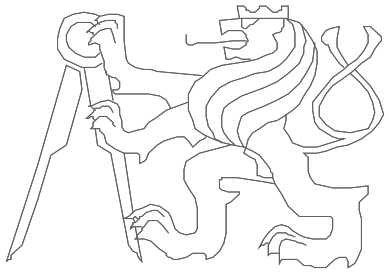
Architektury počítačů

Počítačová aritmetika

Richard Šusta, Pavel Píša



České vysoké učení technické, Fakulta elektrotechnická

Důležitá poznámka úvodem

- Cíl je porozumět struktuře počítače, abyste mohli lépe využít jeho možností k dosažení jeho vyššího výkonu.
- Dále je probíraná návaznostech/propojení HW/SW (periferie)
- Stránky předmětu:
  https://cw.fel.cvut.cz/wiki/courses/a0b36apo/start
  https://dcenet.felk.cvut.cz/apo/ - *budou teprve otevřené*
- Některé navazující předměty:

  B4M35PAP - Pokročilé architektury počítačů

  B3B38VSY - Vestavné systémy

  B4M38AVS - Aplikace vestavných systémů

  B4B35OSY - Operační systémy (OI)

  B0B35LSP - Logické systémy a procesory (KyR + část OI)
- Prerekvizita: Šusta, R.: APOLOS , ČVUT-FEL 2016, 51 s.

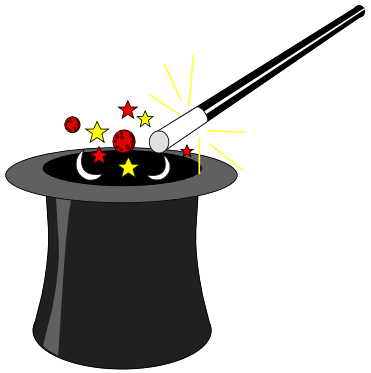
Důležitá poznámka úvodem

- Vychází ze světově uznávané knihy autorů
  Paterson, D., Henessy, V.: **Computer Organization and Design**, The HW/SW Interface. Elsevier, ISBN: 978-0-12-370606-5

David Andrew Patterson
University of California, Berkeley
Vývoj: RISC procesor MIPS,
RAID, Clusters

John Leroy Hennessy
10th President of Stanford University
Vývoj: RISC procesory MIPS,
DLX a MMIX

Počítače



a n e b

kam směřují?…

Budoucnost počítačů

Lze předpovědět vývoj počasí?

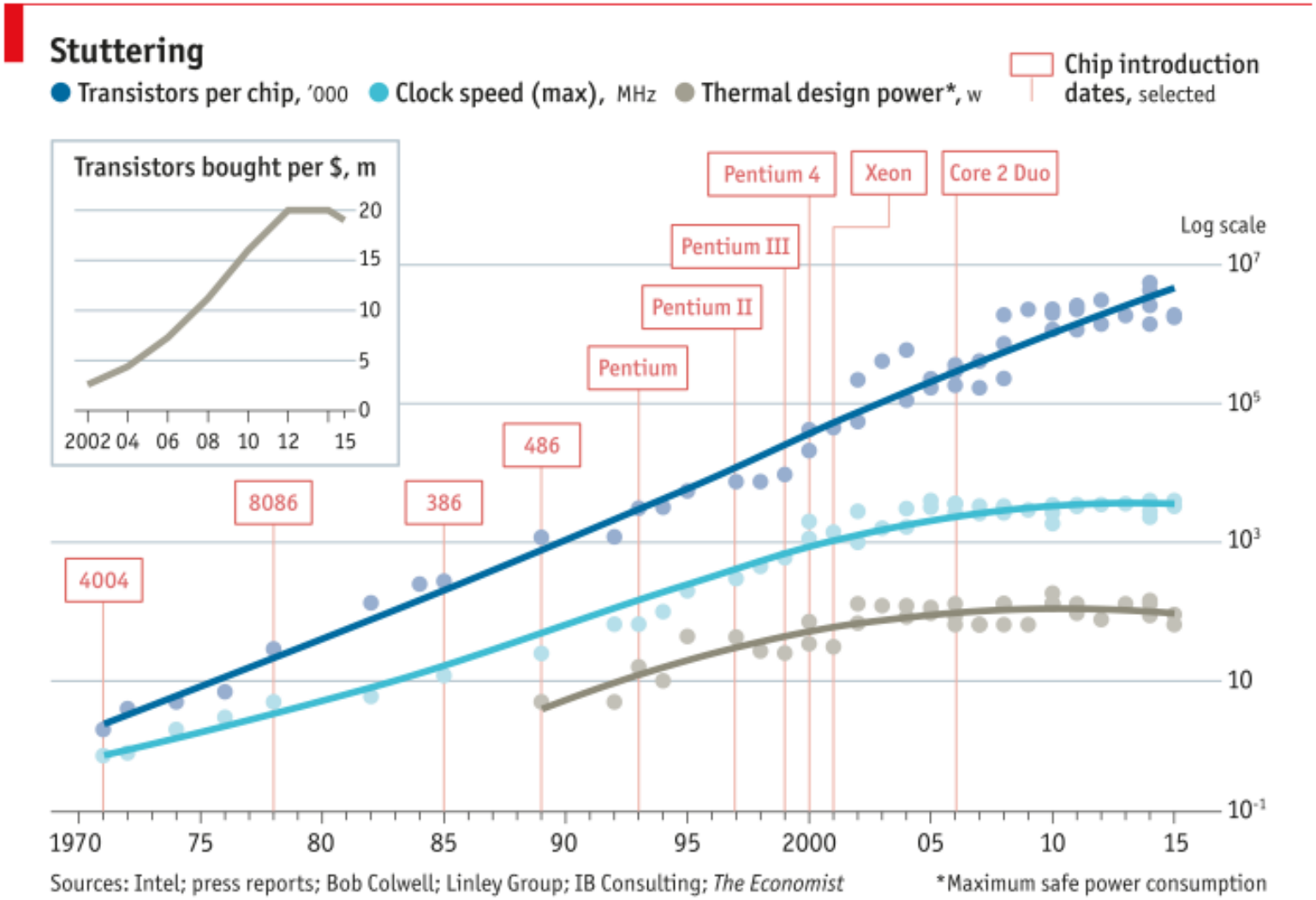
Lze předpovědět vývoj výpočetní techniky?

PÁNŮV HLAS (1899)
Během 30 až 200 veškerá hudba, kterou znáte, bude pouze na gramodeskách z vinylu

Zdroj obrázku: https://www.hmv.com/

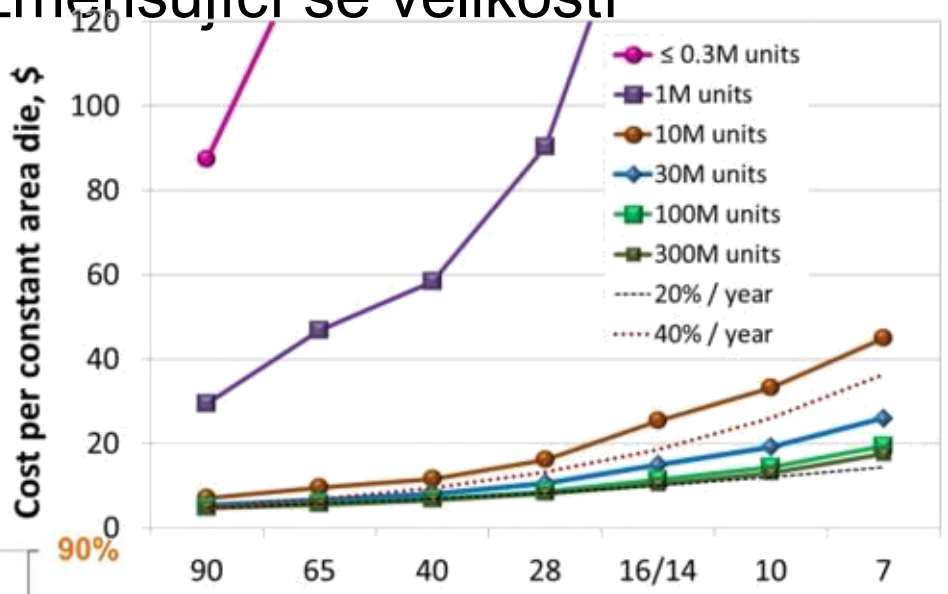
Moore's Law

Gordon Moore, zakladatel Intelu, v roce 1965: " *The number of transistors on integrated circuits doubles approximately every two years* "

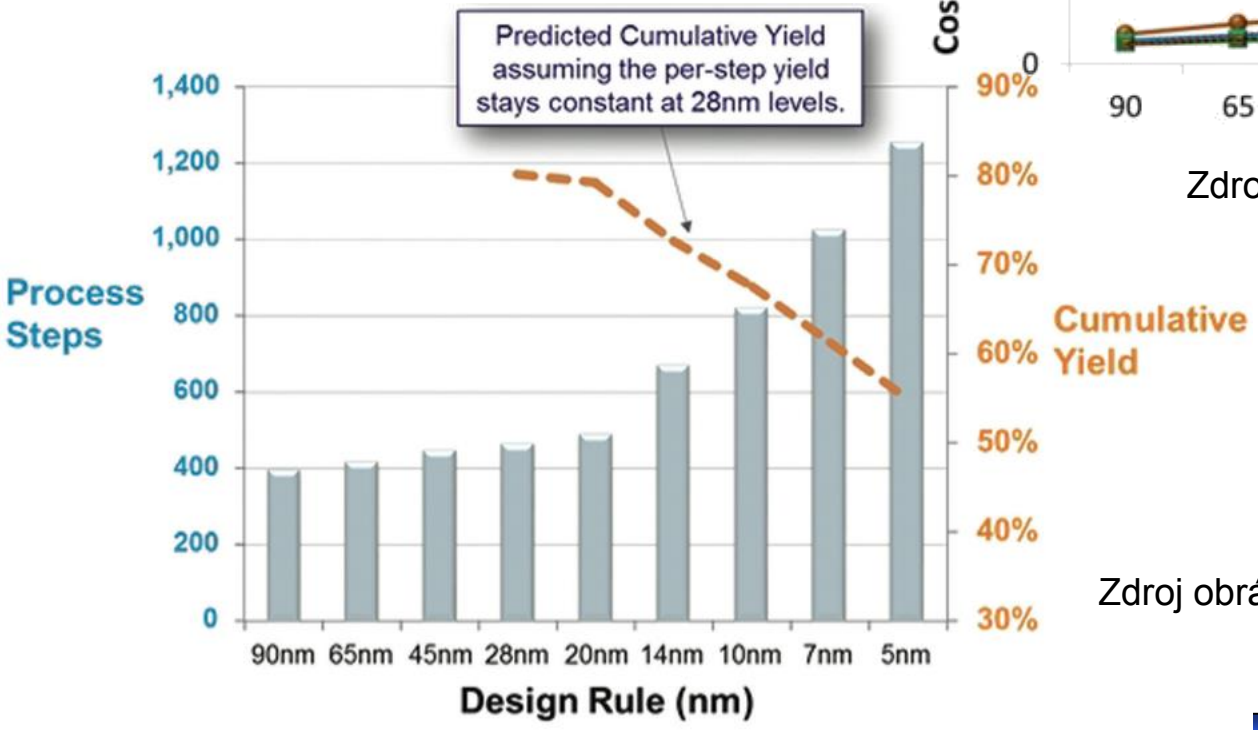


Cena výroby roste se zmenšující se velikostí

Moore's Law nejspíš zastaví cena



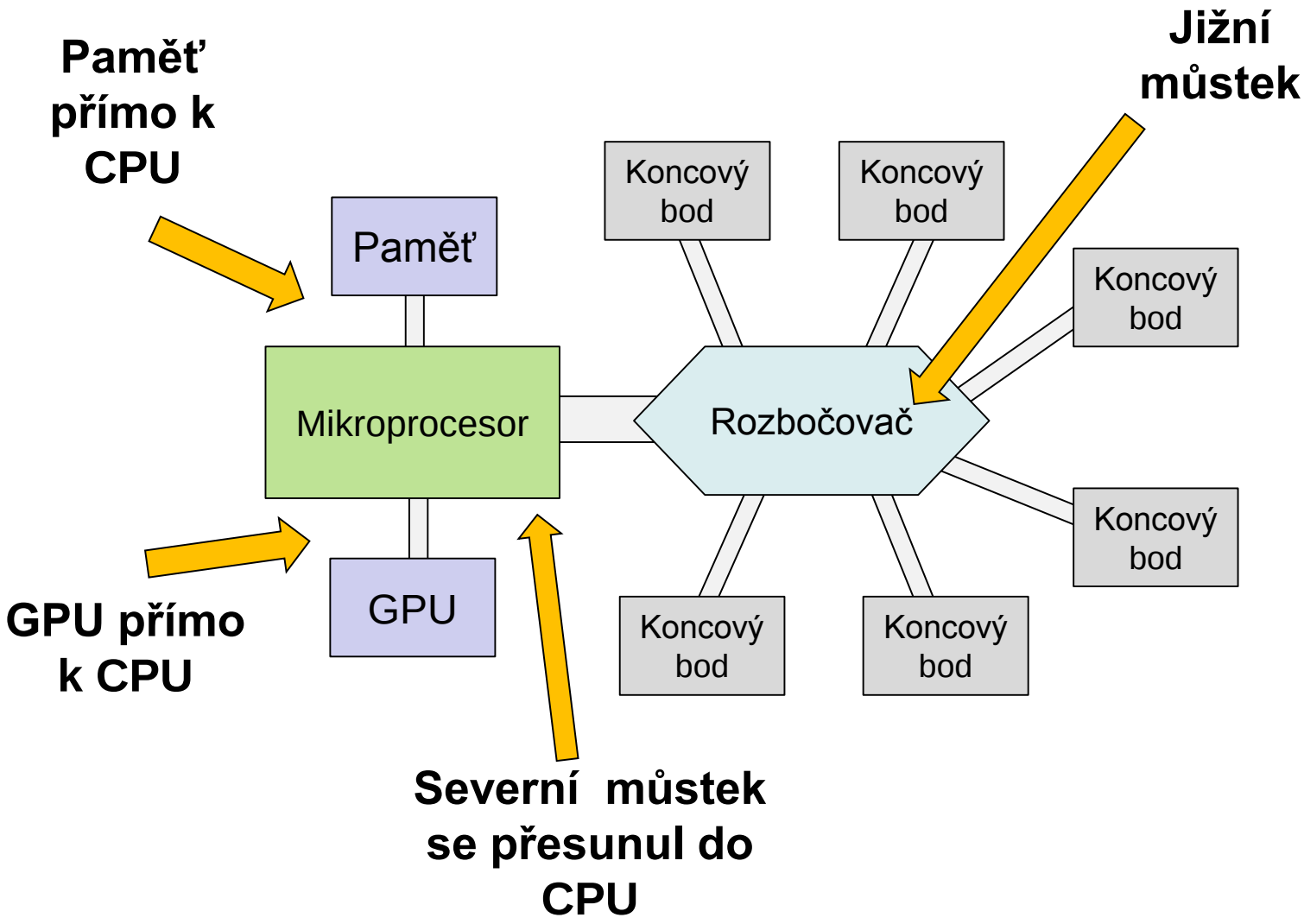
Zdroj obrázku: http://www.eetimes.com/



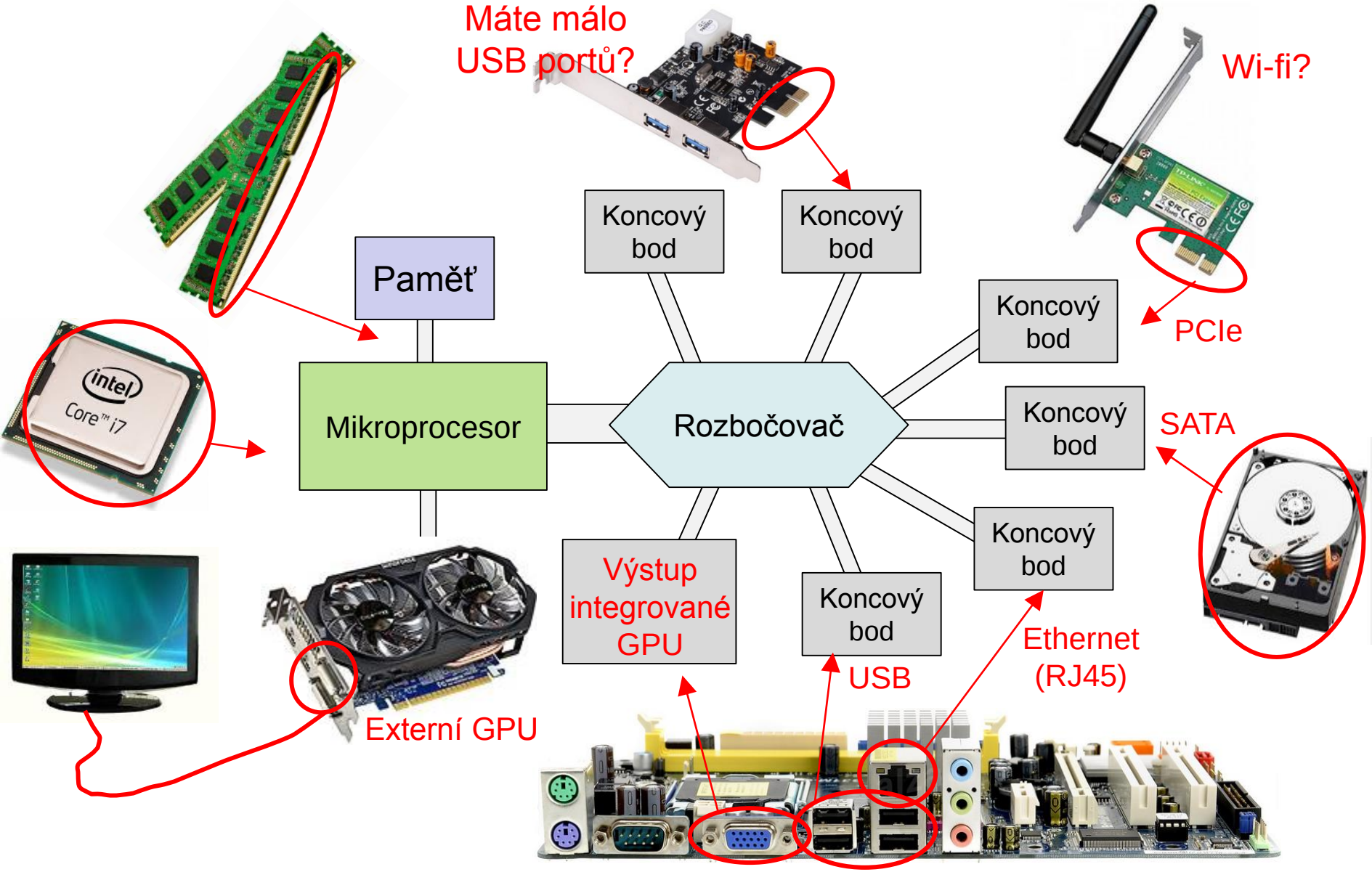
Zdroj obrázku: http://electroiq.com/

Architektura dnešního PC ??? – motherboard –

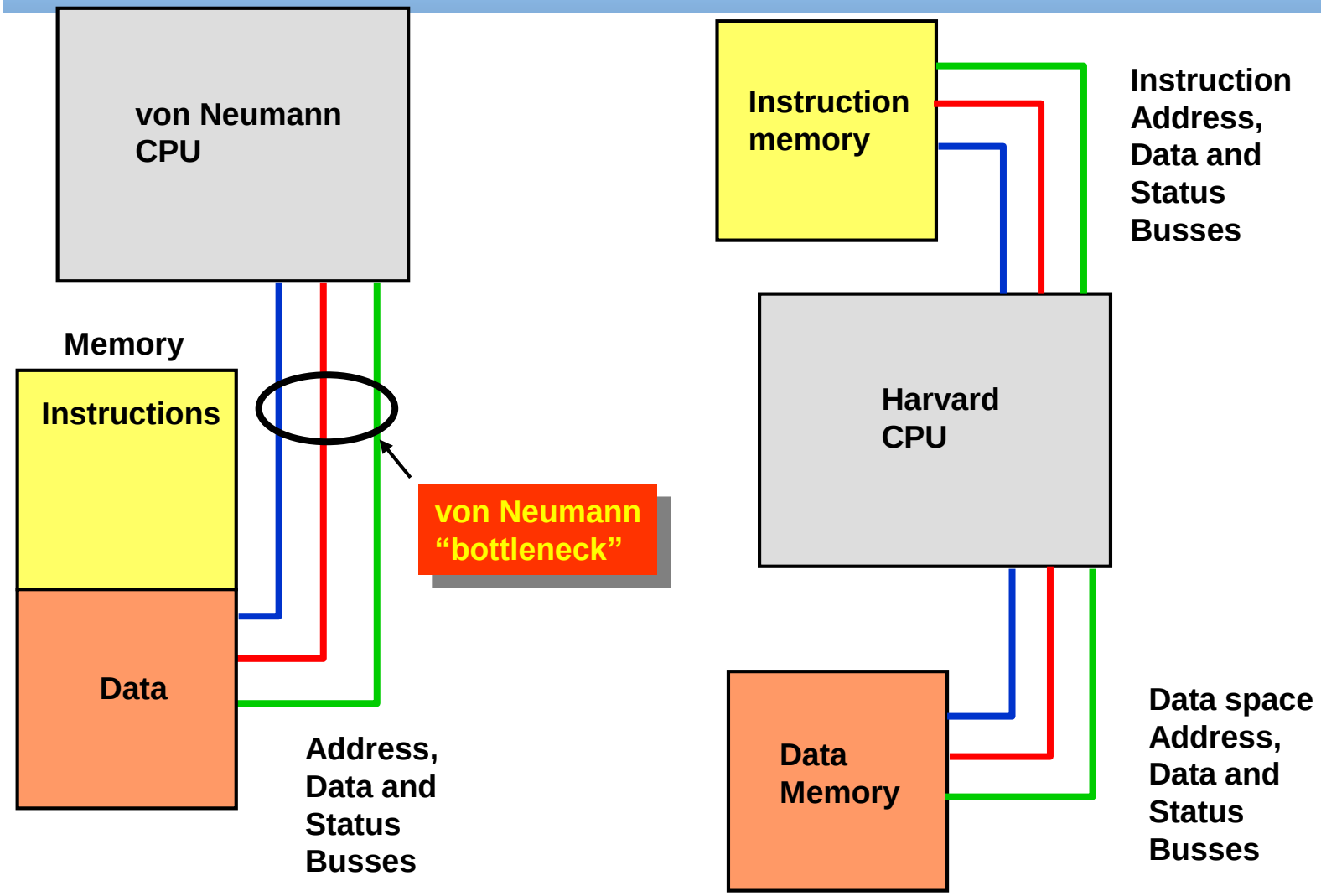
Architektura dnešního PC ??? – motherboard –



