Principy počítačů a operačních systémů

Úvod

Zimní semestr 2011/2012

Proč jsou počítače zajímavé?

Nesmírně dynamický obor

- zrod elektronických počítačů kolem r. 1940
- o 60 let později počítače všudypřítomné
  - 3. revoluce (vedle průmyslové a zemědělské)
- nové technologie nahrazeny dříve než stačí zestárnout

Ohromný vliv na každodenní život

- automobily, mobilní telefony, lidský genom, výpočetní chemie, WWW, vyhledávače, ...
- každém řádové snížení ceny nebo zvýšení výkonnosti poskytuje nové příležitosti

Čemu vlastně říkáme počítač?

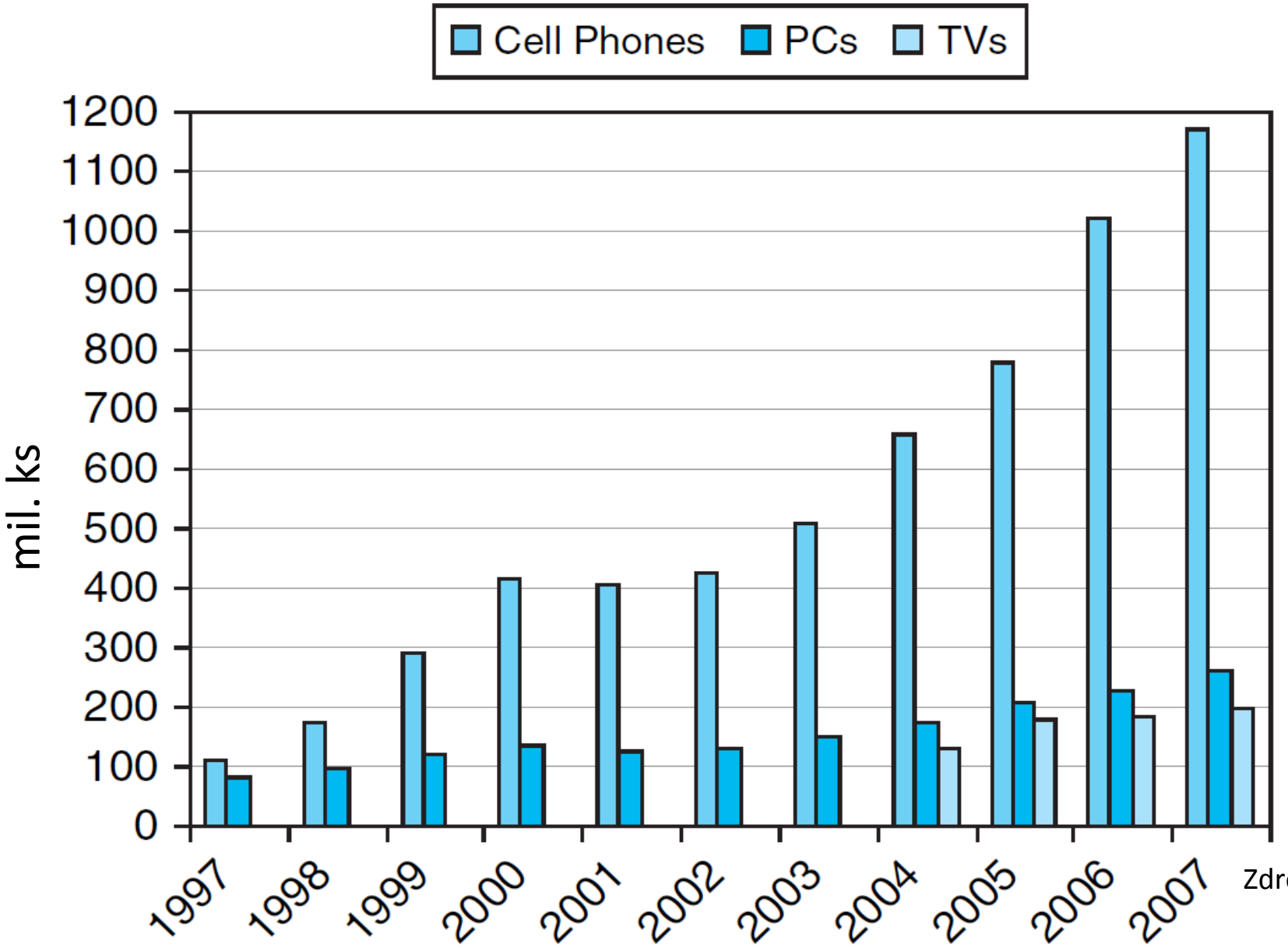
Počítač je široký pojem...

- sdílí řadu společných technologií, ale jejich uspořádání se značně liší podle požadavků a využití

Hlavní třídy počítačů

- stolní počítače
  - optimální poměr cena/výkon ⇨ tlak na vývoj
- servery, superpočítače, mainframy
  - vyšší propustnost, spolehlivost, výpočetní výkon
  - vědecko-technické výpočty vs. velké množství požadavků
- vestavěné (*embedded*) počítače
  - omezené zdroje (paměť, výkon, energie, cena
  - často další nároky, např. na fyzickou odolnost

Osobní vs. vestavěné počítače



Zdroj: P&H

Co byste se měli naučit?

Odpovědět na následující otázky...

- Jak procesory vykonávají programy napsané v high-level jazyce (C, C++, Java, ...)?
- Jaké je rozhraní mezi software and hardware a jak software říká hardwaru co má dělat?
- Jakým způsobem se komunikuje se vstupně/výstupními zařízeními?
- Co určuje výkonnost programu a jak ji může programátor ovlivnit?
- Jak ovlivňuje architektura procesoru jeho výkon?
- Proč nejde jen zvyšovat pracovní frekvenci?
- Jaké jsou důvody a důsledky přechodu od jednojádrových k vícejádrovým procesorům?

Proč jsou tyto odpovědi důležité?

Pochopení otvírá cestu k ...

- zlepšování výkonnosti programů na moderních procesorech a lepšímu využití dostupných prostředků počítače
- schopnosti porovnávat vlastnosti a výkon různých počítačů a posoudit jejich vhodnost pro danou úlohu

... systematicky a nikoliv metodou pokus/omyl

Co a jak ovliňuje výkon programu?

Co?	Jak?
Algoritmus	Počet příkazů ve zdrojovém textu a počet V/V operací
Programovací jazyk, překladač, architektura	Počet strojových instrukcí na každý příkaz ve zdrojovém textu
Procesor a paměť	Rychlost provádění instrukcí
V/V subsystém (hardware + operační systém)	Počet a rychlost provádění V/V operací

Co nevidíte na svém (oblíbeném) programu?

Stovky tisíc řádků kódu...

Aplikační software

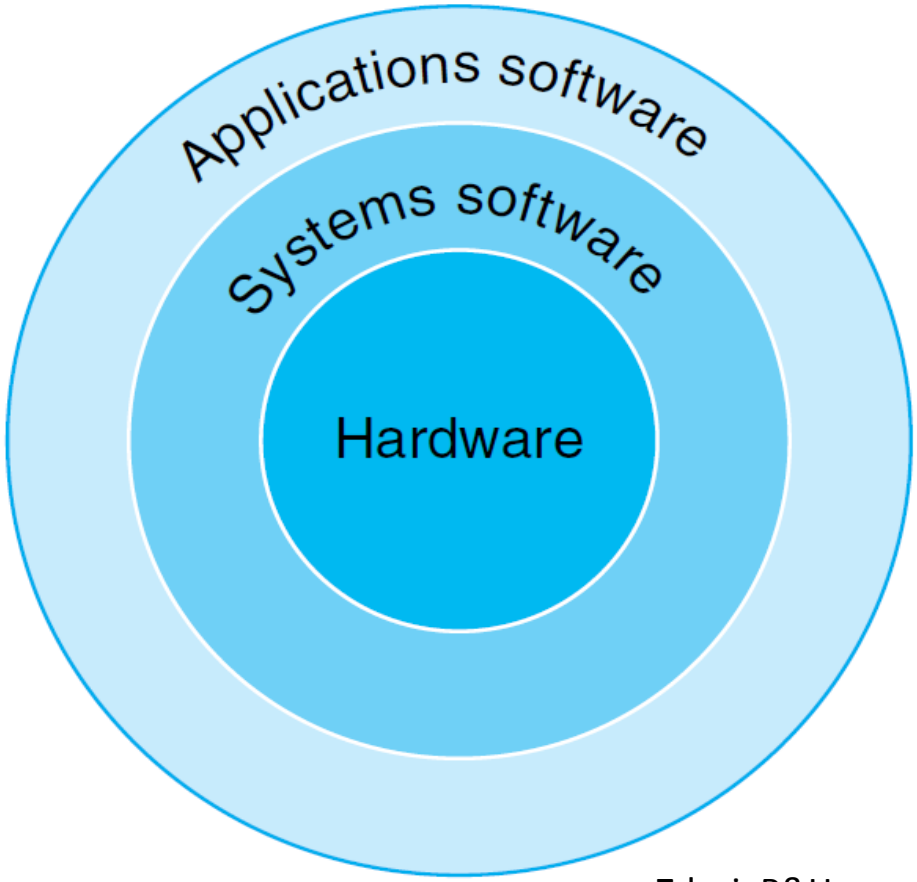
- textový procesor
- knihovny pro UI

Systémový software

- operační systém
  - vstupně/výstupní operace
  - alokace paměti a úložného prostoru
  - sdílení prostředků mezi aplikacemi

