Principy počítačů a operačních systémů

Aritmetika v počítači

Zimní semestr 2011/2012

Úvod

Jak hardware provádí aritmetické operace?

- sčítání/odčítání, násobení a dělení

Co když výsledek operace nelze reprezentovat?

- přetečení, podtečení a jejich detekce

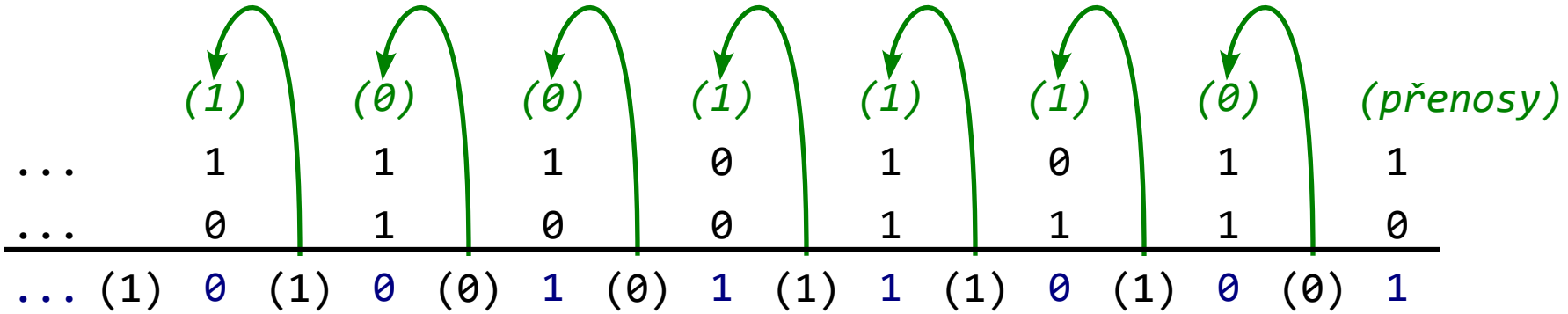
Jak je to s jinými než celými čísly?

- reprezentace zlomků a jiných reálných čísel

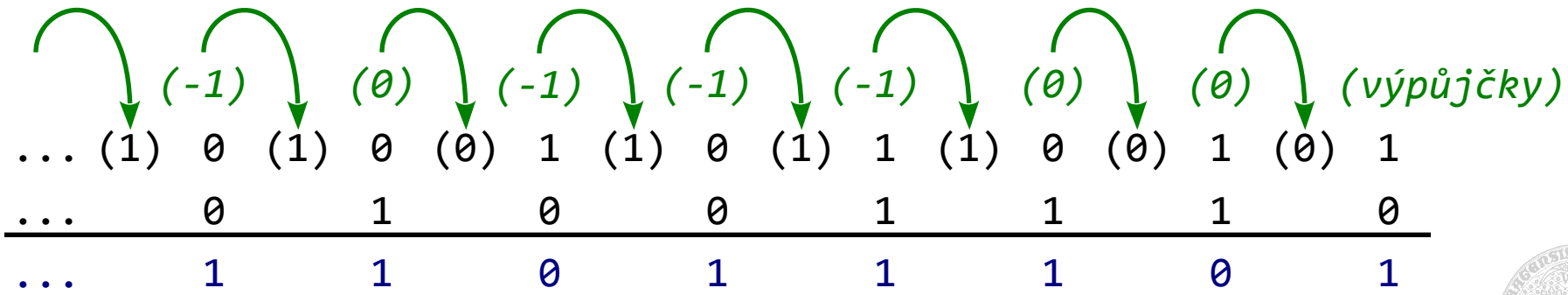
Sčítání a odčítání ve dvojkové soustavě

Sčítání a odčítání

Přenosy do vyššího řádu při sčítání



Výpůjčky z vyššího řádu při odčítání



Příklad: sčítání a odčítání

Zkusme sečíst 107_{10} a 78_{10}

$$\begin{array}{rl} & 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0110\ 1011_2 = 107_{10} \\ + & 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0100\ 1110_2 = 78_{10} \\ \hline = & 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1011\ 1001_2 = 185_{10} \end{array}$$

... a odečíst 78_{10} od 107_{10} přímo

$$\begin{array}{rl} & 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0110\ 1011_2 = 107_{10} \\ - & 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0100\ 1110_2 = 78_{10} \\ \hline = & 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 1101_2 = 29_{10} \end{array}$$

... a pomocí součtu s -78_{10} ve dvojkovém doplňku

$$\begin{array}{rl} & 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0110\ 1011_2 = 107_{10} \\ + & 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1011\ 0010_2 = -78_{10} \\ \hline = & 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 1101_2 = 29_{10} \end{array}$$

Přetečení (aneb když se výsledek nevejde)

Kdy nastává?

- při sčítání (odčítání) operandů s různými (stejnými) znaménky k přetečení ***nemůže*** dojít
  - výsledek nemůže být větší než žádný z operandů
- v opačném případě nastat ***může***

Jak poznat, že došlo k přetečení?

- při přenosu do (výpůjčce ze) znaménkového bitu
  - na výsledek je potřeba o 1 bit více ⇒ znaménkový bit místo znaménka reprezentuje bit výsledku
- jak to pozná HW?
  - přenos do nejvyššího bitu ≠ přenos z nejvyššího bitu
  - výpůjčka z 1. bitu za nejvyšším bitem ≠ výp. z nejvyššího bitu

Přetečení a vyšší programovací jazyky

Diametrálně odlišné přístupy

- Java, C – přetečení ignorováno ⇒ starost programátora
- Fortran, Ada – přetečení vyvolá výjimku

Výjimka (*exception*) resp. přerušení (*interrupt*)

- neplánované "volání funkce" - obslužná rutina
- adresa instrukce, která způsobila přetečení uložena do speciálního registru EPC (exception program counter)
  - po případné korekci může program běžet dál

Jak se program dozví o přetečení?

- kontrola po každé operaci nebo speciální instrukce CPU
  - **add**, **addi**, **sub** vs. **addu** (*add unsigned*), **addiu** a **subu** (MIPS)

Číslicové systémy

Kombinační obvody

Digitální počítač

2 úrovně napětí

- vyšší (logická 1, *high*, *true*) a nižší (logická 0, *low*, *false*)
  - signál je buď v log. 1 (asserted) nebo v log. 0 (*deasserted*)
  - hodnoty jsou vzájemně inverzní (doplňkové)

Kombinační obvody

- neobsahují paměť ⇒ výstup závisí pouze na vstupu
- reprezentují logické funkce

Sekvenční obvody

- obsahují paměť (mají vnitřní stav) ⇒ výstup závisí na obsahu paměti (vnitřním stavu) a vstupu
- umožnují zachytit posloupnost kroků výpočtu

Logické funkce a pravdivostní tabulky

Logická funkce

- výstupní hodnota je funkcí vstupních hodnot
  - Příklad: výstup Y bude v log. 1 pouze pokud právě jeden ze vstupů A a B bude v log. 1

Pravdivostní tabulka

- definice log. funkce výčtem hodnot vstupů a výstupů
  - pro ***n*** vstupů bude mít tabulka 2^n řádků
- Př.: ***X***, ***Y***, ***Z*** funkce 3 proměnných ***A***, ***B***, ***C***
  - ***X=true***, pokud ***alespoň 1*** vstup je ***true***
  - ***Y=true***, pokud ***právě 2*** vstupy jsou ***true***
  - ***Z=true***, pokud ***všechny*** vstupy jsou ***true***

Vstupy			Výstupy		
A	B	C	X	Y	Z
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	0	1

Booleova algebra

Algebraická reprezentace log. funkcí

- logické proměnné, obor hodnot { 0, 1 }
- základní logické operátory – primitivní log. funkce
  - NOT (negace), značeno $\overline{X}$, ¬X, !X
  - AND (log. součin, konjunkce), značeno $X \cdot Y$, $X \wedge Y$, X && Y
  - OR (log. součet, disjunkce), značeno X + Y, $X \vee Y$, X || Y
- další operátory/funkce (pro 2 proměnné celkem 16)
  - NAND, NOR, XOR, ...

Vstupy		Základní operátory			Univerzální operátory		Další operátory						
A	B	NOT A ¬	AND ∧	OR ∨	NAND ↑	NOR ↓	XOR ⊕	XNOR ↔	→	←	...	⊥	⊤
0	0	1	0	0	1	1	0	1	1	1	...	0	1
0	1	1	0	1	1	0	1	0	1	0	...	0	1
1	0	0	0	1	1	0	1	0	0	1	...	0	1
1	1	0	1	1	0	0	0	1	1	1	...	0	1

Vlastnosti booleovy algebry

Axiomy a odvozené vlastnosti

- idempotence: $A + A = A$, $A \cdot A = A$
- komutativita: $A + B = B + A$, $A \cdot B = B \cdot A$
- asociativita: $A + (B + C) = (A + B) + C$, $A \cdot (B \cdot C) = (A \cdot B) \cdot C$
- absorpce: $A \cdot (A + B) = A$, $A + (A \cdot B) = A$
- distributivita: $A \cdot (B + C) = (A \cdot B) + (B \cdot C)$, $A + (B \cdot C) = (A + B) \cdot (A + C)$
- neutralita 0 a 1: $A + 0 = A$, $A \cdot 1 = 1$
- agresivita 0 a 1: $A + 1 = 1$, $A \cdot 0 = 0$
- komplementarita: $A + \overline{A} = 1$, $A \cdot \overline{A} = 0$
- absorpce negace: $A \cdot (\overline{A} + B) = A \cdot B$, $A + (\overline{A} \cdot B) = A + B$
- DeMorganovy zákony: $\overline{A + B} = \overline{A} \cdot \overline{B}$, $\overline{A \cdot B} = \overline{A} + \overline{B}$
- dvojitá negace: $\overline{\overline{A}} = A$

Manipulace s logickými funkcemi

- zjednodušení zápisu, použití omezené množiny operátorů, ...

