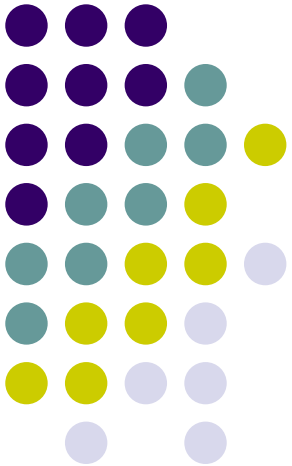
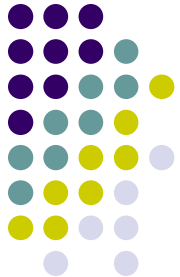
Multicore
Vícejádrové mikroprocesory

Hardware

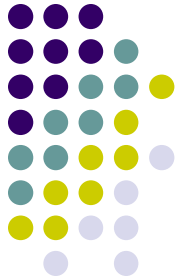


Moorův zákon



- **Gordon Moore** v roce 1965 předpověděl, že „*počet tranzistorů, které mohou být umístěny na integrovaný obvod se při zachování stejné ceny zhruba každých 18 měsíců* ***zdvojnásobí***“
- Moorův zákon **platí** dodnes, ale vývoj se dost zpomalil
- Tedy za stejnou cenu jsme zhruba každé dva roky bylo schopni umístit na stejně velký chip dvojnásobný počet tranzistorů
- V příštích letech to již úplně platit nebude (Moorův zákon přestává platit) – počet tranzistorů dál poroste, ale bude se i zvětšovat rozměr chipů
- U prvních mikroprocesorů současně s počtem tranzistorů **rostl** velmi rychle i jejich **výkon** – dvakrát složitější procesor měl také dvakrát vyšší výpočetní výkon
- S příchodem Pentií výkon procesoru roste již jen **úměrně s odmocninou** počtu tranzistorů – tzn. ke zdvojnásobení výkonu potřebujeme čtyřnásobek tranzistorů
- V současné době roste výkon procesoru s každým přidaným tranzistorem ještě **daleko pomaleji**
- Například aby byl výpočetní výkon procesoru dvojnásobný, potřebujeme osmkrát složitější mikroprocesor