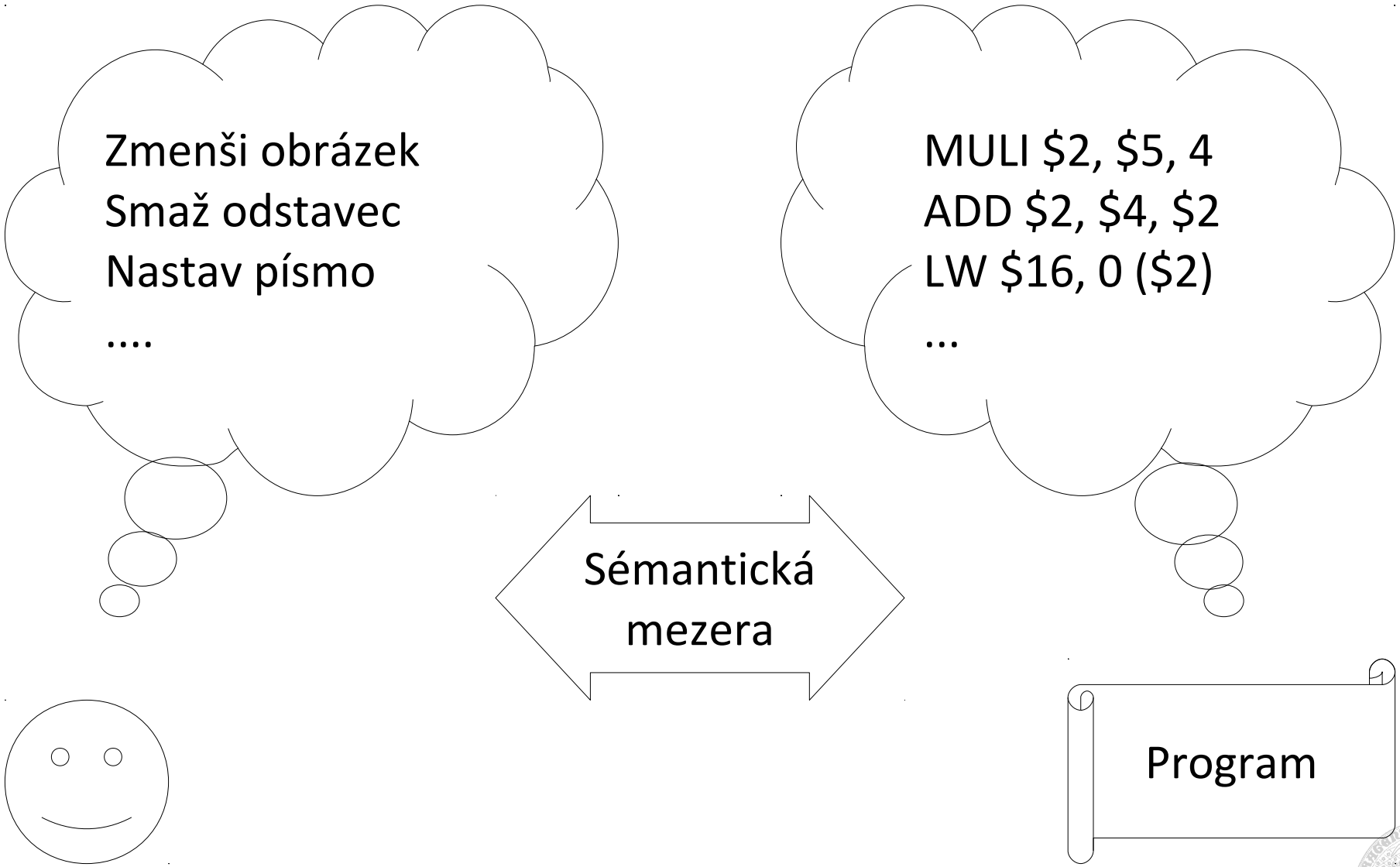
Hardware

- procesor, paměť, zařízení

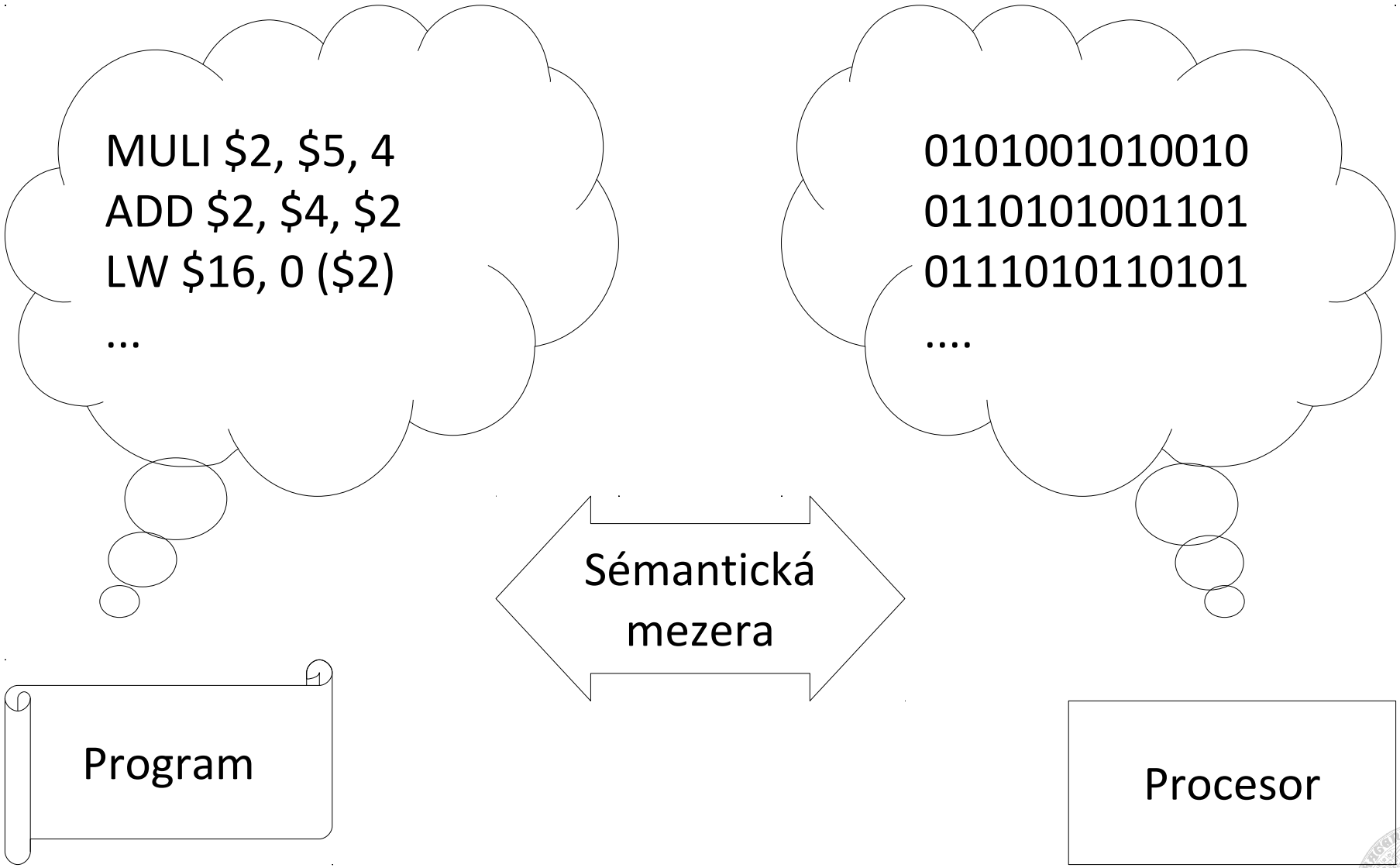


Zdroj: P&H

Od aplikace k instrukcím programu



Od instrukcí ke strojovému kódu



Jak se domluvit s procesorem?

Nutno použít správný jazyk

- slova na abecedou {0, 1}
  - 1000110010100000
- odpovídají příkazům – instrukcím
  - sečti A a B
- v symbolickém zápisu
  - add A, B
- ve vyšším jazyce
  - fruits = fruits + oranges

Co s těmi všemi jazyky?

Překládat z jednoho do druhého

- zmenšení sémantické mezery
- jazyk vyšší úrovně ⇒ vyšší produktivita
  - doménově-specifické jazyky

Překladač

- z jazyka vyšší úrovně do jazyka nižší úrovně, až na úroveň symbolického zápisu instrukcí konkrétního procesoru

Assembler

- překlad symbolického zápisu instrukcí do binárního kódu vykonatelného konkrétním procesorem

Příklad: záměna k a k+1 prvku pole

Zdrojový text ve vyšším jazyce

- vstup překladače

```
void swap (int array [], int k) {
  int old = array [k];
  array [k] = array [k+1];
  array [k+1] = old;
}
```

Výstup překladače

- symbolický zápis instrukcí pro procesor

Příklad: záměna k a k+1 prvku pole

Symbolický zápis pro MIPS

- vstup pro assembler, výstupem je strojový kód

```
swap:
  sll  $a1, $a1, 2
  addu $a1, $a1, $a0
  lw   $v0, 0 ($a1)
  lw   $v1, 4 ($a1)
  sw   $v1, 0 ($a1)
  sw   $v0, 4 ($a1)
  jr   $ra
```

Příklad: záměna k a k+1 prvku pole

Symbolický zápis pro x86_64

- vstup pro assembler, výstupem je strojový kód

```
swap:
  movslq %esi, %rsi
  leaq   (%rdi, %rsi, 4), %rdx
  leaq   4 (%rdi, %rsi, 4), %rax
  movl   (%rdx), %ecx
  movl   (%rax), %esi
  movl   %esi, (%rdx)
  movl   %ecx, (%rax)
  retq
```

Příklad: záměna k a k+1 prvku pole

Zápis ve strojovém kódu pro MIPS

```
00000000000010100101000010000000
00000000101001000010100000100001
10001100101000100000000000000000
10001100101000110000000000000100
10101100101000100000000000000100
10101100101000110000000000000000
00000011111000000000000000001000
```