Příklad: logické funkce

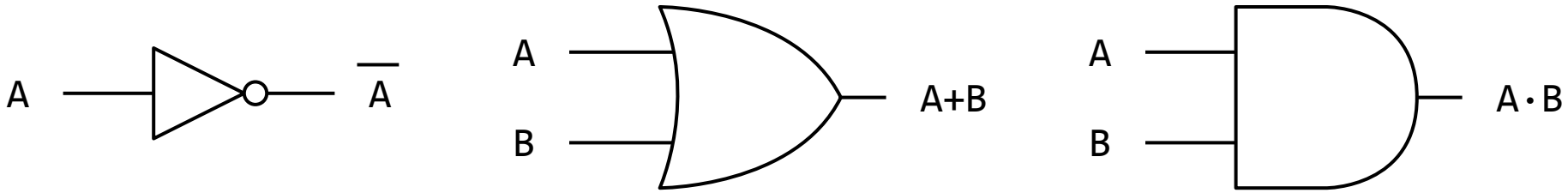
X, *Y*, *Z* funkce 3 proměnných *A*, *B*, *C*

- X=true, pokud alespoň 1 vstup je true
  $X = A + B + C$
- Z=true, pokud všechny vstupy jsou true
  $Z = A \cdot B \cdot C$
- Y=true, pokud právě 2 vstupy jsou true
  $Y = ((A \cdot B) + (A \cdot C) + (B \cdot C)) \cdot \overline{(A \cdot B \cdot C)}$
  $Y = (A \cdot B \cdot \overline{C}) + (A \cdot C \cdot \overline{B}) + (B \cdot C \cdot \overline{A})$

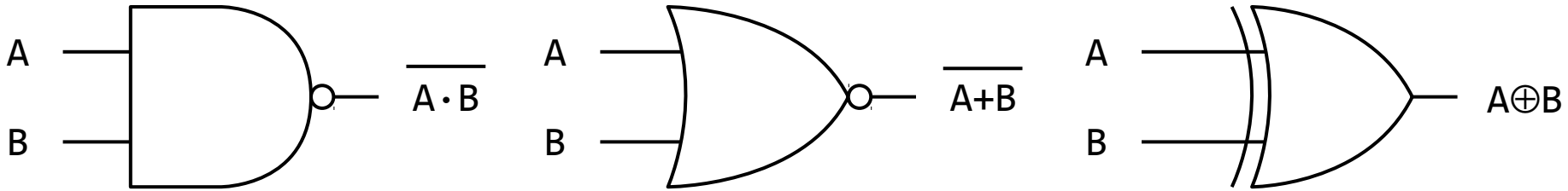
Hradla – základní prvek logických obvodů

Realizují logické operátory

- základní NOT, OR, AND



- odvozené NAND, NOR, XOR



Logické obvody – realizace logických funkcí

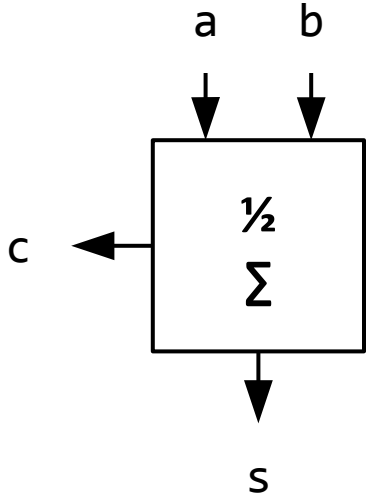
- logické signály jako proměnné, hradla jako operátory
- nejčastěji hradla NAND/NOR

HW realizace sčítání a odčítání

Nejjednodušší případ sčítání...

Součet dvou 1-bitových čísel

- vstupy: operand ***a***, operand ***b***
- výstupy: součet ***s***, přenos ***c***



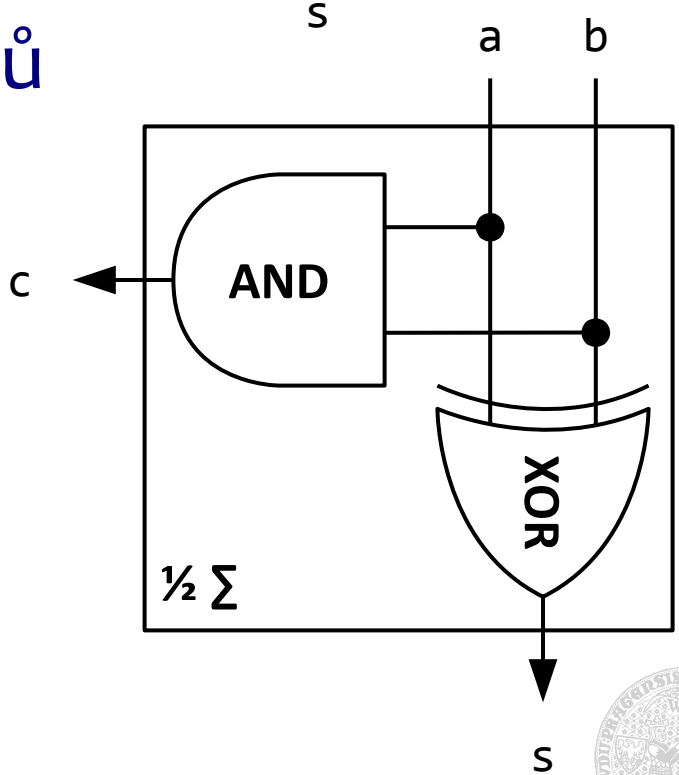
Výstupy jako logická funkce vstupů

- $s = a \cdot \overline{b} + \overline{a} \cdot b$

  $s = a \ XOR \ b$
- $c = a \cdot b$

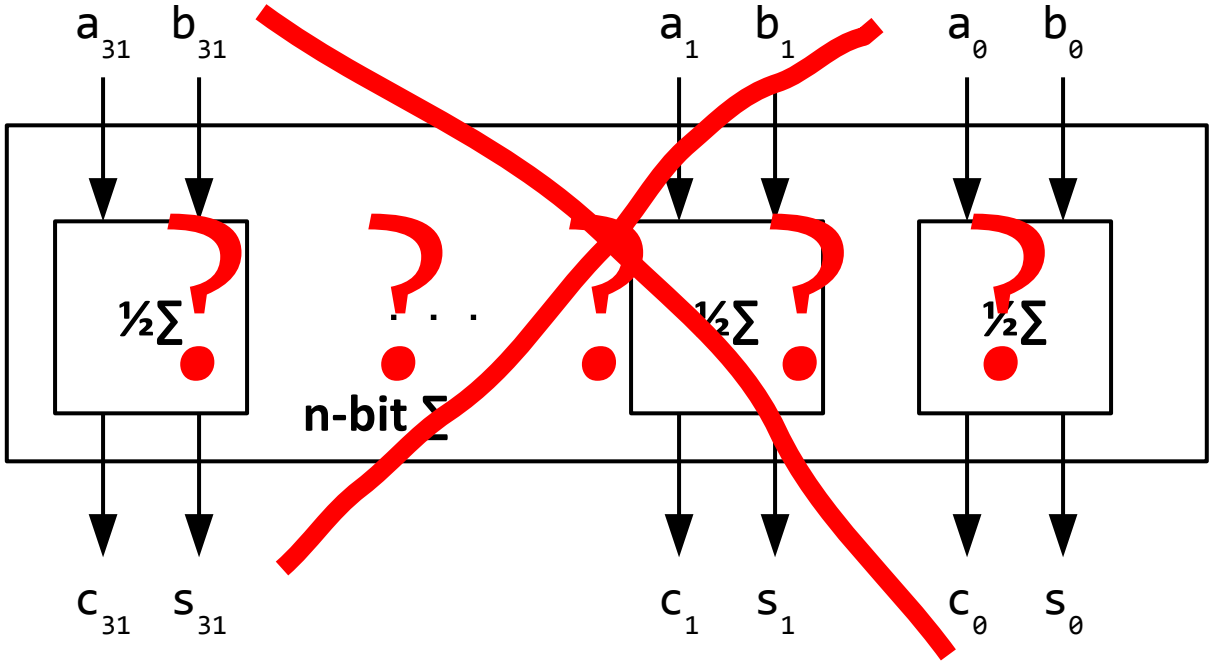
  $c = a \ AND \ b$

Vstupy		Výstupy	
a	b	s	c
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1



... ale my chceme sčítat n-bitová čísla

Spojíme 32 ½ sčítaček pro jednotlivé bity?



Co nám chybí?

- potřebujeme propagovat přenosy mezi řády
- poloviční sčítačka to neumí – má málo vstupů

Přidáme vstup pro přenos z nižšího řádu...

Součet dvou čísel

- 3 vstupy: čísla ***a***
  přenos z nižšíh
- 2 výstupy: souče
  do vyššího řádu

$$s = \overline{c_i}\cdot(\overline{a}\cdot b + a\cdot\overline{b}) + c_i\cdot(\overline{a}\cdot\overline{b} + a\cdot b)$$

$$s = \overline{c_i}\cdot(\overline{a}\cdot b + a\cdot\overline{b}) + c_i\cdot\overline{(\overline{a}\cdot\overline{b} + a\cdot b)}$$

$$s = \overline{c_i}\cdot(\overline{a}\cdot b + a\cdot\overline{b}) + c_i\cdot\overline{((\overline{a}\cdot\overline{b})\cdot(a\cdot b))}$$

$$s = \overline{c_i}\cdot(\overline{a}\cdot b + a\cdot\overline{b}) + c_i\cdot\overline{((a+b)\cdot(\overline{a}+\overline{b}))}$$

$$s = \overline{c_i}\cdot(\overline{a}\cdot b + a\cdot\overline{b}) + c_i\cdot\overline{(a\cdot\overline{a} + a\cdot\overline{b} + \overline{a}\cdot b + b\cdot\overline{b})}$$

$$s = \overline{c_i}\cdot(\overline{a}\cdot b + a\cdot\overline{b}) + c_i\cdot\overline{(a\cdot\overline{b} + \overline{a}\cdot b)}$$

Výstup jako logická fu

- $s = \overline{c_i}\cdot(\overline{a}\cdot b + a\cdot\overline{b}) + c_i\cdot(\overline{a}$ $a\cdot b)$