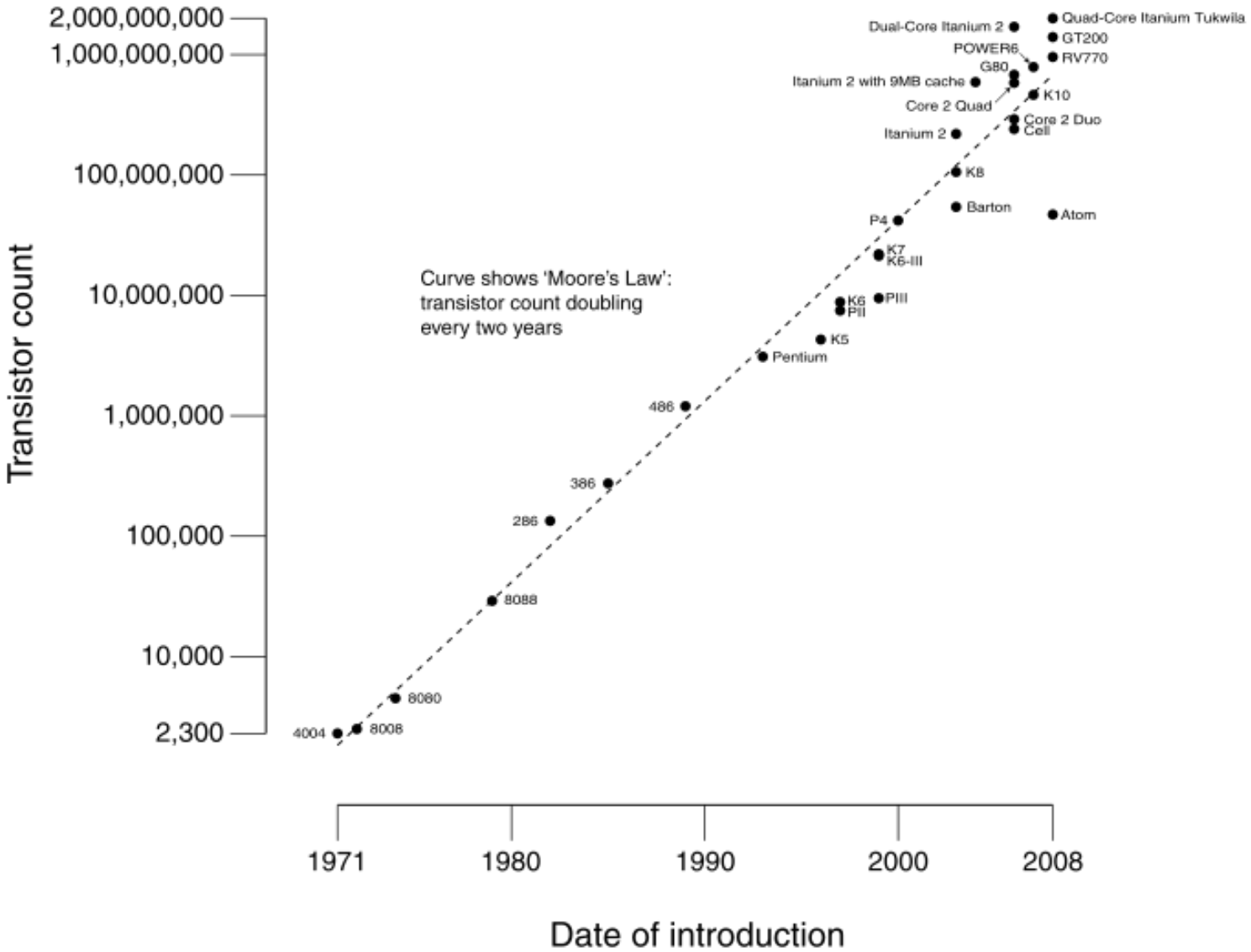
Moorův zákon



- Moorův zákon **přestává platit**, počet **tranzistorů** v mikroprocesorech sice **roste**, ale výkon mikroprocesorů za posledních deset let vzrostl pouze **velmi málo**, narozdíl například od období 1993-2003, kdy výkon vzrostl téměř 20x
- Moorův zákon nemůže platit navždy
- Postupně narážíme na fyzikální limity a tranzistory nebude možné dále zmenšovat
- Současné výrobní technologie (7nm a 5nm) jsou již dost problematické – tranzistory jsou tak malé, že elektrony dokáží „tunelovat“ i přes zavřený tranzistor
- Uzavřeným tranzistorem „prosakuje“ proud (leakage current), což zvyšuje spotřebu energie – od moderní výrobní technologie s malými tranzistory bychom ale čekali pravý opak
- Ukazuje se, že další zmenšování tranzistorů, se zřejmě nevyplatí
- Rozměr atomu křemíku je cca 0,4 nm – tranzistory jsou tvořeny několika málo atomy (např. 10 atomů) a začínají se projevovat podivné efekty kvantové fyziky

- Videa:
- https://www.youtube.com/watch?v=CUnQNTwmHHo
- https://www.youtube.com/watch?v=aWLBmapcJRU
- https://www.youtube.com/watch?v=AohW5jUrR20

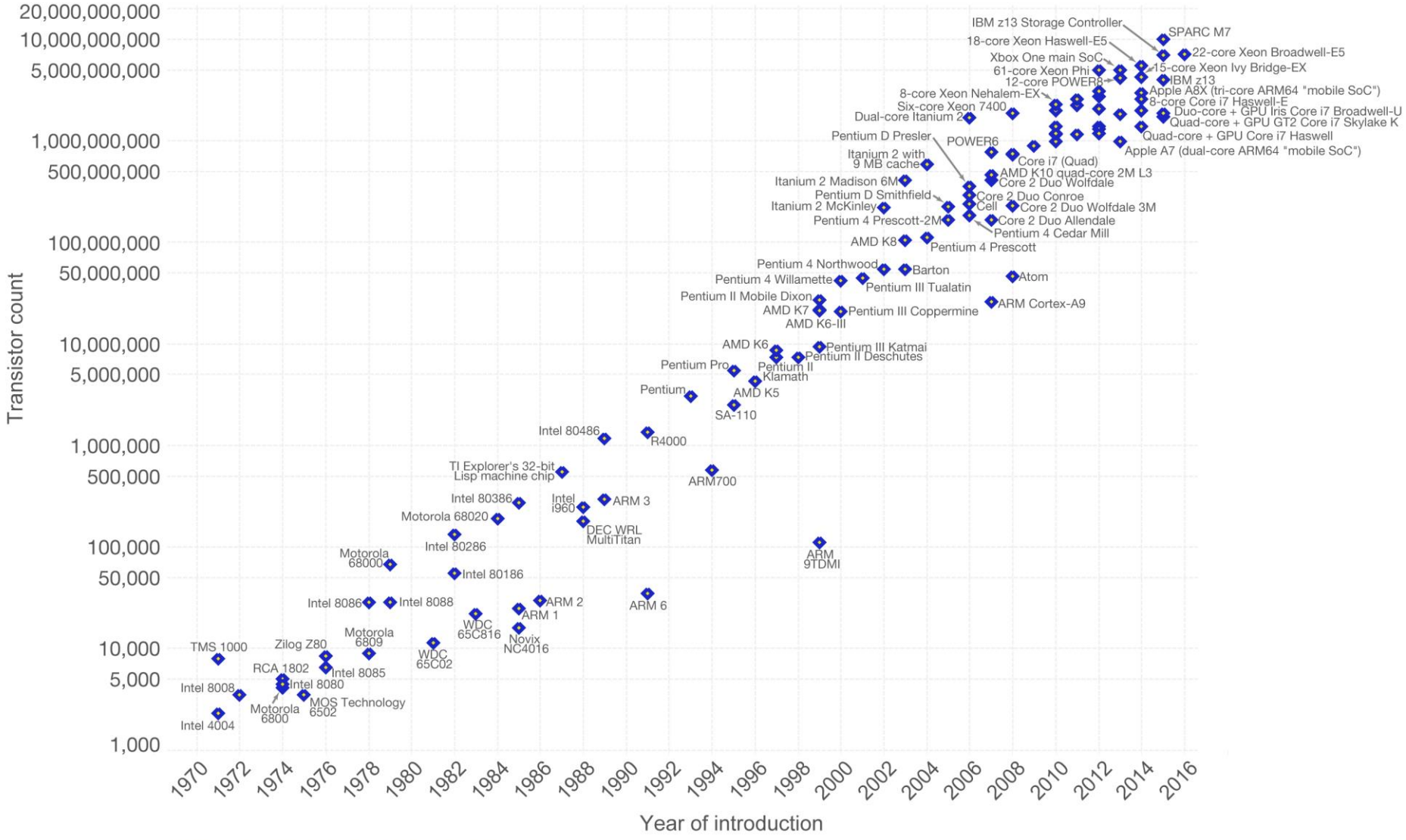
CPU Transistor Counts 1971-2008 & Moore's Law



