Vývoj architektur počítače

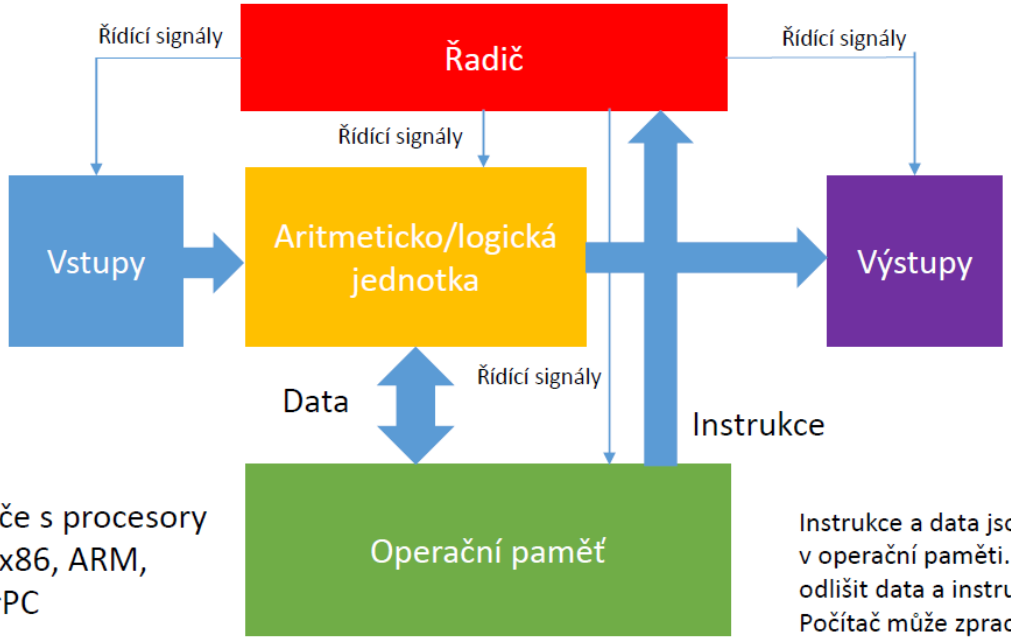
Břetislav Bakala

Od automatu k RISC architektuře

- Z počátku byly počítače **specializované na určité výpočty** a daly se přeprogramovat pouze **změnou zapojení – automaty**

- Koncept Johna **Von Neumanna** položil základ pro univerzální programovatelné stroje. Díky tomuto konceptu vznikl procesor, který **vykonává zadaný program**.

Od automatu k RISC architektuře



Počítače s procesory např. x86, ARM, PowerPC

Instrukce a data jsou uložena v operační paměti. Nelze odlišit data a instrukce. Počítač může zpracovat data jako instrukci a naopak instrukci jako data.

Od automatu k RISC architektuře

- Využití standardizovaného **operačního systému** (UNIX, Linux) umožnilo využít **rozdílné architektury** a jejich rozvoj, celkové **rozšíření a zlevnění**.
- Tyto změny umožnily **využití architektury RISC** (r.1980) – využití **paralelního zpracování instrukcí** (zřetězení, efektivnější organizace cache VAX-11).

CISC – RISC – ARM architektury

- Intel (původně CISC procesor) využíval nejprve RISC architekturu **vnitřním překladem instrukcí 80x86** aby mohl využívat inovace architektury RISC, což vedlo s nárůstem počtu tranzistorů realizujících překlad k přehřívání.
- U dnešních laciných aplikací se využívají architektury RISC, kde dominuje **architektura ARM** (Advanced RISC Machine).

Architektury pro různá využití

- **Mobilní zařízení** (Personal mobile device - **PMD**): cena, spotřeba, zpracování médií, doba odezvy
- **Pracovní stanice** (desktop): cena, výkon, spotřeba, grafika
- **Server**: propustnost, dostupnost, škálovatelnost, spotřeba
- **Cluster, datové úložiště**: cena - výkon, propustnost v poměru se spotřebou

Nároky na výkon počítačů

- Rostoucí výkon počítačů umožnil v oblasti programování **přechod od** nižších **kompilačních programovacích jazyků** (C, C++) s tradičním kompilátorem a linkerem **k interpretačním a skriptovacím jazykům** (Java, C#, Python, …) s použitím frameworku, **just-in-time překladače** a trace-based překladu. Vývoj software se tak změnil na **„Software as a Service (SaaS)**.
- Nová oblast dnešních aplikací (řeč, zvuk, obraz, video) vyžaduje **předvídatelnou odezvu v krátkém čase**.