  $s = c_i$ XOR $(a$ XOR $b)$

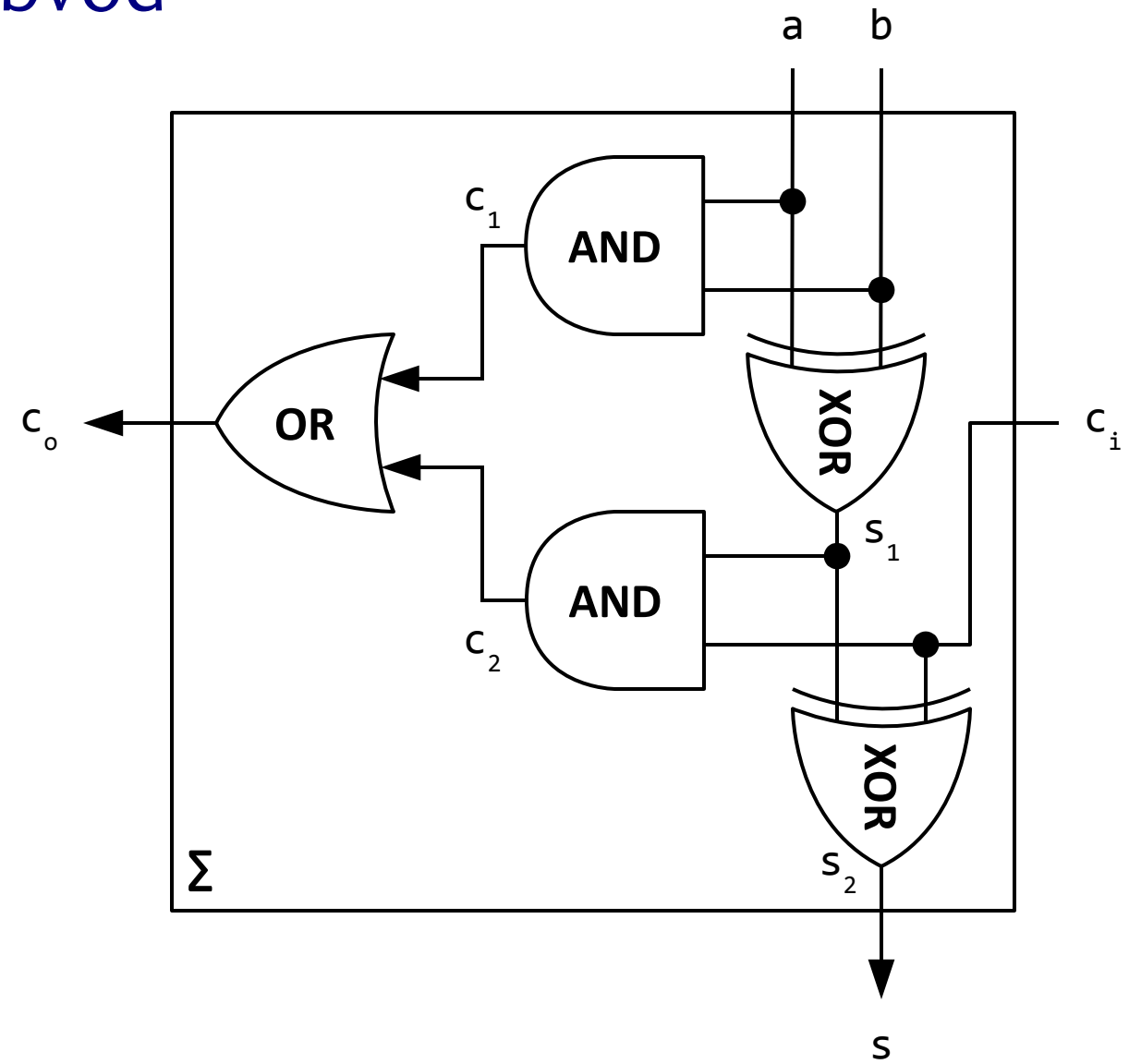
- $c_o = a\cdot b + c_i\cdot(\overline{a}\cdot b + a\cdot\overline{b})$

  $c_o = a$ AND $b + c_i$ AND $(a$ XOR $b)$

Vstupy			Výstupy	
c_i	a	b	s	c_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Úplná sčítačka

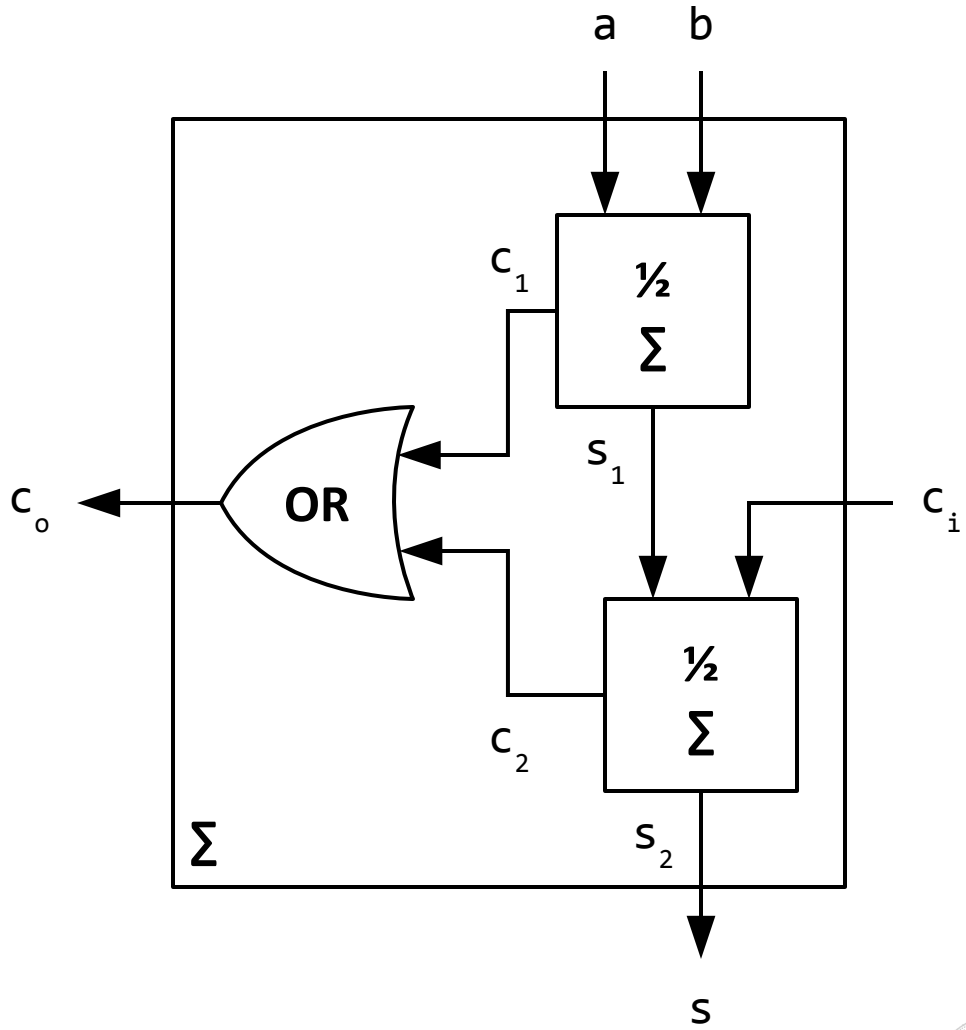
Logický obvod



Úplná sčítačka

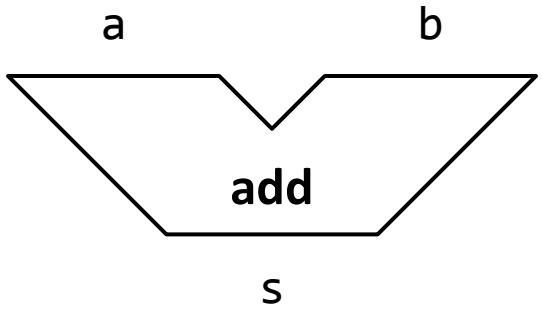
Abstrakce

- $a + b + c_i = (a + b) + c_i$
- 2 poloviční sčítačky
- přenosy vznikají při obou součtech