Moore's Law – The number of transistors on integrated circuit chips (1971-2016)

Our World in Data

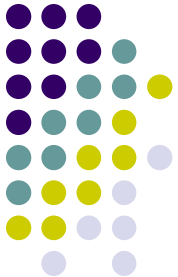
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)

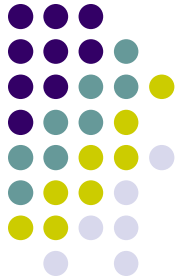
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

Růst frekvence



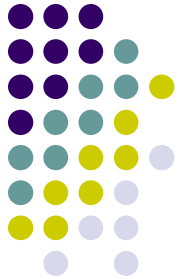
- Po velmi dlouhou dobu se neustále také zvyšovala frekvence procesorů
- Ta rostla také v průměru na **dvojnásobek** každé **dva roky**
- Počítač s procesorem běžícím na dvojnásobné frekvencí může být dvakrát výkonnější, ale nemusí to tak docela platit, protože záleží na tom, jak hodně ho brzdí jeho pomalé okolí (paměti, sběrnice…)
- Růst frekvence spolu s růstem počtů tranzistorů v období 1972 – 2003 způsoboval růst výkonu počítačů na **dvojnásobek** každých **18 měsíců**
- Od té doby se bohužel růst frekvence zastavil a výkon procesoru se dále zvyšuje pouze zvyšováním **IPC**
- Parametr **IPC** je ovšem velmi problematický – není konstantní, silně závisí na prováděném typu aplikace, počtu vláken, počtu podmíněných skoků, množství dat nedostupných v cache…

Murphyho zákon (vtip)



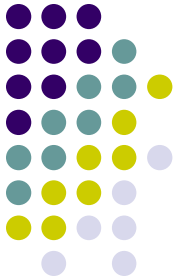
- **„Náročnost aplikací roste na dvojnásobek zhruba každé dva roky“**
- Složitost aplikací roste, mají stále větší paměťové nároky a jsou výpočetně komplikovanější
- Programátoři programují čím dál hůř a nesnaží se vůbec dělat malé rychlé optimalizované programy
- Používají se stále novější jazyky, které usnadňují vývoj aplikací, ale produkují programy obludných rozměrů s velmi neefektivním strojovým kódem
- Program napsaný v assembleru špičkovým programátorem, který zná detailně chování mikroprocesoru, může být až 1000x rychlejší než stejný program napsaný v Javě
- Výhodou Javy bude naopak to, že program bude vytvořen mnohem rychleji, půjdou v něm snadno hledat chyby a bude snadno přenositelný na různé platformy
- Kupujeme tedy stále výkonnější počítače, abychom na nich mohli svižně provozovat stále složitější a pomalejší aplikace

Zvyšování IPC



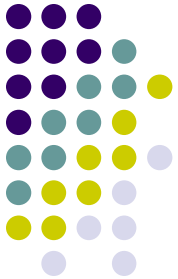
- V současné době tedy **frekvence procesorů neroste**, ale procesory jsou **stále složitější** a přidávají se do nich další a další tranzistory, aby byly schopny vykonat co nejvíce instrukcí naráz v jednom taktu (roste IPC)
- Z hlediska výrobce je nejjednodušší a nejefektivnější zdvojnásobit počet tranzistorů a IPC **zdvojnásobením celého mikroprocesoru** – vzniká tak **dvoujádrový mikroprocesor** (vlastně dva mikroprocesory v jednom)
- Pro výrobce by naopak bylo **velmi složité** dosáhnout zdvojnásobení počtu tranzistorů neustálým zdokonalováním a **zvyšováním složitosti** původního mikroprocesoru (jednoho jádra)

