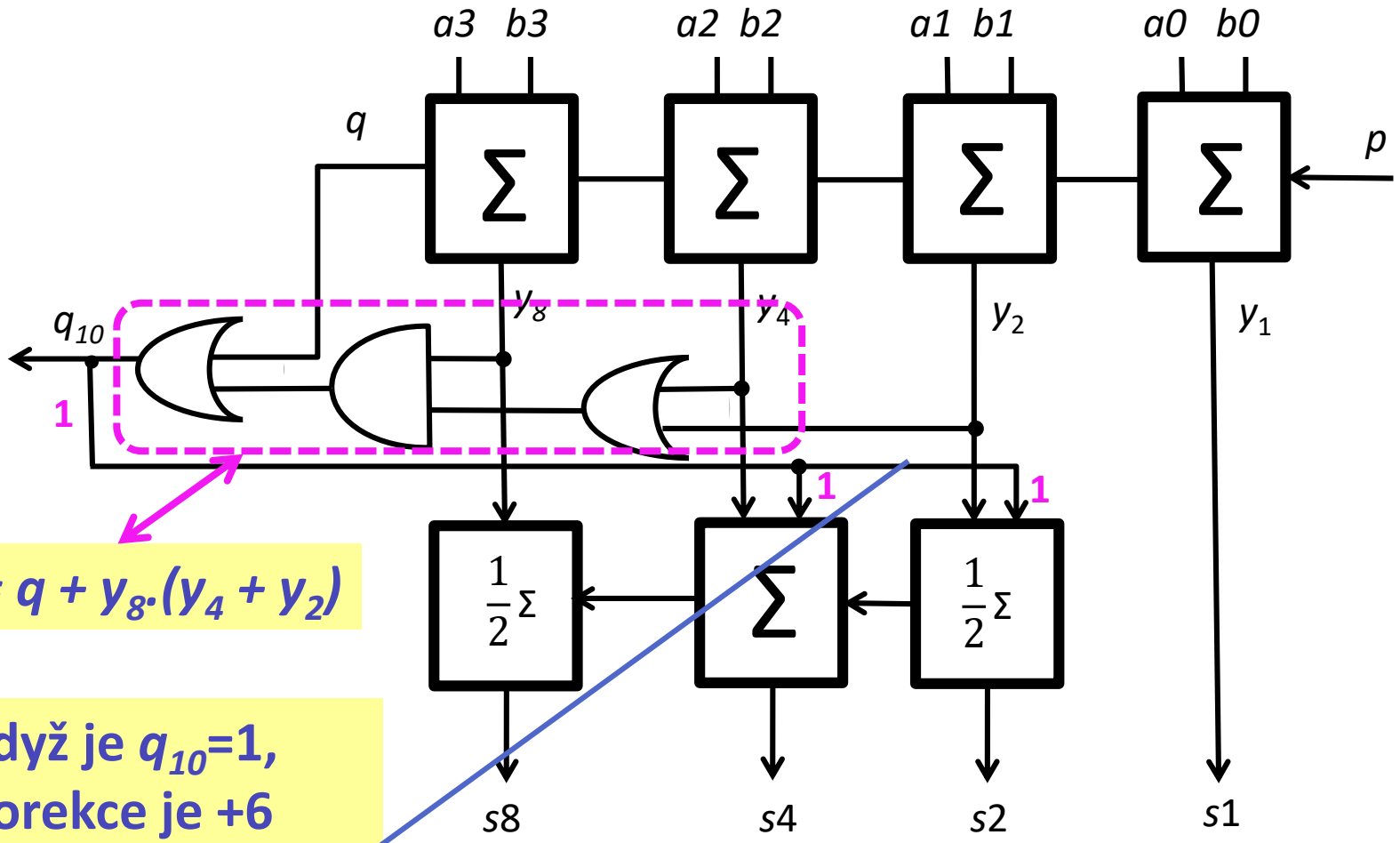
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