Architektura dnešního PC ??? – motherboard –

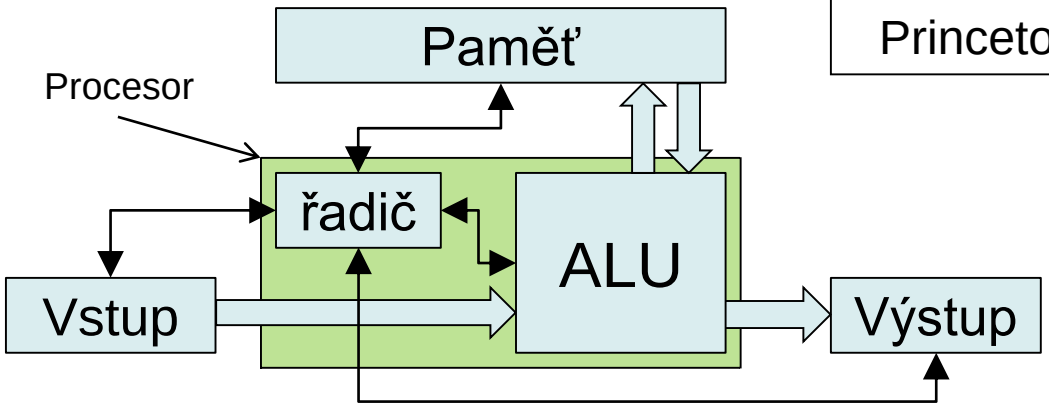


Von Neumann and Harvard Architectures



[Arnold S. Berger: Hardware Computer Organization for the Software Professional]

John von Neumann, maďarský fyzik



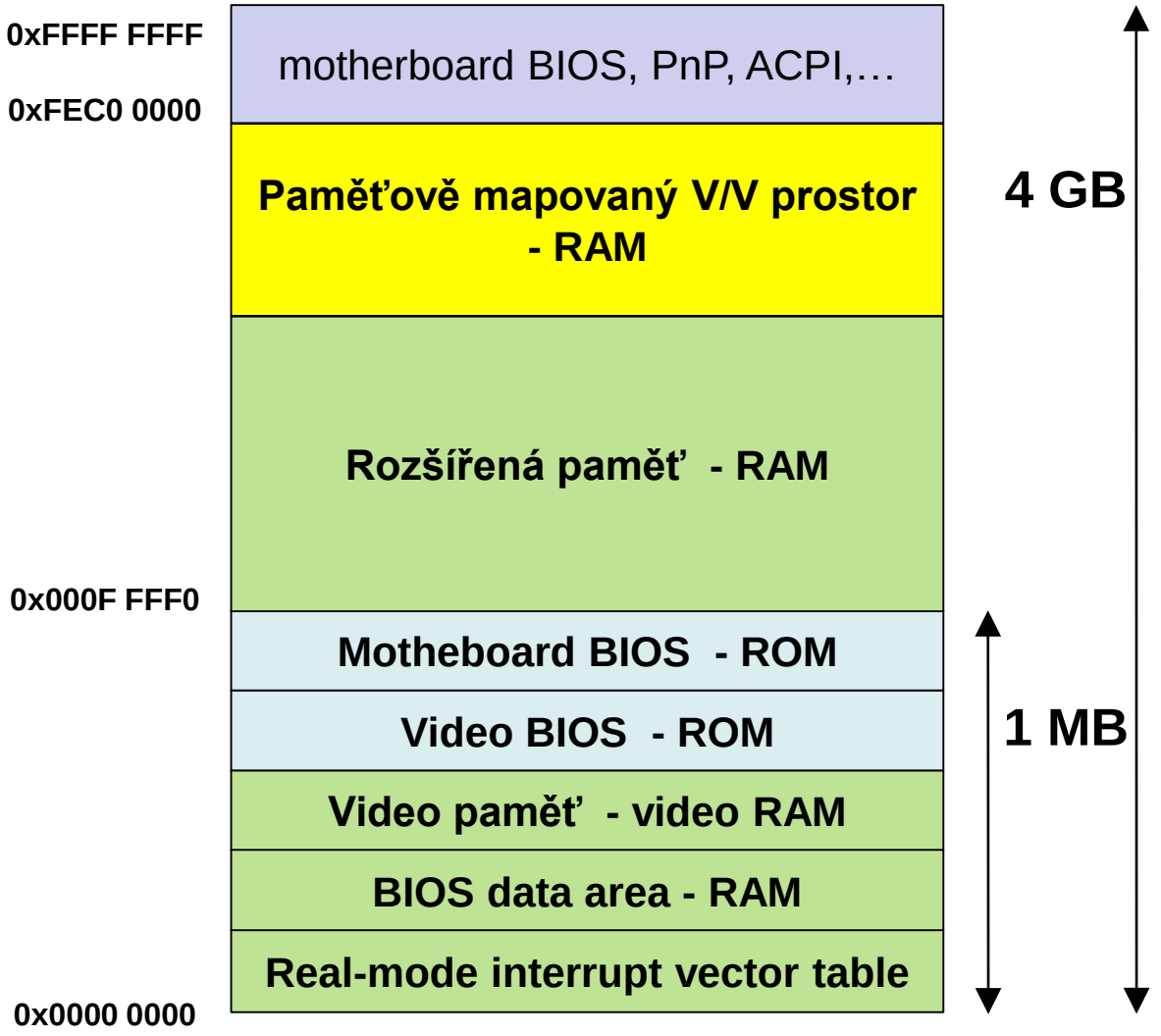
Architektura počítače podle JvN +
Princeton Institute for Advanced Studies

28. 12. 1903 -
8. 2. 1957

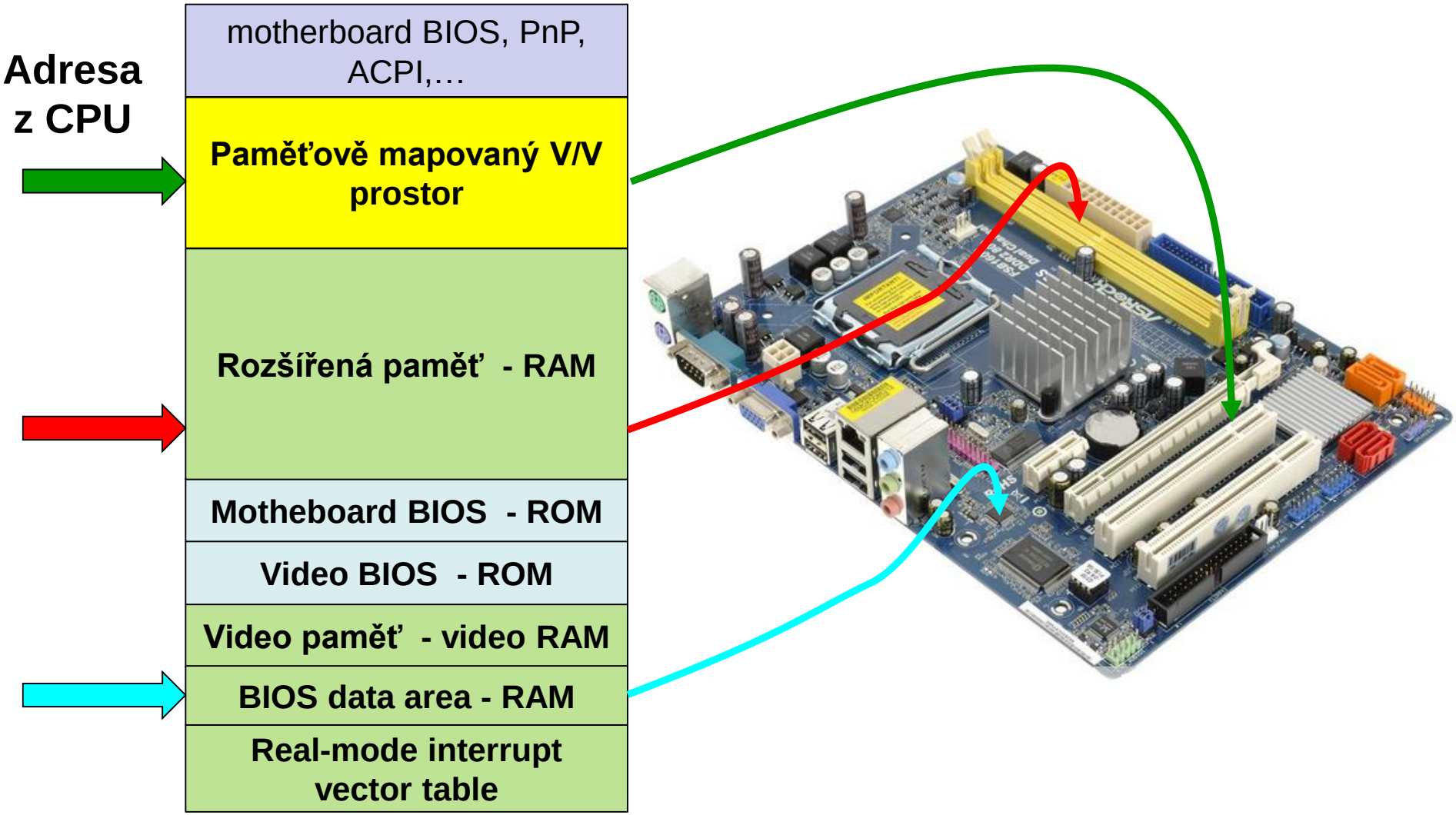
- 5 funkčních jednotek – řídicí jednotka (řadič), aritmeticko-logická jednotka, paměť, vstupní zařízení, výstupní zařízení
- Nezávislost struktury počítače na zpracovávaných problémech. Musí se zavést program a musí se uložit do paměti. Ten řídí činnost počítače.
- **Programy a výsledky (data) se ukládají do téže paměti.** Ta je rozdělena na stejně velké části (buňky), které jsou průběžně očíslované – adresa.
- Po sobě jdoucí instrukce se ukládají do po sobě jdoucích buněk.
- Existují instrukce aritmetické, logické, přenosu, skokové a ostatní.

Fyzický adresní prostor a jeho význam

- Fyzický adresní prostor je prostor, který přímo adresuje samotný procesor.
- Tento prostor může procesor adresovat na jeho adresní sběrnici.



Fyzický adresní prostor a jeho význam



Jak vypadá smartphone uvnitř? Samsung Galaxy S4

- **Android 5.0 (Lollipop)**
- **2 GB RAM**
- **16 GB pro uživatele**
- **1920 x 1080 display**
- **8-jádrový CPU (**čip Exynos 5410**):**
  - **čtyři** 1.6 GHz ARM Cortex-A15
  - **čtyři** 1.2 GHz ARM Cortex-A7

Jak vypadá smartphone uvnitř? Samsung Galaxy S4

Zdroj: http://www.techinsights.com/about-techinsights/overview/blog/samsung-galaxy-s4-teardown/

Jak vypadá smartphone uvnitř? Samsung Galaxy S4

Power management

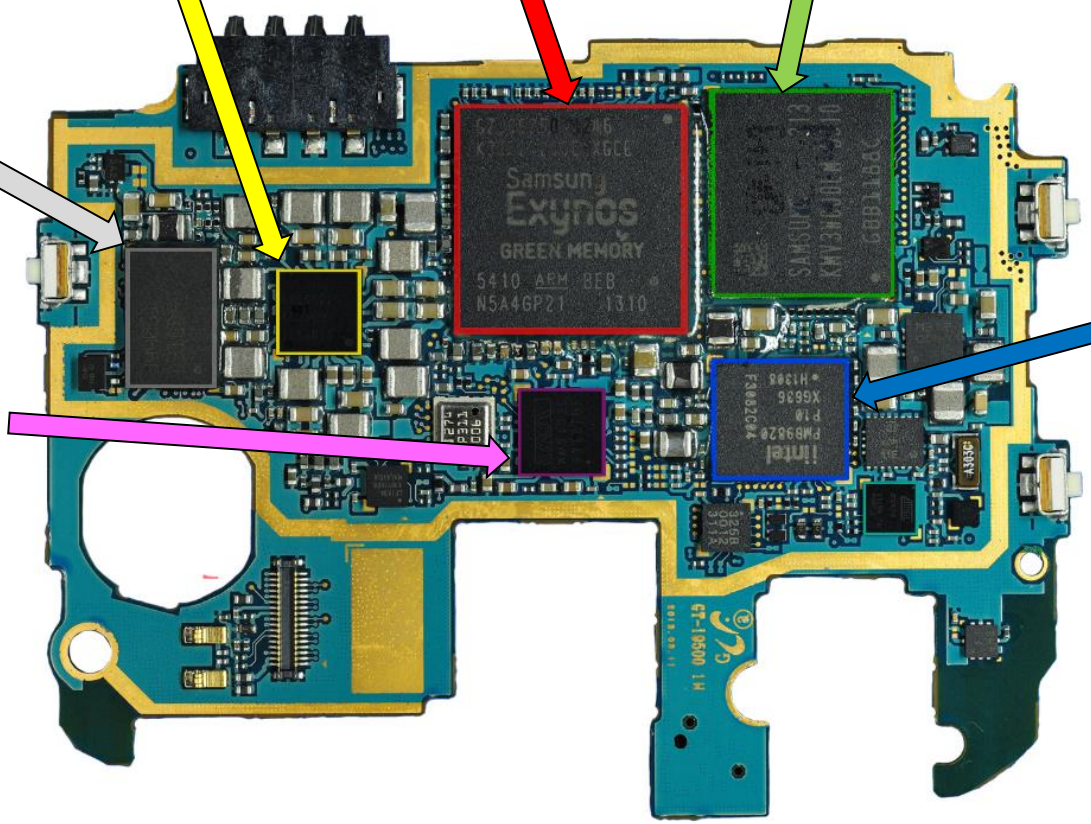
Exynos 5410 (**8-core CPU + 2GB DRAM**)

Multichip memory: 64 MB DDR SDRAM, **16GB NAND Flash**, Controler

Wi-fi (broadcom BCM4335)

DSP procesor pro zpracování hlasu, audio codec

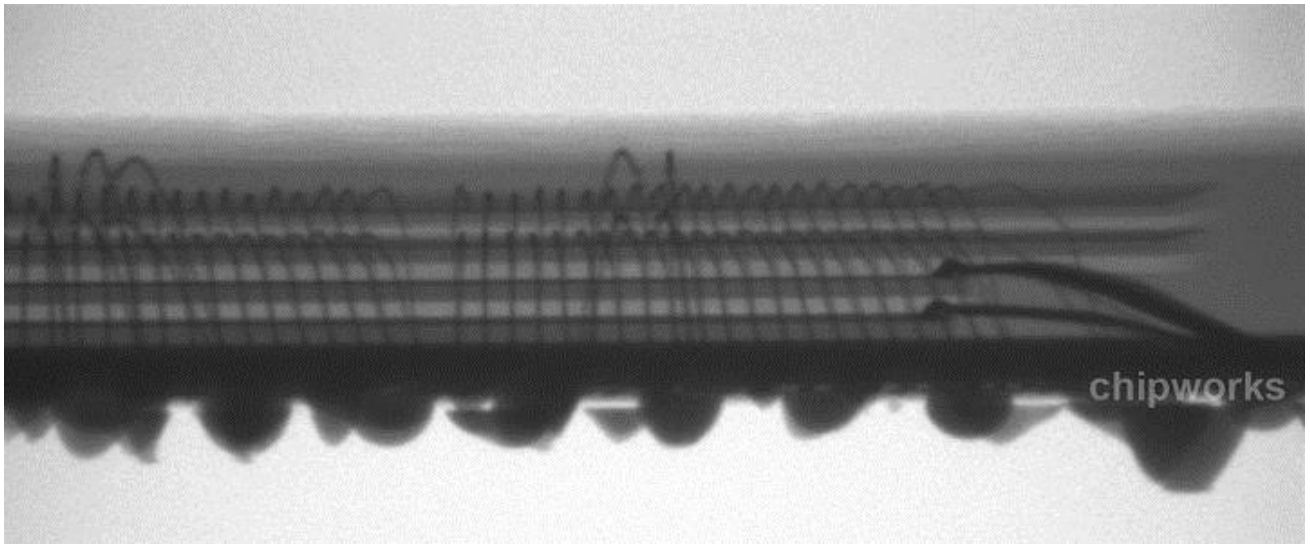
Intel PMB9820 baseband processor (funkce rádia přes anténu - EDGE, WCDMA, HSDPA/HSUPA)



Zdroj: http://www.techinsights.com/about-techinsights/overview/blog/samsung-galaxy-s4-teardown/

Jak vypadá smartphone uvnitř? Samsung Galaxy S4

Rentgenový snímek čipu Exynos 5410 ze strany:



•Vidíme, že se jedná o QDP (Quad die package)