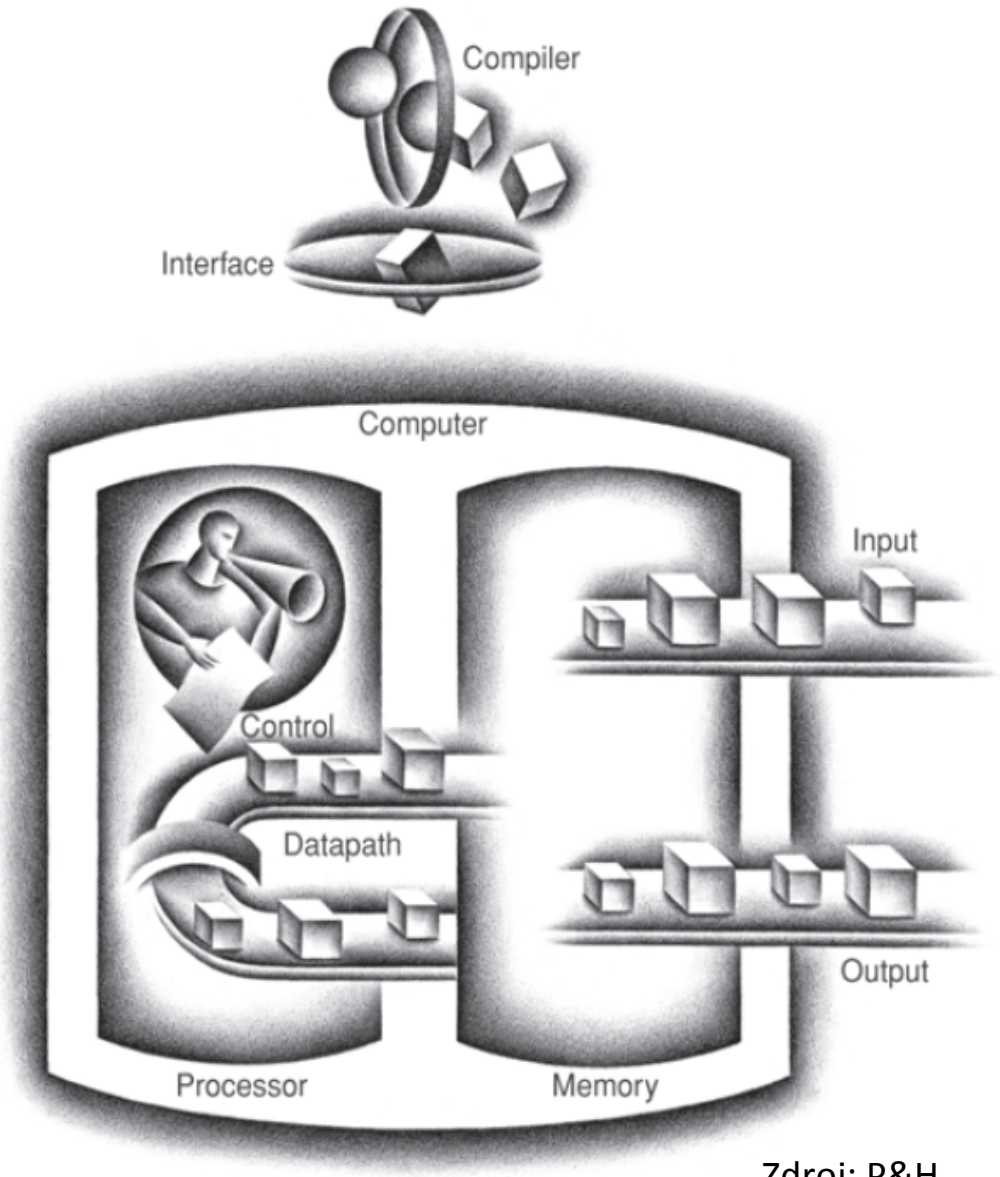
Příklad: záměna k a k+1 prvku pole

Zápis ve strojovém kódu pro x86_64

```
0100100001100111111110110
0100100010001101000101001010110111
0100100010001101010001001011011100000100
1000101100001010
1000101101110000
1000100101110010
1000100100001000
11000111
```

Co je pod kapotou?

Základní organizace počítače



Zdroj: P&H

Počítač

- vstup
- výstup
- paměť
- procesor
  - datová cesta
  - řízení

Nezávisí na technologii

- pasuje na současné i minulé počítače

Interakce s okolím

Vstupní zařízení

- klávesnice, myš, tablet, snímač otisků, joystick

Výstupní zařízení

- CRT monitor, LCD panel, grafická karta, tiskárna

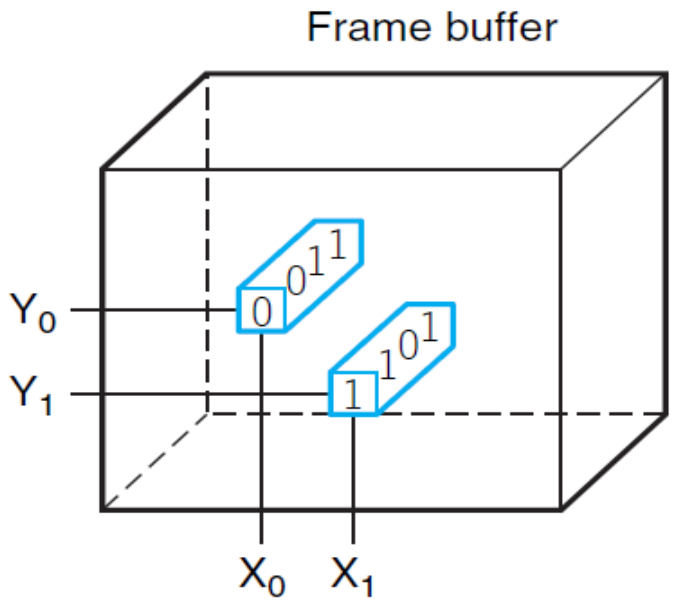
Vstupně/výstupní zařízení

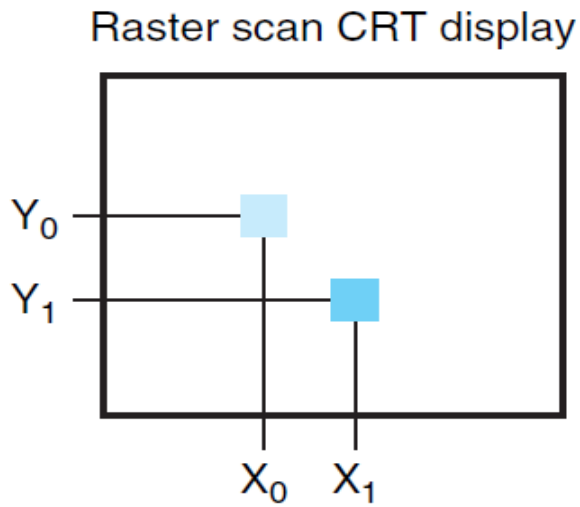
- síťová karta, pevný disk, zvuková karta, kamera, volant + pedály se zpětnou vazbou (force-feedback), ...

Grafický výstup na obrazovku

Paměť na grafické kartě (framebuffer)

- každé místo v paměti (nebo shluk několika míst) odpovídá jednomu obrazovému bodu
- obsah místa v paměti reprezentuje barvu
- velikost místa určuje barevné rozlišení





Zdroj: P&H

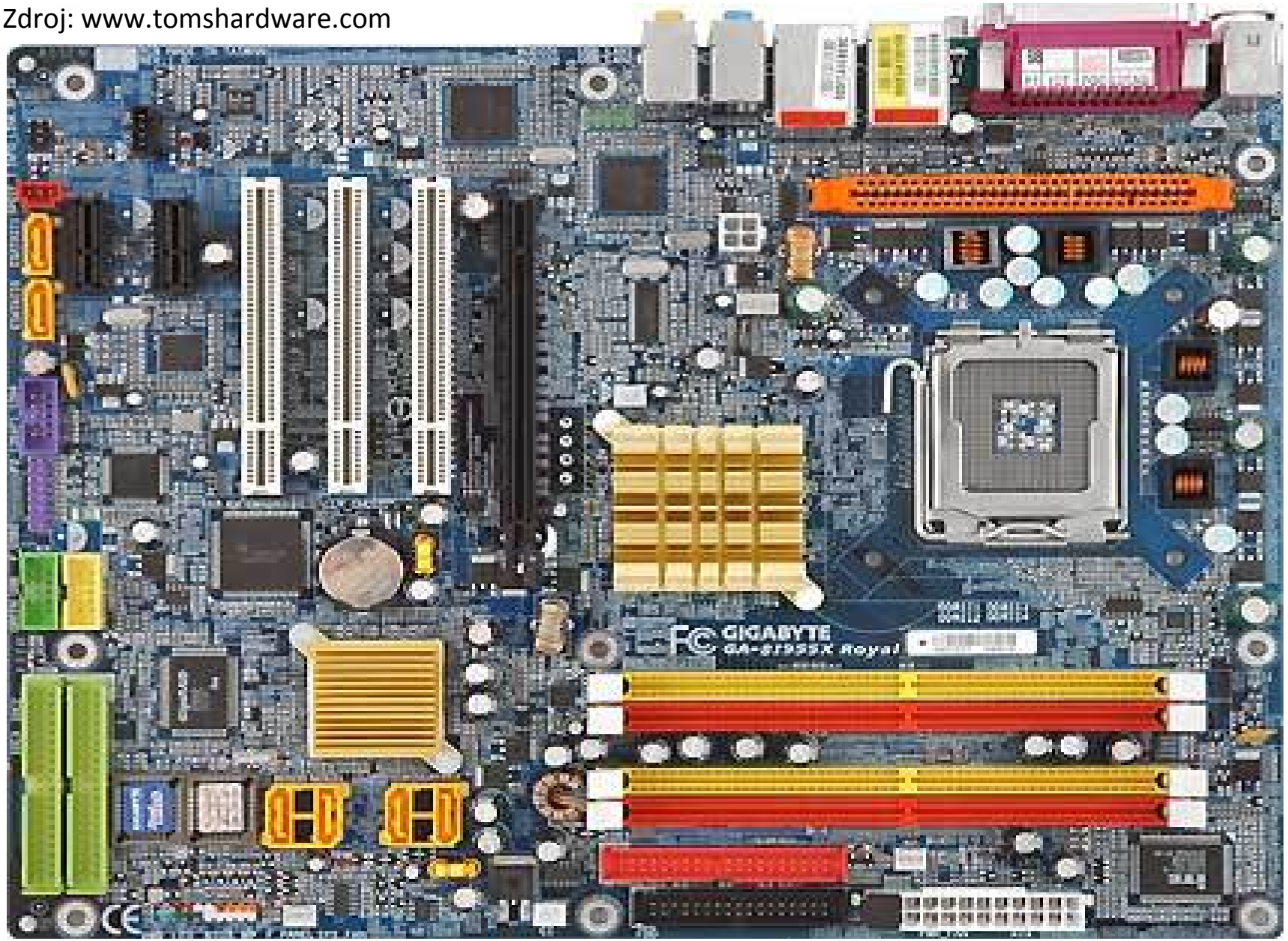
Copak je v té krabici?