8	1000
9	1001

V nejvyšším řádu y_8 je jednička a zároveň je alespoň jedna jednička ve dvou prostředních (y_4 *nebo* y_2) plus korekce při přenosu, když je součet větší než 15 (q):

$$q_{10} = q + y_8.(y_4 + y_2)$$

Sčítačka v kódu BCD (realizace)

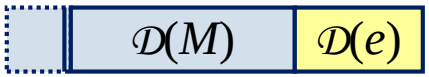


$q_{10} = q + y_8.(y_4 + y_2)$

Když je q_{10}=1, korekce je +6 + 0110 „vyrobeno“ z q_{10}

Pohyblivá řádová čárka

- Řádová mřížka má 2 části (podmřížky):
  - **mantisa** (*m*) - informace o „hodnotě“ čísla, často zlomkový tvar
  - **exponent** (*e*) - informace o pozici řád. čárky, celé číslo
- *m* i *e* používají kódy pro zobrazení čísel se znaménkem
- Ukázky možných formátů ř. m.



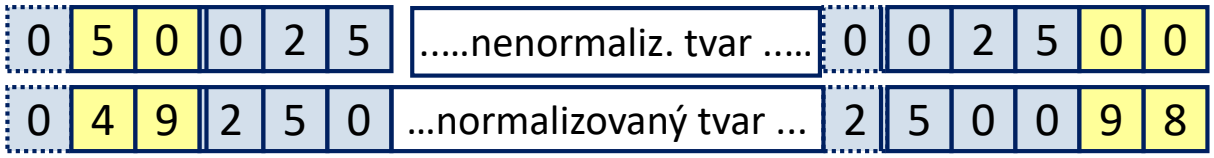
Př. $(-25 \cdot 10^2)_{10}$

Př. $(0{,}02 \cdot 10^{-5})_{10}$

...pohyblivá řádová čárka

- **Normalizovaný tvar**
  - je **tvar čísla, kdy už nelze mantisu posunout více doleva**
  - zjednodušuje aritmetické operace
- Normalizovaný tvar operandů nezaručí normalizovaný tvar výsledku

  $\Rightarrow$ **normalizace**
  - tj. **úprava výsledku na normalizovaný tvar**
  - nutno provádět po každé operaci
- Př.



$A = 0{,}025_{10}$

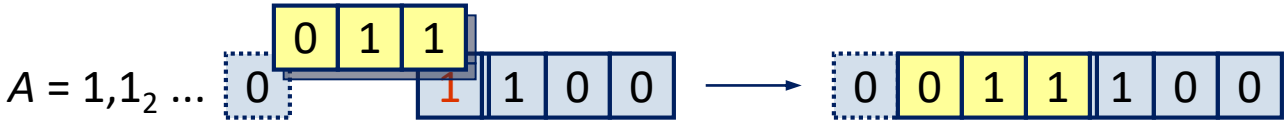
...pohyblivá řádová čárka

- **Skrytá jednička**
  - pro $z = 2$, normalizovaný tvar, **přímý kód mantisy**, $M \neq 0$,

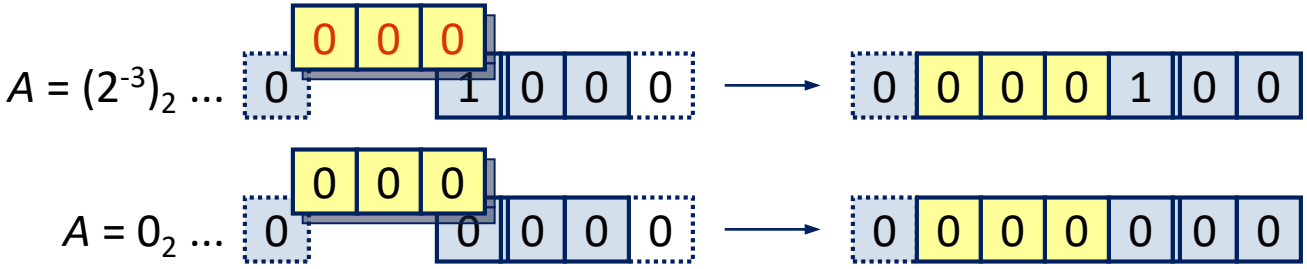
    $\mathcal{A}(e) \neq 0$, aditivní konstanta 11 (3_{10})