Zdroj: http://gamma0burst.tistory.com/m/600

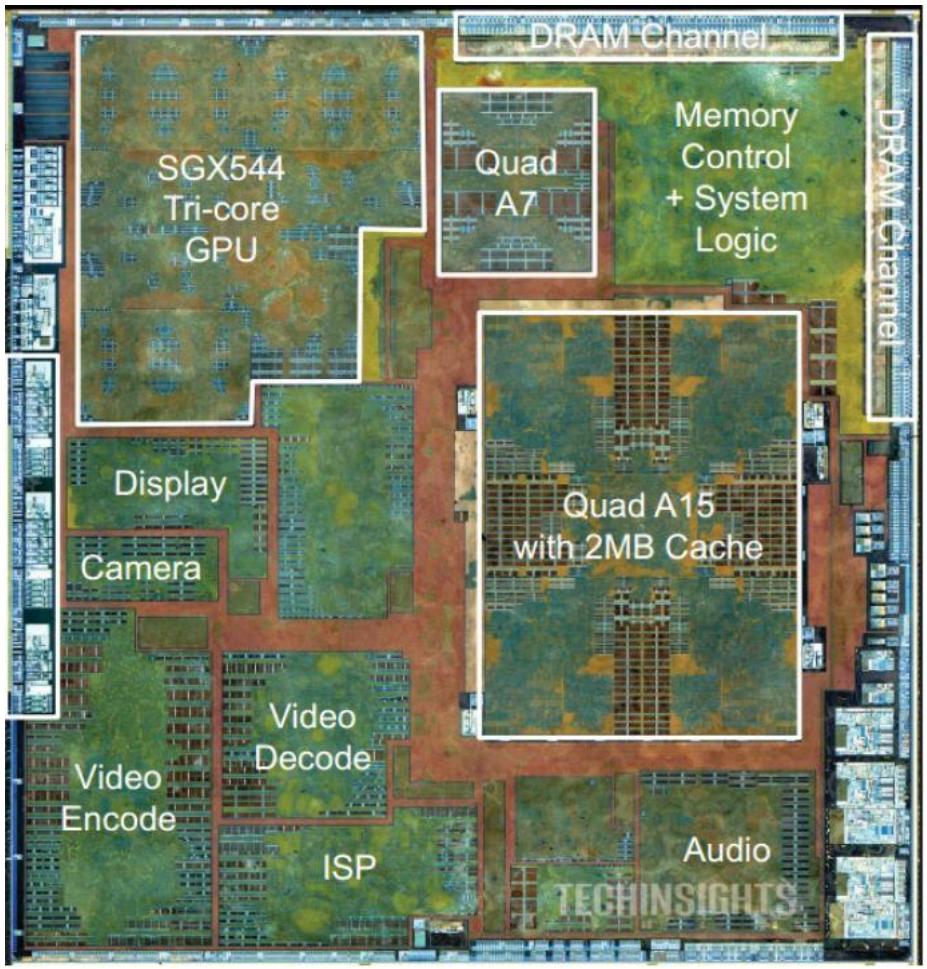
Jak vypadá smartphone uvnitř? Samsung Galaxy S4

Řez čipem Exynos 5410 – zde vidíme DRAM

Zdroj: http://www.embedded-vision.com/platinum-members/embedded-vision-alliance/embedded-vision-training/documents/pages/computational-photography-part-2

Jak vypadá smartphone uvnitř? Samsung Galaxy S4

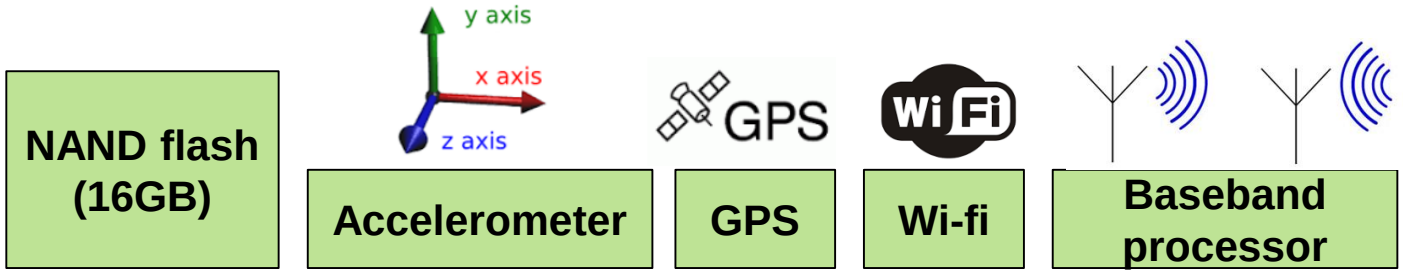
Řez čipem Exynos 5410 (v jiné úrovni)

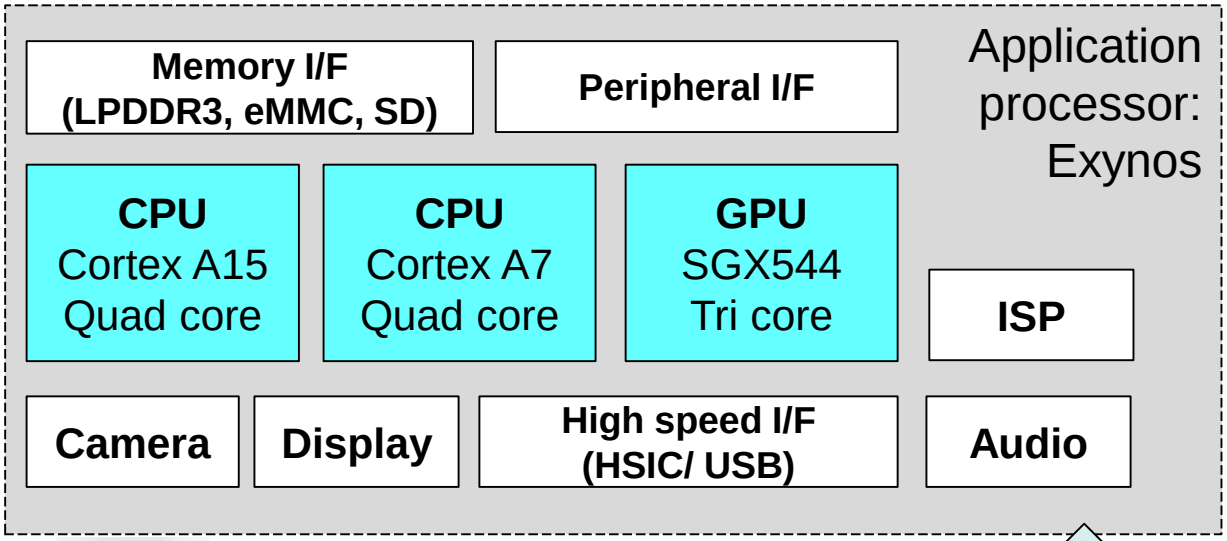


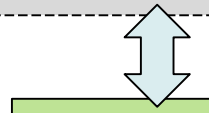
- Všimněte si rozdílných velikostí 4 jáder A7 a 4 jáder A15
- Na čipu jsou mimo vlastního procesoru integrovány i další součásti: GPU, Video coder a decoder a další. Jedná se tedy o **SoC** (System on Chip)

Zdroj: http://www.embedded-vision.com/platinum-members/embedded-vision-alliance/embedded-vision-training/documents/pages/computational-photography-part-2, http://gamma0burst.tistory.com/m/600

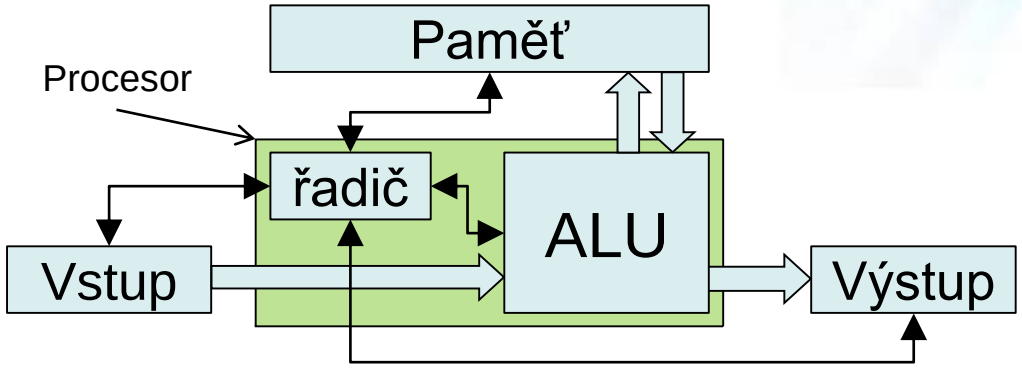
Jak vypadá smartphone uvnitř? Samsung Galaxy S4







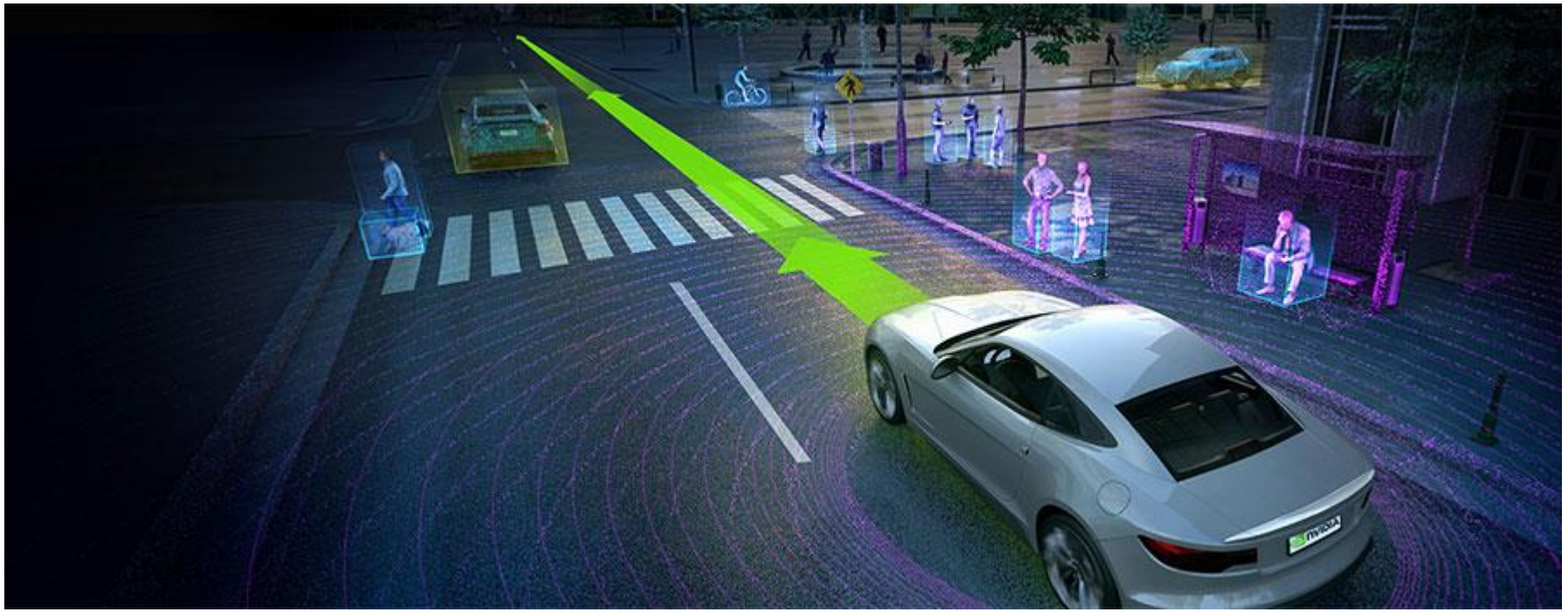
Společný koncept



- Procesor vykonává instrukce uložené v paměti (ROM, RAM) tak, aby obsluhoval periferie – reagoval na vnější události a zpracovával data.

Motivační příklad: (neformální uvedení do probíraných témat)

Autonomní řízení automobilů



Zdroj: http://www.nvidia.com/object/autonomous-cars.html

- Mnoho úloh z oblasti umělé inteligence založeno na hlubokých neuronových sítích (deep neural networks)
- Průchod neuronové sítě – maticové násobení

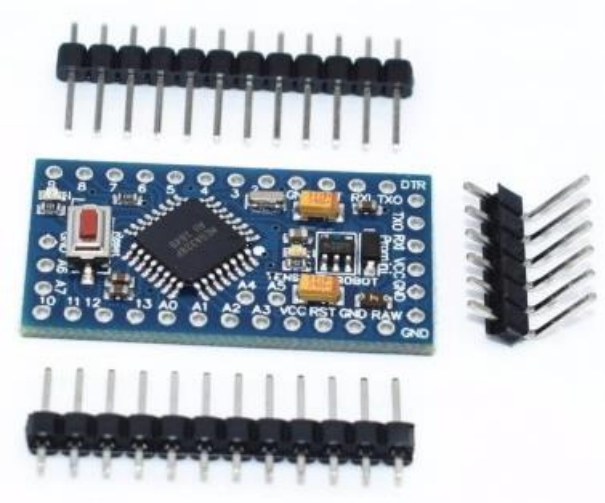
Průchod neuronové sítě – maticové násobení

- Jak docílit zrychlení?
- Výsledky jednoho z mnoha experimentů
  - Naivní algoritmus (3 × for) – 3.6 s = 0.28 FPS
  - Optimalizace přístupů k paměti – 195 ms = 5.13 FPS
    (bezpodmínečně nutná znalost HW)
  - Čtyři jádra – 114 ms = 8.77 FPS
    (nutnost výběru minimální nutné synchronizace)
  - GPU (256 procesorů) — 25 ms = 40 FPS
    (znalost předávání dat mezi hlavním CPU a koprocesory)
- Naivním algoritmus, mat. knihovnou Eigen (1 jádro a 4 jádra (2 fyzické) na i7-2520M, kompilace s -O3), GPU na základě měření Joela Matějky ze zkupiny http://industrialinformatics.cz/ kde se v rámci evropských projektů vývojem operačních systémů a budoucích SW platforem pro autonomní řízení zabýváme

A co jiné systémy?

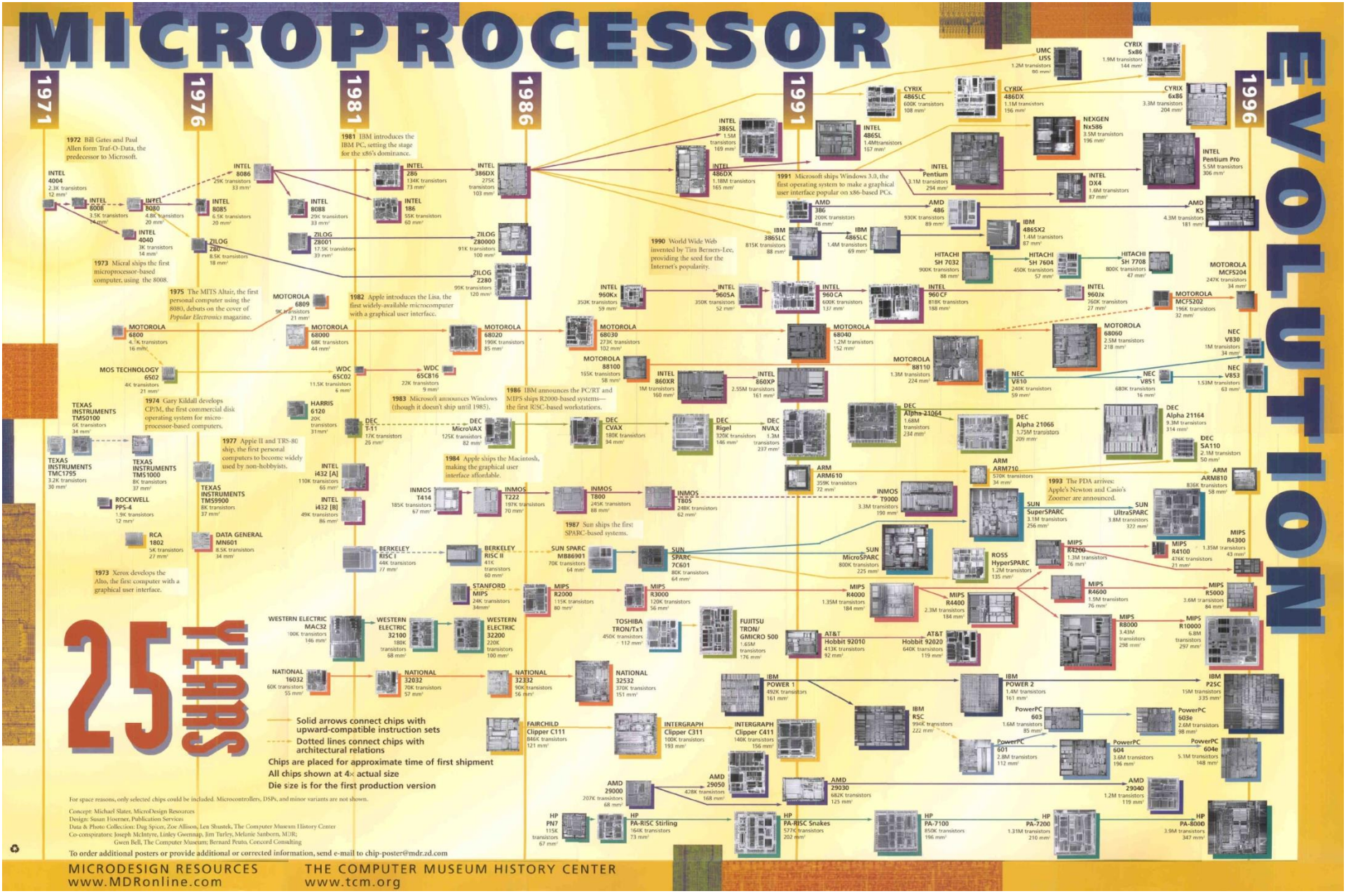
- Při využití GPU zpracujeme i 40 fps.
- Auto má ale dost energie na jejich napájení....

- Ale v „embedded“ (vestavěném) zařízení je někdy třeba snížit spotřebu a cenu. Používají se zde velmi jednoduché procesory či mikrokontroléry, někdy i bez operací s reálnými čísly, a programované nízkoúrovňovým jazykem C.