Vícejádrové procesory



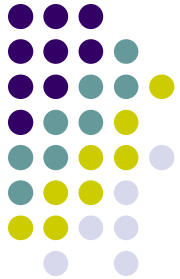
- **Multicore** = **vícejádrový** procesor
- Jedná se vlastně o paralelní zapojení několika mikroprocesorů, přičemž jsou umístěny na jednom jediném čipu a tváří se jako jeden celistvý procesor
- **dual-core** jsou v podstatě dva procesory v jednom (bez zdvojení těch částí, které mohou být využity oběma jádry společně)
- Pro využití potenciálu je zapotřebí speciálních aplikací – tzv. **multithreaded** aplikace
- jediné vlákno nedokáže využít potenciálu dvoujádrového procesoru - k tomu jsou totiž zapotřebí vlákna aspoň dvě
- Kde se tedy druhé vlákno vezme? Programátor ho musí vytvořit
- Programátor musí označit části programu a ty přidělit různým vláknům
- **hyperthreading** byla jakási simulace multicore na jednom procesoru
- Typickému programátorovi ale dosud bylo úplně lhostejné, zda jím vytvořený program poběží na zrovna vašem počítači dostatečně rychle
- Neběží-li rychle, vzkáže vám, abyste si prostě pořídili rychlejší – do budoucna je třeba **změnit filozofii programování** a vývoje nového softwaru

Threads - Vlákna



- Vlákna musí být vytvořena programátorem
- Programátor se musí při psaní program zamyslet, které jeho části by mohly probíhat nezávisle na sobě paralelně
- **Příklad:**
- Úkolem je uvařit k obědu smažený řízek, brambory, okurkový salát
- Činnost lze rozdělit do několika vláken, která mohou být vykonávána současně
- **První vlákno** bude posloupnost instrukcí popisující výrobu okurkového salátu (oškrábej okurku, nastrouhej ji, přidej cukr, přidej ocet…)
- **Druhé vlákno** je sled povelů pro uvaření brambor (oškrábej brambory, nakrájej je na kostičky, nalej vodu, přidej sůl, vař….)
- **Třetí vlákno** je sled povelů, který popisuje, jak nakrájet maso a obalit ho ve strouhance…
- Toto vlákno by šlo po nakrájení masa dála rozdělit na několik vláken, ve kterých by se mohl smažit každý řízek – vytvoříme tolik vláken, kolik máme prostředků (pánví a volných hořáků na sporáku), abychom mohli smažit více řízků paralelně
- Nakonec výsledky všech vláken spojíme – vše servírujeme na talíři dohromady

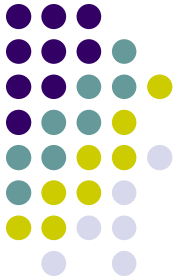
Threads - Vlákna



- Některé programy nelze žádným způsobem rozdělit na vlákna, která by mohla probíhat paralelně
- Takové programy, které nelze rozdělit na vlákna, poběží vždy stejně rychle bez ohledu na počet jader

- **Příklad:**
- Úkolem je dojít ze školy zpět domů
- Program lze napsat pouze jako jediný sled po sobě jdoucích kroků
  - Jdi 100 metrů na sever
  - Odboč doprava
  - Pokračuj 200 metrů rovně
  - Zahni doleva
  - Atd…

Threads - Vlákna



- Některé programy lze dělit na vlákna téměř neomezeně

Příklad:

- Hledání výsledku „hrubou silou“
- Najděte klíč o délce 5 znaků potřebný k odemčení zašifrovaného dokumentu
- První vlákno bude zkoušet všechny kombinací začínající A…..
- Druhé vlákno bude zkoušet všechny kombinací začínající B…..
- Třetí vlákno bude zkoušet všechny kombinací začínající C…..
- ……..

Paralelní programování

- Programování paralelních aplikací je intelektuálně mimořádně náročná činnost
- https://cs.wikipedia.org/wiki/Paraleln%C3%AD_v%C3%BDpo%C4%8Dty
- Přináší řadu úskalí –
  - Vymyslet zcela nový paralelní algoritmus - to je to úplně nejtěžší a ty nejdokonalejší paralelní algoritmy chápe jen pár lidí na celém světě (např. paralelní násobení matic nebo řazení polí s logaritmickou složitostí)
  - Jak rozdělit program na nezávislá vlákna a jaký bude optimální počet vláken
  - Synchronizace vláken (některé akce prováděné v jednom vlákně lze provádět až po dokončení akcí v jiném vlákně)
  - Sdílení společných prostředků (soubory, paměť, vstupy a výstupy)
  - Slučování mezivýsledků jednotlivých vláken do finálního výsledku
  - Zamezení **deadlocků**

Paralelní programování

Paralelní systémy a programování bývají nejobtížnějším předmětem vysokoškolského studia IT oborů

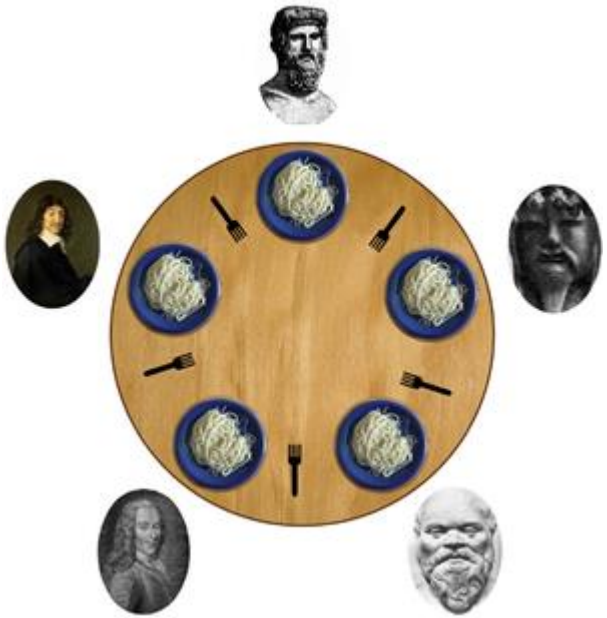
- Pár odkazů pro inspiraci při úvahách o vašem budoucím studiu
  - https://www.fel.cvut.cz/cz/education/bk/predmety/47/51/p4751806.html
  - http://bilakniha.cvut.cz/cs/predmet4643006.html
  - https://www.vutbr.cz/studenti/predmety/detail/218349