Více jader - pralelní zpracování

- **Do roku 2003** byly dostupné pouze **jednojádrové procesory** (do té doby byl zvyšován výkon zvyšováním taktovací frekvence procesoru až na hranici technologických možností (viz pokusy přetaktování). Tento technologický limit zahájil vývoj **vícejádrových procesorů** s **paralelním zpracováním,** je možné je v základu rozdělit na:
- **Data level parallelism** DLP – více dat zpracovaných ve stejnou dobu
- **Task level parallelism** TLP – více úloh najednou

Paralelní zpracování - hardware

- **Paralelní zpracování instrukcí -** Instruction level parallelism (**ILP**)
- **Vektorová grafika -** Graphic Processor Unit (**GPU**)
- **Paralelní běh vláken -** Thread level parallelism (**TLP**)
- **Paralelní zpracování požadavků** - Request level parallelism (**RLP**) v datovém úložišti na úrovni operačního systému nebo programu

Flynnova klasifikace I.

- **Michael Flynn (1966) rozdělil zpracování dat a instrukcí do čtyř kategorií:**

- Single instruction, single data stream (**SISD**) – uniprocesor, standardní sekvenční procesor, může využívat ILP – superskalární architekturu a předvídání instrukcí
- Single instruction, multiple data stream (**SIMD**) – stejná instrukce se současně vykonává nad různými daty, využívá DLP, každý procesor má svou datovou paměť, ale společnou paměť instrukcí. (Vektorová architektura, multimediální rozšíření – MMX).

Flynnova klasifikace II.

- Multiple instruction, single data stream (MISD) – nevyužívá se.
- Multiple instruction, multiple data stream (MIMD) – každý procesor načítá své instrukce a operuje se svými daty – TLP. V základu je pružnější, než SIMD, je ale z principu dražší

Tato klasifikace je pouze modelová, mnoho procesorů jsou hybridní.

Architektura instrukční sady I.

Instruction set architecture (ISA) určuje hranici mezi HW a SW

1. **Třída ISA -** dnes převážně architektura univerzálních registrů, operand je registr nebo adresa v paměti

- Register-memmory ISA: 80x86, adresa paměti v operandu instrukce
- Load-store ISA: ARM, přístup do paměti pouze pomocí instrukcí load, store

2. **Adresace paměti**

- Virtuálně – všechny PC a servery 80x86, ARM s využití byte adresace
- Některé architektury ARM vyžadují zarovnání objektů – aligned (rychlejší)

Architektura instrukční sady II.

3. **Adresovací mód** – určuje adresu v paměťovém objektu. MIPS adresuje registr, konstantu, a Displacement – offset je sečten s registrem. 80x86 má navíc displacement: bez registru, dva registry – báze+index. ARM má navíc k MIPS relativní adresování pomocí dvou registrů, umožňuje auto in a dekrementaci. dnes převážně architektura univerzálních registrů, operand je registr nebo adresa v paměti
4. **Typ a velikost perandu**

- 8 bit ASCII, 16bit Unicode, 32 bit Integer, 64 bit dubble, IEEE754 floatpoint 32 a 64 bit, 80x86 podporuje navíc 80bit floatpoint.

Architektura instrukční sady III.

5. **Operace** – přesun dat, ALU, fp. **MIPS – jednoduché zřetězení** operací - RISC, **80x86 velká instrukční sada**,
6. **Řízení toku instrukcí:** podmíněný a nepodmíněný **skok, procedury**, relativní adresování skoku, **MIPS testuje podmíněný skok na obsah registru**, **80x86 a ARM testuje výsledek stavového registru** po provedení předchozí ALU operaci. **ARM a MIPS** umisťuje **návratovou adresu do registru, 80x86 do zásobníku.**
7. **Kódování ISA**: **pevná délka** instrukce – 32bit ARM a MIPS (Thumb 16bit), zjednodušuje dekódování instrukce. 80x86 - proměnná délka instrukce 1 až 18 byte, program zabírá méně místa. Záleží přitom na možnostech kompilace - počtu registrů a adresovacích módů.

Stavební prvky architektury počítače

- **Logická struktura (mikroarchitektura)** – paměťový systém, sběrnicové propojení, struktura CPU – ALU.
  Např:
  AMD Opteron a Intel Core i7 mají **stejnou instrukční sadu** x86, ale velmi **rozdílné zřetězení instrukcí a organizaci cache.**

- **Hardware** – realizace logické struktury danou technologií. Např:
  Intel Core i7 a Intel Xeon 7560 mají stejnou instrukční sadu a velmi podobnou logickou strukturu, nabízí však rozdílnou taktovací frekvenci a paměťový systém, čímž předurčují Xeon 7560 více efektivní pro serverové aplikace.