Existovalo a dodnes existuje mnoho typů od různých výrobců – principy zůstávají společné

http://research.microsoft.com/en-us/um/people/gbell/CyberMuseum_contents/Microprocessor_Evolution_Poster.jpg

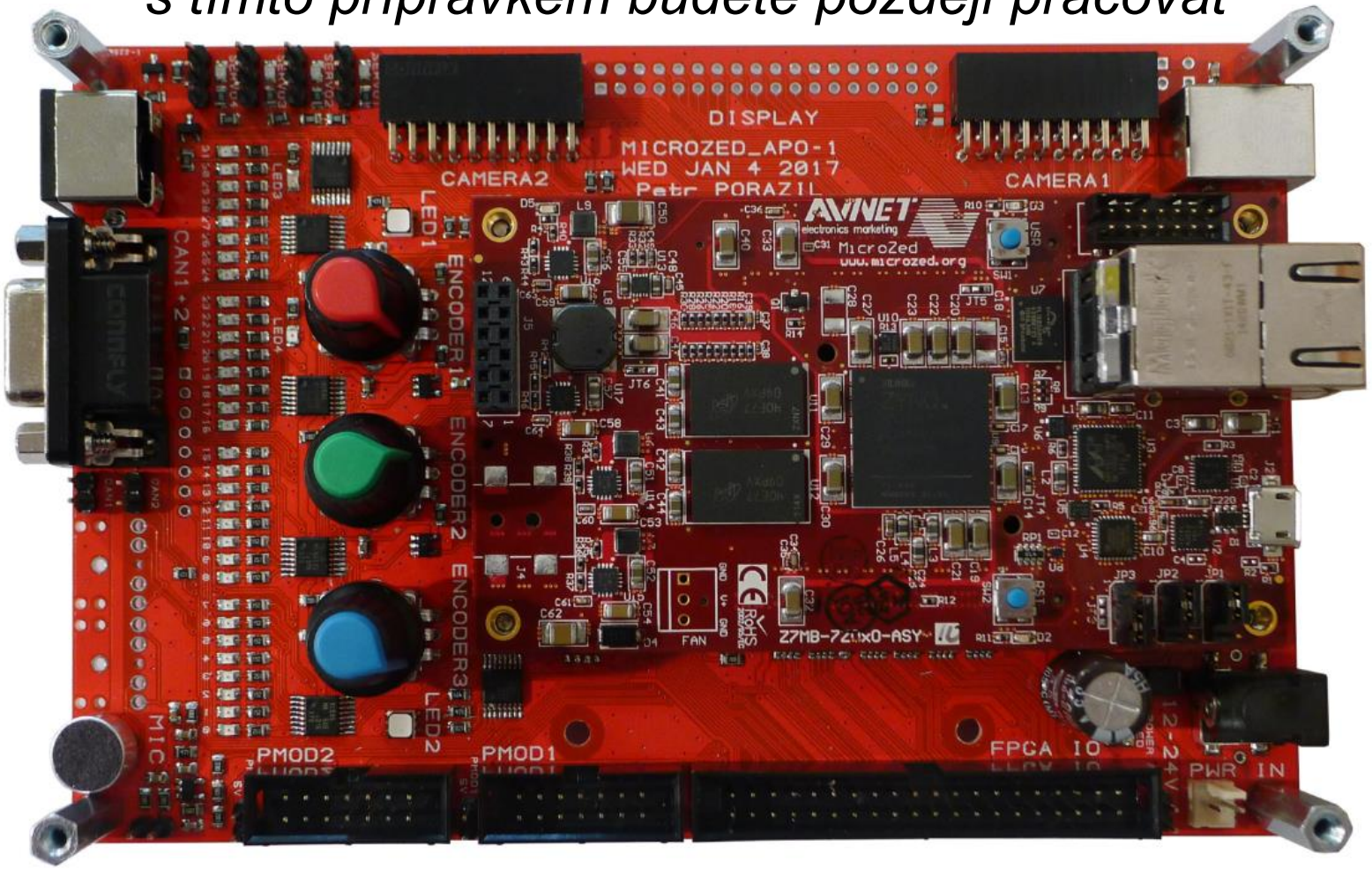


Proč je potřeba studovat i nízkoúrovňovou konstrukci počítače

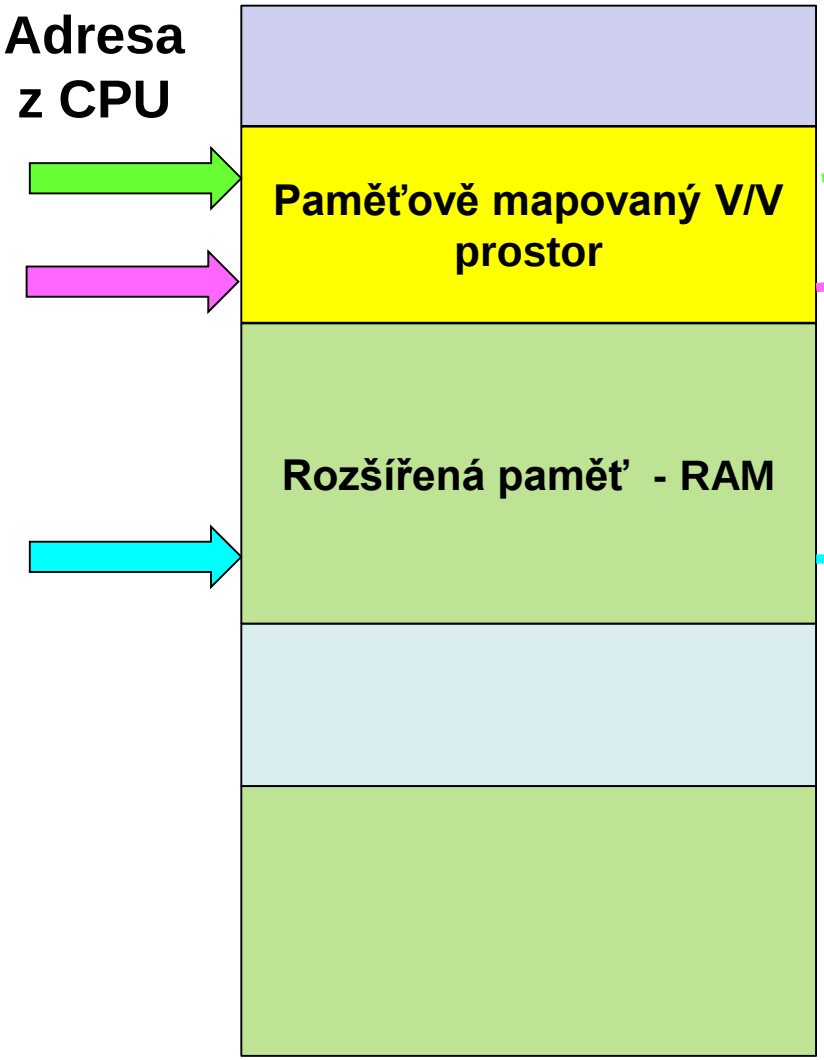
- Pro návrh nové počítačové/procesorové architektury
- Pro implementaci vybrané architektury v integrovaném obvodu/FPGA
- Pro obvodový návrh hardware/systému (velké nebo vestavné systémy)
- Pro porozumění obecným otázkám a problémům ohledně počítačů, jejich architektur a výkonnosti
- **Pro to jak efektivně využívat existující hardware** (to znamená, jak psát kvalitní software)
  - Bez přehledu a pochopení chování, možností, omezení a limitace zdrojů není možné efektivně využít žádný hardware (pro moderní HW s více jádry a výpočetními subsystému to platí dvojnásob)
  - Určitě lze vytvořit dobře placené programy i bez těchto znalostí, ale budou vyžadovat mnohonásobně silnější hardware a budou plýtvat zdroji. Není však takto možné vytvořit žádné náročné aplikace ať již na výkon nebo na ušetřen energie. Přitom to je oblast kde probíhá skutečný vývoj a mají z dlouhodobějšího technologického a i vědeckého pohledu smysl.

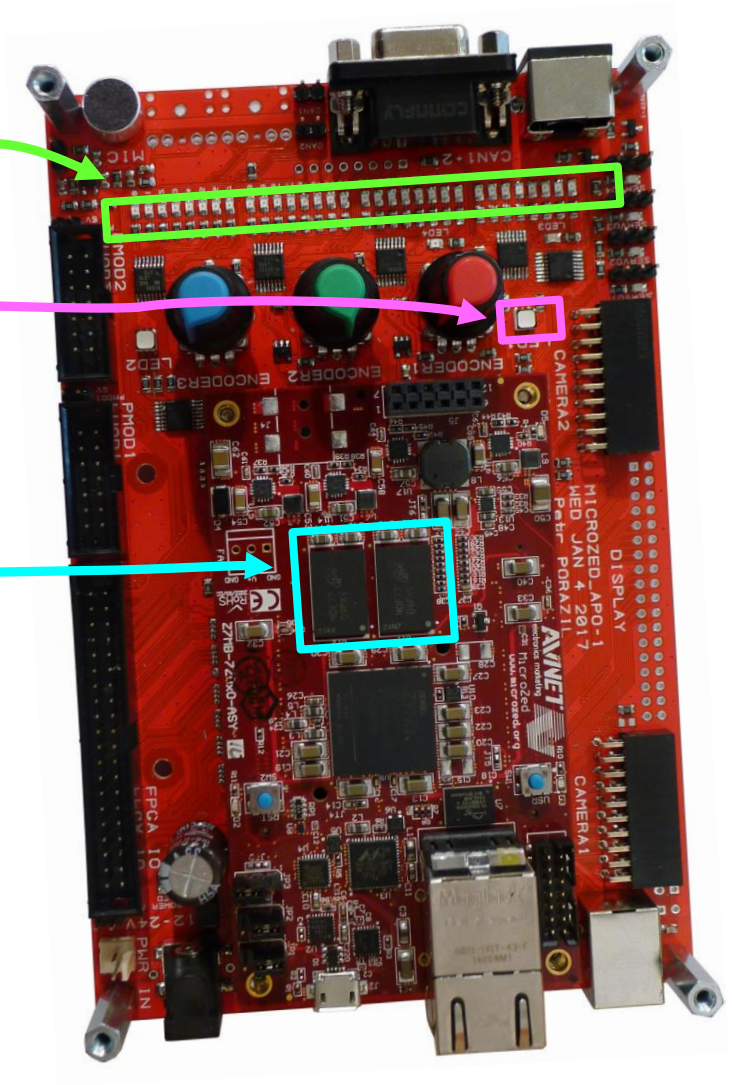
Další motivace a příklady

s tímto přípravkem budete později pracovat

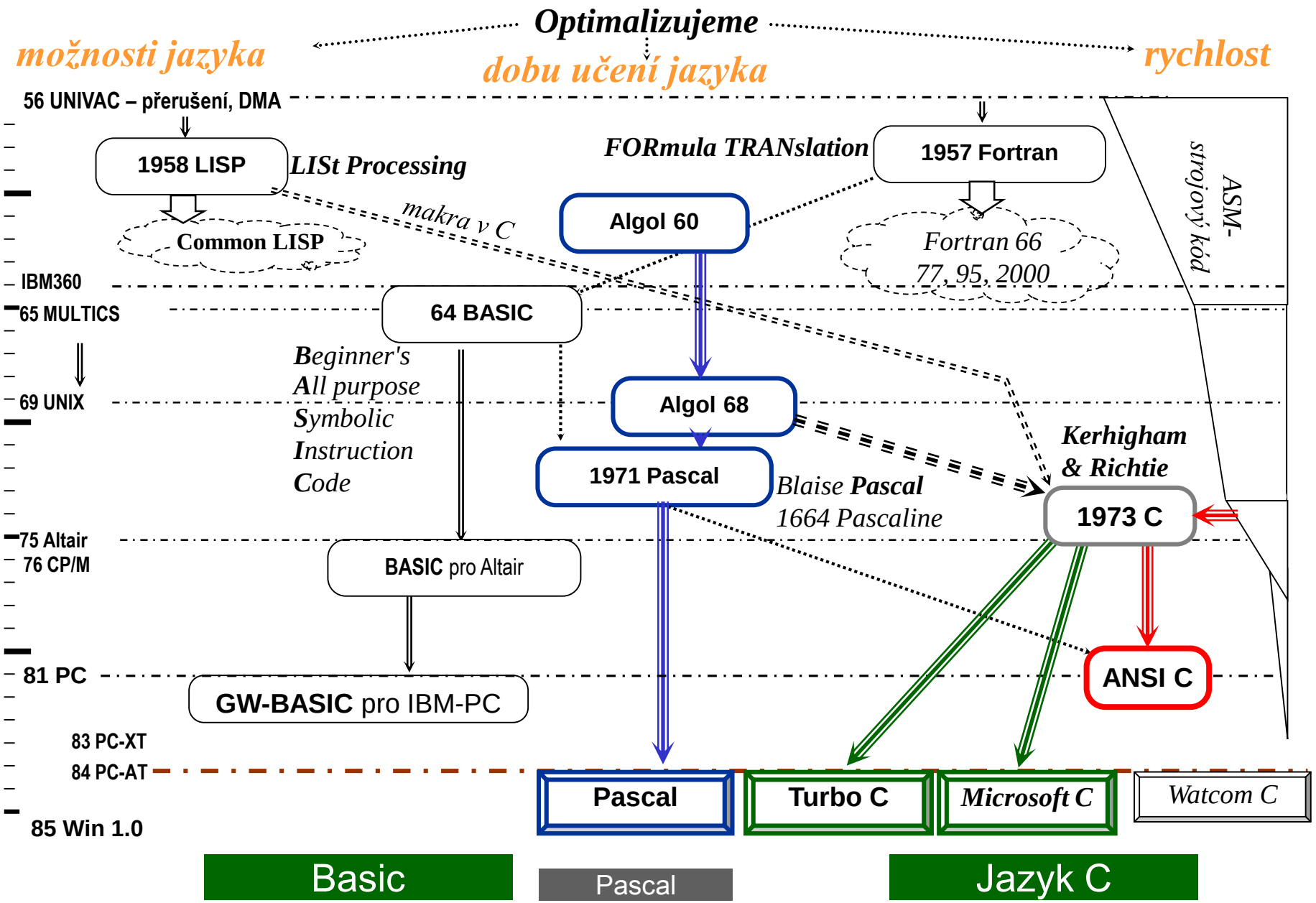


Připomeňme si… Fyzický adresní prostor

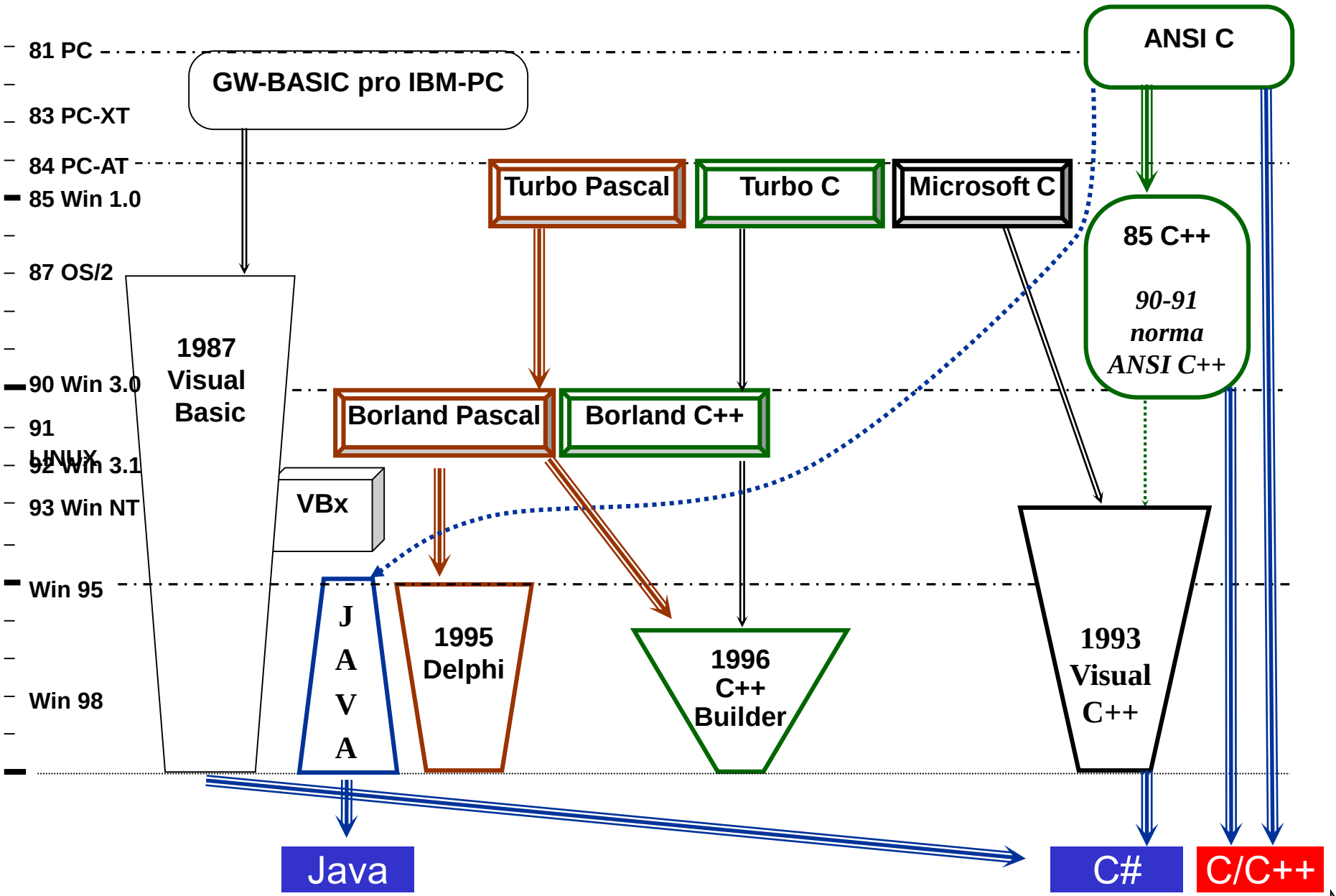




Hledání programovacího jazyka



Jazyk C pořád zůstává ve hře

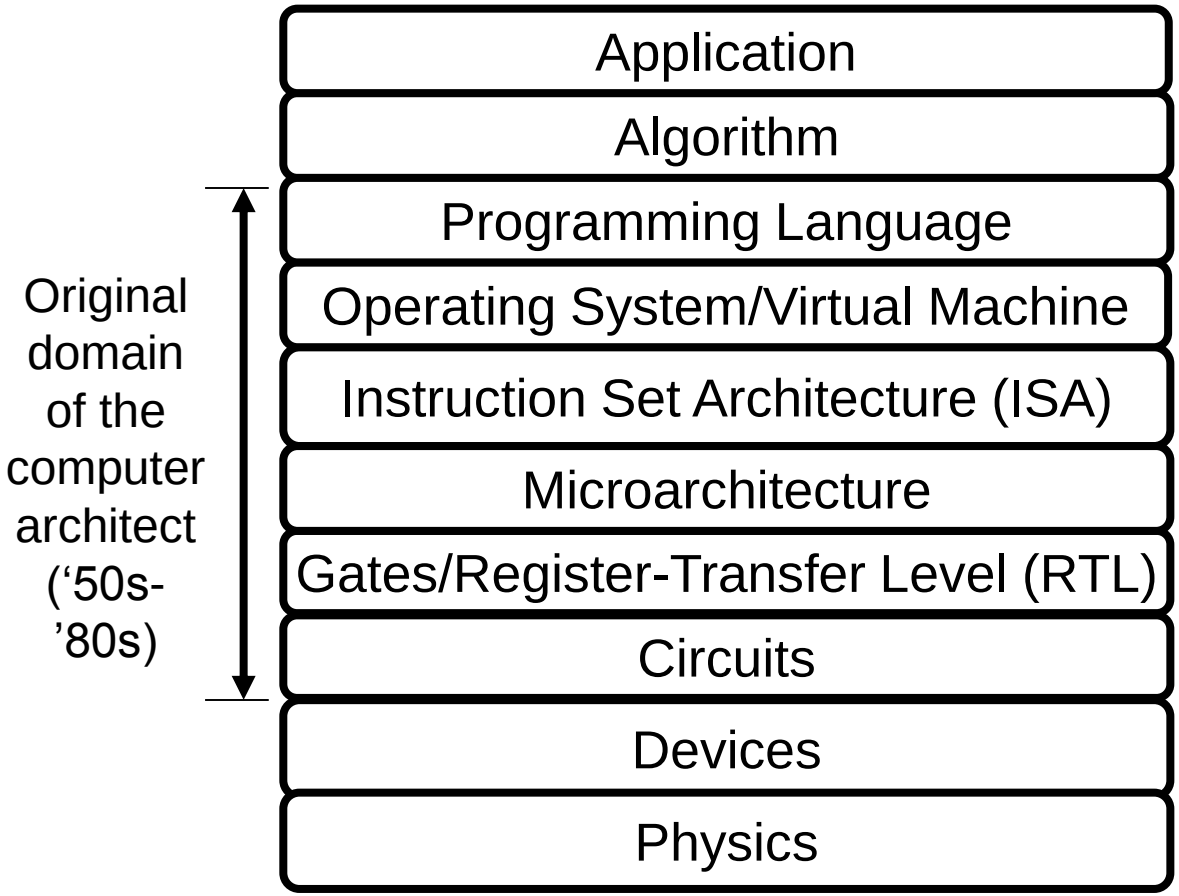


It is easy to see by formal-logical methods that there exist certain [instruction sets] that are in abstract adequate to control and cause the execution of any sequence of operation. The really decisive considerations from the present point of view, in selecting an [instruction set], are more of a practical nature: simplicity of the equipment demanded by the [instruction set], and ***the clarity of its application to the actually important problems together with the speed of its handling of those problems****.*

[Burks, Goldstine, and von Neumann, 1947]