- Největším odborník na toto téma je u nás prof. Tvrdík (děkan FIT ČVUT)
  - https://www.youtube.com/watch?v=GoPvT7MZIzo
  - https://www.youtube.com/watch?v=rDTVK06Kpsc
  - https://www.youtube.com/watch?v=6Li7ConE_uk (zde například od cca 38. minuty uvidíte trochu lepší MergeSort)

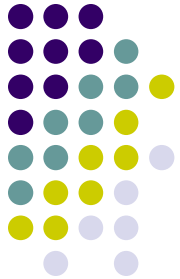
Zde zmíněná videa jsou jen na ukázku – nepouštějte si je celá, pouze si udělejte představu…

Deadlock

- Deadlock neboli uváznutí je jedním z nejnepříjemnějších fenoménů paralelního programování
- Nejčastěji bývá demonstrován na příkladu tzv. „večeřících filozofů“
- Kolem kulatého stolu sedí několik osob
- Mezi osobami leží stejný počet vidliček
- Aby se „filozof“ mohl najíst, potřebuje dvě vidličky (do každé ruky jednu)
- Fungování každého filozofa je popsáno jedním vláknem programu
- První instrukcí všech vláken je povel „uchop vidličku levou rukou“
- Vlákna se provádějí paralelně
- Všichni tak uchopí vidličku nalevo do levé ruky, všechny vidličky jsou zvednuté, na stole nezůstala žádná volná vidlička
- Program všech vláken pokračuje povelem „uchop druhou vidličku do pravé ruky“
- V tuto chvíli se všechna vlákna „zaseknou“, protože nelze pokračovat, na stole není žádná volná vidlička a všichni filozofové jednu drží v ruce, čímž se **navzájem blokují**
- Nastal **Deadlock**



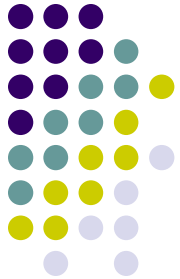
Deadlock



- Jiný příklad deadlocku:
- Paralelní program je rozdělen na vlákno A a vlákno B
- Obě vlákna jsou zpracovávána paralelně
- **Vlákno A**, potřebuje přístup k **tiskárně** a k **souboru** XY.TXT, aby mohlo pokračovat dále. Soubor XY.TXT je tímto vláknem otevřen a lze do něj zapisovat, ale tiskárnu používá vlákno B, takže vlákno A musí čekat na uvolnění přístupu k tiskárně (obě vlákna nemůže posílat data tiskárně současně – vznikl by chaotický výtisk)
- **Vlákno B**, potřebuje přístup k **tiskárně** a k **souboru** XY.TXT, aby mohlo pokračovat dále. Vlákno právě používá tiskárnu, ale nemůže zapisovat do souboru XY.TXT, protože tento soubor je otevřen vláknem A, takže vlákno B musí čekat na uvolnění přístupu k souboru XY.TXT
- Ani jedno z vláken nemůže pokračovat dále
- Je zřejmě, že obě vlákna budou čekat donekonečna
- Došlo k **deadlocku**

- Programátor paralelního programu musí mít velmi dobře promyšleno, aby k takovým situacím nedocházelo

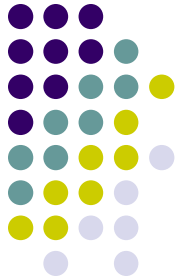
Deadlock



- Aby nastal deadlock, musí být splněny tyto podmínky

- **Vzájemné vyloučení** (Mutual exclusion) – Sdílený prostředek může v jednom okamžiku používat jenom jedno vlákno (např. soubor, aby nedošlo k porušení konzistence dat nebo tiskárna, aby se netiskla směs dvou dokumentů)
- **Drž a čekej** (Hold & waiit) - Vlákno může žádat o další prostředky, i když už má nějaké přiděleny.
- **Neodnímatelnost** (No-preemption) - Jakmile vlákno zmíněný prostředek vlastní, nelze mu ho bezpečně odejmout, musí ho samo vrátit.
- **Cyklické čekání** (Circular wait) – každé z vláken čeká na nějaký prostředek, nikdo nemůže pokračovat