Zdroj: www.soundonsound.com

- Zdroj
- Základní deska
  - Paměť
  - Procesor
  - Síťové rozhraní
- Pevný disk
- CD/DVD/BD mechanika
- Rozšiřující karty
  - Grafická karta
  - Zvuková karta

Základní deska

Zdroj: www.tomshardware.com



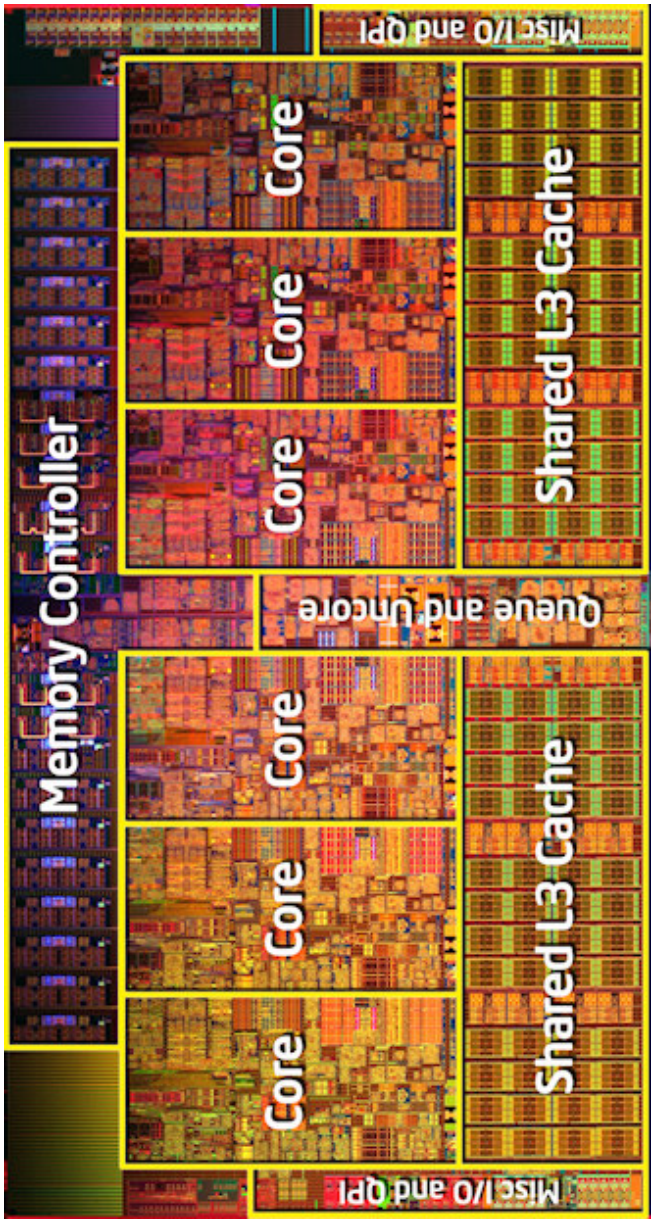
Hlavní/primární/operační paměť (dočasná)

DRAM

- dynamic random access memory
- konstantní doba přístupu na libovolné místo v paměti v řádu ns
- obsahuje běžící programy a jejich data
- s touto pamětí pracuje procesor
- kapacita 1-10 GiB
- při odpojení napájení jsou data ztracena, tj. neslouží jako permanentní úložiště

Zdroj: www.slashgear.com

Procesor



Zdroj: www.intel.com

Intel Core i7-980X

- 6 jader, 12MB L3 cache
- 32nm proces, 1170 mil. tranzistorů, plocha 248mm^2
- taktovací frekvence 3.33GHz

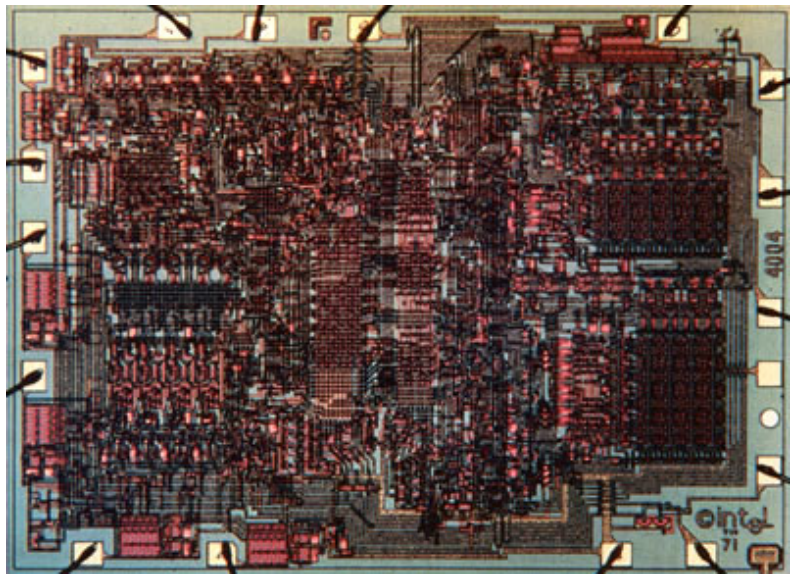
Součástí procesoru

- datová cesta (operace s daty)
- řízení (datové cesty)
- cache (SRAM)
  - static random access memory
  - rychlejší, ale menší (řádově desítky MiB) než DRAM

Procesor (pro srovnání)

Intel 4004

- první mikroprocesor, 4-bitový
- 10µm proces, 2300 tranzistorů
- taktovací frekvence 108kHz



Zdroj: www.intel.com

Technologie pro výrobu procesorů a pamětí

Tranzistor

- základní stavební prvek
- použit jako diskrétní prvek : elektronicky řízený spínač
  - nikoliv jako analogový prvek: zesilovač

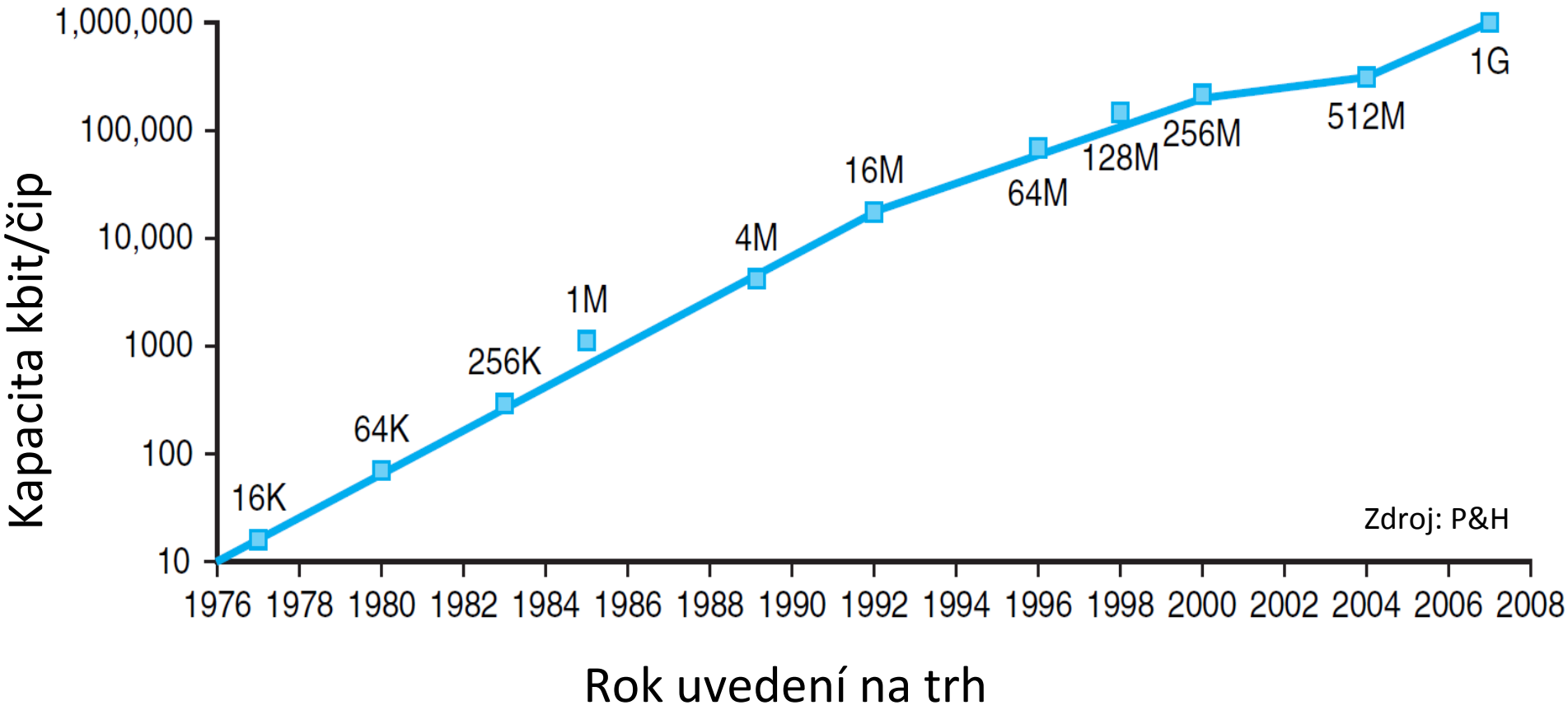
Integrovaný obvod

- kombinace desítek až stovek tranzistorů na čipu
- lepší technologie ⇨ menší rozměry ⇨ vyšší stupeň integrace
  - malá, střední, vysoká, velmi vysoká, ultra vysoká
- důsledkem je vyšší rychlost procesorů a vyšší kapacita pamětí

Relativní výkon s ohledem na technologii

Rok	Technologie	Relativní výkon/jednotková cena
1951	Elektronka	1
1965	Tranzistor	35
1975	Integrovaný obvod	900
1995	IO velmi vysoké integrace	2400000
2005	IO ultra vysoké integrace	6200000000

Růst kapacity DRAM čipů



Moorův "zákon" / Moore's Law

Gordon Moore, 60. léta 20. století

- zakladatel společnosti Intel
- předpověď: množství tranzistorů integrovaných na jednom čipu se zdvojnásobí každých 18-24 měsíců
  - zmenšování velikosti vedoucí k nárůstu rychlosti
- exponenciální růst posledních 40 let!
  - udržení vyžadovalo **velmi netriviální** technický pokrok

Často "platí" i pro jiné oblasti

- disková kapacita, přenosové pásmo

Důsledky?