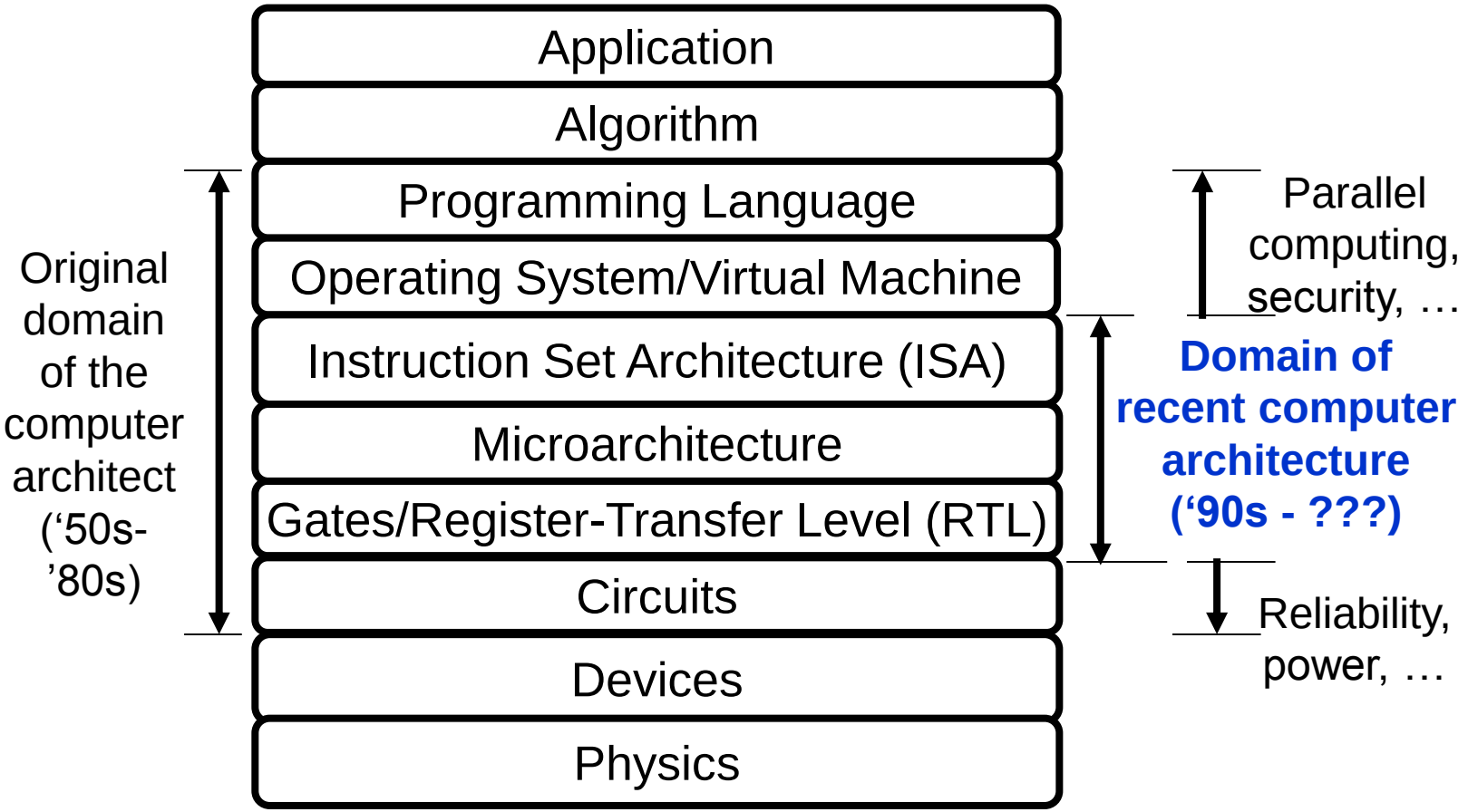
Formálně-logickými metodami lze dokázat, že existují jisté [instrukční sady], které jsou abstraktně adekvátní pro provedení jakékoliv sekvence operací. Ze současného hlediska rozhoduje při jejich výběru spíše praktické hledisko: ***jednoduchost [instrukční sady] a jasná aplikace na skutečně důležité problémy spolu s rychlostí jejich vyřešení****.*

Co je to architektura počítače?



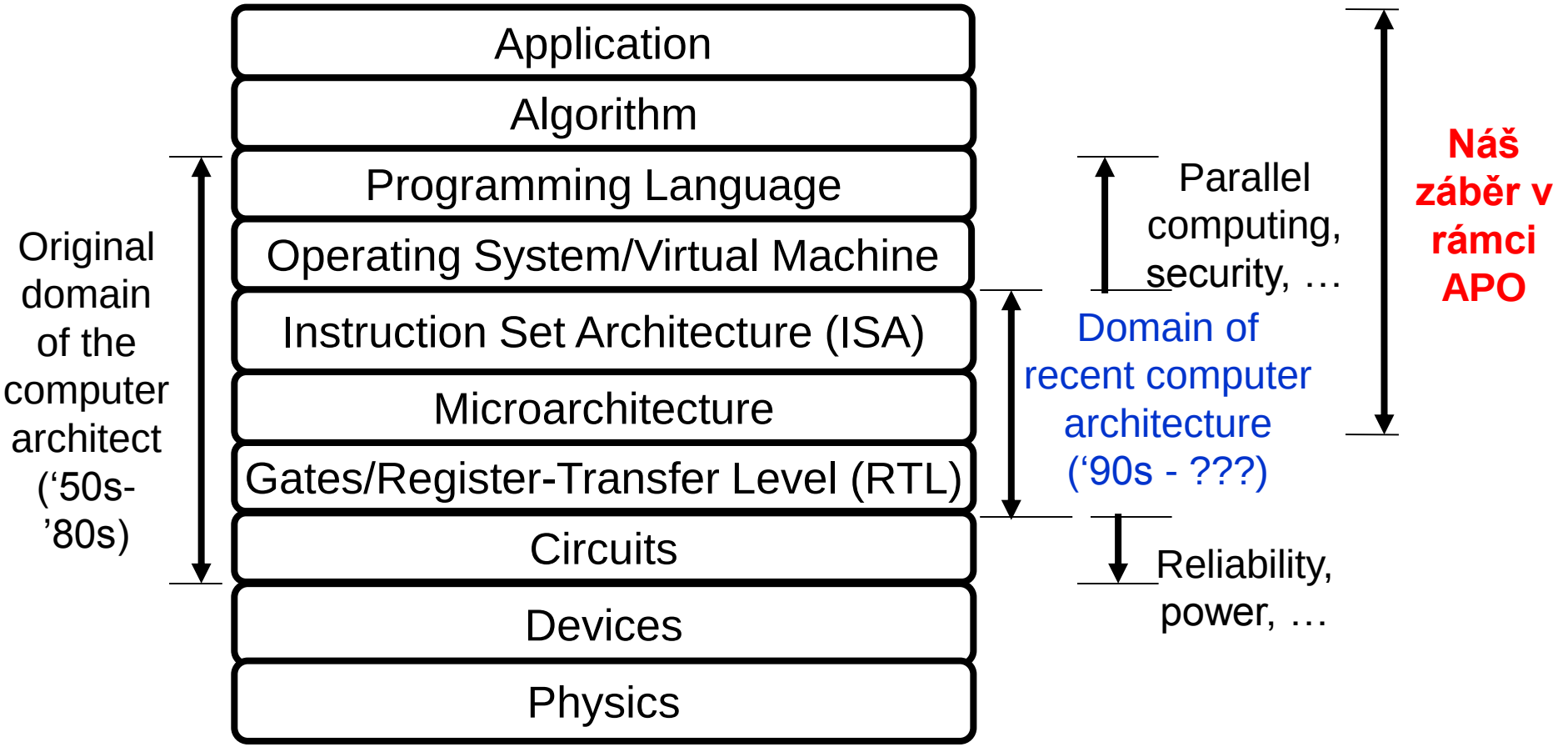
Reference: John Kubiatowicz: EECS 252 Graduate Computer Architecture, Lecture 1. University of California, Berkeley

Co je to architektura počítače?



Reference: John Kubiatowicz: EECS 252 Graduate Computer Architecture, Lecture 1. University of California, Berkeley

Co je to architektura počítače?



Reference: John Kubiatowicz: EECS 252 Graduate Computer Architecture, Lecture 1. University of California, Berkeley

Obsah 1. přednášky

- Jak se v počítači ukládají
  - Čísla typu INTEGER, bez i se znaménkem,
  - Hodnoty typu LOGICAL?
- Jak se realizují základní operace
  - Sčítání, odčítání,
  - Posuny,
  - Násobení, dělení,

Základní terminologie

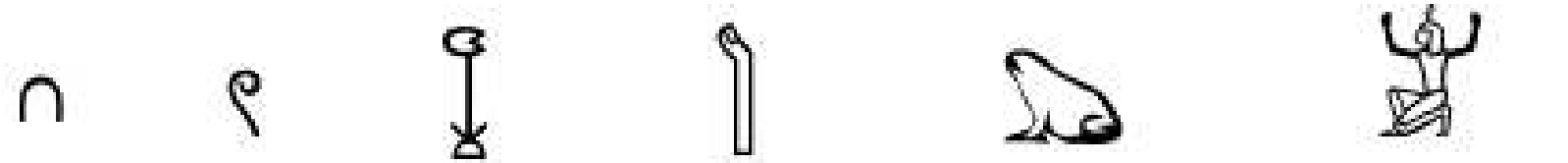
Číselná soustava:

- **Nepoziční číselná soustava -** hodnota číslice není dána jejím umístěním v dané sekvenci číslic, ale jejím vzhledem (zápisem/symbolem)

http://diameter.si/sciquest/E1.htm

- Hodnota čísla může být dána prostým součtem hodnot jednotlivých číslic (Egyptské číslice) nebo je zapotřebí použít nějaká pravidla (Římské číslice)

Základní terminologie



Hodnota čísla tedy je: 1 333 331

Základní terminologie

Číselná soustava:

- **Poziční číselná soustava -** hodnota každé číslice je dána její pozicí v sekvenci symbolů
- Množina všech symbolů (číslic) se nazývá **abeceda**
- Celá část je oddělena od zlomkové speciálním znakem (řádová čárka/tečka)

Λ – abeceda - Například {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} pro desítkovou soust.

z – základ (radix) soustavy – obvykle přirozené číslo >1

$a \in \Lambda$ - číslice

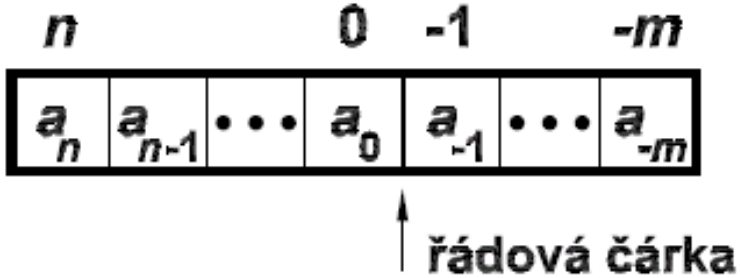
$$A \sim a_n a_{n-1} \ldots a_0, a_1 \ldots a_{-m}$$
$$A = a_n z^n + a_{n-1} z^{n-1} + \ldots + a_0 + a_1 z^{-1} \ldots a_{-m} z^{-m}$$

Základní terminologie

Například dekadické číslo 348,31

má hodnotu $\mathbf{3}*10^2 + \mathbf{4}*10^1 + \mathbf{8}*10^0 + \mathbf{3}*10^{-1} + \mathbf{1}*10^{-2}$

- Pokud si **zvolíme** fixně počet pozic pro celočíselnou část (n+1) a počet pozic pro zlomkovou část (m), bude:



- Nejmenší zobrazitelné číslo: $\varepsilon = z^{-m}$,
- **Modul** - nejmenší hodnota, kterou už neumíme zobrazit: $M = z^{n+1}$
- Zobrazitelná čísla tedy leží v rozsahu: $0 \leq A < M$

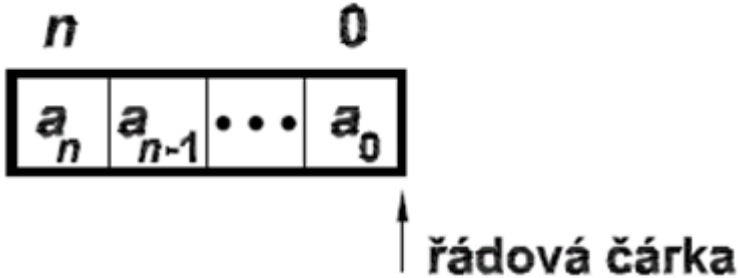
Čísla bez znaménka

Jazyk C:

unsigned int

Uložení čísel typu INTEGER **bez znaménka**

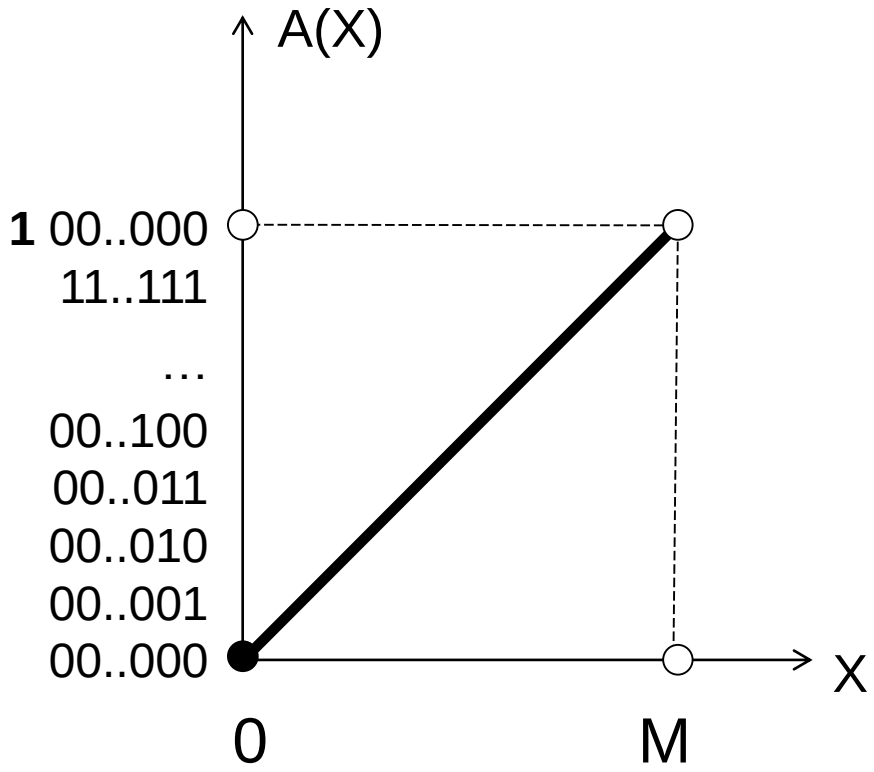
- Zvolme si tedy celkově například **8 pozic**, z toho všechny pro celočíselnou část a základ soustavy roven 2.



- Kolik je různých stavů položky?
  - 2↑8 = 256D (desítkově). To není mnoho, že?
  - Řešení: proč nepoužít více bajtů?
  - 4B = 2↑32 = 4 294 976 296D,
  - Rozsah tedy je: <0, 2^{n+1}-1>, nebo pokud **N** bude **počet bitů: <0, 2^N-1>**

Binární hodnota	Neznaménková reprezentace
00000000	$0_{(10)}$
00000001	$1_{(10)}$
⋮	⋮
01111101	$125_{(10)}$
01111110	$126_{(10)}$
01111111	$127_{(10)}$
10000000	$128_{(10)}$
10000001	$129_{(10)}$
10000010	$130_{(10)}$
⋮	⋮
11111101	$253_{(10)}$
11111110	$254_{(10)}$
11111111	$255_{(10)}$

Uložení čísel typu INTEGER **bez znaménka**

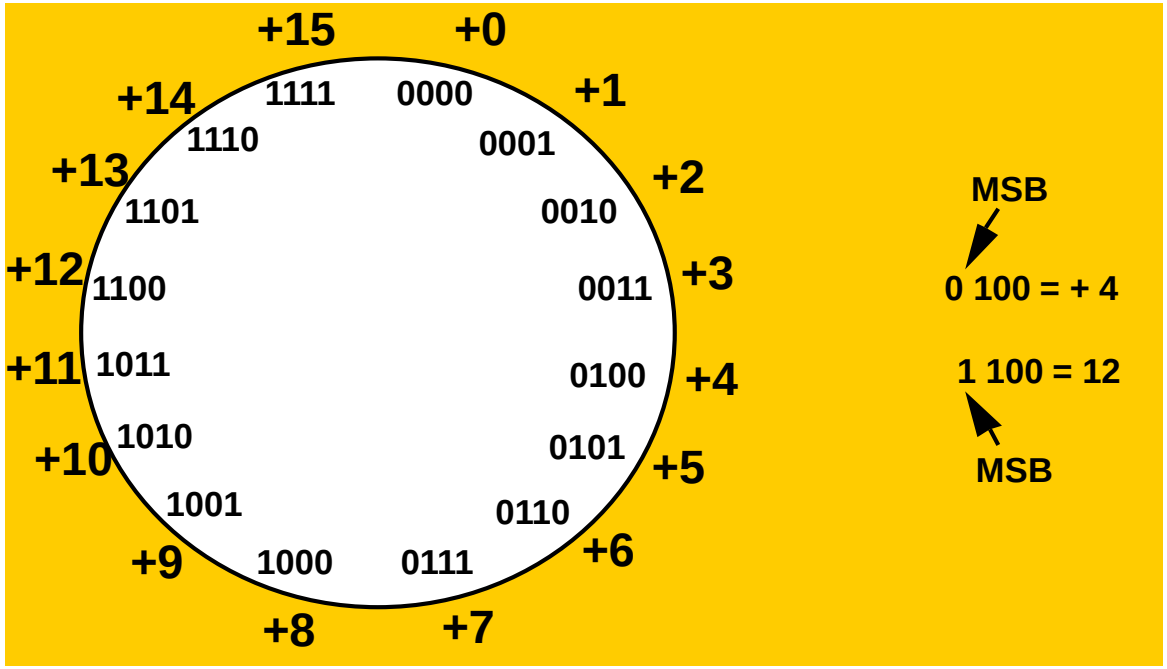


Binární hodnota	Neznaménková reprezentace
00000000	$0_{(10)}$
00000001	$1_{(10)}$
⋮	⋮
01111101	$125_{(10)}$
01111110	$126_{(10)}$
01111111	$127_{(10)}$
10000000	$128_{(10)}$
10000001	$129_{(10)}$
10000010	$130_{(10)}$
⋮	⋮
11111101	$253_{(10)}$
11111110	$254_{(10)}$
11111111	$255_{(10)}$

Reprezentace čísel bez znaménka

Předpokládejme 4bitový počítač

Unsigned 4-bit numbers - 4bitové číslo bez znaménka



- Výhoda: velký rozsah kladných čísel,
  Nevýhoda: Nepřehledné odčítání
- *Běžný kód v počítačích*

[Seungryoul Maeng:Digital Systems]

Čísla se znaménkem

Jazyk C:

`int`

`signed int`

Čísla INTEGER se znaménkem I.

Dvojkový doplněk (two's complement):

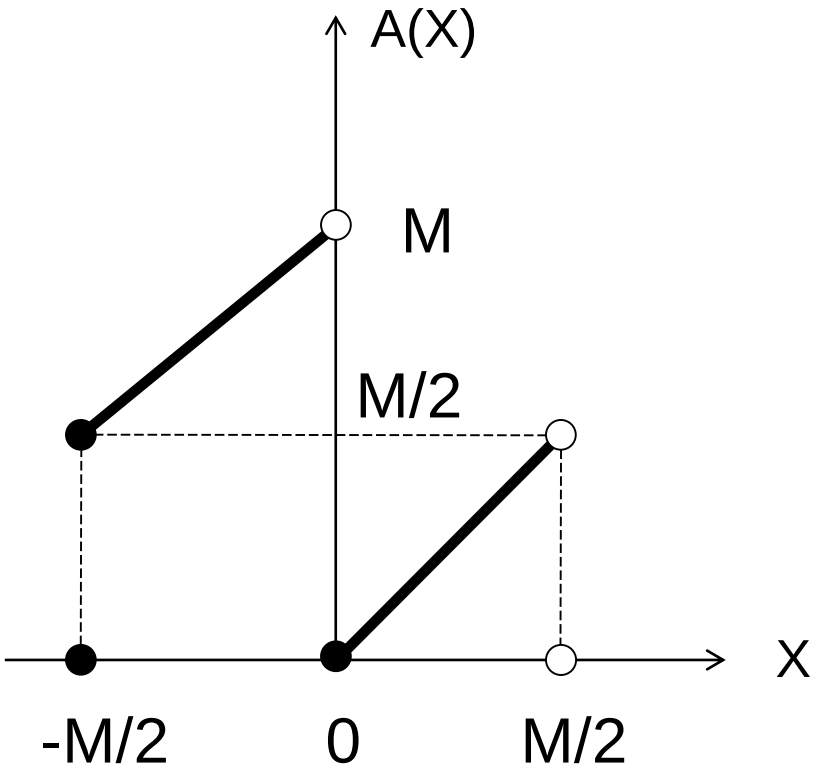
- Nejčastější varianta
- **Často se označuje nepřesně jako „doplňkový kód“**
- Dvojkový doplněk **záporného** čísla se (ve dvojkové soustavě) připočtením 1 k nejnižšímu bitu záporného čísla reprezentovaného v inverzním kódu
- Součet dvou opačných čísel se stejnou absolutní hodnotou je 00000000_H !