Funkční požadavky a jejich podpora

- **Oblast nasazení** – PMD, PC, server, cluster, embedded
- **Úroveň SW komatibility** – programovací jazyky, binární kódy.
- Požadavky na **operační systém** – adresový prostor, správa paměti (segmentace, stránkování), ochrana (úrovně oprávnění, podpora virtuálních strojů)
- **Standardy** – float point (IEEE 754), I/O rozhraní (serial ATA, PCI Express), operační systém (UNIX, Windows, Linux, CISCO IOS), síť (Ethernet, Infiniband), programovací jazyky (ANSI C, C++, Java, Fortran)

Trendy v technologii

- **Integrované obvody** – hustota integrace roste v souladu s Mooreovým zákonem (hustota tranzistorů se zdvojnásobí za každých 18 měsíců při zachování stejné ceny), rychlost obvodů takto rapidně neroste, naráží se na technologické hranice – velikost molekul polovodičů
- **Paměti DRAM** – podobně stoupá kapacita a rychlost DRAM čipů, .
- **Paměti Flash** – standard pro PMD, využití se značně rozšiřuje, kapacita stoupá každým rokem na dvojnásobek,
- **Magnetické disky** – 20x levnější než Flash, a 400x levnější než DRAM. Vhodné především pro servery, a paměťové sklady.
- **Síťové technologie** – dáno vývojem přenosových systémů

Napájení a spotřeba obvodů

- **Maximální příkon procesoru** závisí na prováděné činnosti a špičkové proudy by mohly překročit mezní hranici. Moderní procesory provádí v krajním případě regulaci napětí, která ale vede ke snížení výpočetního výkonu.
- **Trvalá spotřeba** – thermal design power (**TDP**). Určuje nároky na chlazení procesoru. Je to průměrná spotřeba při běžném výpočtu (výkonové špičky jsou až 1,5x vyšší). Napájení a chlazení procesoru se navrhuje o mnoho vyšší hodnotu příkonu, než je TDP. Moderní procesory jsou chráněny:
  - Snížením taktovací frekvence procesoru a tím snížení příkonu
  - Detekcí překročení maximální teploty a vypnutí napájení

Příkon a spotřebovaná energie

Který parametr je vhodnější? Příkon TDP nebo spotřebovaná energie?
Příklad:
Procesor A má o 20% vyšší TDP než procesor B
Procesor A provede úlohu v 70% času než procesor B

Energie spotřebovaná procesorem A je 1,2 x 0,7 = 0,84 energie procesoru B

Procesor A je tedy pro danou úlohu lepší, i když má větší TDP

Mohlo by se zdát, že pro server nebo cloud stačí uvažovat procesor B s nižším TDP, poněvadž úloha není ohraničená. Procesor B ale vykoná méně práce za stejné množství vynaložené energie.

TDP jako měřítko - pokud jsme limitováni maximálním příkonem.

Dynamická spotřeba energie

- Primární spotřeba energie je ve spínacích tranzistorech (CMOS)
- Energie při přepnutí je závislá na kapacitě zátěže a napětí pulzu:

energie při přepnutí ~ ½ x kapacita zátěže x napětí ^{2} (polovina impulzu)

Příkon při přepnutí ~ ½ x kapacita zátěže x napětí ^{2} x přepínací frekvence

Při stejné úloze snížení frekvence sníží příkon, nikoliv spotřebovanou energii.

Opatření ke snížení příkonu I.

V poslední době jsou možnosti jak zlepšit energetickou účinnost i při nezvyšující se taktovací frekvenci a konstantním napájecím napětí:

- **Uspání neaktivních modulů procesoru -** vypnutím taktovací frekvence, např do float point modulu, pokud se nevykonává instrukce fp.
- **Dynamic Voltage Frequency Scaling** – **DVFS** – při nízké aktivitě modulu, kdy není potřeba maximální výkon, se snižuje taktovací frekvence a napětí.
- **Návrh pro typické případy** – např. PMD jsou často v nečinnosti. Pak je možné přepnout paměť do low power módu. DRAM pouze uchovává obsah – nelze z ní číst ani do ní zapisovat, HD vypne otáčení desek.

Opatření ke snížení příkonu II.

- **Přetaktování** - technika používaná ručně již dříve, přetaktování bylo pevné
  - **Turbo mode** – obvody čipu rozhodnou, že je bezpečné pracovat na vyšší frekvenci po krátkou dobu, případně na několika jádrech, dokud teplota nezačne stoupat. (Intel nabízí od r. 2008, zvýšení frekvence okolo 10%) Operační systém může Turbo mode vypnout, takže programátoři mohou být překvapeni, že jejich programy se liší ve výkonu.
- Kromě dynamických ztrát dochází také ke **ztrátám statickým**, daným přímo **úměrně počtu zařízení**.
- **Race-to-halt** strategie, kdy může mít smysl **použít rychlejší a méně energeticky úsporný procesor**, aby se zbytek systému **dostal dříve do režimu spánku**

Spolehlivost architektury počítače

- **Spolehlivost** (reliability) se dá vyjádřit střední dobou mezi poruchami (Mean Time To Failure - MTTF). Poruchovost (failure rate) je převrácenou hodnotou MTTF.

$$poruchovost = 1/spolehlivost$$

- **Dostupnost** (availability) modulu vyjadřuje střídání mezi dostupným stavem a střední dobou opravy (Mean Time To Repair - MTTR)

$$dostupnost\ modulu = \frac{MTTF}{(MTTF+MTTR)}$$

- **Spolehlivost systému**: $MTTF_{\ systém} = \frac{1}{\sum poruchovost\ komponent} = \frac{1}{\sum \frac{1}{MTTF_{komponent}}}$