Vedlejší/sekundární/permanentní paměť

Zdroj:neznámý

Pevný disk

- data zapisována na magnetické médium
- uchovává data i po odpojení napájení
- obsahuje datové soubory a spustitelné programy
- s pevným diskem pracuje OS, nikoliv CPU
- průměrná doba přístupu v řádu jednotek ms
- kapacita v jednotkách TB

Od spuštění počítače k běžící aplikaci

Co se stane, když zapnu počítač?

- procesor začne vykonávat kód (program) BIOSu (Basic Input/Output System)
- BIOS zjistí, jaký hardware je instalován, provede inicializaci grafické karty a z uživatelem definovaného disku načte a spustí boot sektor
- boot sektor obsahuje kód, který s pomocí služeb BIOSu přečte z disku a spustí zavaděč operačního systému
- zavaděč operačního systému přečte z disku kód operačního systému a spustí ho
- operační systém nastartuje systémové služby a uživatelské rozhraní

Výkonnost počítače

K čemu je dobré znát výkonnost?

Porovnávání různých počítačů

- obzvlášť když se jich objevuje tolik a tak často
- důležité pro kupující ⇨ důležité pro návrháře a výrobce
- vyhrává levnější a/nebo lepší produkt (paměť, procesor)
  - alespoň v oblasti osobních počítačů – velmi tvrdá konkurence

K ověření přínosu změn v architektuře

- co nefunguje, nemá šanci přežít

Co je vlastně výkonnost?

- co znamená, že počítač A je "lepší" než počítač B?
- můžeme říct, že nákladní vůz je "lepší" než sportovní?
- záleží na tom co porovnáváme a co potřebujeme

Základní měřítka

Doba odezvy/běhu (*execution/response time*)

- jak dlouho počítač vykonává konkrétní úlohu
- zajímavé pro konkrétního uživatele počítače

Propustnost (*throughput*)

- kolik práce počítač vykoná za jednotku času
- zajímavé pro správce serveru nebo datového centra

Jak se změní odezva a propustnost...

- pokud vyměníme procesor za rychlejší?
- pokud přidáme do systému další procesory?

Výkonnost určená dobou odezvy

Výkonnost systému X

$$Performance_X = \frac{1}{Execution_X}$$

- maximalizace výkonu ⇔ minimalizace odezvy
- pokud ***Performance*****$_X$** > ***Perfomance*****$_Y$**
  pak zjevně ***Execution*****$_X$** < ***Execution*****$_Y$**
- pokud X je ***n***-krát "rychlejší" (výkonnější) než Y, pak

$$n = \frac{Performance_X}{Performance_Y} = \frac{Execution_Y}{Execution_X}$$

Kolikrát je A rychlejší než B...

- pokud A provede program za 10 sekund a B stejný program za 15 sekund?

Jak měřit výkonnost z pohledu uživatele?

Celkový čas běhu úlohy

- *wall clock time, response time, elapsed time*
- zahrnuje vše, včetně přístupu na disk a režie OS
  - co když je počítač sdílen více uživateli?
- odráží **výkonnost systému** jako celku

Procesorový čas

- *CPU execution time, CPU time*
- čas, po který procesor program skutečně vykonával
- nezahrnuje čekání na V/V zařízení a čas kdy neběžel
- zahrnuje čas strávený uvnitř OS – režii OS
  - **uživatelský** a **systémový** procesorový čas
- odráží **výkonnost procesoru**

Jak měřit výkonnost z pohledu návrháře?

Rychlost provádění základních operací

- všechny události řízeny hodinovým signálem
- frekvence hodin (*clock rate*), např. 3 GHz
  - taktovací frekvence, frekvence hodinového signálu, ...
- délka hodinového cyklu (*clock cycle*), např. 250 ps
  - doba periody hodinového signálu

Jak souvisí metriky uživatelů a návrhářů HW?

$$\begin{matrix}\textit{CPU execution time} \\ \textit{for a program}\end{matrix} = \begin{matrix}\textit{CPU clock cycles} \\ \textit{for a program}\end{matrix} \times \textit{Clock cycle time}$$

$$\begin{matrix}\textit{CPU execution time} \\ \textit{for a program}\end{matrix} = \frac{\textit{CPU clock cycles for a program}}{\textit{Clock rate}}$$

- jak může návrhář zlepšit výkon?

Příklad: zlepšování výkonnosti

Jakou taktovací frekvenci potřebujeme, pokud ...

- program běží 10s na počítači A s hod. frekvencí 2 GHz
- cílem je navrhnout počítač B, kde program poběží 6s
  - technologie umožňuje výrazně zvýšit taktovací frekvenci...
  - ... ale pro vykonání programu je potřeba o 20% více cyklů

$$CPU\ time_A = \frac{CPU\ clock\ cycles_A}{Clock\ rate_A} \qquad 10 = \frac{CPU\ clock\ cycles_A}{2 \times 10^9}[s]$$

$$CPU\ clock\ cycles_A = 10 \times 2 \times 10^9 = 20 \times 10^9$$

$$CPU\ time_B = \frac{1.2 \times CPU\ clock\ cycles_A}{Clock\ rate_B} \qquad 6 = \frac{1.2 \times 20 \times 10^9}{Clock\ rate_B}[s]$$

$$Clock\ rate_B = \frac{1.2 \times 20 \times 10^9}{6} = 0.2 \times 20 \times 10^9 = 4 \times 10^9 = 4\,GHz$$

Jak do toho zapadají instrukce?

Zatím o nich nepadla zmínka, ale...

- překladač zcela jistě vygeneroval instrukce k vykonání
- počítač musel instrukce vykonat aby program běžel
- doba běhu musí záviset na počtu instrukcí v programu

Průměrný počet taktů na vykonání 1 instrukce

- *clock cycles per instruction (CPI)*
- vztaženo k programu nebo jeho části
  - program se skládá z instrukcí trvajících různou dobu

$$CPU\ clock\ cycles = \begin{matrix} Instructions \\ for\ a\ program \end{matrix} \times \begin{matrix} Average\ clock\ cycles \\ per\ instruction \end{matrix}$$

- umožňuje srovnávat různé implementace architektury
  - při zachování počtu instrukcí na program

Příklad: různé implementace architektury

Který počítač je rychlejší a o kolik...

- počítač A pracuje s délkou cyklu 250ps a pro nějaký program P má CPI 2.0
- počítač B pracuje s délkou cyklu 500ps a pro stejný program P má CPI 1.2
- oba počítače implementují stejnou architekturu ⇒ oba počítače vykonají stejný počet instrukcí I

$$CPU\ clock\ cycles_A = I \times 2.0 \qquad CPU\ clock\ cycles_B = I \times 1.2$$

$$CPU\ time_A = CPU\ clock\ cycles_A \times Clock\ cycle\ time_A = I \times 2.0 \times 250 = 500 \times I\ [ps]$$
$$CPU\ time_B = CPU\ clock\ cycles_B \times Clock\ cycle\ time_B = I \times 1.2 \times 500 = 600 \times I\ [ps]$$

- počítač A je zjevně rychlejší... jak moc?

$$\frac{CPU\ performance_A}{CPU\ performance_B} = \frac{Execution\ time_B}{Execution\ time_A} = \frac{600 \times I}{500 \times I} = 1.2$$

Základní vztah pro výkonnost procesoru

Zápis pomocí počtu instrukcí, CPI a délky cyklu

$$CPU\ time = Instruction\ count \times CPI \times Clock\ cycle\ time$$

$$CPU\ time = \frac{Instruction\ count \times CPI}{Clock\ rate}$$

3 různé faktory ovlivňující výkonnost

- srovnání různých implementací stejné architektury
- zhodnocení alternativního návrhu
  - pokud známe jeho vliv na uvedené parametry

Příklad: porovnání částí kódu

Kterou posloupnost instrukcí použít?

Posloupnost	Počet instrukcí různého typu			Počet instrukcí
	A	B	C	
P1	2	1	2	5
P2	4	1	1	6

	CPI pro jednotlivé typy instrukcí		
	A	B	C
CPI	1	2	3

$$CPU\ clock\ cycles = \sum_{i=1}^{n} CPI_i \times C_i$$

$$CPI = \frac{CPU\ clock\ cycles}{Instruction\ count}$$

$$CPU\ clock\ cycles_{P1} = (2\times 1)+(1\times 2)+(2\times 3) = 10$$

$$CPI_{P1} = \frac{10}{5} = 2.0$$

$$CPU\ clock\ cycles_{P2} = (4\times 1)+(1\times 2)+(1\times 3) = 9$$

$$CPI_{P2} = \frac{9}{6} = 1.5$$

Na čem závisí výkonnost procesoru?

Výkonnost závisí na řadě faktorů

- počet instrukcí, průměrný počet taktů na instrukci (CPI), délka hodinového cyklu (taktovací frekvence)
- **žádný z faktorů sám o sobě nevypovídá o výkonnosti**
  - snížení počtu instrukcí často vede na architekturu s nižší taktovací frekvencí nebo vyšším CPI
  - CPI závisí na ***instrukčním mixu*** (četnost a typ prováděných instrukcí), proto kód s nejmenším počtem instrukcí nemusí být nejrychlejší

Jediným úplným a spolehlivým měřítkem je **čas**

- umožňuje porovnávat různé počítače
  - stačí porovnávat faktory, které se liší

Na čem závisí výkonnost programu?