Dekadická hodnota	Reprezentace v dvojkovém doplňku na 4 bity
6	0110
-6	1010

Uložení čísel typu INTEGER **se znaménkem I.**

Dvojkový doplněk – pokračování…

- Pokud **N** bude **počet bitů:**
  $<-2^{N-1}, 2^{N-1}-1>$



Binární hodnota	Dvojkový doplněk
00000000	$0_{(10)}$
00000001	$1_{(10)}$
⋮	⋮
01111101	$125_{(10)}$
01111110	$126_{(10)}$
01111111	$127_{(10)}$
10000000	$-128_{(10)}$
10000001	$-127_{(10)}$
10000010	$-126_{(10)}$
⋮	⋮
11111101	$-3_{(10)}$
11111110	$-2_{(10)}$
11111111	$-1_{(10)}$

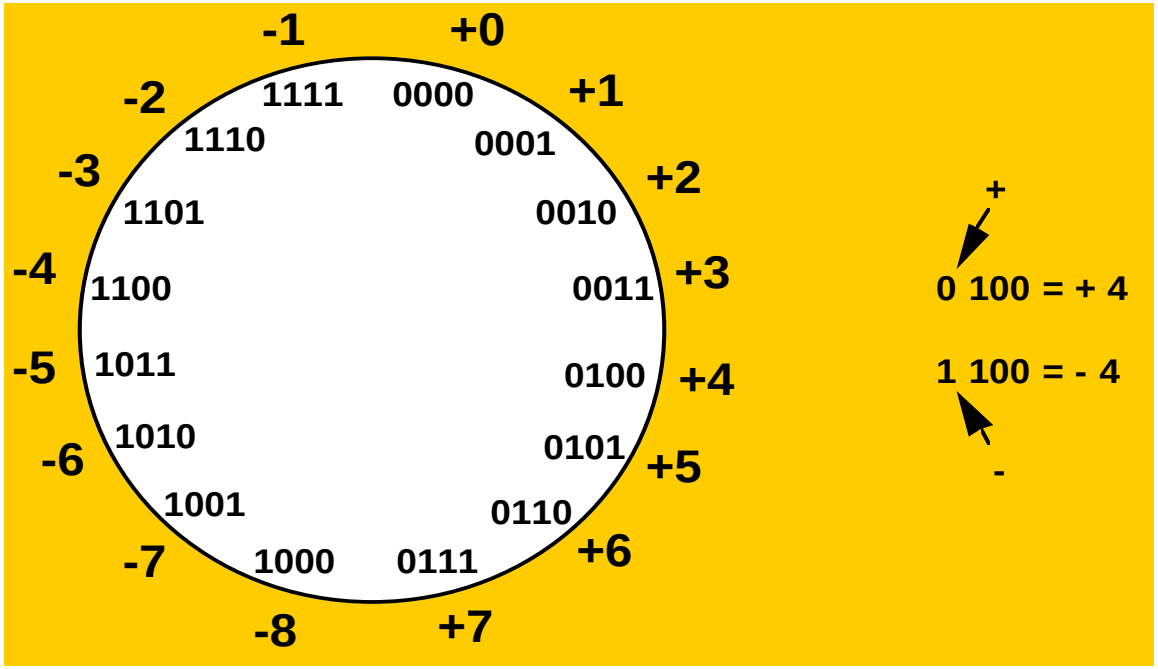
Doplňkový kód - příklady

- Příklady reprezentací:
  - $0_D = 00000000_H$,
  - $1_D = 00000001_H$,
  - $2_D = 00000002_H$,
  - $3_D = 00000003_H$,
  - $-1_D = FFFFFFFF_H$,
  - $-2_D = FFFFFFFE_H$,
  - $-3_D = FFFFFFFD_H$,
- Analogii dvojkového doplňku se v soustavě s jiným základem říká doplněk do modulu. Stejně se postupuje třeba v soustavě desítkové (doplněk do „10“).
- Přenos do vyššího řádu (32.) ignorujeme. Sčítáme vlastně mod $2\uparrow 32$.

Reprezentace čísel

Twos Complement

(cz: Dvojkový doplněk, doplňkový kód)



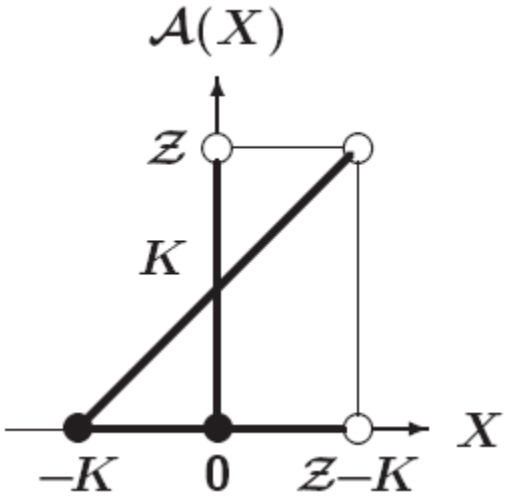
Jedna reprezentace 0, stejná aritmetická jednotka pro čísla bez znaménka i se znaménkem, ale záporných čísel máme o jedno více

Běžné uložení čísel integer se znaménkem

Jiné možnosti

Čísla INTEGER se znaménkem II.

- Je i jiná možnost pro zobrazení čísel se znaménkem?
- Ano, **kód aditivní** (jinak zvaný s posunutou nulou).



$$K = \frac{1}{2}Z - 1$$

Z je modul

Sčítání a odčítání v aditivním kódu

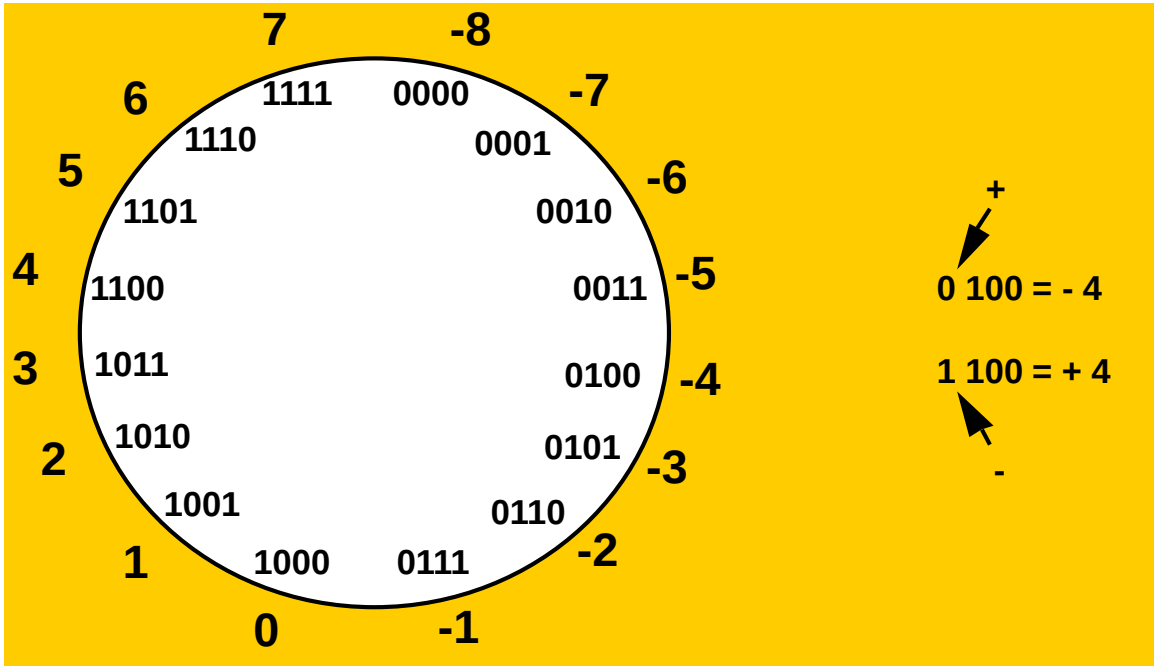
- Platí:

$$\mathcal{A}(A+B) = \mathcal{A}(A) + \mathcal{A}(B) - K$$
$$\mathcal{A}(A-B) = \mathcal{A}(A) - \mathcal{A}(B) + K$$

- Detekce přeplnění
  - sčítání: stejná znaménka sčítanců a jiné znaménko výsledku,
  - odčítání: znaménka menšence a menšitele se liší a liší se znaménka menšence a výsledku.

Reprezentace čísel se znaménkem

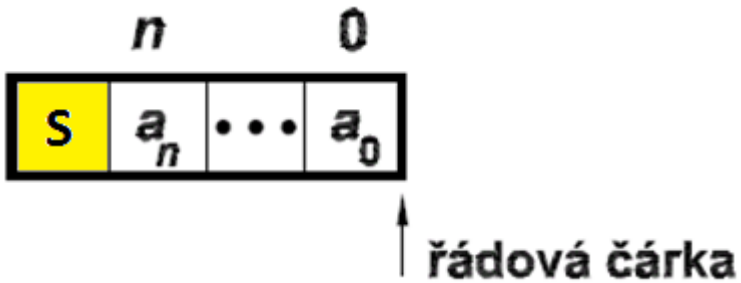
Excess-K, offset binary or biased representation
(cz: Aditivní kód, kód s posunutou nulou)



Jedna reprezentace 0, volbou offsetu lze volit počet záporných čísel.

Běžné aritmetické jednotky umí čísla porovnat, ale nedovedou již snadno provádět výpočty - *kód se používá např. pro exponenty reálných čísel (float, double...) nebo pro zpracování signálů.*

Uložení čísel typu INTEGER **se znaménkem III.**



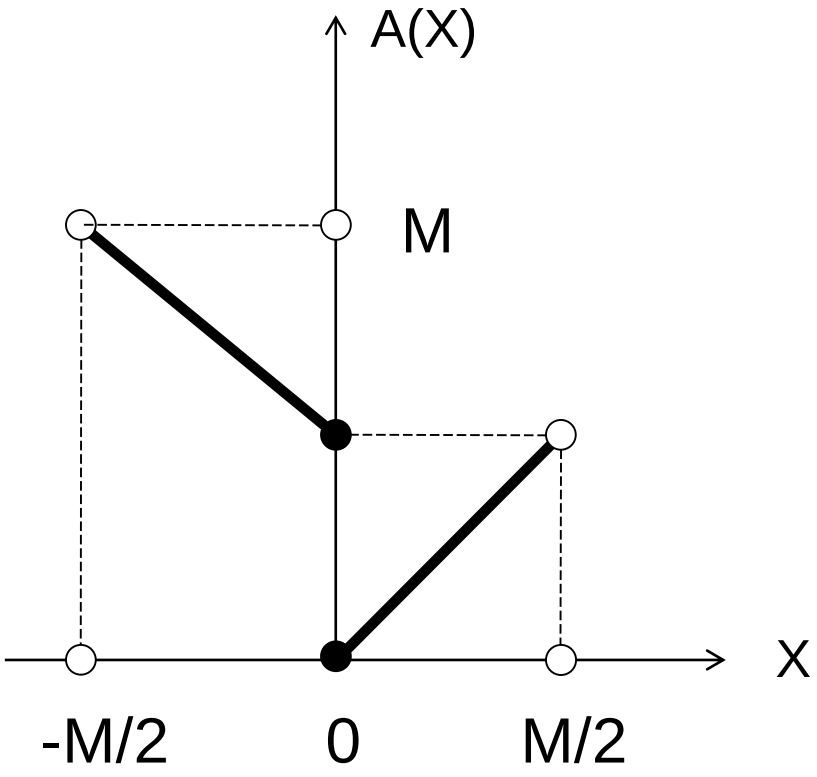
- Znaménko a hodnota. Jde o tzv. **přímý kód**.
- Běžně dodržovaná dohoda:
  - 0 ≈ +, 1 ≈ -.
- Nevýhoda: jinak se musí při aritmetických operacích pracovat se znaménkovým bitem, jinak s bity hodnoty.
- Jiná nevýhoda: máme 2 různá vyjádření nuly.

Binární hodnota	Přímý kód
00000000	$+0_{(10)}$
00000001	$1_{(10)}$
⋮	⋮
01111101	$125_{(10)}$
01111110	$126_{(10)}$
01111111	$127_{(10)}$
10000000	$-0_{(10)}$
10000001	$-1_{(10)}$
10000010	$-2_{(10)}$
⋮	⋮
11111101	$-125_{(10)}$
11111110	$-126_{(10)}$
11111111	$-127_{(10)}$

Uložení čísel typu INTEGER **se znaménkem III.**

Přímý kód – pokračování…

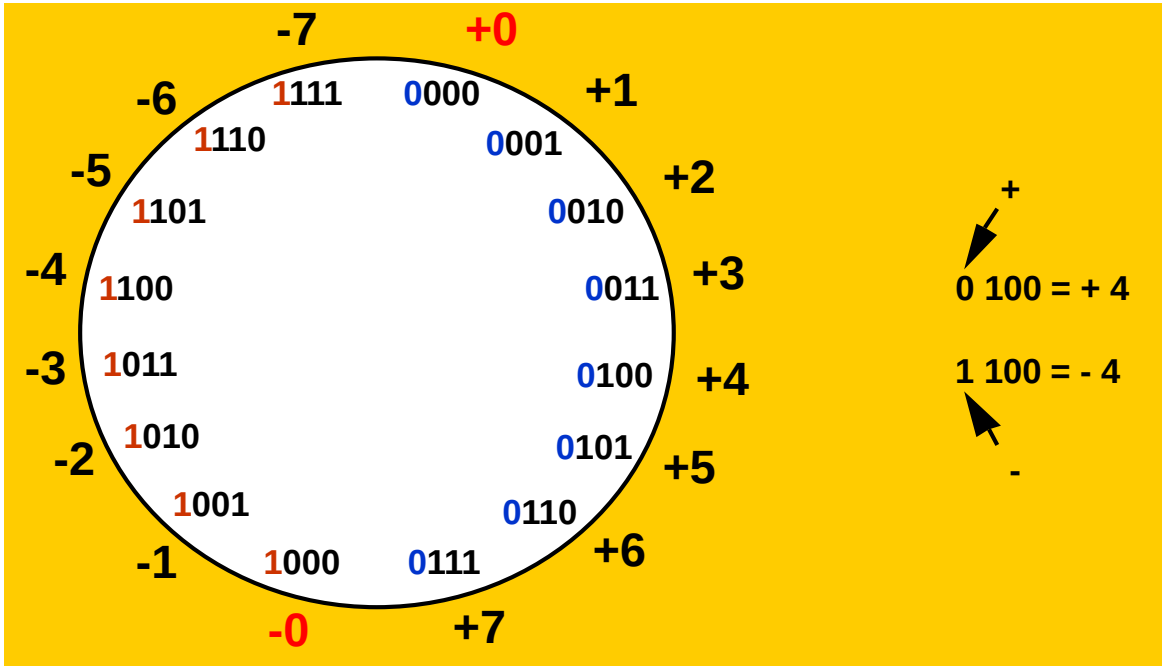
- Pokud **N** bude **počet bitů:**
  $<-2^{N-1}-1, 2^{N-1}-1>$



Binární hodnota	Přímý kód
00000000	$+0_{(10)}$
00000001	$1_{(10)}$
⋮	⋮
01111101	$125_{(10)}$
01111110	$126_{(10)}$
01111111	$127_{(10)}$
10000000	$-0_{(10)}$
10000001	$-1_{(10)}$
10000010	$-2_{(10)}$
⋮	⋮
11111101	$-125_{(10)}$
11111110	$-126_{(10)}$
11111111	$-127_{(10)}$

Reprezentace čísel se znaménkem

Znaménko a hodnota, tzv. přímý kód.
"Sign and Magnitude Representation"



- Nevýhoda: při aritmetických se pracuje se znaménkovým bitem jinak než s hodnotou, existují 2 různá vyjádření nuly
- *Používá se třeba pro mantisy reálných čísel*

[Seungryoul Maeng:Digital Systems]

Čísla INTEGER se znaménkem IV.

Inverzní kód – jedničkový doplněk (one's complement):

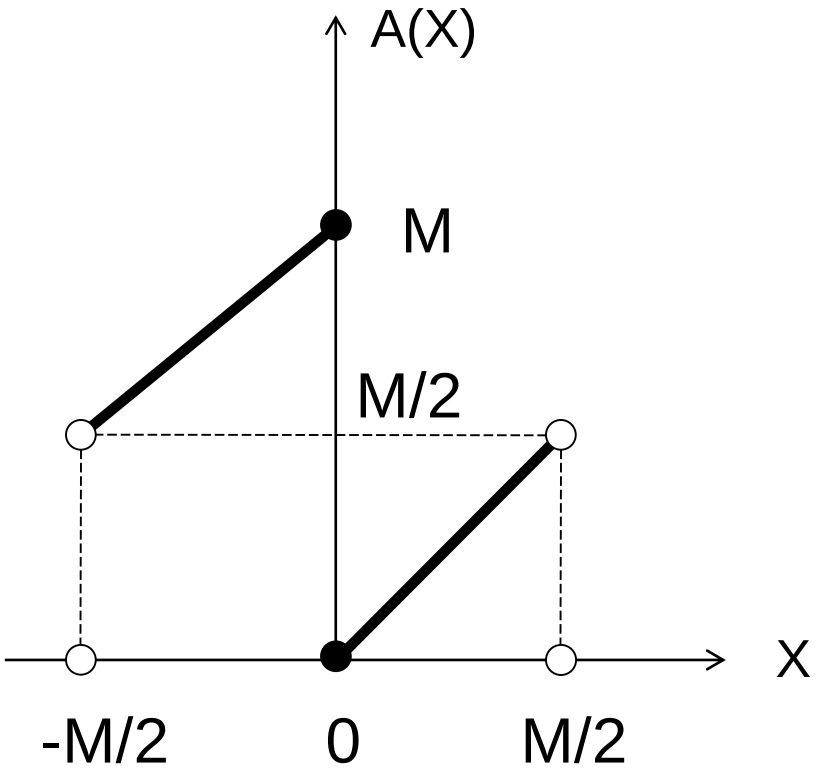
- *Spíš akademický případ, zmíněn jen pro úplnost...*
- Jedničkový doplněk **záporného** čísla se (ve dvojkové soustavě) vytvoří bitovou negací čísla **kladného =>** inverze všech bitů (jedničkový doplněk)

Dekadická hodnota	Reprezentace v inverzím kódu na 4 bity
6	0110
-6	1001

Uložení čísel typu INTEGER **se znaménkem II.**

Inverzní kód – pokračování…

- Pokud **N** bude **počet bitů:**
  $<-2^{N-1}-1, 2^{N-1}-1>$

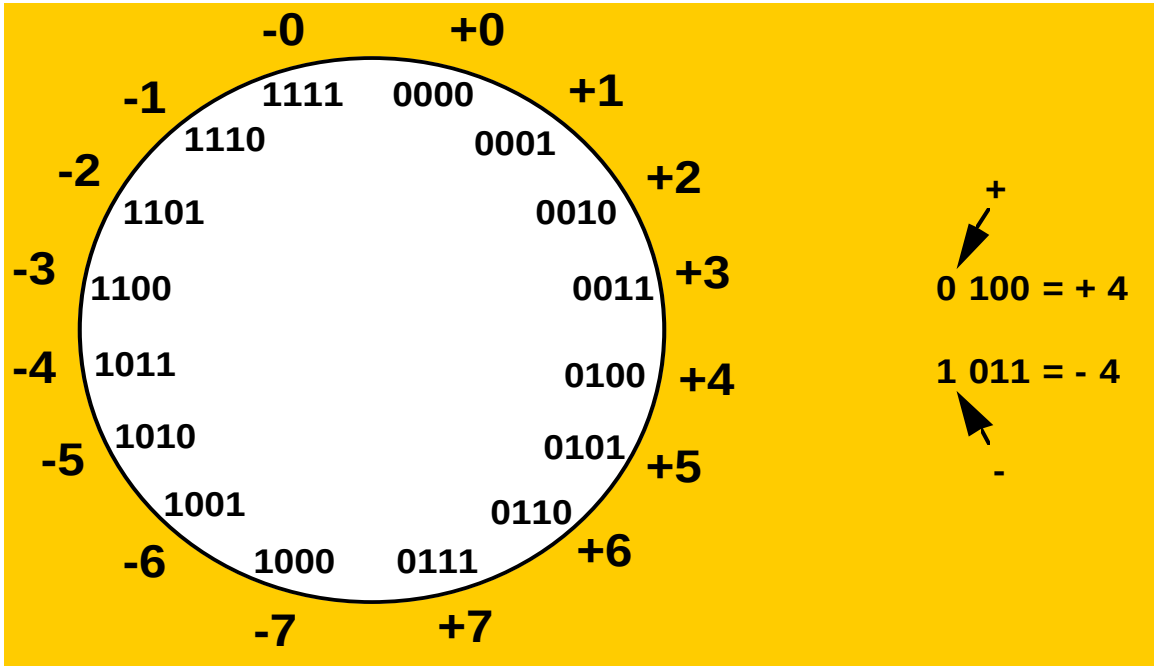


Binární hodnota	Inverzní kód
00000000	$0_{(10)}$
00000001	$1_{(10)}$
⋮	⋮
01111101	$125_{(10)}$
01111110	$126_{(10)}$
01111111	$127_{(10)}$
10000000	$-127_{(10)}$
10000001	$-126_{(10)}$
10000010	$-125_{(10)}$
⋮	⋮
11111101	$-2_{(10)}$
11111110	$-1_{(10)}$
11111111	$-0_{(10)}$

Reprezentace čísel se znaménkem

Ones Complement

(cz: Jedničkový doplněk, inverzní kód)



Dvě reprezentace 0! Obtížnější sčítání

Používá se velmi málo, kromě rychlé tvorby záporných čísel nepřináší žádné výhody.