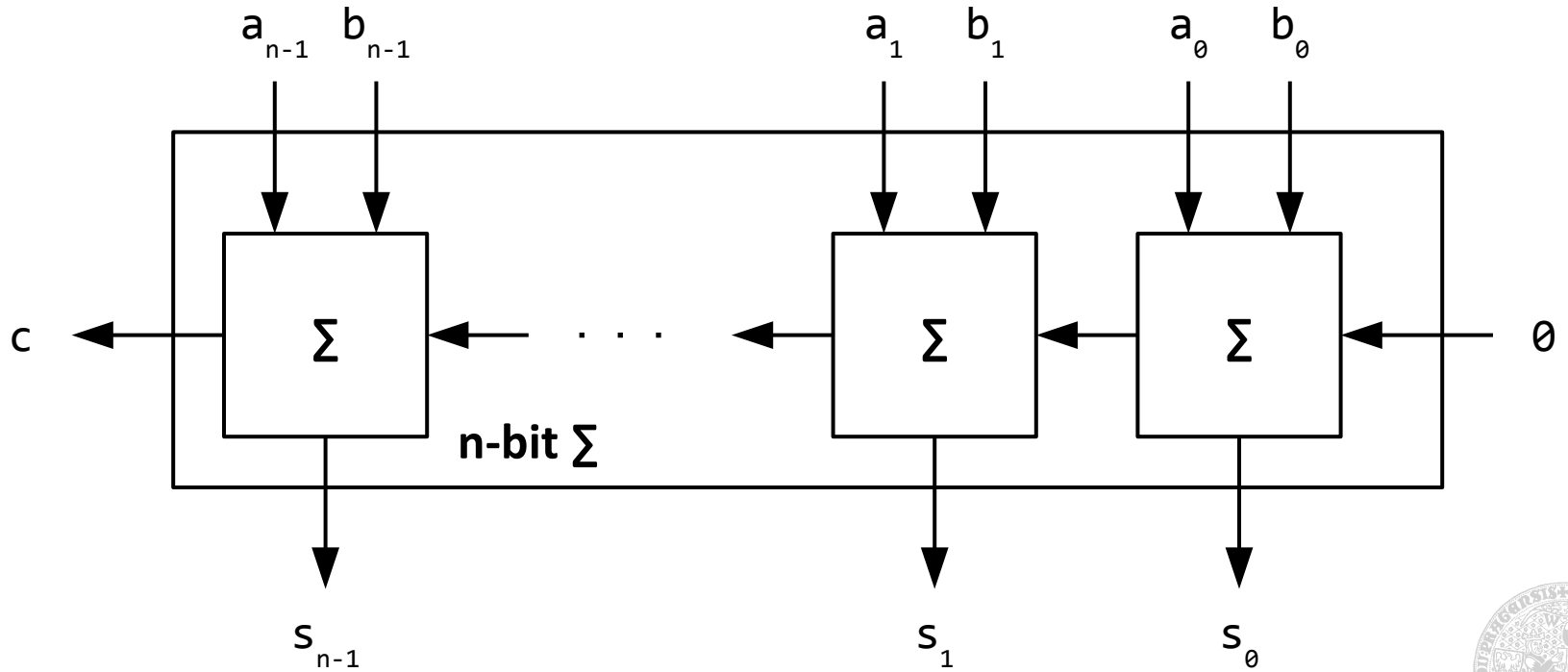


Sčítačka dvou n-bitových čísel

Funkční blok



Vnitřní struktura



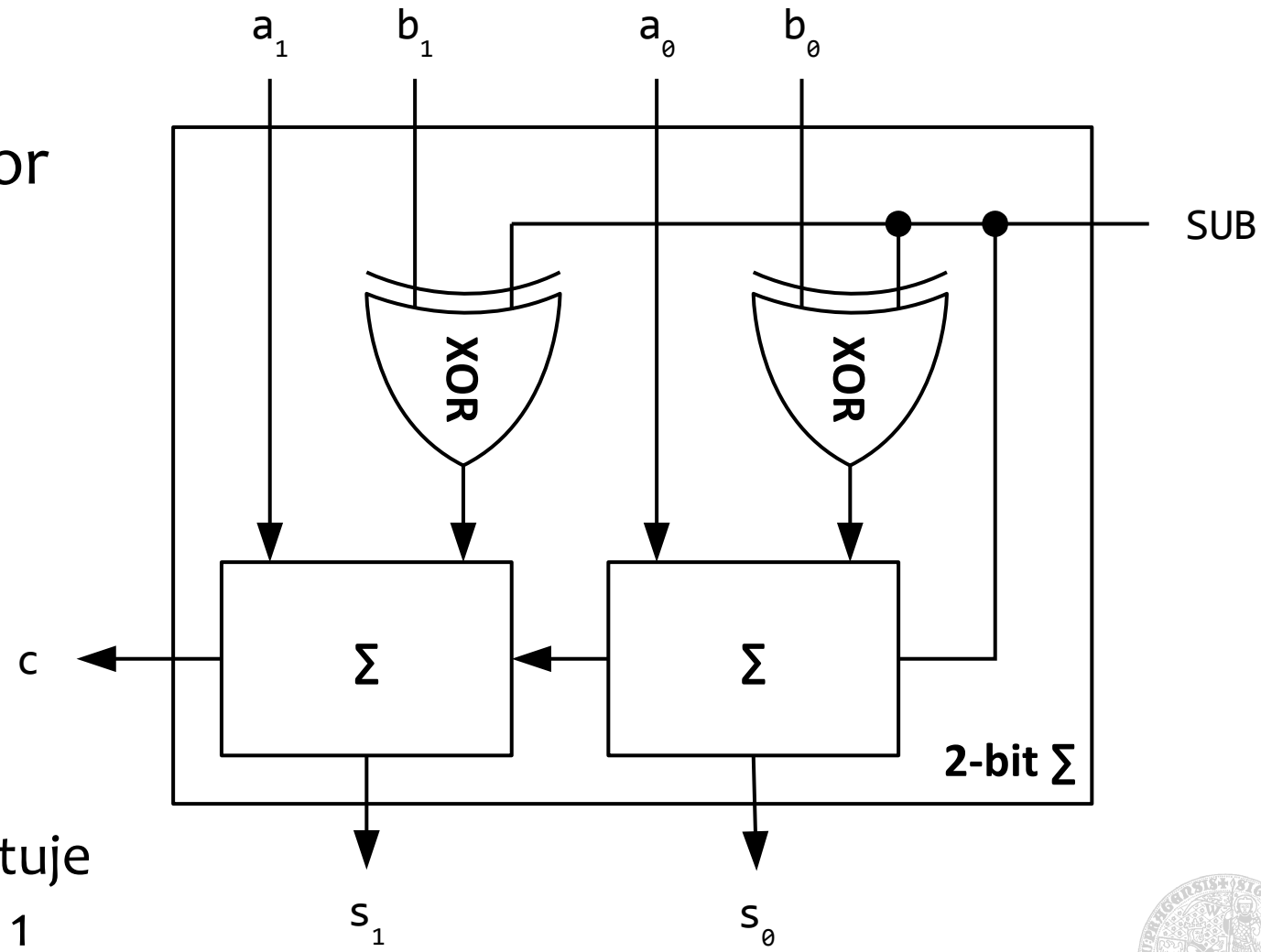
Potřebujeme také odčítat...

2-bitová ALU s podporou sčítání/odčítání

- XOR hradlo funguje jako řízený invertor

a	b	XOR
0	0	0
1	0	1
0	1	1
1	1	0

- řídící vstup určuje typ operace
  - SUB=1 invertuje bity a přičte 1



Násobení a dělení ve dvojkové soustavě

Násobení přirozených čísel

Obecný postup

- stejný jako v 10-kové soustavě
- reprezentace výsledku vyžaduje dvojnásobný počet bitů

Příklad: $6_{10} \times 13_{10}$

$0110_2 = 6_{10}$

$1101_2 = 13_{10}$

```
  0110
 0000
0110
0110
--------
01001110  = 78(10)
```

Co když budeme násobit celá čísla?

Násobení celých čísel

Příklad: $3_{10} \times -5_{10}$

$0011_2 = 3_{10}$

$1011_2 = -5_{10}$

```
  0011
 0011
0000
0011
--------
00100001 = 33
```

Jak to spravit?

- nutno uvažovat čísla s nekonečným počtem cifer v omezené reprezentaci

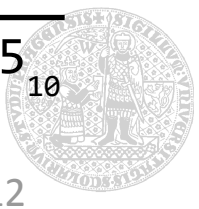
Příklad: $3_{10} \times -5_{10}$

- znaménkové rozšíření na výslednou velikost reprezentace

$00000011_2 = 3_{10}$

$11111011_2 = -5_{10}$

```
  00000011
 00000011
00000000
00000011
 00000011
  00000011
   00000011
    00000011
--------
0000001011110001 = -15
```



Problémy s násobením celých čísel

Znaménkové rozšíření před výpočtem

- dvojnásobný počet součtů
  - velký počet jedniček v násobiteli vede na nutnost provést mnoho sčítání
- čtyřnásobná šířka (dočasného) výsledku

Úspornější postup bez rozšíření

- mezisoučet inicializován na 0
- aritmetický posun mezisoučtu vpravo v každém kroku
- v posledním kroku se násobenec k mezisoučtu přičte/odečte podle znaménka násobitele

Násobení bez rozšíření před operací

Příklad: $3_{10} \times -5_{10}$

```
   0011
 × 1011
------------
   0000 0000 ... počáteční stav
   0011 0000 ... přičíst násobenec
   0001 1000 ... aritmetický posun vpravo
   0100 1000 ... přičíst násobenec
   0010 0100 ... aritmetický posun vpravo
   0010 0100 ... ignorovat násobenec
   0001 0010 ... aritmetický posun vpravo
   1110 0010 ... odečíst násobenec (znaménko)
------------
   1111 0001 ... aritmetický posun vpravo (výsledek)
```

Operace dělení

Pouze přirozená čísla, výsledkem podíl a zbytek

- znaménko výsledku odpovídá znaménkům operandů
- znaménko zbytku odpovídá znaménku dělence

  dělenec = dělitel × podíl + zbytek

- dělení nulou – neplatná operace

Dělení ve dvojkové soustavě

- postupné odečítání dělitele od nejvyššího řádu dělence
- obvodová realizace
  - dělení s návratem, dělení bez návratu
  - rychlé dělení

Příklad: dělení přirozených čísel s návratem

Vypočtěte podíl $11_{10} \div 2_{10}$

```
   1011₂ ÷ 10₂ = q + r
 - 1000          ... odečíst dělitel
 ------
   0011   ≥ 0    ... kvocient << 1, q0 = 1, dělitel >> 1
 - 0100          ... odečíst dělitel
 ------
   1111   < 0    ... kvocient << 1, q0 = 0, přičíst dělitel
   0011          ... zbytek, dělitel >> 1
 - 0010          ... odečíst dělitel
 ------
   0001   ≥ 0 ... kvocient << 1, q0 = 1, dělitel >> 1
   0001          ... dělitel == 1 ⇒ konec, zbytek = 1
```

- $11_{10} \div 2_{10} = 0101_2$ (0001_2)

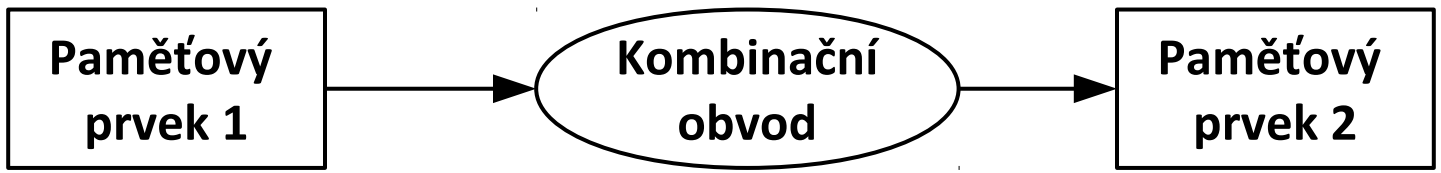
Číslicové systémy

Sekvenční obvody

Sekvenční obvody

Kombinační obvody + paměťové prvky

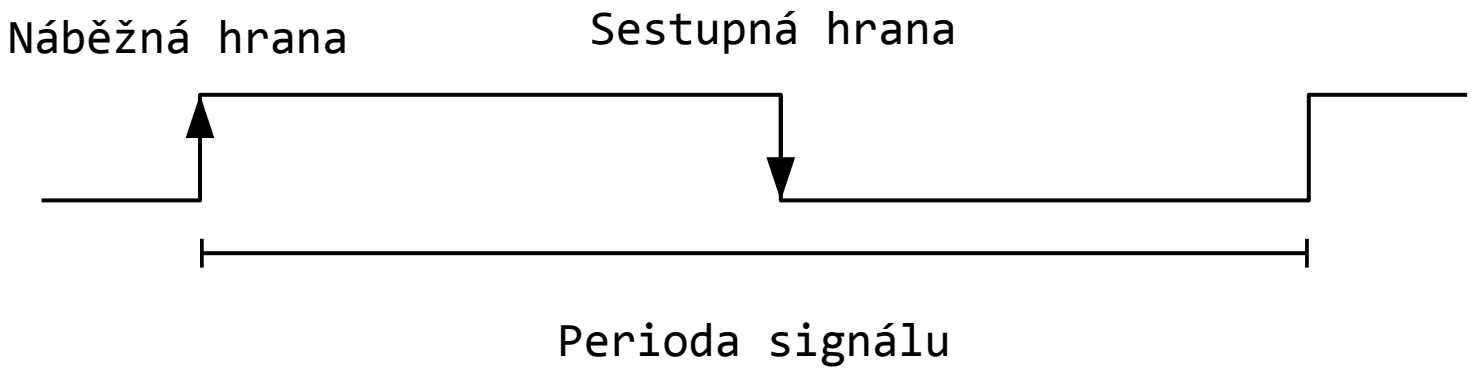
- paměťové prvky udržují stav
- vstup a obsah paměťových prvků určuje výstup a následující stav



