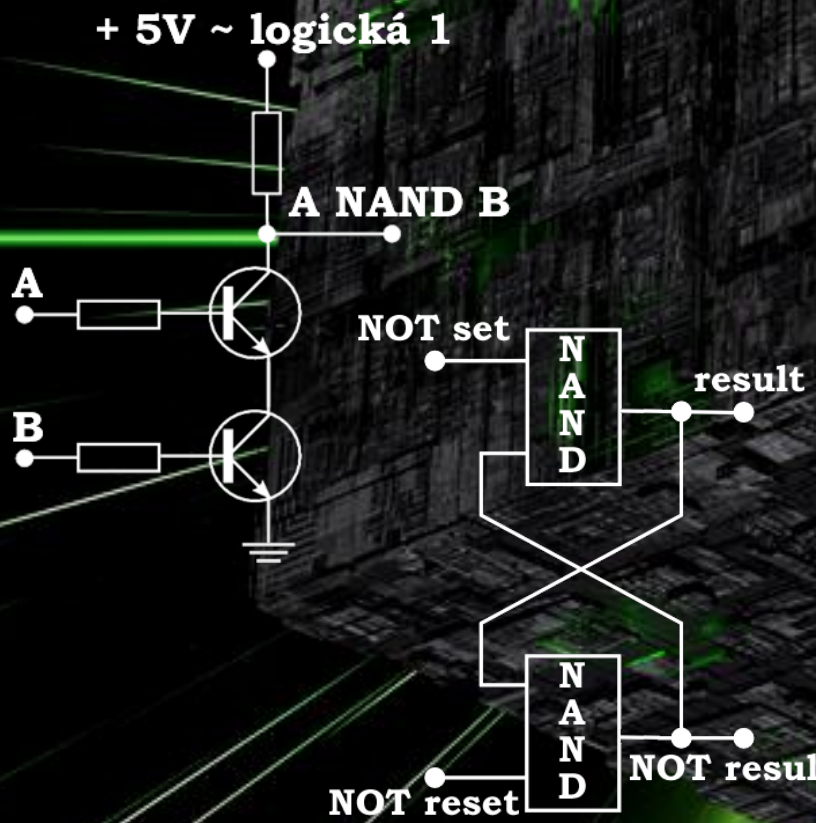
Princípy počítačů

Von Neumannova Architektura



Martin Urza

Co je to architektura počítače?

- Architektura udává, z jakých částí je počítač složen, jakou mají které části funkci a jak jsou mezi sebou propojené, jak komunikují a podobně.
  - Toto není definice, jen vysvětlení selským rozumem.
- Architektura naopak **není**, jestli máte v počítači 2GB nebo 4GB operační paměti, či jak rychlý je procesor.
- Z toho mimo jiné vyplývá, že architektura drtivé většiny dnešních PC je prakticky stejná.
- Naopak architektura palubního počítače v autě bude typicky odlišná od architektury PC.
- Architektura univerzálních počítačů zůstává přes padesát let téměř neměnná (jen malé změny).