Problematika multicore



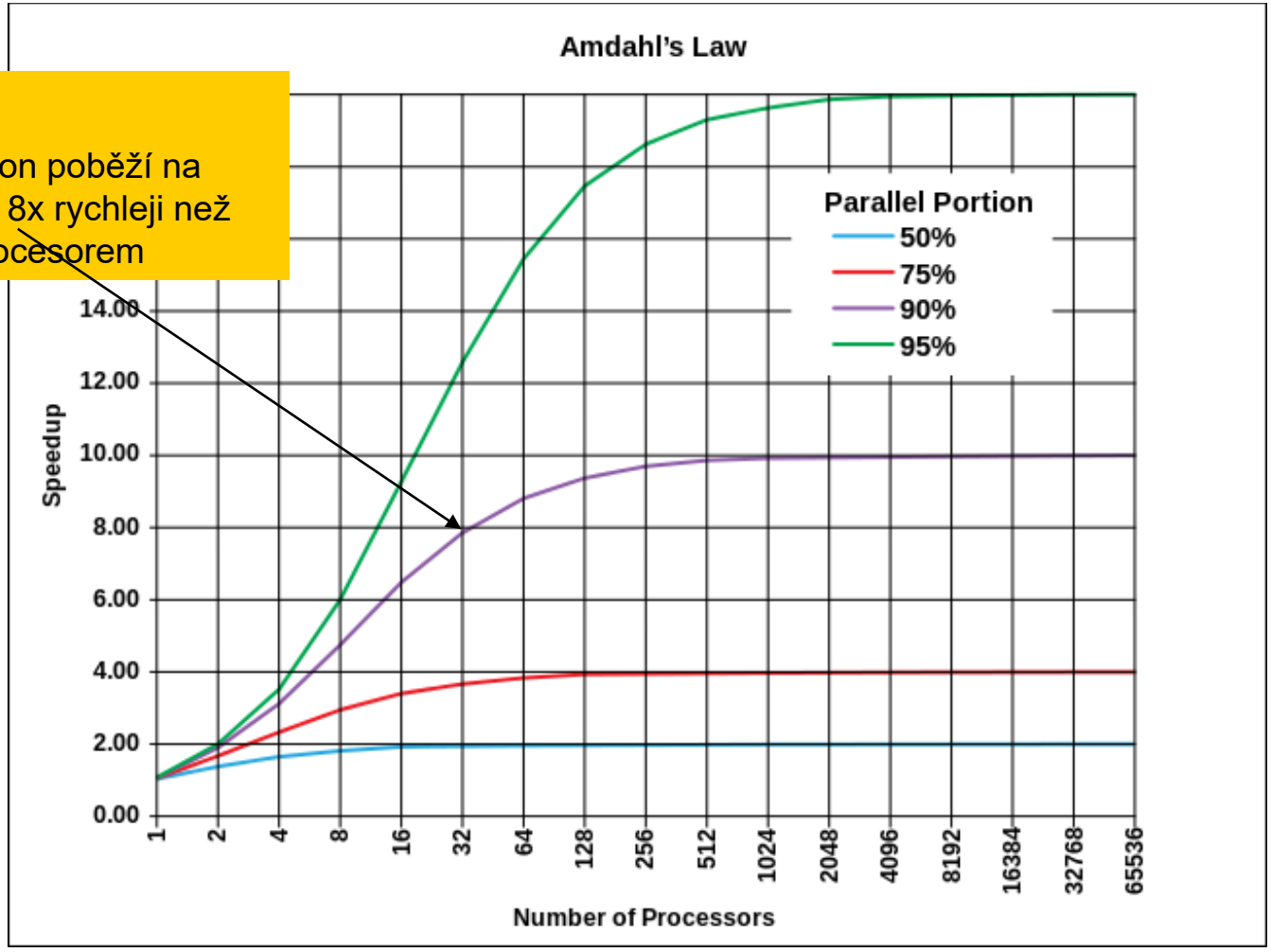
- Zásadní otázka zní: Bude dvoujádrový procesor dvakrát výkonnější než jednojádrový ?
- Odpověď zní: Obvykle ne
- S dvojnásobnou rychlostí by tento procesor vykonal pouze dokonale paralelní program, ale takové neexistují
- **Parallel portion** je podíl paralelizovatelné části programu
- Je-li např. Parallel portion 90 %, znamená to, že zbylých 10% programu nelze rozdělit na paralelní vlákna
- Parallel portion u některých reálných úloh je nula, protože úloha svým charakterem paralelizovat nelze – např. cesta z bodu A do bodu B krok za krokem
- Parallel portion u některých úloh je až 99% - např. komprimace zvuku a videa – každé vlákno zpracovává jiný časový interval. Zbylé jedno procento je třeba na rozdělení úkolů jednotlivým vláknům a spojení výsledků jednotlivých vláken do výsledného souboru
- Pouze programy, které mají velkou Parallel portion poběží na multicore procesoru přiměřeně rychle – cestu z bodu A do bodu B projdete stejně dlouho na jednojádrovém i osmijádrovém procesoru, ale video zkomprimuji na osmijádrovém procesoru možná i sedmkrát rychleji

Amdahlův zákon

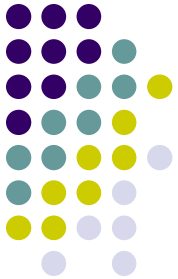
- Závislost **zrychlení výpočtu** na počtu jader a Paralell portion vystihuje tzv. Amdahlův zákon