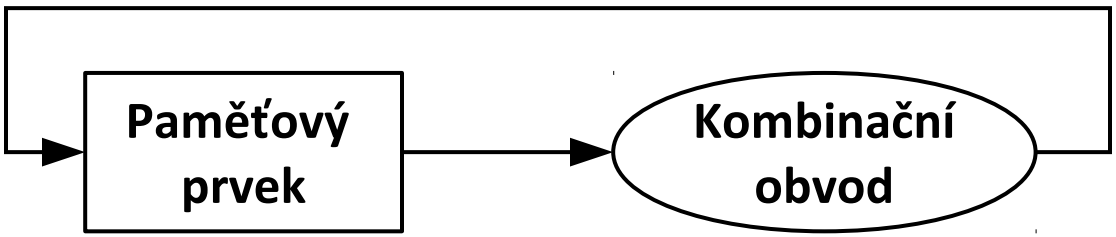
- synchronní/asynchronní
  - způsob a okamžik změny stavu

Synchronní sekvenční obvody

Hodinový signál synchronizuje změny stavu



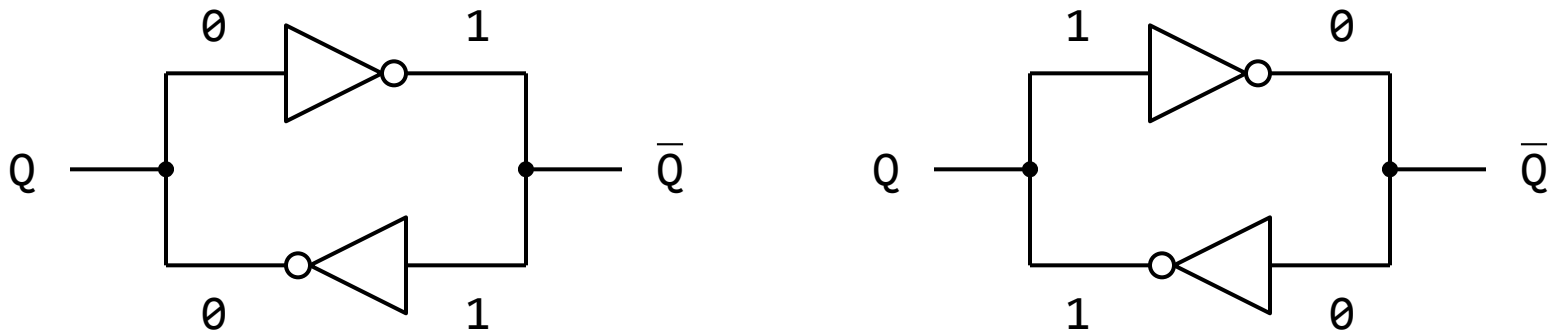
- změna stavu během jednoho cyklu
  - hodnoty pro kombinační logiku se během čtení nemění
  - zápis s náběžnou/sestupnou hranou hodinového signálu



Základ paměťového prvku

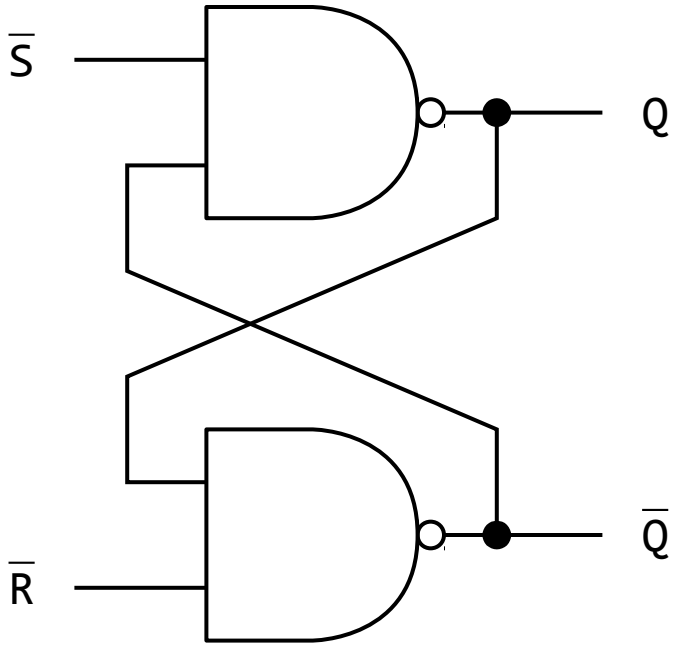
Dvojice invertorů se zpětnou vazbou

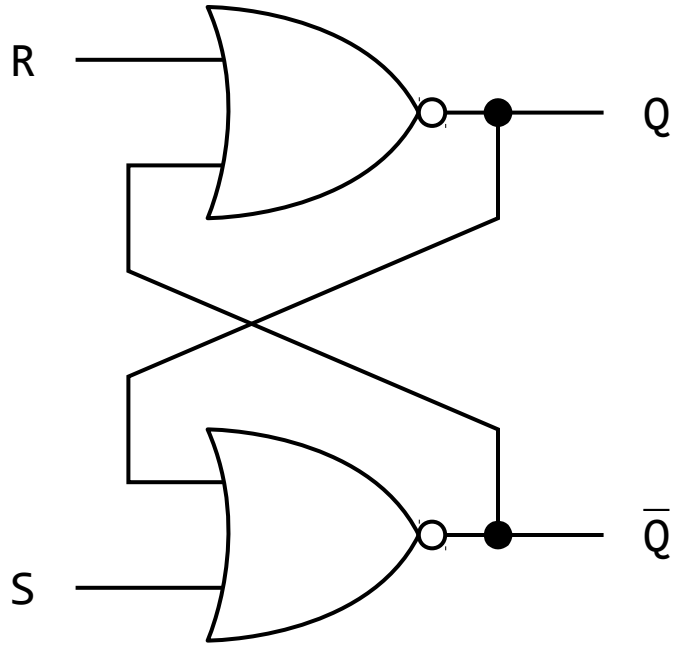
- obvod se může nacházet ve dvou stabilních stavech
- umožňuje záznam 1 bitu informace



Jak se dá zapsat informace?

Klopný obvod typu R-S (latch)





Vstupy		Výstupy	
$\overline{R}$	$\overline{S}$	Q_n	$\overline{Q}_n$
0	0	?	?
0	1	0	1
1	0	1	0
1	1	Q_{n-1}	$\overline{Q}_{n-1}$

Vstupy		Výstupy	
a	b	NAND	NOR
0	0	1	1
0	1	1	0
1	0	1	0
1	1	0	0

Vstupy		Výstupy	
R	S	Q_n	$\overline{Q}_n$
0	0	Q_{n-1}	$\overline{Q}_{n-1}$
0	1	1	0
1	0	0	1
1	1	?	?

Další klopné obvody

R-S s hodinovým vstupem (clocked R-S latch)

- synchronní varianta R-S

R-S master/slave (R-S flip-flop)

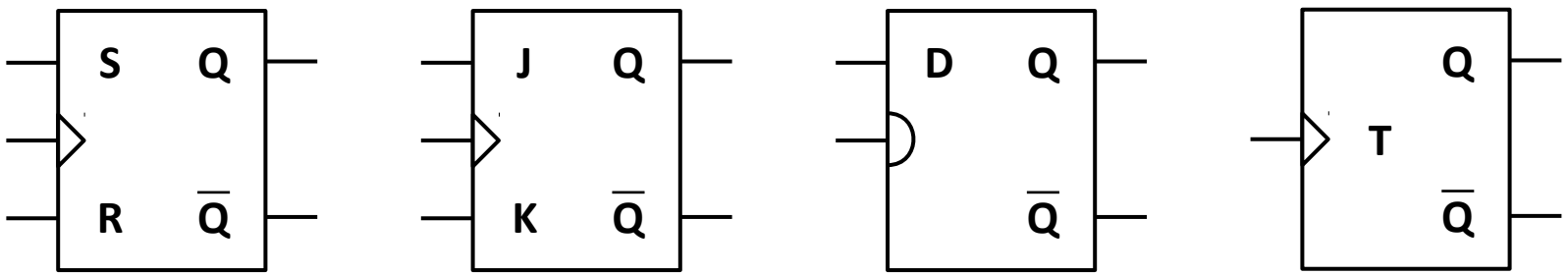
- reaguje na náběžnou/sestupnou hranu hod. signálu

J-K master/slave (J-K flip-flop)