    $\Rightarrow$ **v nejvyšším řádu mantisy bude vždy 1**

  $\Rightarrow$ tuto 1 můžeme „skrýt“ (tj. vynechat ze zápisu čísla v ř.m.)



  - V případě $\mathcal{A}(e) = 0$ se skrytá jednička nepoužívá!!!

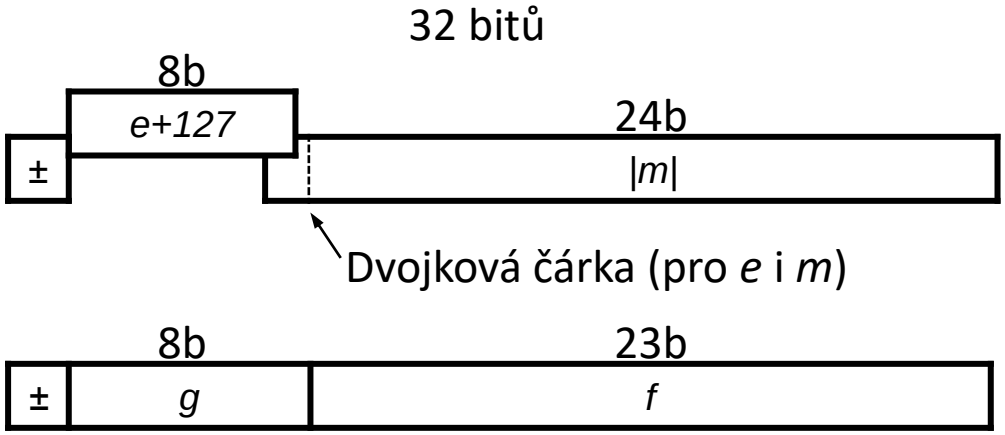


IEEE 754

	znaménko	exponent	mantisa
32 b	1b	8b	23 (24)b
64 b	1b	11b	52 (53)b

exponent – aditivní kód, K=127
mantisa – přímý kód, |M|<2

32 b:



https://cs.wikipedia.org/wiki/IEEE_754

Pohyblivá řádová čárka

Některé singulární případy:

e	M	A
0	$= 0$	0
0	$\neq 0$	$(-1)^s \cdot M \cdot 2^{-126}$
$\langle 1..254 \rangle$	-	$(-1)^s \cdot (M + 1) \cdot 2^{e-127}$
255	$= 0$	$(-1)^s \cdot \infty$
255	$\neq 0$	NaN (*Not a Number*)

Skrytá jednička!

Jak číslo uložit do paměti

Příklad: $\mathbf{-58_{10}} = -11\ 1010_2 = (-1{,}1101\ 0 \times 10^{101})_2$

$|m| = 1{,}1101\ 0 \rightarrow f = 1101\ 00....0$

$e = 101 \rightarrow g = 1000\ 0100$ ..... tzn. 127+5

1 1000 0100 1101 0000 00... 0

| C | 2 | 6 | 8 | 0 | 0 | 0 | 0 |

v little endian je tedy postupně uloženo

ve slabikově organizované paměti od adr1:

adr1 00
adr2 00
adr3 68
adr4 C2

Jaké číslo je uloženo?

| 4 | 3 | 6 | 8 | 0 | 0 | 0 | 0 |

0 100 0011 0110 1 000 0 00... 0

0 10000110 1101000 0 00... 0

znaménko exponent mantisa se skrytou 1

adr1 00
adr2 00
adr3 68
adr4 43

f = 1101 00....0 → |m| = 1,1101 0

g = 1000 0110 → e = 111 → .. tzn. 127+7 (128+6)

$(+1{,}1101\ 0 \times 10^{111})_2 = 11101000_2 = 232_{10}$

Aritmetika v pohyblivé ř.č.