OPERACE S CELÝMI ČÍSLY

Přímá realizace sčítačky jako logické funkce

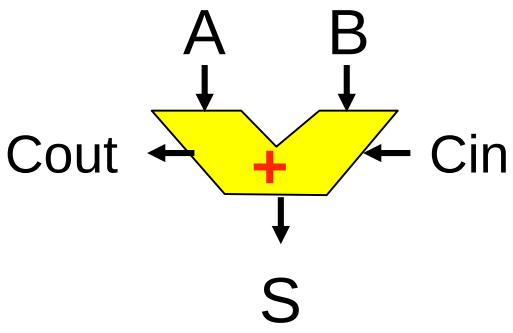
n-bitové sčítance	Celkem logických operací pro výpočet součtu
1	3
2	22
3	89
4	272
5	727
6	1567
7	3287
8	7127
9	17623
10	53465
11	115933

Složitost je vyšší než $O(2^n)$

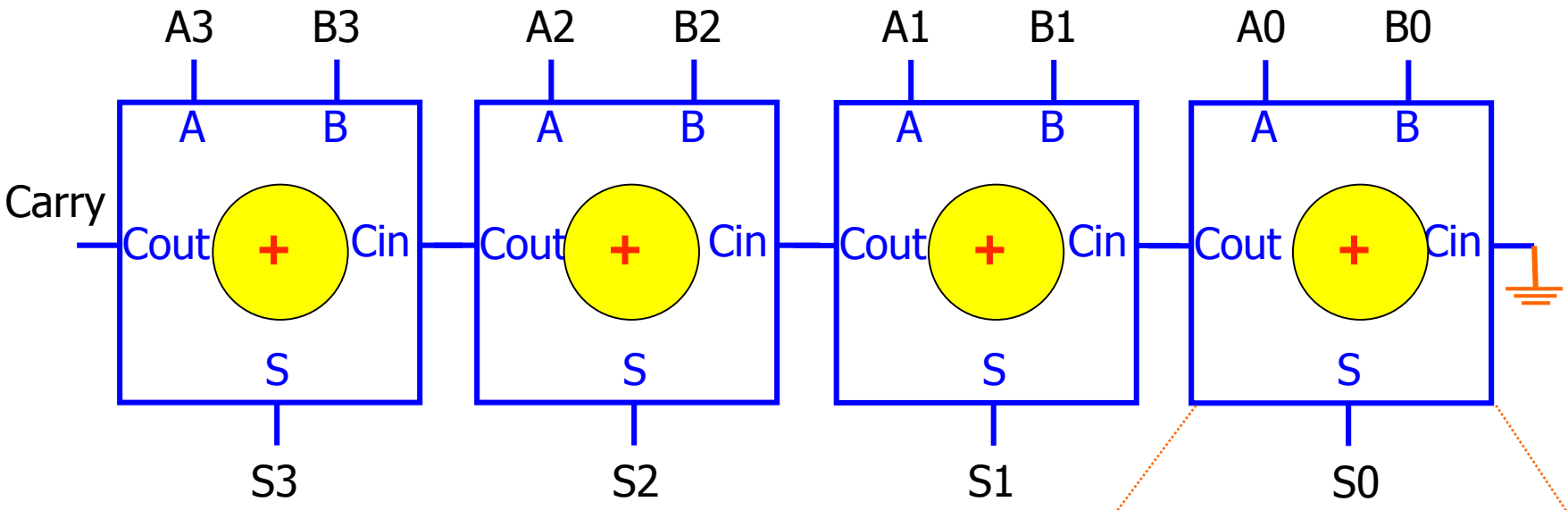
Výpočet provedený logickým minimalizátorem BOOM vytvořeným na Katedře počítačů ČVUT-FEL

Úplná 1bitová sčítačka

A	0	0	1	1	0	0	1	1
+B	0	1	0	1	0	1	0	1
Sum	00	01	01	10	00	01	01	10
+ Carry-In	0	0	0	0	1	1	1	1
CarryOut Sum	00	01	01	10	01	10	10	11

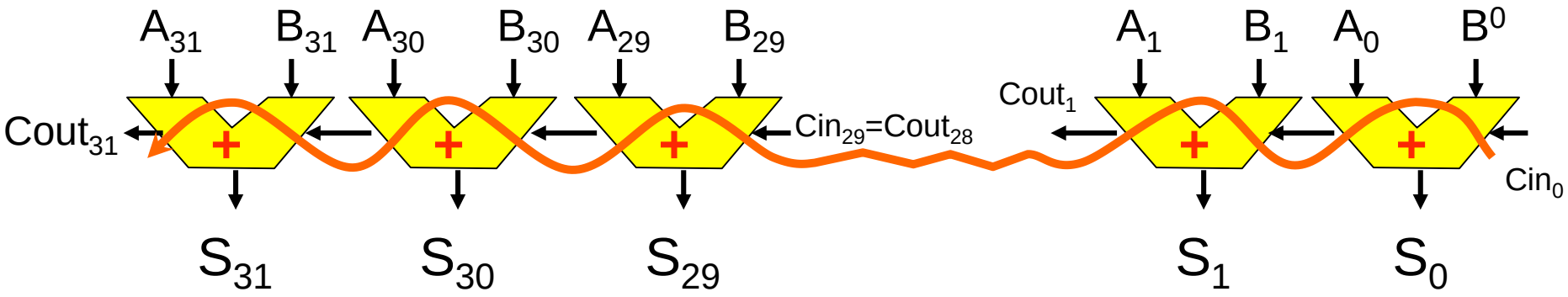


Sčítačka



Jednobitová
úplná sčítačka

Nejjednodušší sčítačka



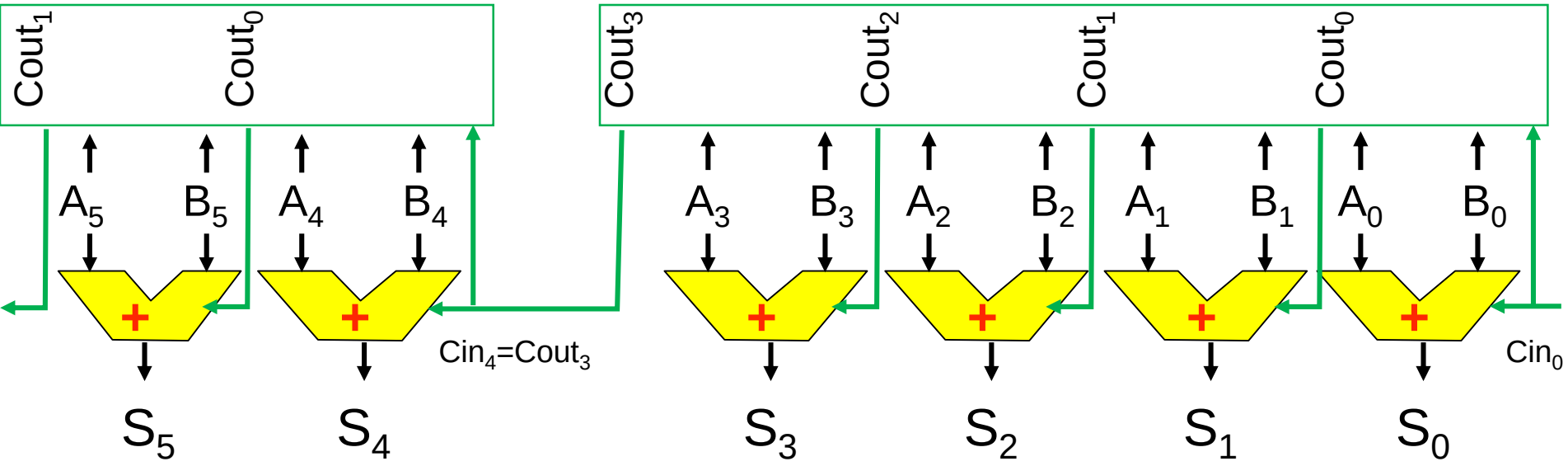
Nejjednodušší N-bitová sčítačka

- Řetězíme 1-bitové úplné sčítačky
- "Carry" (přenos do vyššího řádu) prochází skrz řetězec
- Zpoždění určeno Cout a rovná se (2N+1) krát zpoždění jednoho logického hradla

32bitová CLA "carry look-ahead" sčítačka

po skupinách urychlujeme šíření přenosu do vyššího řádu (Cout)

pomocí "carry-lookahead" (cz: předvídání přenosu)



Statická "carry look ahead (CLA)" jednotka po 4 bitech.

Doba je zhruba = 3*počet čtveřic * zpoždění jednoho hradla

Increment / Decrement

***Velmi rychlé operace, které nepotřebují sčítačku*!**

Poslední bit se vždy neguje, a ty předchozí pak podle podle koncových 1 / 0

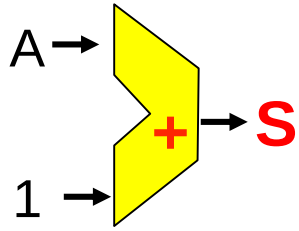
Dec.	Binary 8 4 2 1	+1	Binary 8 4 2 1	-1
0	0000	0001	0000	1111
1	0001	0010	0001	0000
2	0010	0011	0010	0001
3	0011	0100	0011	0010
4	0100	0101	0100	0011
5	0101	0110	0101	0100
6	0110	0111	0110	0101
7	0111	1000	0111	0110
8	1000	1001	1000	0111
9	1001	1010	1001	1000
10	1010	1011	1010	1001
11	1011	1100	1011	1010
12	1100	1101	1100	1011
13	1101	1110	1101	1100
14	1110	1111	1110	1101
15	1111	0000	1111	1110

Speciální případ +1/-1

S0=not A0

S1=A1 xor A0

S2=A2 xor (A1 and A0)

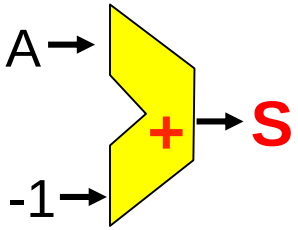


Obecný vztah: $S_i = A_i$ xor (A_{i-1} and A_{i-2} and … A_1 and A_0); i=0..n-1

S0=not A0

S1=A1 xor (not A0)

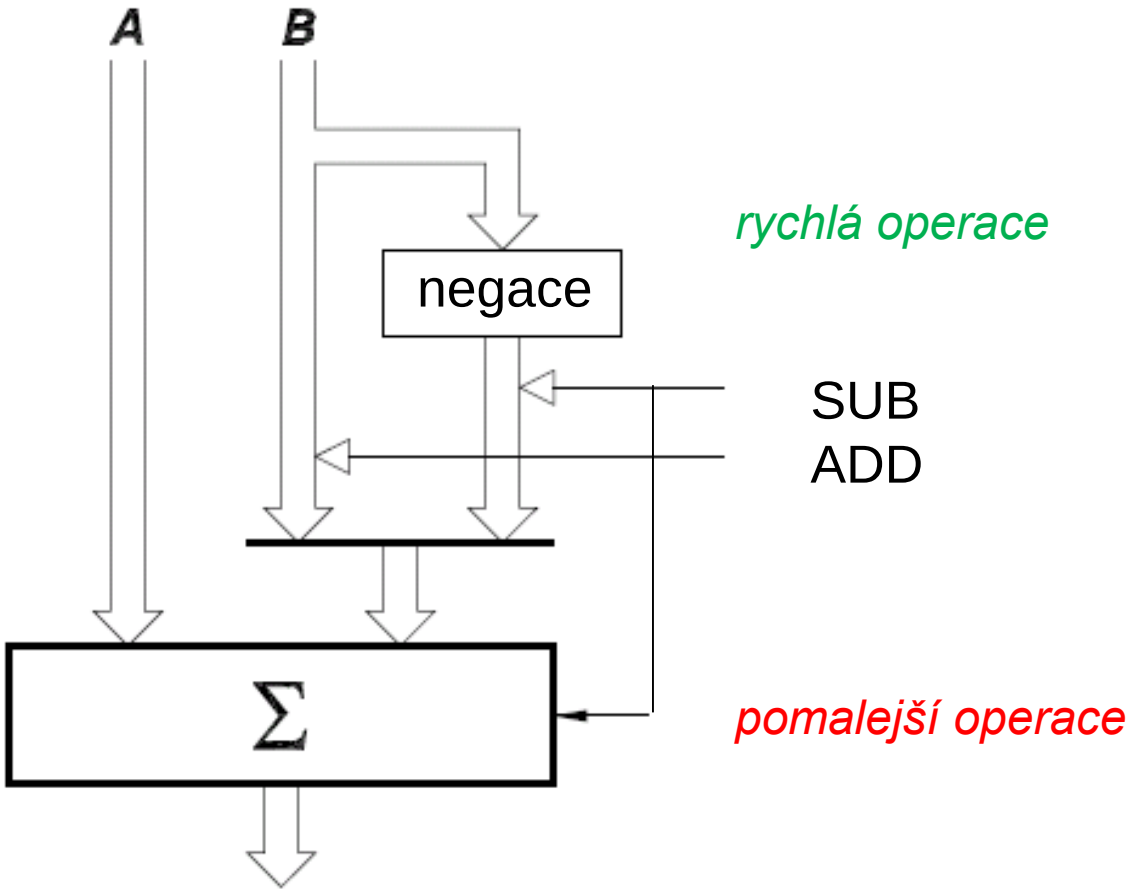
S2=A2 xor (not A1 and not A0)



Obecný vztah: $S_i = A_i$ xor (not A_{i-1} and … and not A_0); i=0..n-1

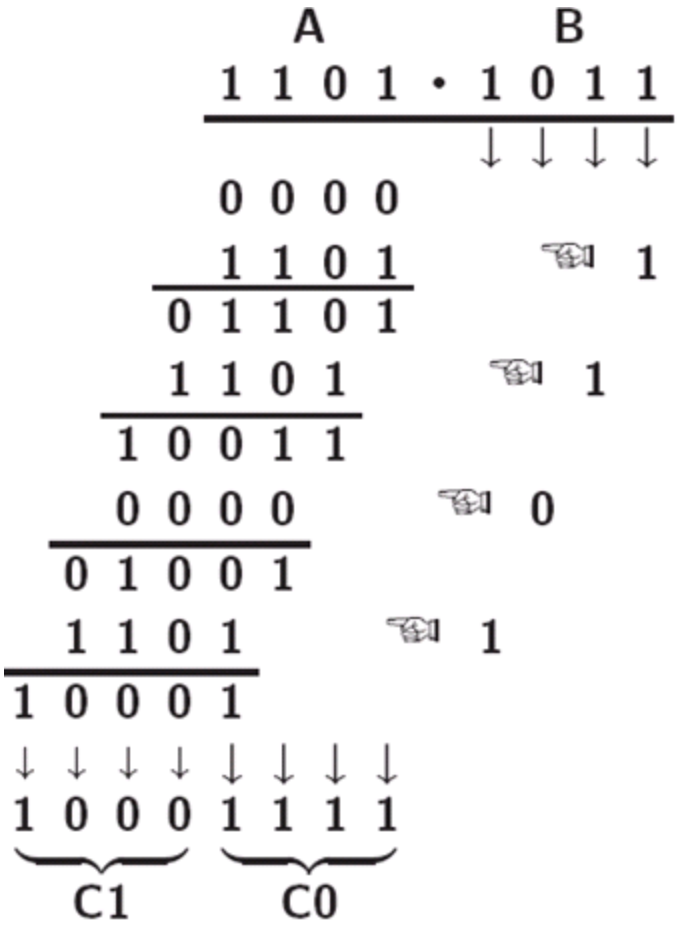
Počet obvodů +1/-1 operace je dán součtem aritmetické řady, složitost je tedy $O(n^2)$, kde n je počet bitů čísla. Operaci lze provést paralelně pro všechny bity a pro obě operace využít obvod lišící se jen negacemi.

Sčítací/odčítací HW

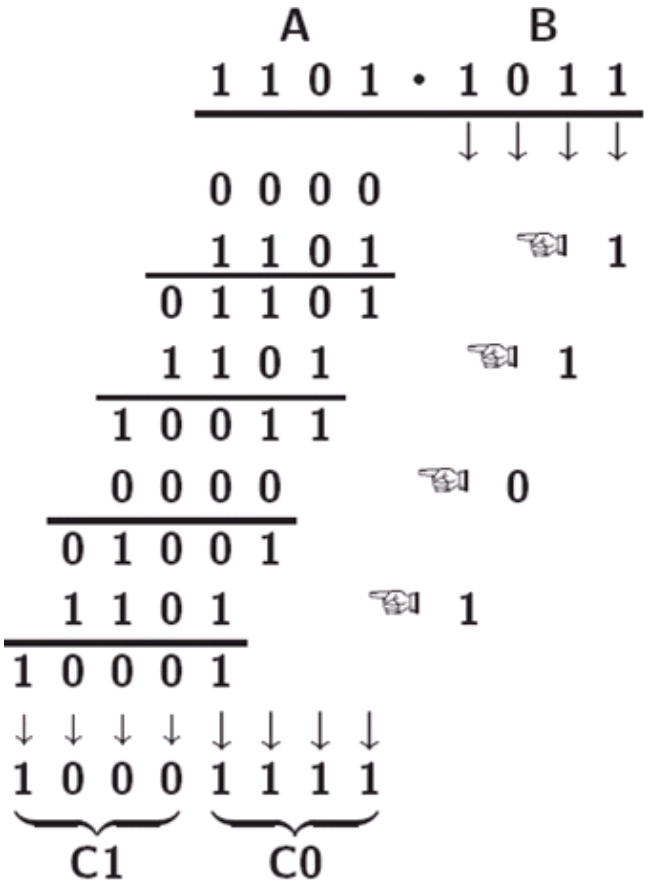


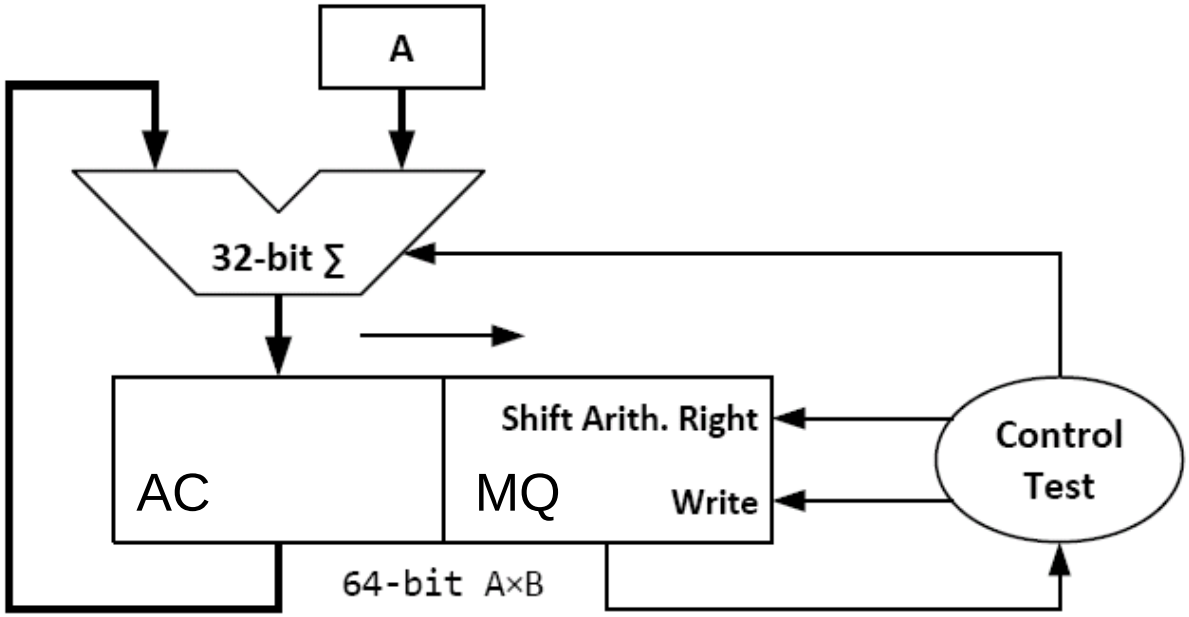
Inspirace: X36JPO, A. Pluháček

Násobení binárních čísel bez znaménka – pro připomenutí



Sekvenční HW násobička (varianta 32b)





Diskuze o rychlosti: ta je ale pomalá, co?