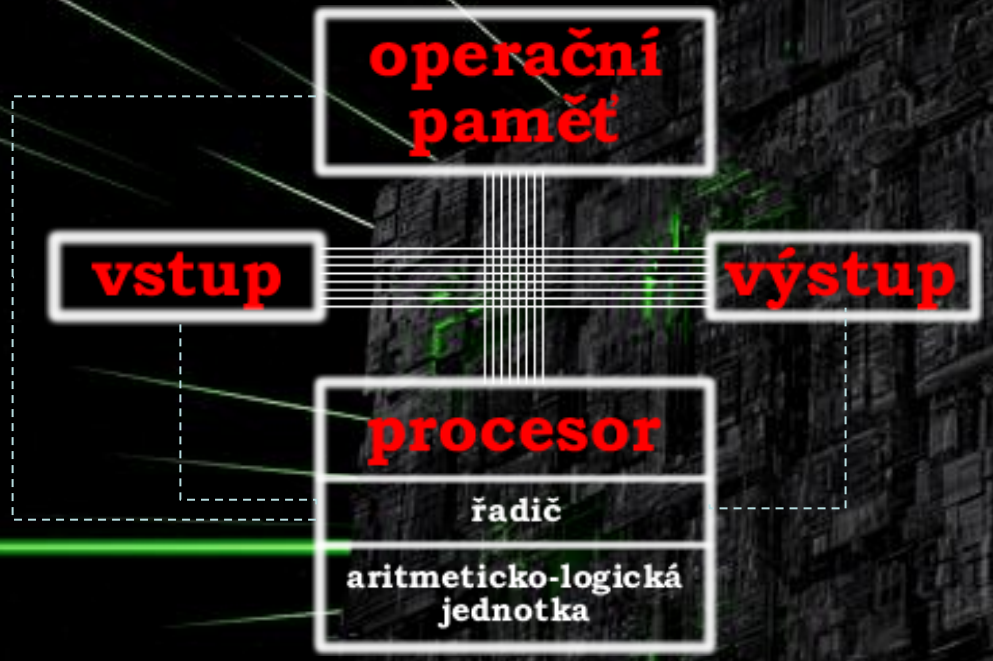
John Ludwig von Neumann

- Geniální matematik a vědec s jazykovým nadáním.
  - V šesti letech uměl zpaměti dělit osmimístnými čísly a žertoval se svým otcem ve starořečtině.
  - V sedmnácti publikoval svou první vědeckou práci.
  - Studoval chemii, ta pro něj byla moc snadná - během studia vypracoval doktorskou práci z matematiky.
  - Ve dvaadvaceti letech se stal nejmladším asistujícím profesorem v historii (na universitě v Berlíně).
  - V pětadvaceti publikoval teorii her, matematický model, který se dodnes využívá v ekonomii.
  - Udělal mnoho zásadních objevů ve fyzice.
  - Spolupracoval s Albertem Einsteinem.

Von Neumannova architektura

- Koncepce, kterou John von Neumann vymyslel v první polovině čtyřicátých let dvacátého století.
- Dodnes se používá architektura velmi podobná této, lze vlastně říci, že von Neumannova architektura vydržela více než půl století, což je téměř tolik, jak starý je celý tento obor (od reléových počítačů).
- Někdy je mylně celá von Neumannova architektura prezentována jako velmi zprofanovaný obrázek von Neumannova schématu. **To zdaleka není úplné, architektura udává princip fungování počítačů!**
  - Ač se na von Neumannově schématu tato architektura dobře vysvětluje, je třeba chápat souvislosti.

Von Neumannovo schéma

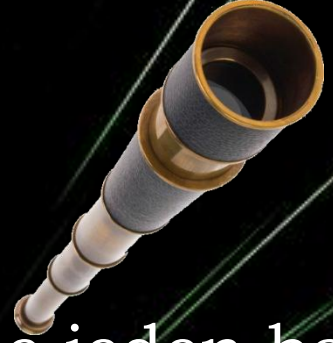


- Počítač je řízen **řadičem** procesoru, ten zejména řídí komunikaci zařízení.
  - Komunikace probíhá po **sběrnicích**, což jsou na obrázku úzké čáry, ve skutečnosti dráty (vně nebo v základní desce).
- Výpočty vykonává **aritmeticko-logická jednotka (ALU)**, pracovní data ukládá do **operační paměti.**
- Všechna ostatní zařízení jsou vstupní, výstupní, nebo vstupně-výstupní, v obrázku jsou zařazena do obdélníků **vstup** a **výstup**.

Jak funguje paměť?

- Paměť je zařízení k uchovávání dat. Pochopitelně je žádoucí, aby do ní šla data nejen ukládat, ale také, aby se z ní tato data dala opět načíst.
- Představte si sklad, do kterého ukládáte věci, které z něj chcete později vyzvedávat. Je-li to velký sklad, typicky dostanete ke svým věcem nějaký lísteček.
- V operační paměti funguje místo systému lístečků systém adres, takzvané adresování (dat).
- Když je potřeba vyzvednout z paměti nějaká data, pošle se dovnitř jejich adresa a ven vypadnou data.
- Při ukládání se do paměti pošlou data i s adresou, na kterou mají být uložena.

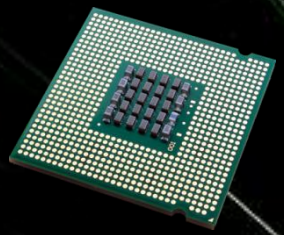
Adresování paměti



- Základní jednotkou operační paměti je jeden bajt.
- První bajt má adresu 0, každý další o 1 vyšší.
  - Má-li paměť 4GB (4,294,967,296 bajtů), jsou její adresy v rozsahu 0 až 4,294,967,295 (o 1 méně).
- Kolik paměti zabere adresa? Záleží na rozsahu.
  - 1b ~ 2^1 stavů, 2b ~ 2^2 stavů, 8b = 2^8 stavů, ….
    - To vyplývá z toho, co to je bit (viz. minulá přednáška).
    - Ukládáme-li celá čísla, pak každý stav je jedno číslo.
  - 8b = 1B (tedy osm bitů je jeden bajt)
    - To z ničeho nevyplývá, je to jen název pro osm bitů.
  - Otázka tedy zní: Kolik bitů (nebo bajtů) je potřeba pro zapamatování N stavů, kde N je počet bajtů paměti?