- Aritmetické operace:
  - **sčítání/odčítání:** Srovnat exponenty a sečíst/odečíst mantisy.
  - **násobení:** Sečíst exponenty a vynásobit mantisy.
  - **dělení:** Odečíst exponenty a vydělit mantisy.
  - **porovnání:** Srovnat exponenty a porovnat mantisy.
  - **posuv:** Posunem mantisy nebo zvětšení/zmenšení exponentu.

- Normalizovaný tvar operandů nezaručí normalizovaný tvar výsledku
  $\Rightarrow$ **normalizace**
  - tj. **úprava výsledku na normalizovaný tvar**
  - nutno provádět po každé operaci

Jiný formát (16 bitů)

Poznámka – spočtěte doma!!!

Příklad 1:

Popis použitého formátu pro 16 bitová čísla

Y je obrazem čísla *X* v pohyblivé řádové čárce, kde:

- prvních 12 bitů (zleva) obrazu *Y* je obrazem *D(M)* mantisy *M* v doplňkovém kódu; modul řádové mřížky pro mantisu je 2, tzn.: -1 <= *M* < 1,
- zbývající 4 bity jsou rovny exponentu *E* zvýšenému o 8 (aditivní kód), tedy *A(E)* = *E*+8.
- Princip skryté jedničky není použit!

- Určete hodnotu čísla *X*, je-li *Y* = 0C0A (šestnáctkově)!

- Má obraz *Y* normalizovaný tvar? Pokud nemá, najděte obraz čísla *X* v normalizovaném tvaru!

Výsledky: $X=0{,}375_{10}$, normalizovaný tvar: *Y*= 6007

Ještě jiný formát (16 bitů)

Příklad 2:

Poznámka – spočtěte doma!!!

Popis použitého formátu pro 16 bitová čísla

Y je obrazem čísla *X* v pohyblivé řádové čárce, kde:

- první bit (zleva) je znaménko mantisy,
- následuje 4 bitový exponent zvětšený o 7 (aditivní kód)
- zbývajících 11 bitů je použito k uložení absolutní hodnoty mantisy při využití principu skryté jedničky. Modul řádové mřížky pro absolutní hodnotu mantisy je roven 2 tzn. $-2 < M < 2$.

- Určete hodnotu čísla *X*, je-li *Y*=ED00 (šestnáctkově).
- K číslu *X* přičtěte $-1000\ 0010{,}1_2$ a výsledný součet uložte ve stejném formátu jako číslo *X*.

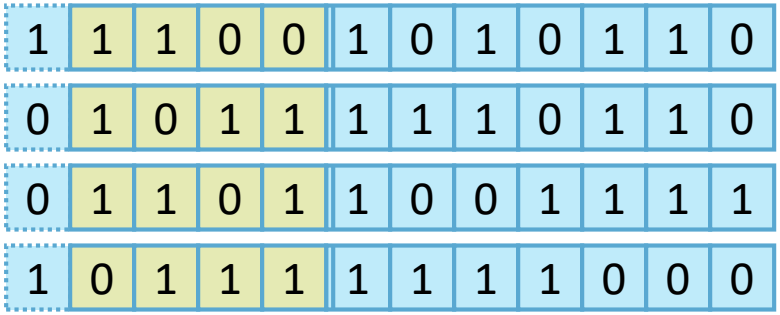
Výsledky: $X=-104_{10}$, normalizovaný tvar: *Y*= F6A8

Úloha: Zapište v normalizovaném tvaru

Poznámka – spočtěte doma!!!