Například:

Program s 90% Paralell portion poběží na počítači s 32 mikroprocesory 8x rychleji než na počítači s jedním mikroprocesorem



Amdahlův zákon

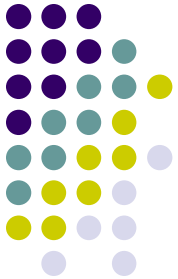


- Amdahlův zákon je čistě matematicky odvozen

Příklad:

- Vyřešení úkolu vyžaduje **100 pracovních hodin**
- 90 procent činnosti lze provádět paralelně (parallel portion 90 %)
- Úkol lze tedy rozdělit na činnosti, které zaberou 10 hodin a nelze je paralelizovat a na činnosti, které vyžadují 90 hodin a lze je paralelizovat
- Na úkolu bude pracovat 10 pracovníků současně
- Činnosti, které vyžadují 90 hodin, tedy vykoná 10 pracovníků za 9 hodin
- K tomu je potřeba přičíst 10 hodin na činnosti, které nelze paralelizovat
- Celkově tedy úkol bude vykonán za 9+10 hodin = 19 hodin
- Zrychlení vykonání úkolu při provádění 10 pracovníky je tedy 100/19 = 5,26 x (viz graf, fialová křivka pro parallel portion 90)

Amdahlův zákon

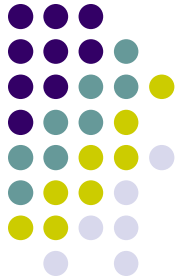


- Amdahlův zákon říká, kolikrát se teoreticky **může maximálně** zrychlit provedení programu – ve skutečnosti může být zrychlení nižší (nikdy ne vyšší)

Příklad:

- Několik dělníků má za úkol vykopat 20 metrů dlouhý výkop
- Dělnící mohou pracovat paralelně, ale před zahájením výkopu je potřeba rozdělit jim práci a po skončení práce odvézt nářadí – tuto část úkolu nelze paralelizovat
- Paralell portion by mohla být okolo 75 %
- Dle Amdahlova zákona by při použití 64 dělníků mohlo dojít téměř ke čtyřnásobnému zrychlení
- Amdahlův zákon však nevystihuje jednou podstatnou věc – dělníci si budou ve výkopu překážet – Čím více vláken, tím více se mohou různě blokovat při přístupu do paměti, při čtení dat z disku, množství dat, která lze cachovat pro každé vlákno klesá atd…
- Čím více dělníků a menší výkop, tím hůře se práce porcuje
- Další problém spočívá v tom, že rychlý procesor má pomalé okolí a sběrnice – vykopaná zemina nebude odvážena dostatečnou rychlostí (Pokud rozdělíte kódování videa na sto vláken, nebudete pro ně stíhat načítat data z disku a zapisovat výsledná data)