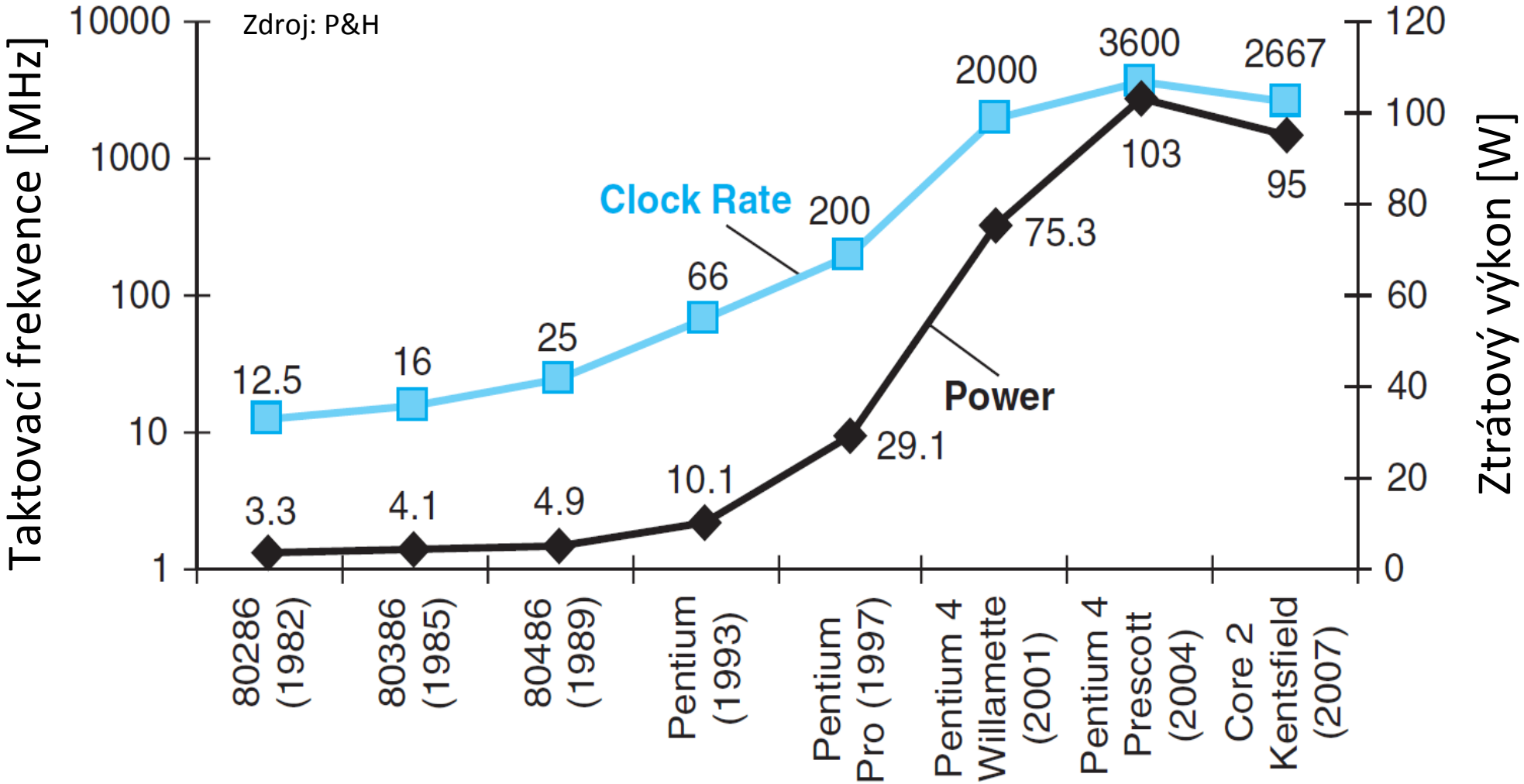
Komponenta	Co ovlivňuje?	Jak?
Algoritmus	• počet instrukcí • potenciálně CPI	• počet příkazů ve zdrojovém kódu • datové typy (integer vs. FP)
Programovací jazyk	• počet instrukcí • CPI	• typ příkazů ve zdrojovém kódu • abstraktní datové typy (nepřímá volání)
Překladač	• počet instrukcí • CPI	• způsob překladu příkazů na instrukce
Architektura	• počet instrukcí • taktovací frekvenci • CPI	• instrukce potřebné pro vyjádření funkce • taktovací frekvence daná technologií • doba provádění instrukcí v taktech

Jak rychle poběží program...

- po přeložení překladačem A běží program 15s
- překladač B použije 60% instrukcí, ale CPI je o 10% vyšší

Konec zlatých časů

Taktovací frekvence a ztrátový výkon procesorů Intel



Ztrátový výkon v CMOS obvodech

CMOS (Complementary Metal Oxide Semiconductor)

- dominantní technologie pro výrobu IO
- minimální spotřeba v klidovém stavu
- dynamický ztrátový výkon
  - kapacitní zátěž (vodiče, tranzistory, zátěž na výstupu)
  - provozní napětí (souvisí s rychlostí přepínání)
  - frekvence přepínání (odvozená od taktovací frekvence)

$$P \approx C \times U^2 \times f$$

1000x nárůst frekvence vs. 30x ztrátového výkonu

- pokles napětí o 15% s každou generací (5V → 1V)

Příklad: porovnání ztrátového výkonu

Jak se projeví nový návrh procesoru na ztrátách?

- starý procesor A, nový procesor B
- kapacitní zátěž B je pouze 85% zátěže A
- provozní napětí B je o 15% nižší než u A, čemuž odpovídá také o 15% nižší taktovací frekvence

$$\frac{P_B}{P_A} = \frac{(C_A \times 0.85) \times (U_A \times 0.85)^2 \times (f_A \times 0.85)}{C_A \times U_A{}^2 \times f_A} = 0.85^4 = 0.52$$

Energetická zeď

Napětí není možné pořád snižovat

- začínají se projevovat statické ztráty
- cca 40% spotřeby v roce 2008

Chlazení už nelze jednoduše zlepšovat

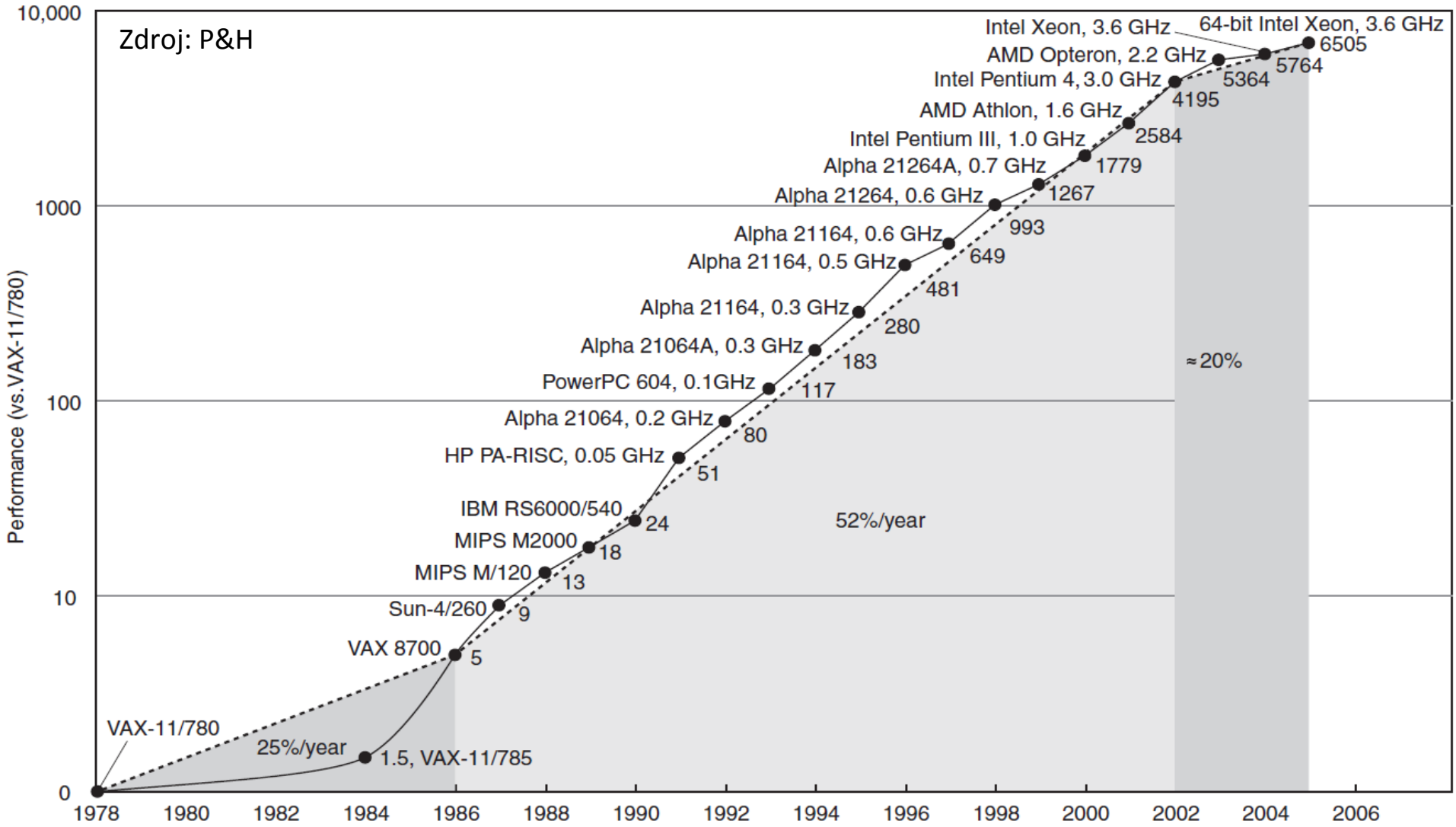
- v daném taktu nepoužívané části čipu se vypínají
- vodní a jiné chlazení není praktické (stolní počítače)

Nutno nalézt jinou cestu ke zvýšení výkonu

- odlišnou od té, kterou se vývoj ubíral posledních 30 let

Přechod od jednoprocesorových k víceprocesorovým strojům

Nárůst výkonu osobních počítačů



Co s tím?