- umí invertovat vlastní stav

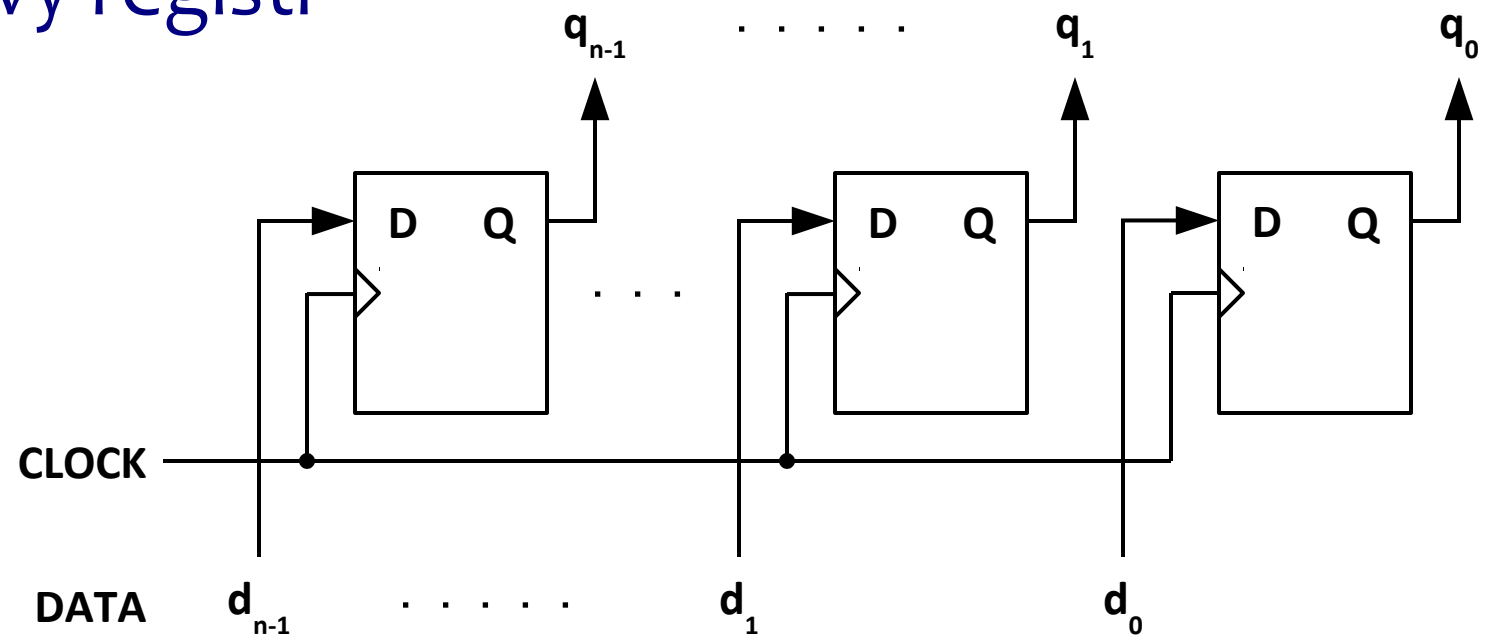
Odvozené obvody a značení

- D latch, D flip-flop, T flip-flop

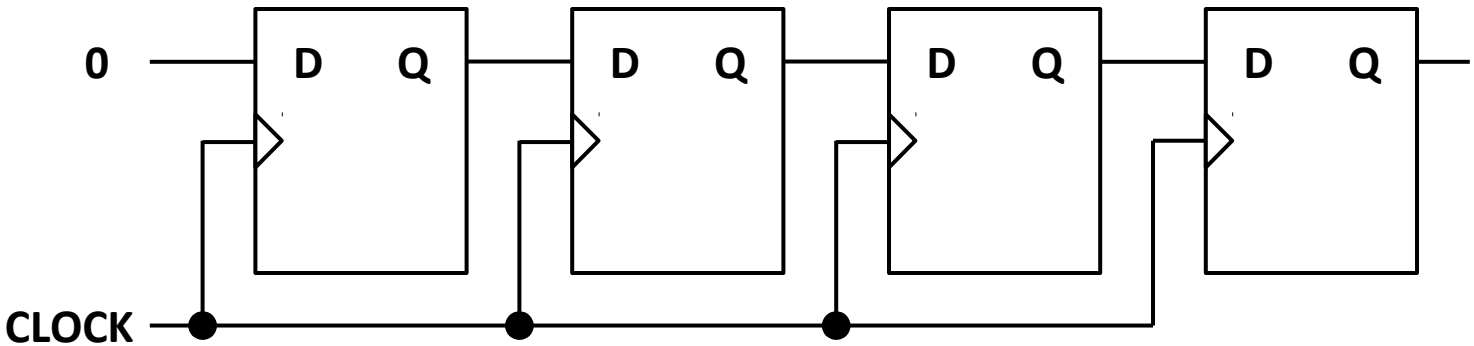


Registry

n-bitový registr

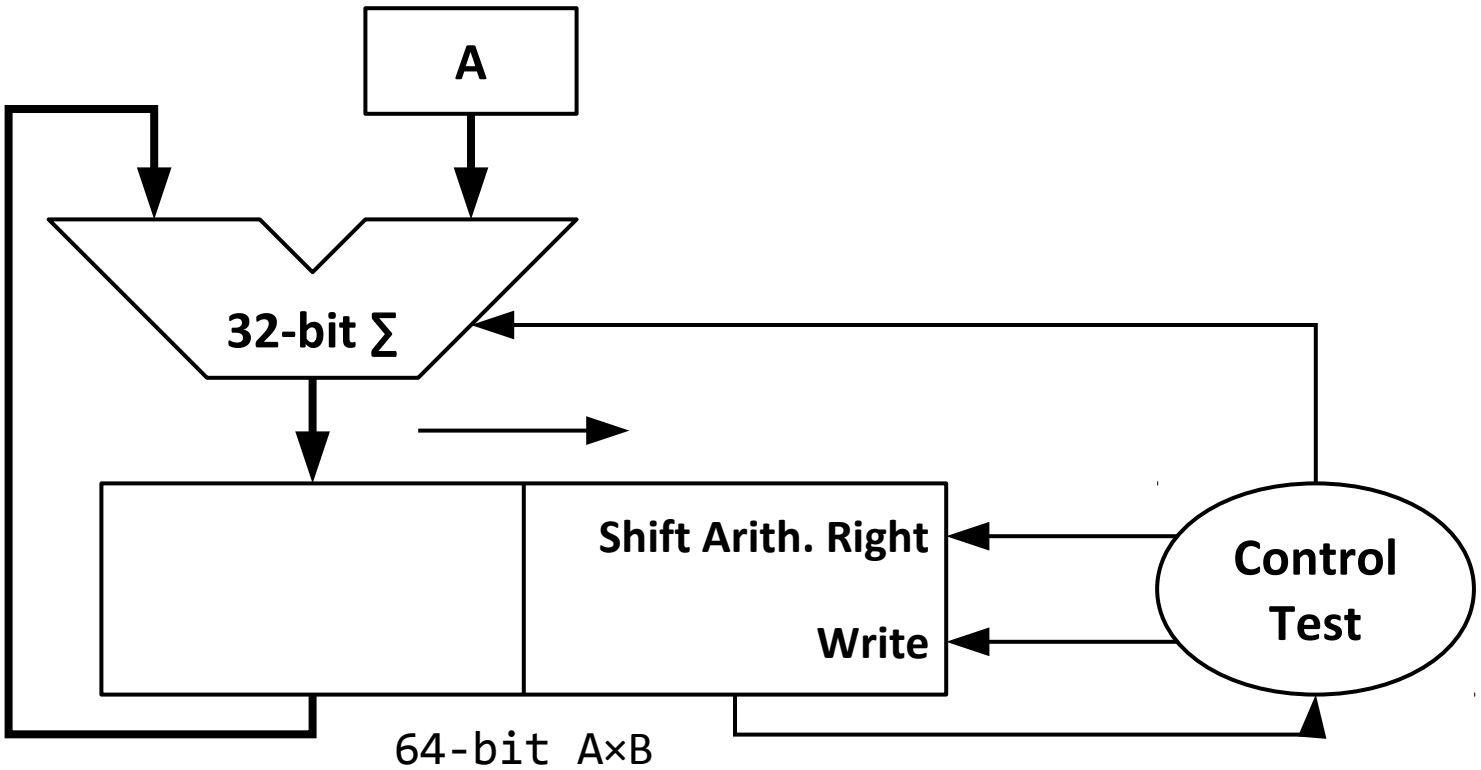


Posuvný registr

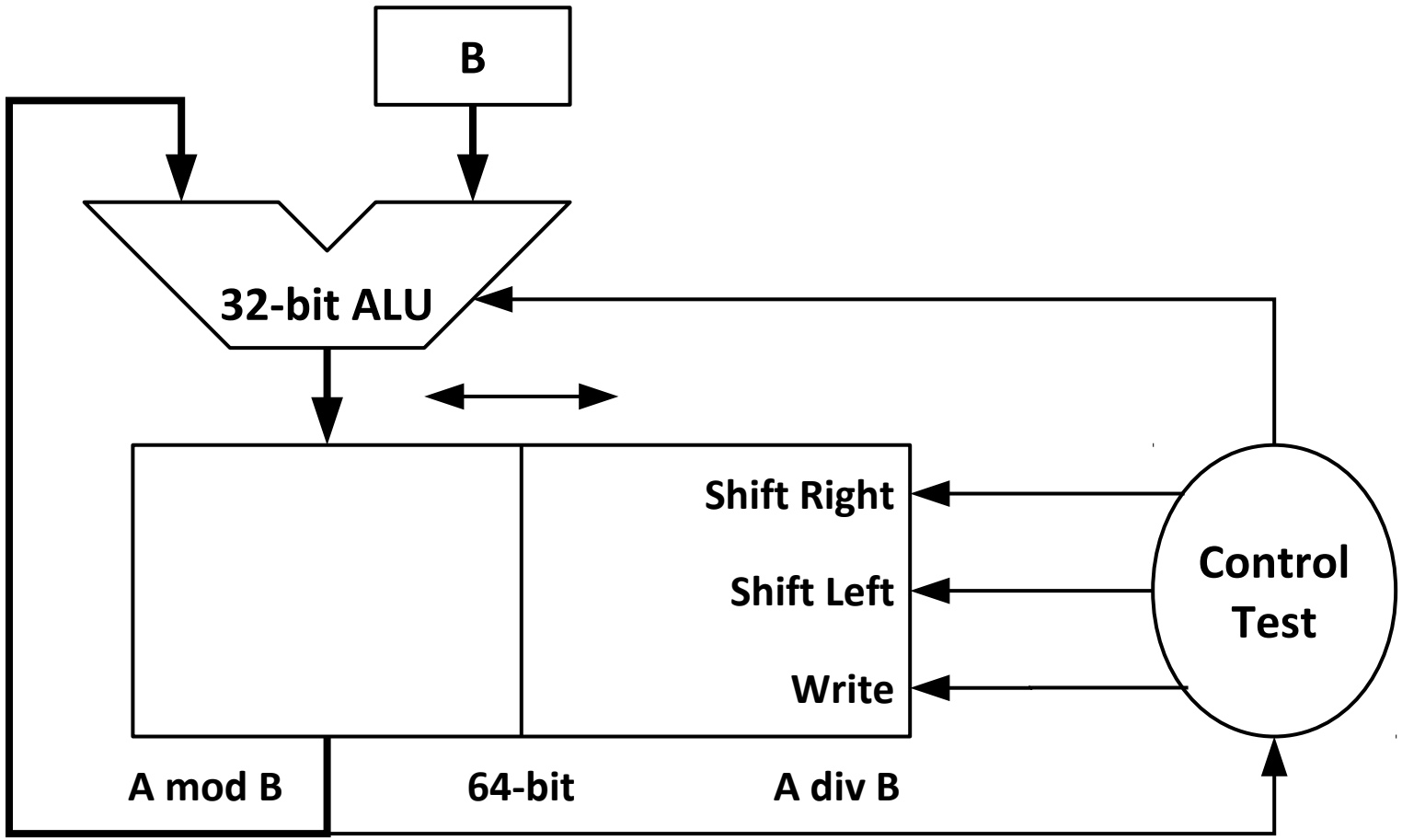


HW realizace násobení a dělení

32-bitová sekvenční násobička



32-bitová sekvenční dělička

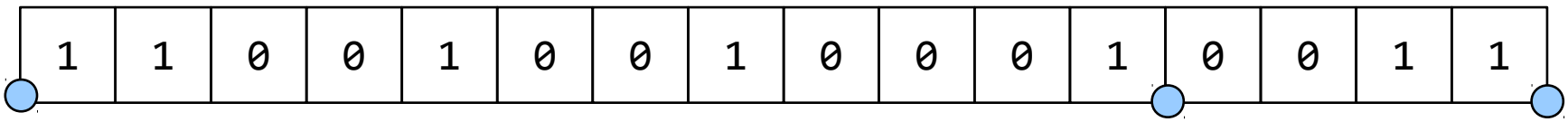


Reprezentace čísel v zobrazení s pevnou a plovoucí řádovou čárkou

Zobrazení s pevnou řádovou čárkou

Pozice řádové čárky určuje přesnost zobrazení

- ***n*** cifer celé části, ***m*** cifer zlomkové části
  - interval $0 \leq x \leq 2^n - 2^{-m}$
- celá čísla ~ speciální případ pro $m = 0$