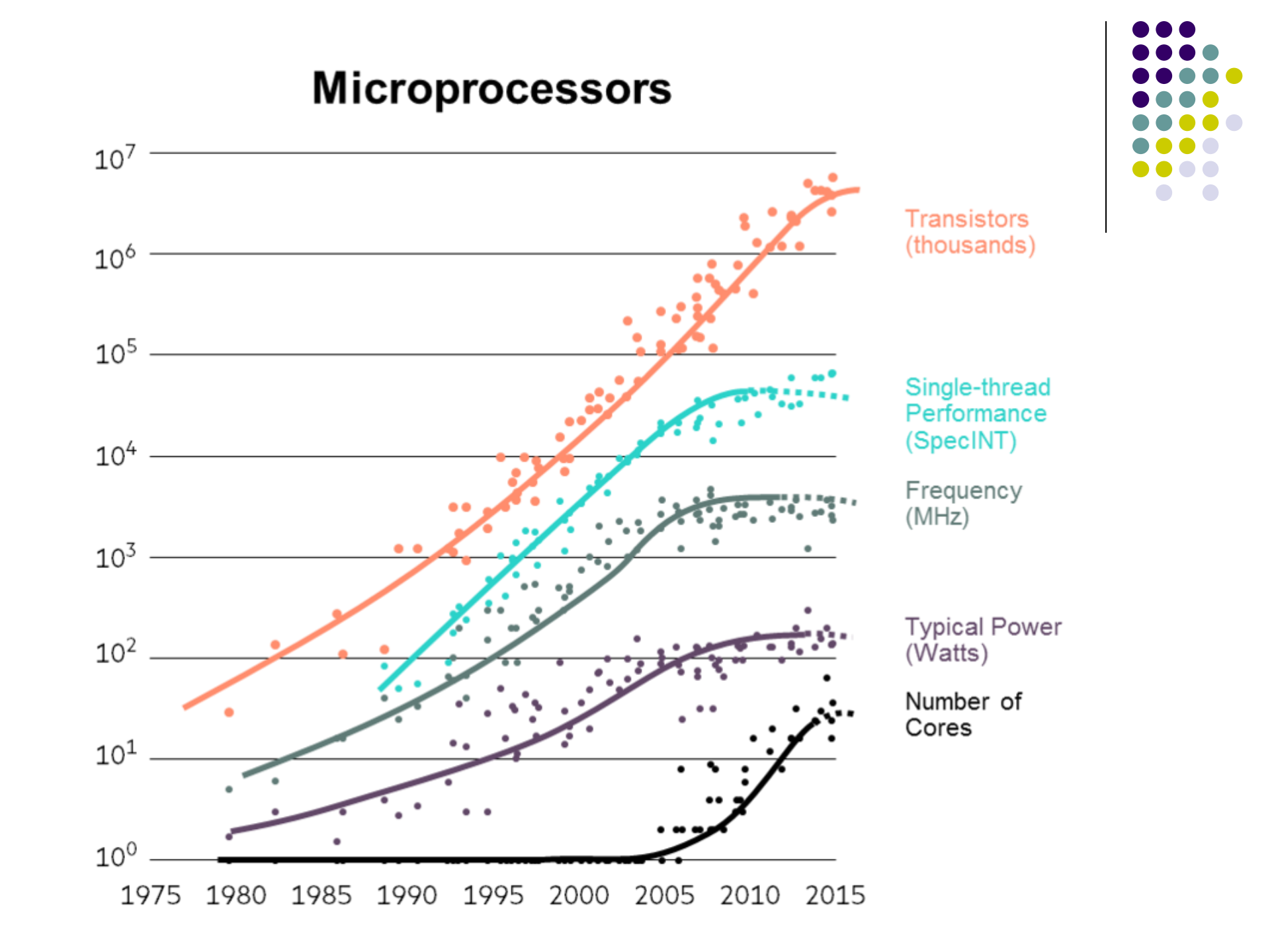
Zákon klesajících marginálních výnosů



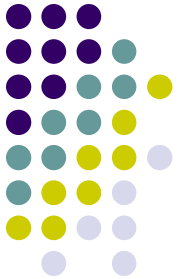
- **Celkový výkon** počítače není dán jen výkonem mikroprocesoru, ale závisí také na kapacitě paměti, rychlosti paměti, rychlosti sběrnice, rychlosti pevného disku….
- Pokud budeme zrychlovat **pouze jeden z parametrů**, např. rychlost paměti, poroste výkon celého počítače zpočátku rychle, ale od určité hranice další zvyšování rychlosti paměti nepřinese **žádný účinek**
- Pořídíte-li si **dvakrát rychlejší pevný disk**, je možné, že výkon vašeho počítače jako celku vzroste **pouze o 5%,** protože závisí na mnoha dalších parametrech
- To samé platí pro **výkon procesoru** – zpočátku výkon celého počítače prudce rostl se zvyšujícím se výkonem stále modernějších procesorů, ale v dnešní době se právě nacházíme na hranici, za kterou další zvyšování výkonu procesoru nepřináší žádný efekt, protože celkový výkon počítače závisející i na dalších parametrech již v podstatě neroste

- Další zvyšování výkonu procesorů je velmi problematické, protože ho lze dosáhnout pouze **vyšším IPC**.
- Navíc i když se výkon procesoru o pár procent zvýší, nezvýší se odpovídajícím způsobem výkon počítače jako celku

- Počet tranzistorů v mikroprocesoru se zvýší na dvojnásobek → Výpočetní výkon procesoru vzroste pouze o 30 % → Výkon počítače se tím zvýší jen o 10 %


Microprocessors
10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
1975
1980
1985
1990
1995
2000
2005
2010
2015
Transistors (thousands)
Single-thread Performance (SpecINT)
Frequency (MHz)
Typical Power (Watts)
Number of Cores

Kontrolní otázky



- Co vystihuje Amdahlův zákon ? **kolikrát se teoreticky může maximálně zrychlit provedení paralelního programu**
- Co vystihuje Moorův zákon ? – **počet tranzistorů se za každých 18 měsíců při stejné ceně zdvojnásobí**
- Platí Moorův zákon i dnes ? – **přestává platit**
- Platí Amdahlův zákon i dnes ? **ano**
- Roste výkon počítače na dvojnásobek každé dva roky i dnes ? **Ne**
- Výkon přepočtený na jeden tranzistor se neustále zvyšuje nebo snižuje? **snižuje**
- Výkon mikroprocesorů v letech 2010-2020 roste především díky zvyšování frekvence nebo zvyšováním IPC ? **Zvyšování IPC**
- Co je to deadlock ? – **pokud dojde k tomu, že například vlákno 1 potřebuje data z vlákna 2 a vlákno 2 data z vlákna 1**
- Co je to parallel portion ? **Podíl paralelizovatelné části programu**
- Vysvětlete zákon klesajících marginálních výnosů – **to, že výkon nezávisí jen procesoru, ale i na dalších komponentách**
- Na jaké taktovací frekvenci pracovaly první historické mikroprocesory ? **Jednotky až desítky MHz**
- Na jaké taktovací frekvenci pracují současné mikroprocesory ? **4 GHz**
- Kolikrát rychleji vykoná osmijádrový procesor program, jehož parallel portion je 80 % ? **100 vs 30 – cca 3.3x**