Vícejádrové (multicore) procesory

- výrobcům se zatím daří zlepšovat technologii ⇒ pojďme s každou generací (~ každé 2 roky) zdvojnásobit počet procesorů/jader na čipu

Jaký to má vliv na výkon systému?

- primárně roste propustnost (througput)

Jaký je důsledek pro programátory?

- programy se (samy) nezrychlí díky novým technologiím
- programy musí začít využívat více procesorů/jader
  - každé dva roky jich bude nutno využít více

Proč tolik povyku?

Fundamentální změna v rozhraní HW/SW

- paralelizmus byl vždy důležitý, ale dařilo se ho skrývat
  - paralelizmus na úrovni instrukcí (*instruction-level parallelism*)
  - zřetězené zpracování instrukcí a další techniky (později)
  - překladač i programátor pracují se sekvenčním strojem
- najednou musí být programy explicitně paralelní
  - programátor musí o paralelizmu vědět

Celý IT průmysl na takovou změnu vsadil...

- doposud slepá větev vývoje architektur
  - společnosti spoléhající na změnu programovacího paradigmatu nepřežily

Proč je paralelní programování těžké?

Programování zaměřené na výkon

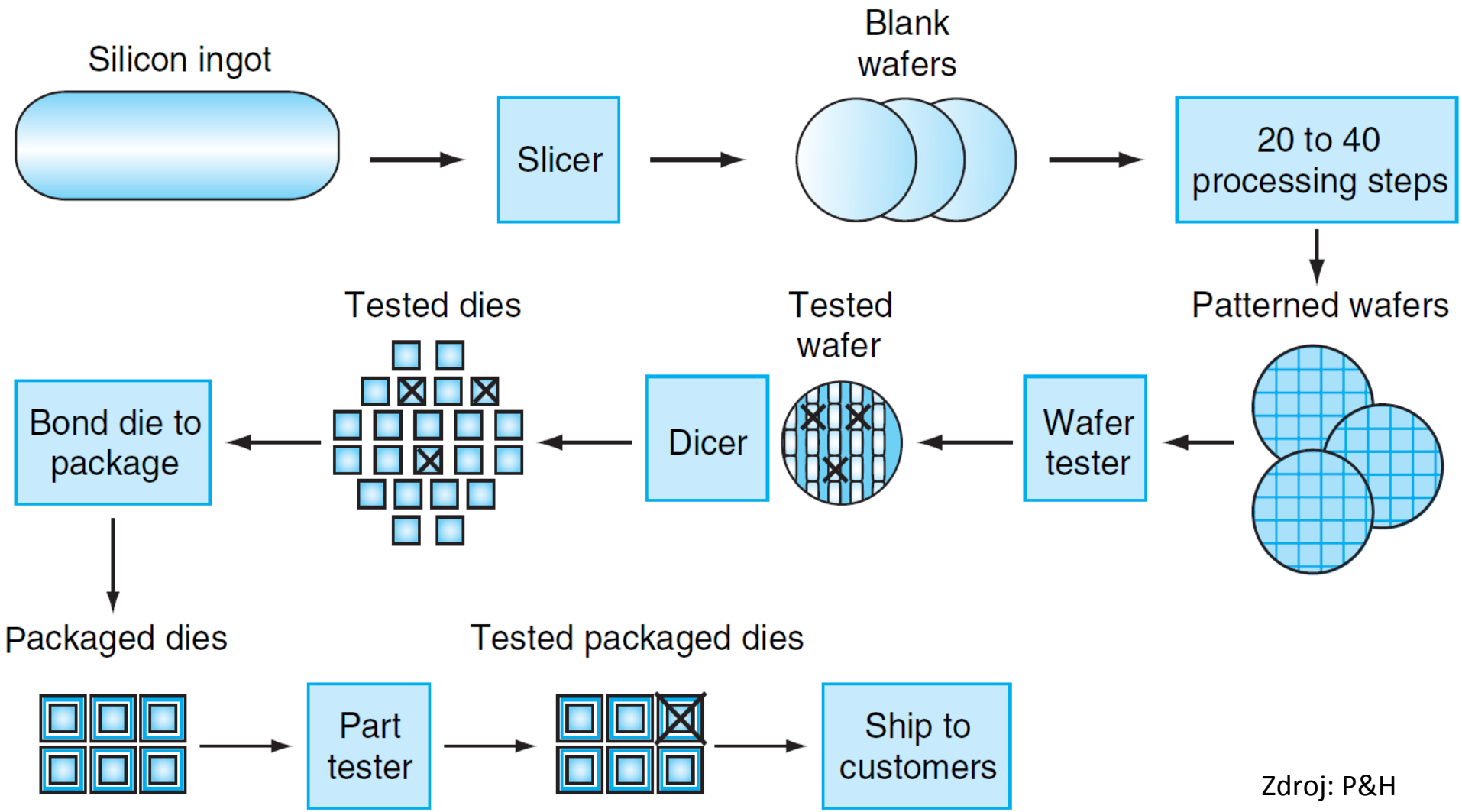
- řádově obtížnější, vedle funkce důležitá i rychlost
  - v opačné případě stačí napsat sekvenční program
- práci je nutno rozdělit mezi procesory
- plánování a koordinace nesmí mít vysokou režii

Analogie: 1 reportér = článek/2h, 8 reportérů = ???

- plánování (*scheduling*)
- vyvažování zátěže (*load balancing*)
- režie na komunikaci a synchronizaci (*communication and synchronization overhead*)

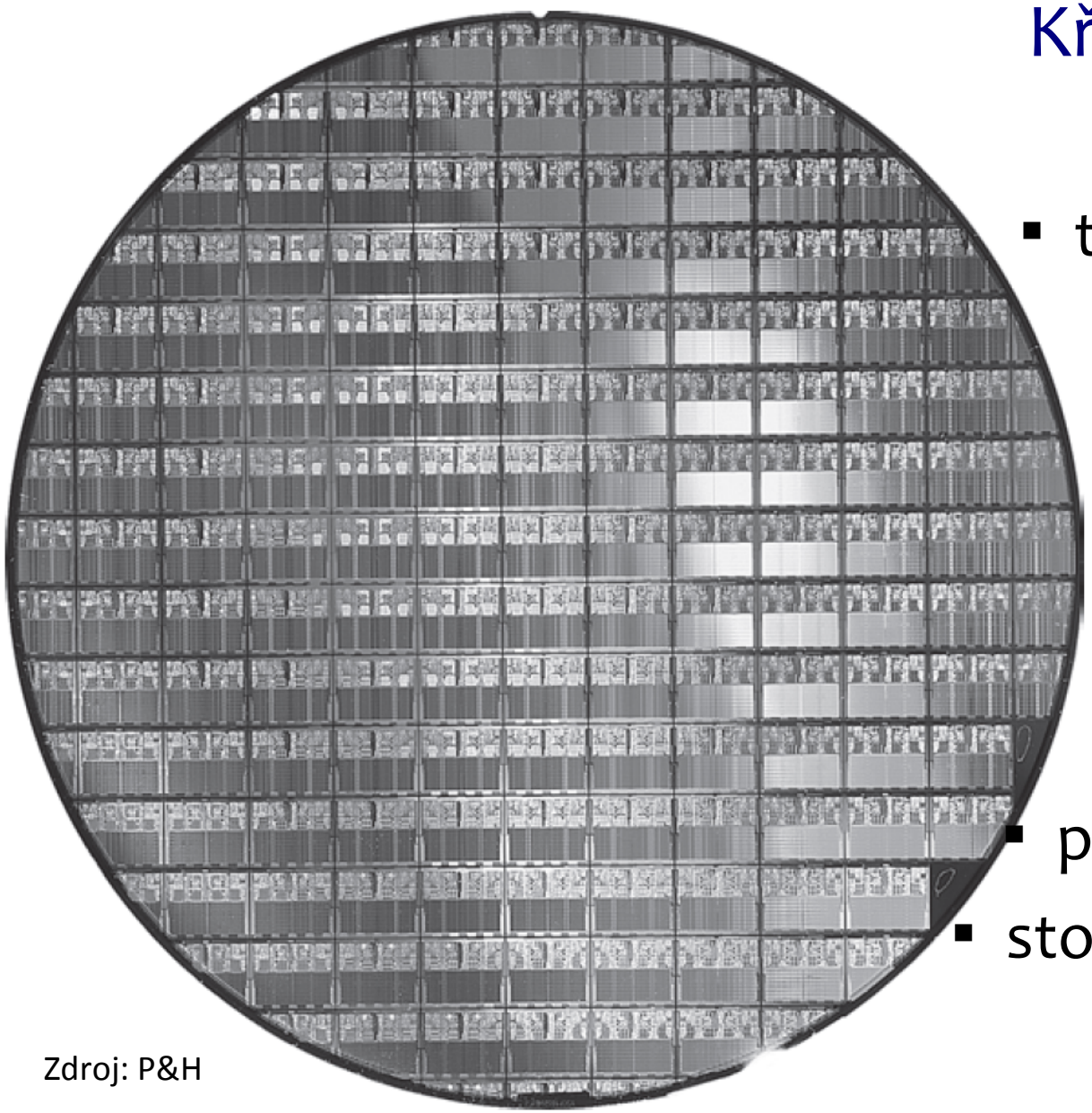
Výroba a testování výkonnosti procesoru AMD Opteron X4

Jak se vyrábí procesor?