Použití

- úlohy s omezenými nároky na přesnost
- hardware bez podpory ***floating point*** operací

Vědecká notace

Standardizovaný zápis desítkových čísel

- zápis ve formě $m \times 10^e$
- mantisa m
  - pouze platné číslice, počet číslic odpovídá přesnosti
  - normalizovaný zápis ~ $1 \leq m < 10$
- exponent e
  - reprezentuje pozici desetinné čárky
- pro $\beta \neq 10$ ~ ***floating point***

Zápis čísla 0,000000004945

- normalizovaný zápis ~ $4{,}945 \cdot 10^{-9}$

Zobrazení s plovoucí řádovou čárkou

"Vědecká notace" v nedesítkových soustavách

- mantisa ***m***, exponent ***e***, základ **β**, přesnost ***p***
  - základ **β** určuje základ mantisy i exponentu
  - přesnost ***p*** určuje počet platných míst mantisy

Příklady

- 0,9 pro **β = *10*, *p* = *3***
  - $9{,}00 \times 10^{-1}$
- 0,1 pro **β = *2*, *p* = *24*** (nelze přesně)
  - $1{,}10011001100110011001101 \times 2^{-4}$

Zápis čísla

Obecný tvar

$$\pm d_0, d_1 d_2 \ldots d_{p-1} \times \beta^e \qquad 0 \leq d_i < \beta$$

Reprezentované číslo

$$\pm \left(d_0 + d_1 \beta^{-1} + d_2 \beta^{-2} + \ldots + d_{p-1} \beta^{-(p-1)} \right) \beta^e$$

Mantisa $\boldsymbol{m = d_0, d_1 d_2 \ldots d_{p-1}}$

- pokud $\boldsymbol{d_0 = 0}$, reprezentace není ***normalizovaná***
- pokud $\boldsymbol{d_0 \neq 0}$, reprezentace je ***normalizovaná***

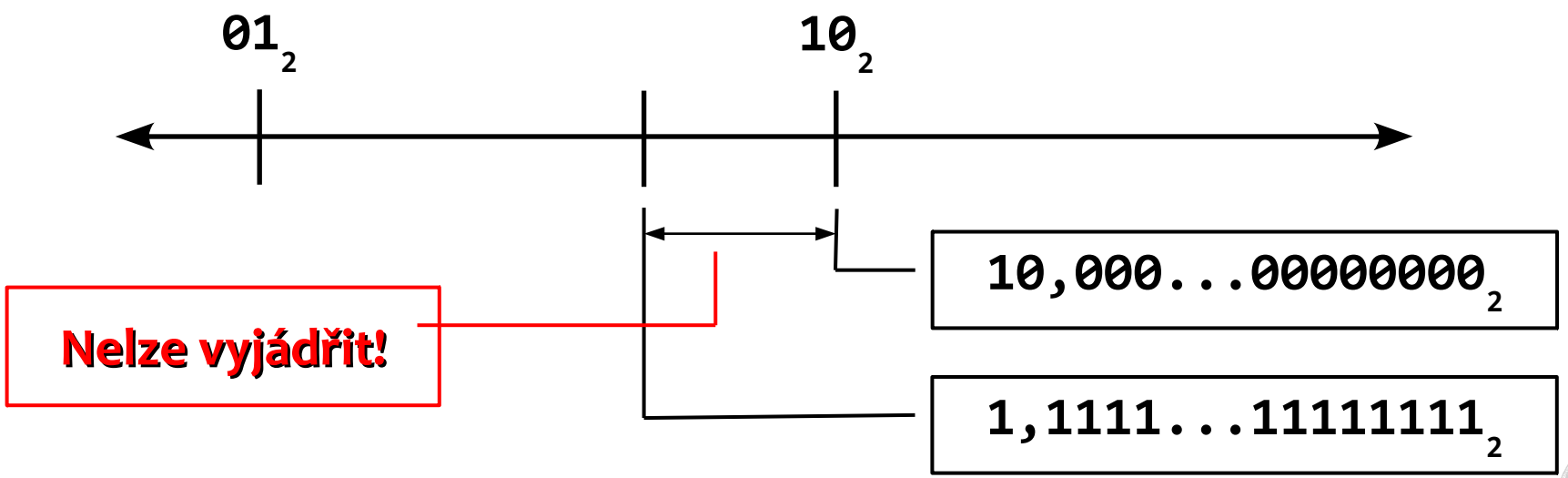
Hlavní vlastnosti

Velký rozsah zobrazitelných čísel

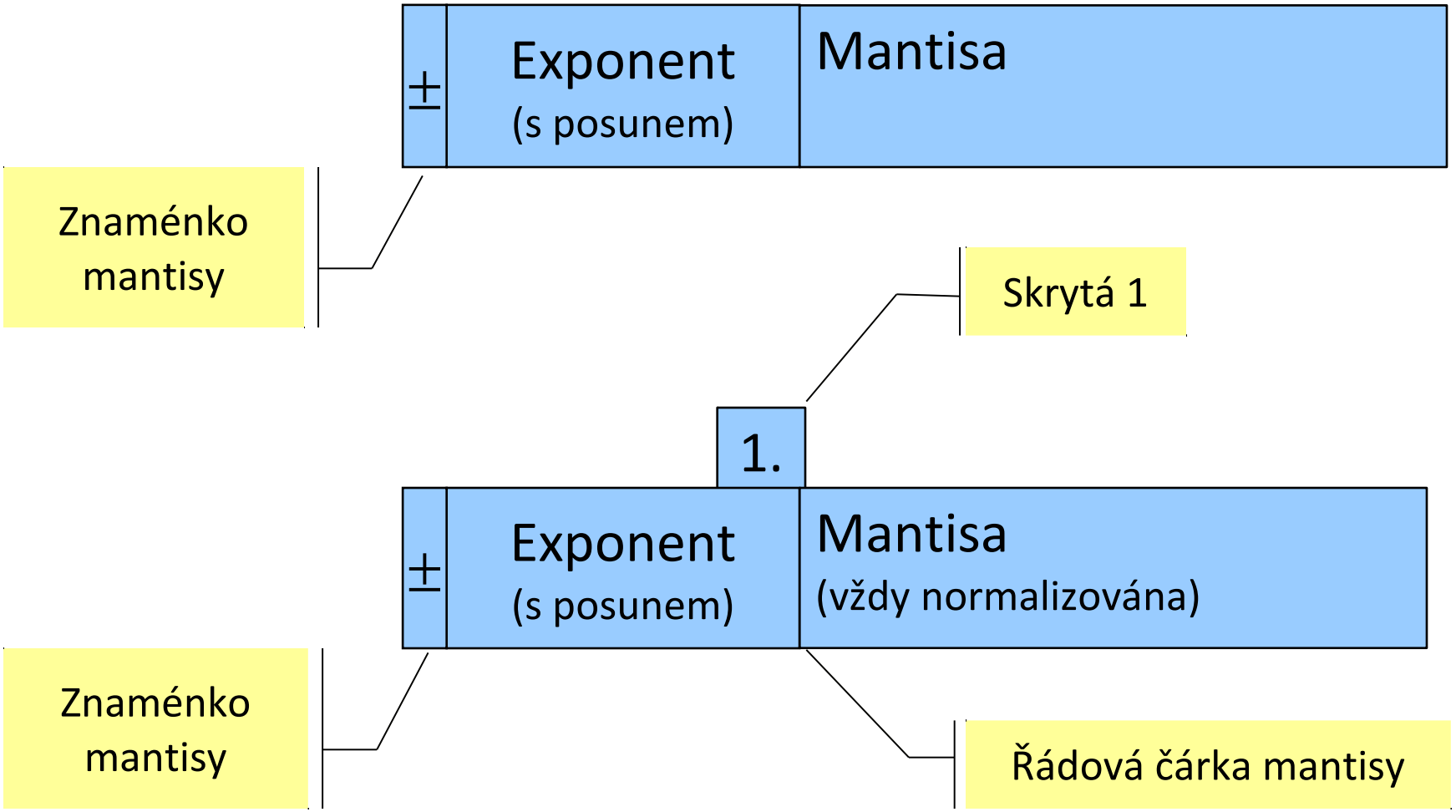
- stejná přesnost všech čísel

Čísla netvoří kontinuum

- pouze přibližné vyjádření některých čísel



Hlavní formáty



Převod do reprezentace s plovoucí řádovou čárkou

Obecný postup

1. převést absolutní hodnotu čísla do dvojkové reprezentace
   - celou i zlomkovou část
2. normalizovat dvojkovou reprezentaci
   - výsledkem je mantisa m_2 a exponent e_{10}
3. zaokrouhlit mantisu na přesnost p
   - při použití skryté 1 nejprve "odtrhnout" úvodní 1
4. převést exponent do požadovaného zobrazení
   - posun nebo znaménkové rozšíření
5. nastavit znaménkový bit mantisy

IEEE Standard 754-1985

IEEE Standard for
Binary Floating-Point Arithmetics

Standardizované formáty čísel

Single precision: $\beta = 2, p = 24, |e| = 8\ (+127)$

- rozsah ~ $\pm 10^{38}$, přesnost ~ 7 desetinných míst

Double precision: $\beta = 2, p = 53, |e| = 11\ (+1023)$

- rozsah ~ $\pm 10^{308}$, přesnost ~ 17 desetinných míst

Vnitřní reprezentace (bez skrytého bitu)

- single extended: $\beta = 2, p \geq 32, |e| \geq 11$ **bitů**
- double extended: $\beta = 2, p \geq 64, |e| \geq 15$ **bitů**

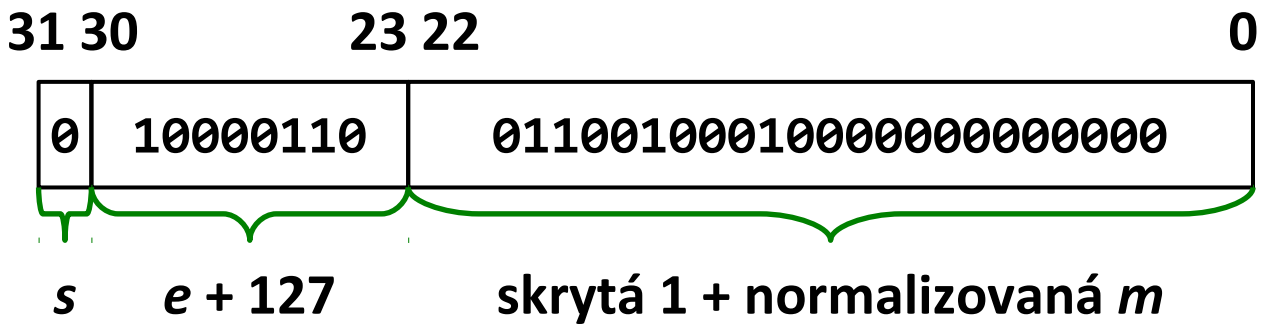
Podobné (nestandardní) formáty

- quad precision: $\beta = 2, p = 113, |e| = 15$ **bitů**
- half precision: $\beta = 2, p = 11, |e| = 5$ **bitů**

Příklad: zobrazení čísla

Jak zapsat číslo 178,125?