- Předpokládejte délku délku ř.m. l = 12, přitom délka podmřížky exponentu je l_e = 4. Exponent v aditivním kódu, mantisa v přímém kódu (modul *M*=1,0), aditivní konstanta pro exponent je 8. Skrytá jednička není použita.

1. $-(1010{,}11_2)$
2. $7{,}375_{10}$
3. $13{,}C_{16}$
4. $-(46{,}875 \cdot 10^{-2})_{10}$



Příklad: Sčítání v pohyblivé ř.č.

- Zapište čísla $3{,}5_{10}$ a $0{,}625_{10}$ v normalizovaném tvaru a pak je sečtěte.

$$3{,}5_{10} = 11{,}1_2 = 0{,}111_2 \cdot 2^2$$

$$0{,}625_{10} = 0{,}101_2 \cdot 2^0$$

0	1	0	1	0	1	1	1	0	0	0	0

0	1	0	0	0	1	0	1	0	0	0	0

$$3{,}5_{10} + 0{,}625_{10} = 0{,}111_2 \cdot 2^2 + 0{,}101_2 \cdot 2^0 =$$
$$= (0{,}111_2 + 0{,}00101_2) \cdot 2^2 =$$
$$= (1{,}00001_2) \cdot 2^2 = 0{,}100001_2 \cdot 2^3$$

+

0	1	0	1	1	1	0	0	0	0	1	0

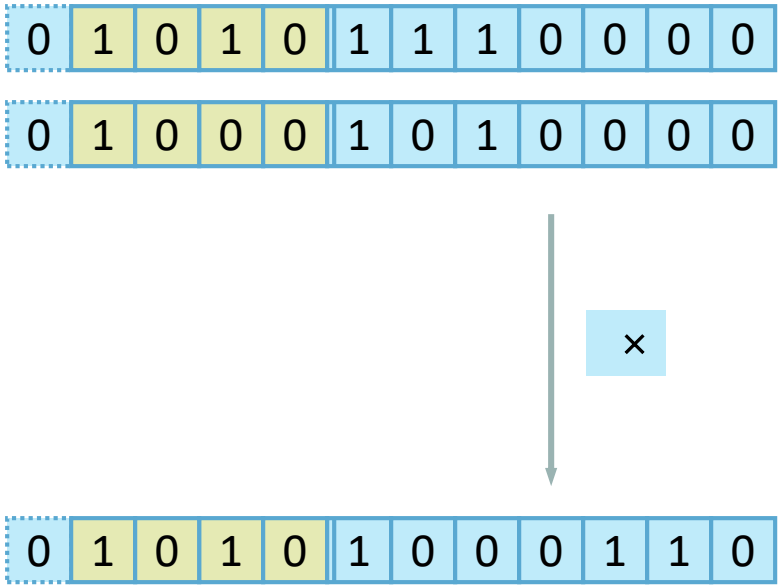
Příklad: Násobení v pohyblivé ř.č.

- Zapište čísla $3{,}5_{10}$ a $0{,}625_{10}$ v normalizovaném tvaru a pak je vynásobte.

$3{,}5_{10} = 0{,}111_2 \cdot 2^2$

$0{,}625_{10} = 0{,}101_2 \cdot 2^0$

$$
\begin{array}{rl}
 & 0{,}111_2 \\
\times & 0{,}101_2 \\
\hline
 & 0{,}0111_2 \\
+ & 0{,}00000_2 \\
+ & 0{,}000111_2 \\
\hline
 & 0{,}100011_2
\end{array}
$$



Úloha: Spočítejte v pohyblivé ř.č.

- Předpokládejte délku délku ř.m. $l = 12$, přitom délka podmřížky exponentu je $l_e = 4$. Exponent v aditivním kódu, mantisa v přímém kódu.

1. $10{,}375_{10} \times 0{,}125_{10}$
2. $13{,}625_{10} + 1{,}375_{10}$
3. $4{,}C_{16} - 3_{16}$
4. $(-0{,}40625_{10} \cdot 2^{-3}) \times (-0{,}28125_{10})$