Zdroj: P&H

Opteron X2, předchůdce X4



Zdroj: P&H

Křemíkový plátek

- průměr 300mm
- technologie 90nm
- 117 čipů při 100% výtěžnosti

Opteron X4

- ztráty 120W
- povrch čipu ~ 1 cm^2
- stovky "nožiček" jen pro napájení

Měření výkonnosti: SPEC CPU Benchmark

Workload

- typická pracovní zátěž – sada spouštěných programů
- porovnání počítačů ⇔ porovnání doby běhu programů
- časově náročné, špatně se automatizuje

Benchmark

- program specificky určený k měření výkonnosti
- sada benchmarků ≈ typická pracovní zátěž procesoru
  - doufáme, že výsledky předpoví chování při aktuální zátěži

SPEC (Standard Perf. Evaluation Corporation)

- snaha výrobců o možnost jednoduše porovnat počítače
- původně hlavně CPU, dnes i GPU, Java, mail, ...

SPEC CPU2006

Hodnocení výkonnosti procesoru

- CINT2006, 12 benchmarků, celá čísla
  - překladač C, šachy, simulace kvantového počítače, ...
- CFP2006, 17 benchmarků, plovoucí řádová čárka
  - metoda konečných prvků, molekulární dynamika, ...

SPECratio

- podíl referenční a změřené doby běhu benchmarku
  - větší číslo reprezentuje vyšší výkon

Souhrné hodnocení

- geometrický průměr SPECratio z celočíselných benchmarků

$$\sqrt[n]{\prod_{i=1}^{n} SPECratio_i}$$

SPEC CINT2006 pro AMD Opteron X4 (Barcelona)

Description	Name	Instruction Count × 10^9	CPI	Clock cycle time (seconds × 10^9)	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2,118	0.75	0.4	637	9,770	15.3
Block-sorting compression	bzip2	2,389	0.85	0.4	817	9,650	11.8
GNU C compiler	gcc	1,050	1.72	0.4	724	8,050	11.1
Combinatorial optimization	mcf	336	10.00	0.4	1,345	9,120	6.8
Go game (AI)	go	1,658	1.09	0.4	721	10,490	14.6
Search gene sequence	hmmer	2,783	0.80	0.4	890	9,330	10.5
Chess game (AI)	sjeng	2,176	0.96	0.4	837	12,100	14.5
Quantum computer simulation	libquantum	1,623	1.61	0.4	1,047	20,720	19.8
Video compression	h264avc	3,102	0.80	0.4	993	22,130	22.3
Discrete event simulation library	omnetpp	587	2.94	0.4	690	6,250	9.1
Games/path finding	astar	1,082	1.79	0.4	773	7,020	9.1
XML parsing	xalancbmk	1,058	2.70	0.4	1,143	6,900	6.0
Geometric Mean							11.7

Zdroj: P&H

Bludy a pasti

Past: planá očekávání

Očekáváme, že zlepšením části systému zvýšíme výkonnost celého systému úměrně velikost zlepšení.

- Program běží 100s, z toho 80s času spotřebují instrukce násobení. Kolikrát musím zrychlit násobení, aby byl program 5x rychlejší?

$$Execution_{Slow} = 100 = 80 + 20 \qquad Execution_{Fast} = \frac{80}{n} + 20$$

$$Execution_{Fast} = \frac{Execution_{Slow}}{5} = 20 = \frac{80}{n} + 20 \qquad 0 = \frac{80}{n}$$

- **Pouze zrychlením násobení není možné 5x zrychlit celý program.**

Amdahlův zákon

Zvýšení výkonnosti dosažitelné nějakým zlepšením je omezené mírou používání tohoto zlepšení.

- Zákon klesajících výnosů

$$Performance_{Improved} = \frac{Performance_{Affected}}{Improvement\ ratio} + Performance_{Unaffected}$$

Důsledek pro návrh (nejen) hardware

- *Make common case fast/Optimize for the common case*
- Optimalizace má největší užitek pro nejčastější případy
  - nejčastější případy bývají často výrazně jednodušší než ty výjimečné ⇒ snadno se optimalizují

Past: špatné měřítko výkonnosti

Používáme podmnožinu aspektů ze vztahu pro výkonnost procesoru jako měřítko pro srovnání.

- jeden se použít nedá, dva někdy (ale často špatně)
  - téměř všechny alternativy k času selhaly

MIPS (Million Instructions Per Second)

- rychlost vykonávání instrukcí

$$MIPS = \frac{Instruction\ count}{10^6 \times Execution\ time}$$

- intuitivní, vyšší číslo ⇒ rychlejší, ale ...

Co je špatného na MIPS?

Nebere v úvahu možnosti instrukcí

- nelze porovnávat počítače s různou instrukční sadou, protože počty instrukcí budou různé

Liší se pro různé programy na stejném počítači

- AMD Opteron X4, SPEC CINT2006: rozdíl až 13x
- jedna hodnota MIPS nereprezentuje výkon počítače

$$MIPS = \frac{Instruction\ count}{10^6 \times \frac{Instruction\ count \times CPI}{Clock\ rate}} = \frac{Clock\ rate}{CPI \times 10^6}$$

Může se měnit bez ohledu na výkonnost

- pokud nový program vykoná více instrukcí rychleji

Příklad: rychlost vs. MIPS

Porovnejte následující dva počítače

Metrika	Počítač A	Počítač B
Počet instrukcí	10×10^9	8×10^9
Taktovací frekvence	4 GHz	4 GHz
CPI	1.0	1.1

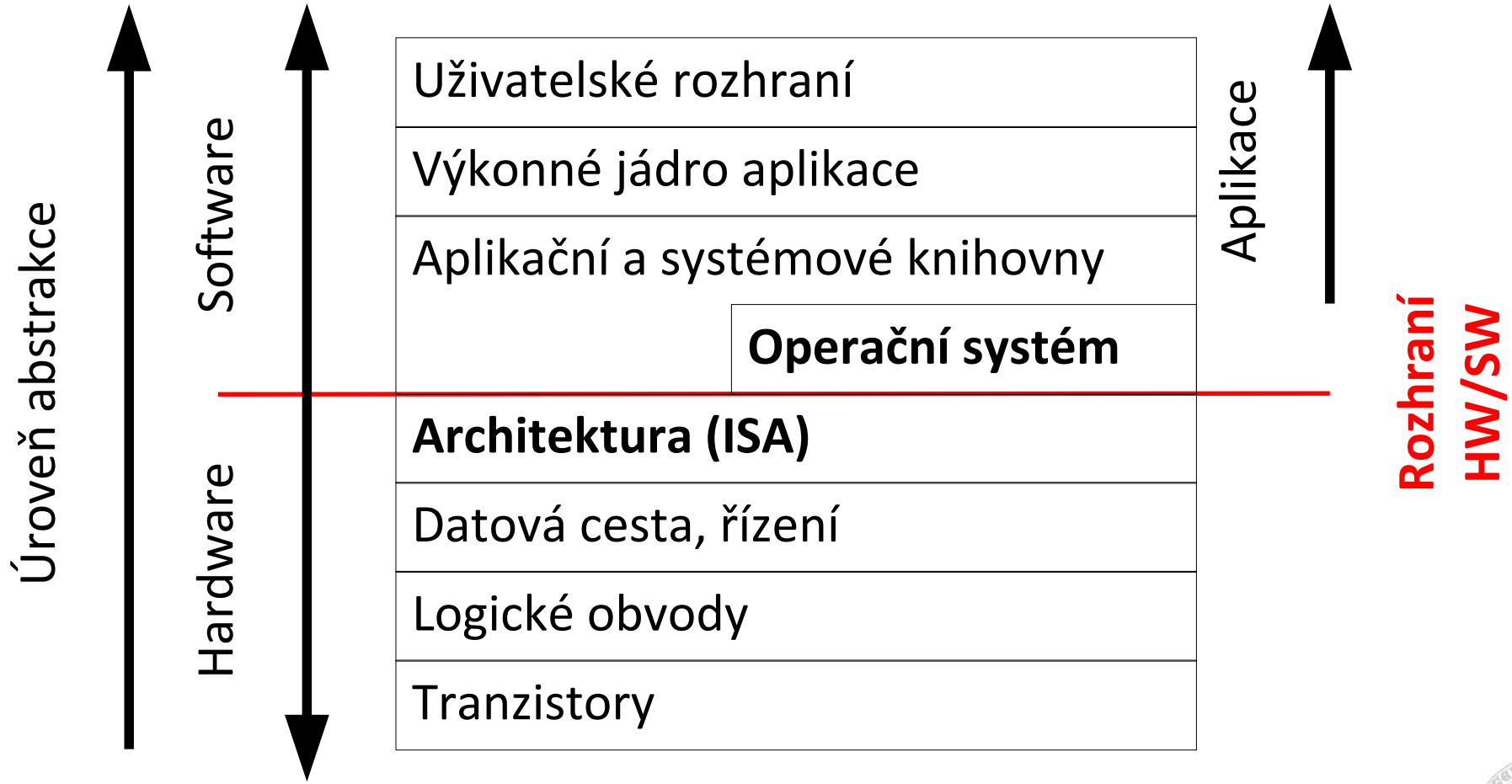
- který z nich má více MIPS?
- který z nich je rychlejší?

Shrnutí

Abstrakce a vrstvy

Nástroj návrhářů HW i SW

- umožňuje zvládat velký rozsah složitosti



Organizace a výkonnost

Základní organizace počítače nezávisí na technologii

- vstup, výstup, paměť, procesor (řadič + datová cesta)

U počítačů se vždy hodnotila cena a výkonnost

- spolehlivost, provozní náklady, škálovatelnost, ...

Čas je jediné spolehlivé měřítko výkonnosti

- alternativní měřítka fungují špatně nebo vůbec

$$\frac{Seconds}{Program} = \frac{Instructions}{Program} \times \frac{Clock\ cycles}{Instruction} \times \frac{Seconds}{Clock\ cycle}$$

Výkonnost se odvíjí technologie a architektury

- využití paralelizmu v programu
- využití lokality přístupu do paměti

Technologie

Integrované obvody a polovodiče na bázi křemíku

- rychlost změny ~ Moorův "zákon"

Energetická obálka silně ovlivňuje návrh procesoru

- s frekvencí a napětím již nelze příliš hýbat
- plocha čipu a počet tranzistorů už nejsou tak kritické

Snaha o zvýšení výkonu v rámci energetické obálky si vynutila přechod na vícejádrové procesory

- programátoři musí využívat paralelizmus explicitně