- dvojkový zápis: 10110010,001
- normalizovaný dvojkový zápis: $1{,}0110010001_2 \times 2^7$
  - exponent $e = 7_{10} = 111_2$
- zápis ve formátu ***single precision***
  - přesnost 24b, exponent s posunem +127, skrytá 1



Definované hodnoty a symboly

exponent	zobrazení exponentu	mantisa	zobrazení mantisy	význam
$e_{min} - 1$	000 ... 000	m = 0	000 ... 000	± 0
$e_{min} - 1$	000 ... 000	m ≠ 0	...	$0,m \times 2^{e_{min}}$
$<e_{min}, e_{max}>$	...	m	...	$1,m \times 2^{e}$
$e_{max} + 1$	111 ... 111	m = 0	000 ... 000	± ∞
$e_{max} + 1$	111 ... 111	m ≠ 0	1?? ... ???	QNaN
$e_{max} + 1$	111 ... 111	m ≠ 0	0?? ... ???	SNaN

Příčinu vzniku NaN

NaN nemusí generovat přímo procesor

Operace	Vznik NaN
+, -	$\infty + (-\infty)$
$\times$	$0 \times \pm\infty$
/	$0/0, \pm\infty/\pm\infty$
REM	x REM 0, ∞ REM y
$\sqrt{}$	$\sqrt{}$x pro x<0

Propagace NaN v operacích

- zjednodušení řízení výpočtu, pouze QNaN

IEEE Standard 754-2008/754r

Náhrada IEEE 754-1985 a IEEE 854-1987

- sjednocení a vyjasnění terminologie
- výměnné (interchange) formáty
  - 16, 32, 64 a 128 bitů pro dvojkové kódování
  - 32, 64, a 128 bitů pro desítkové kódování (DPD)
- vnitřní (non-interchange) formáty
  - nejsou určeny pro výměnu dat
  - doporučené minimální hodnoty $\boldsymbol{\beta}$, $\boldsymbol{p}$ a $\boldsymbol{e}_{max}$
- povinné operace
  - aritmetické (fused-multiply-add), konverzní, škálování a kvantizace, manipulace se znaménkem, porovnávání a úplné uspořádání, klasifikace a testování na NaN, ...

Aritmetika v zobrazení s plovoucí řádovou čárkou

Sčítání a odčítání čísel ve vědecké notaci

Čísla ***A*** a ***B***, základ **β**

$$A = M_A \times \beta^{e_A} \qquad B = M_B \times \beta^{e_B}$$

- pokud $e_A \neq e_B$
  - denormalizovat menší z operandů
- pokud $e = e_A = e_B$
  - provést operaci s mantisami

$$A + B = \left(M_A + M_B \right) \times \beta^{e}$$

$$A - B = \left(M_A - M_B \right) \times \beta^{e}$$

- normalizovat výsledek

Příklad: součet čísel ve vědecké notaci

Sečtěme $3{,}25 \times 10^{3} + 2{,}63 \times 10^{-1}$

$$3{,}25 \times 10^{3}$$
$$+\ 2{,}63 \times 10^{-1}$$

- denormalizace

$$3{,}250000 \times 10^{3}$$
$$+\ 0{,}000263 \times 10^{3}$$

- součet

$$3{,}250263 \times 10^{3}$$

- normalizace není třeba

Operace sčítání a odčítání (FP)

Dvojková reprezentace s plovoucí ř. čárkou

- zvyšovat exponent menšího čísla
  - posun řádové čárky vlevo, resp. operandu vpravo
  - při ztrátě přesnosti se ztrácí nejméně významné bity
  - co když se exponenty liší o více než přesnost ***p***?
- po normalizaci nutno zaokrouhlit
  - může číslo denormalizovat ⇒ opakovat normalizaci
- kontrola přetečení/podtečení mantisy a exponentu
  - operace s mantisami, denormalizace, normalizace

Hlavní odlišnosti od operací s reálnými čísly

Ztráta asociativity a distributivity

- na rozdíl od reálných čísel závisí na pořadí operací, algebraické úpravy výrazu mohou ovlivnit výsledek

Ztráta přesnosti při operacích

- zobrazení nekonečně přesných čísel v zobrazení s omezenou přesností

Příklad: sčítání v binární rep. s plov. řádovou čárkou

Spočtěme 0,5 – 0,4375 při přesnosti ***p=4***

$1{,}000 \times 2^{-1}$

$-1{,}110 \times 2^{-2}$

- denormalizace

  $1{,}000 \times 2^{-1}$

  $-0{,}111 \times 2^{-1}$

- součet

  $0{,}001 \times 2^{-1}$

- normalizace, zaokrouhlení

  $1{,}000 \times 2^{-4}$

- výsledek 0.0625

Chyby při operaci odčítání (FP)

Odčítání FP čísel s parametry β, p

- při odečítání v přesnosti p může být relativní chyba výsledku až $\beta - 1$
- ve dvojkové soustavě může být relativní chyba stejně velká jako výsledek
  - pozor na odčítání (sčítání čísel s opačnými znaménky) podobně velkých čísel

Guard digits

- vyšší přesnost mezivýsledku
- 1 číslice snižuje chybu, nezaručuje přesné zaokrouhlení
  $\Rightarrow$ vyžaduje 3 číslice

Operace násobení a dělení

Obecný postup

$$A = M_A \times \beta^{e_A} \qquad B = M_B \times \beta^{e_B}$$

- násobení

$$A \cdot B = \left(M_A \cdot M_B \right) \times \beta^{e_A + e_B}$$

- dělení

$$A / B = \left(M_A / M_B \right) \times \beta^{e_A - e_B}$$

- normalizace

Příklad: násobení v rep. s plov. řádovou čárkou

Spočtěme $1{,}11 \times 10^{10} \cdot 9{,}2 \times 10^{-5}$

- součet exponentů

  `exp = 10 + (−5) = 5`

- součin mantis

```
    1,11
  ×  9,20
 ---------
  10,2120
```

- výsledek

  $10{,}212 \times 10^{5}$

- normalizace

  $1{,}0212 \times 10^{6}$

Operace násobení a dělení (FP)

Dvojková reprezentace s pohyblivou ř. čárkou

- po normalizaci nutno zaokrouhlit
  - zaokrouhlení může číslo denormalizovat
- kontrola přetečení/podtečení mantisy a exponentu
  - operace s mantisami a exponenty, normalizace

- aproximace dělení
  - místo **a** / **b** se provádí **a** · 1 / **b**

Příklad operace násobení (FP)

Dvojková reprezentace s pohyblivou ř. čárkou

- 0,5 × 0,4375 při přesnosti ***p=4***

  $1{,}000 \times 2^{-1}$

  $-1{,}110 \times 2^{-2}$

- součet exponentů

  $exp = -3$

- součin mantis

  $$\begin{array}{r} 1{,}000 \\ \times\ 1{,}110 \\ \hline 1{,}110000 \end{array} \times 2^{-3}$$

- normalizace, zaokrouhlení, znaménko

  $-1{,}110 \times 2^{-3} = -0{,}21875$

Zaokrouhlování při FP operacích

Různé způsoby pro různé situace

- směrem k $+\infty$
- směrem k $-\infty$
- směrem k nule

- směrem k nejbližší hodnotě

Vznik chyb při FP operacích

Výjimečné stavy

- overflow – přetečení
- underflow – podtečení
- divide by zero – dělení nulou
- invalid – neplatná operace
- inexact – nepřesný výsledek

Numerické chyby

- truncation error – zanedbání části čísla
- rounding error – zaokrouhlení

Bludy a pasti

Bludy

Když logický posun vlevo funguje jako násobení mocninou 2, tak logický posun vpravo funguje jako dělení mocninou 2

- posun -5_{10} o 2 bity vpravo by mělo odpovídat dělení 4 (2^2), výsledek by tedy měl být -1

  $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1011_2 = -5_{10}$

- po posunu vpravo o dva bity dostáváme

  $0011\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_2 = 1073741822_{10}$

- pomůže aritmetický posun vpravo?
  - zachovává znaménkový bit

  $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_2 = -2_{10}$

- jsme blízko, ale správný výsledek to není...

Bludy

Přesnost výpočtů v plovoucí řádové čárce zajímá jen matematiky

- chyba v implementaci dělení u procesoru Pentium
  - Intel: chyba se může projevit až na 9. místě
  - skutečnost: chyby v 12. až 52. bitu, tj. ve 4. až 15. desítkové číslici
- nepřesnost se projeví i při běžném používání tabulkového procesou
- konečná cena: $500 mil. za výměnu

Shrnutí

Závěrečné poznámky

Kombinace bitů nemá sama o sobě význam

- význam určuje program a instrukce, které s daty pracují

Počítačová aritmetika má omezený rozsah

- výsledkem operací může být přetečení, podtečení, výjimky apod.

Čísla v plovoucí ř. čárce **nejsou** reálná čísla

- netriviální numerické algoritmy, ztráta komutativity a asociativity komplikuje paralelizaci algoritmů

Standardní reprezentace zlepšují přenositelnost

- dvojkový doplněk u celých čísel, IEEE 754 u čísel v plovoucí řádové čárce