Algoritmus

A = násobenec;
MQ = násobitel;
AC = 0;

for(int i=1; i <= *n*; i++) // *n* je počet bitů
{
  if(MQ_0 = = 1) AC = AC + A; // MQ_0 = nejnižší bit MQ

  SR (posuň registr AC MQ o jedno místo doprava a doplň případný přenos z nejvyššího řádu z předchozího kroku)
}
end.

Nyní je výsledek v AC MQ.

Příklad x.y

Násobenec x=110 a násobitel y=101.

i	operace	AC	MQ	A	komentář
		000	101	110	prvotní nastavení
1	AC = AC+MB	110	101		začátek cyklu
	SR	011	010		
2	nic	011	010		protože MQ_0 = = 0
	SR	001	101		
3	AC = AC+MB	111	101		
	SR	011	110		konec cyklu

Tedy: x×y = 110×101 = 011110, (6×5 = 30)

Násobení v doplňkovém kódu

Lze realizovat, ale je tu problém…

Obraz součinu obecně není roven součinu obrazů!

Řešení spočívá v modifikaci dvojkové soustavy na soustavu s relativními číslicemi $0,\ 1,\ \widehat{1} = -1$

Podrobnosti už jsou mimo zamýšlený rozsah APO.

$$\begin{aligned}
\text{př.:}\quad & 1\,\widehat{1}\,1\,0 \;\sim\; 1\cdot 8 + (-1)\cdot 4 + 1\cdot 2 + 0\cdot 1 \;=\; 6 \\
& \widehat{1}\,1\,\widehat{1}\,0 \;\sim\; (-1)\cdot 8 + 1\cdot 4 + (-1)\cdot 2 + 0\cdot 1 \;=\; -6 \\
& 1\,1\,0 = 1\,0\,\widehat{1}\,0 = 1\,\widehat{1}\,1\,0 \;\sim\; 6
\end{aligned}$$

Nebo v znaménkovém rozšíření na 2N bitů a násobení obvyklým způsobem. Z výsledku bereme pouze 2N bitů. -> „ruční“ násobení

Rychlá násobička podle Walaceova stromu

Q=X .Y, X a Y nechť jsou 8b čísla

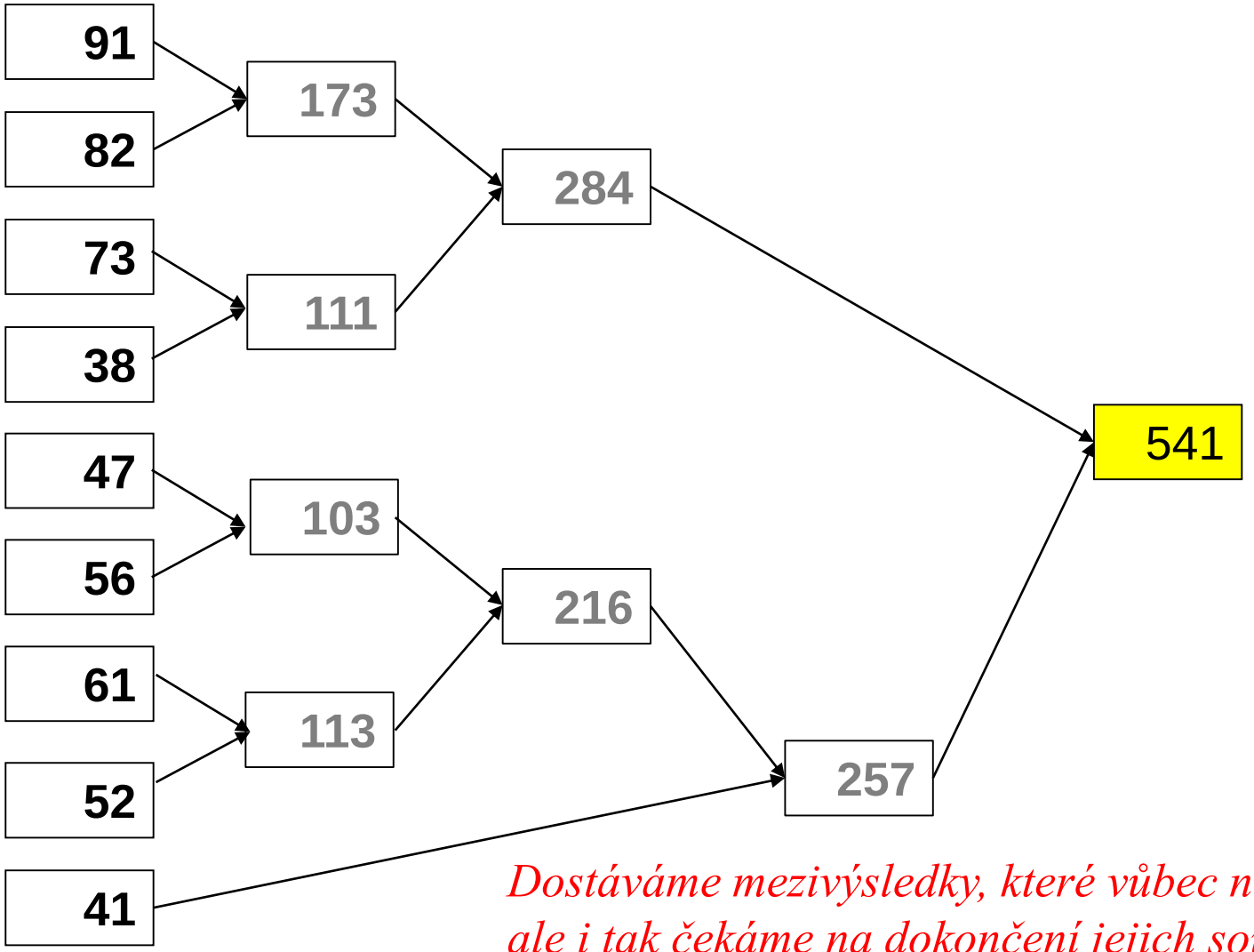
(x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0). (y_7 y_6 y_5 y_4 y_3 y_2 y_1 y_0) =

0	0	0	0	0	0	0	0	x_7y_0	x_6y_0	x_5y_0	x_4y_0	x_3y_0	x_2y_0	x_1y_0	x_0y_0	P0
0	0	0	0	0	0	0	x_7y_1	x_6y_1	x_5y_1	x_4y_1	x_3y_1	x_2y_1	x_1y_1	x_0y_1	0	P1
0	0	0	0	0	0	x_7y_2	x_6y_2	x_5y_2	x_4y_2	x_3y_2	x_2y_2	x_1y_2	x_0y_2	0	0	P2
0	0	0	0	0	x_7y_3	x_6y_3	x_5y_3	x_4y_3	x_3y_3	x_2y_3	x_1y_3	x_0y_3	0	0	0	P3
0	0	0	0	x_7y_4	x_6y_4	x_5y_4	x_4y_4	x_3y_4	x_2y_4	x_1y_4	x_0y_4	0	0	0	0	P4
0	0	0	x_7y_5	x_6y_5	x_5y_5	x_4y_5	x_3y_5	x_2y_5	x_1y_5	x_0y_5	0	0	0	0	0	P5
0	0	x_7y_6	x_6y_6	x_5y_6	x_4y_6	x_3y_6	x_2y_6	x_1y_6	x_0y_6	0	0	0	0	0	0	P6
0	x_7y_7	x_6y_7	x_5y_7	x_4y_7	x_3y_7	x_2y_7	x_1y_7	x_0y_7	0	0	0	0	0	0	0	P7
Q_{15}	Q_{14}	Q_{13}	Q_{12}	Q_{11}	Q_{10}	Q_9	Q_8	Q_7	Q_6	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0	

Součtem P0+P1+...+P7 získáme výsledek součinu X a Y.

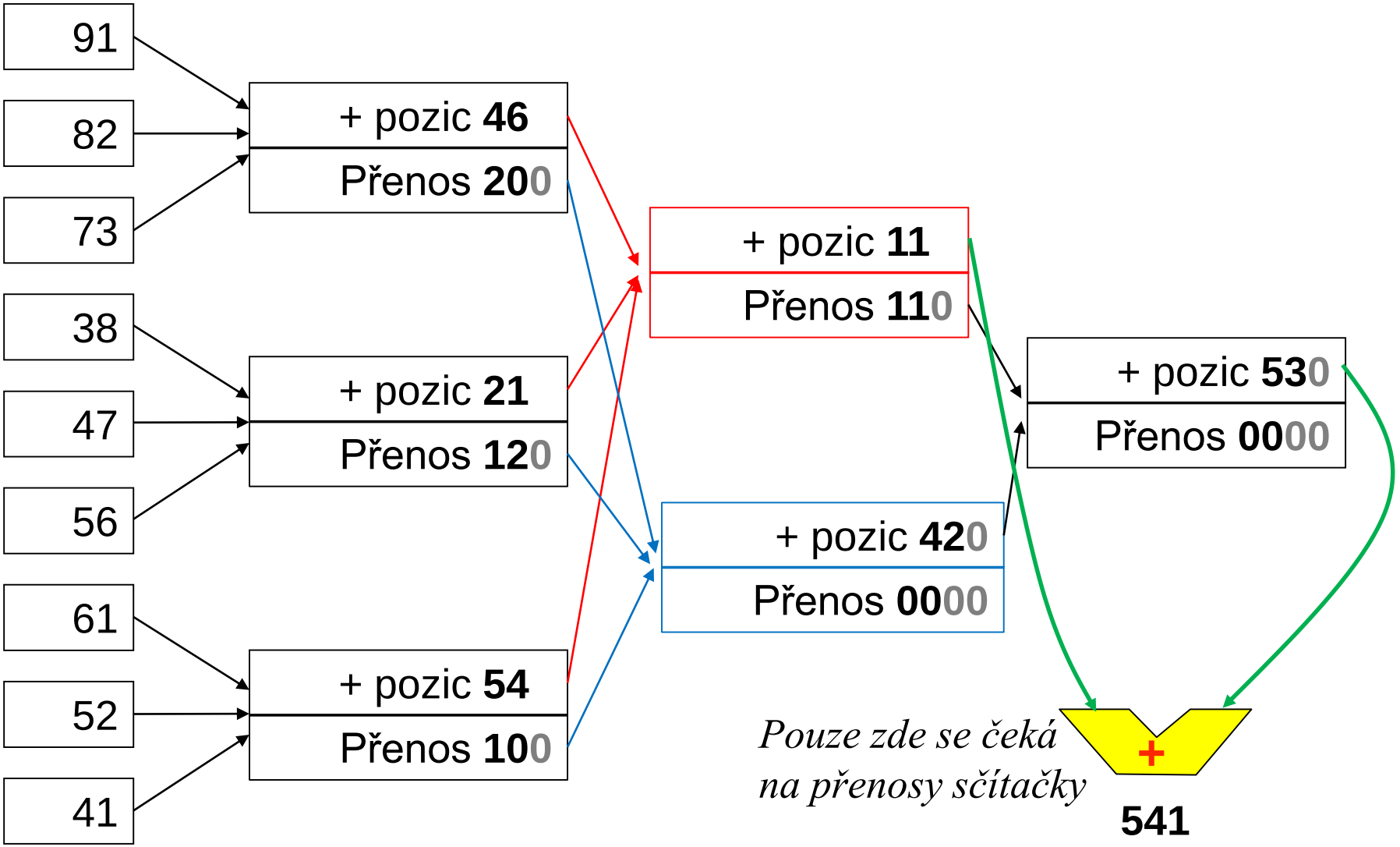
Q = X .Y = P0 + P1 + ... + P7 - **potřebuje ale sčítat hodně čísel!**

Paralelní sečtení 9 dekadických čísel



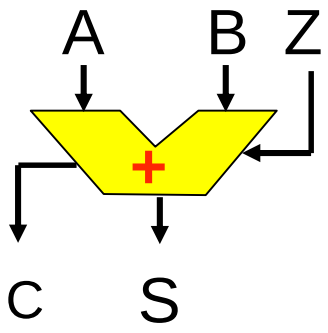
Dostáváme mezivýsledky, které vůbec nepotřebujeme, ale i tak čekáme na dokončení jejich součtu!

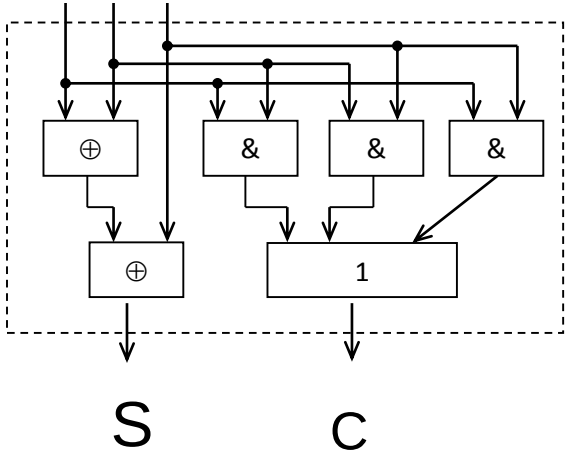
Dekadický Carry-save adder



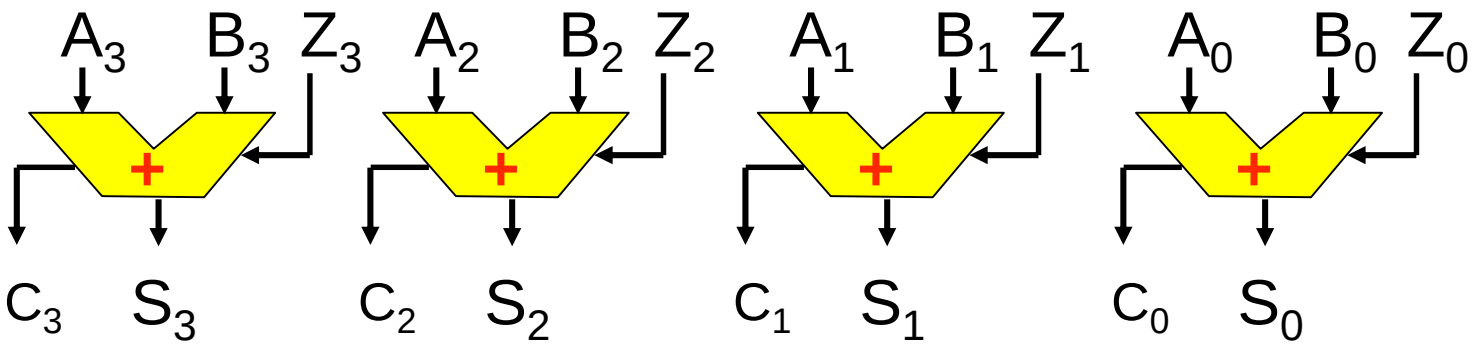
1bitová sčítačka Carry Save Adder

A	0	0	1	1	0	0	1	1
+B	0	1	0	1	0	1	0	1
Z=Carry-In	0	0	0	0	1	1	1	1
Sum	0	1	1	0	1	0	0	1
C=Cout	0	0	0	1	0	1	1	1



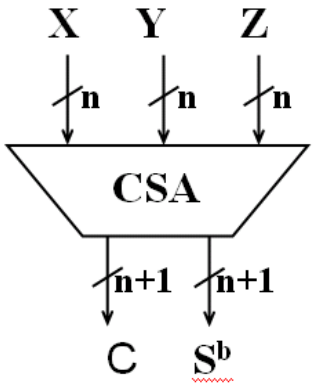


3-bitová Carry-save sčítačka



Rychlá násobička podle Walaceova stromu

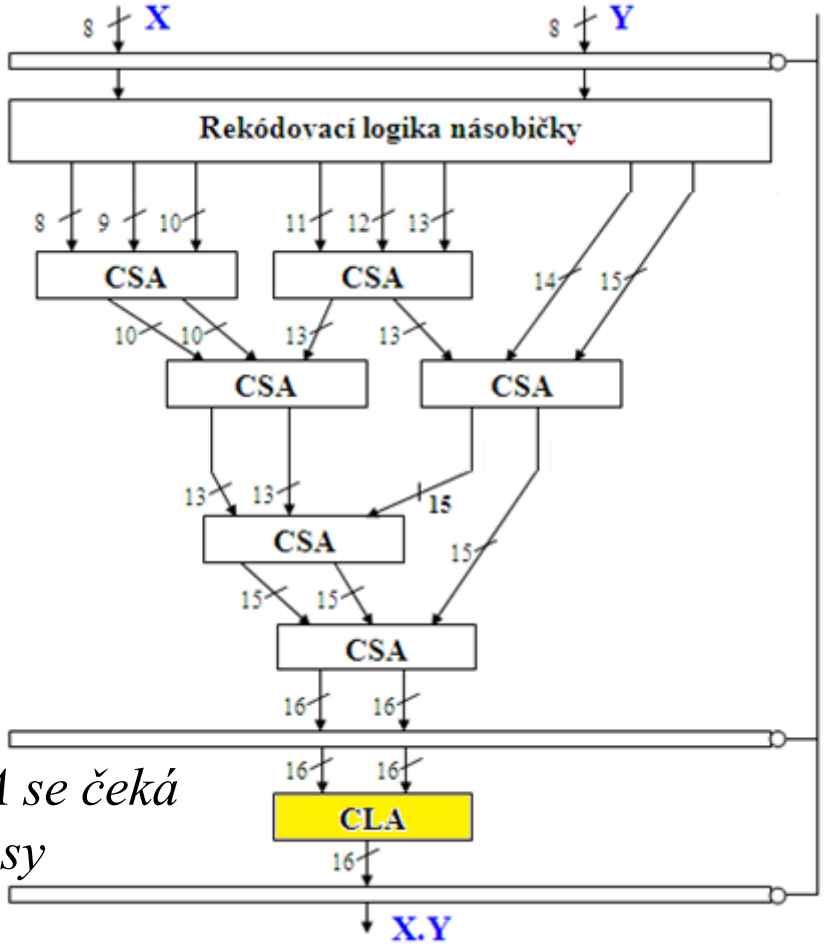
Jejím stavebním prvkem je sčítačka s uchováním přenosu CSA (Carry Save Adder)



$$S = S^b + C$$

$$S^b_i = x_i \oplus y_i \oplus z_i$$

$$C_{i+1} = x_i y_i + y_i z_i + z_i x_i$$



Až v CLA se čeká na přenosy