Příklady adresování paměti

- Nejprve je třeba pochopit, proč do n bitů lze uložit 2^n stavů. V binární soustavě je to jako v dekadické.
  - Bity jsou jako ciferné pozice, jenže v binární soustavě.
    - Do 4 cifer lze v dekadické soustavě uložit 10^4 čísel (jsou to čísla 0000 až 9999 a těch je v součtu právě 10^4).
    - Do 4 cifer lze v binární soustavě uložit 2^4 čísel (jsou to čísla 0000 až 1111 a těch je v součtu právě 2^4).
      - Binární počítání do patnácti: 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111.
- To znamená, že například do 2 bajtů (tedy 16 bitů) lze uložit 2^{16} (= 65,536) stavů, tedy i tolik čísel a tím pádem i přesně stejně adres. Adresa o velikosti 2B tedy stačí pro paměti do velikosti 65,536B.

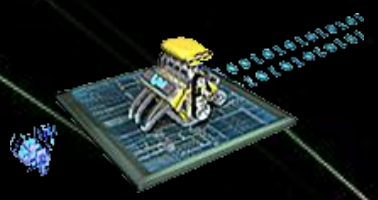


Jak funguje procesor?

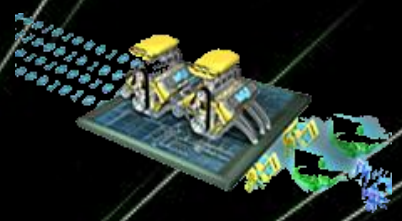
- Procesor je velmi hloupý a extrémně rychlý stroj, který pouze vykonává instrukce (které čte z paměti).
- Procesor pracuje v krocích, které jsou řízeny tiky hodin. Zjednodušeně platí, že v každém tiku vykoná jednu instrukci. **Nemůže nikdy přestat (až na velmi vzácné výjimky, které nebudeme řešit).**
- Rychlost, kterou hodiny tikají, odpovídá frekvenci procesoru. Hodiny tiknou za vteřinu tolikrát, jaká je jejich frekvence. Při frekvenci 1Hz tiknou jednou za vteřinu, při frekvenci 2GHz tiknou dvě miliardy krát.
- Taktovací frekvence (hodin procesoru) říká, kolik instrukcí stihne procesor za jednu sekundu vykonat.

Kde bere procesor instrukce?

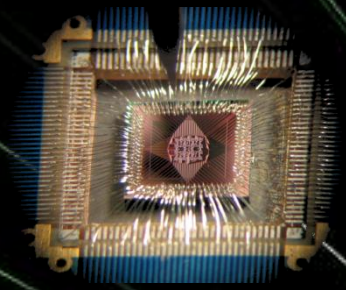
- Instrukce jsou uloženy v paměti (zkompilovaný kód).
- Jako vše v paměti, i instrukce mají své adresy.
- Procesor má v sobě číslo (instruction/program counter register), ve kterém je uložena adresa další instrukce, která má být vykonána v příštím tiku.
- Při každém vykonání instrukce se toto číslo zvyšuje tak, aby byla „na řadě“ hned následující instrukce. Tedy za předpokladu, že jsou instrukce dlouhé například 4 bajty, přičte se v každém tiku čtyřka.
- Skoky (tedy i cykly) jsou realizovány tím, že existují instrukce, které program counter register změní. Procesor pak vykonává instrukce z jiného místa.



Příklad vykonávání instrukcí



- Uvažujme procesor, který má instrukce dlouhé čtyři bajty (skutečnost bývá typicky složitější, například procesory na většině dnešních počítačů nemají pevně danou délku instrukcí, každá je jinak dlouhá).
- Je-li program counter register (PCR) nastaven na 16,384, znamená to, že v dalším tiku bude načtena instrukce z adresy 16,384 a PCR se zvýší na 16,388, další instrukce jej zvýší na 16,392 a tak dále.
- Je-li v programu while-cyklus, který obsahuje osm instrukcí, na jeho konci se vyhodnotí podmínka a pokud je třeba návrat na začátek (cyklu), odečte se od PCR 32 (= 8 instrukcí * 4 bajty délky instrukce).



Vnitřní stavba procesoru

- Informace na tomto slajdu jsou velmi zjednodušené a slouží pouze k pochopení slajdu následujícího.
- Procesor je navržen pro práci s adresami určité délky/velikosti (délka = velikost), např. 32b či 64b.
- Aby mohl procesor fungovat, musí si pamatovat několik čísel (třeba adresu následující instrukce, viz. PCR na minulém slajdu, ale i několik dalších čísel).
- Tato čísla si procesor ukládá do takzvaných registrů.
  - Délka registrů typicky odpovídá velikosti adres.
    - To není bezpodmínečně nutné, ale VELMI žádoucí.
  - Počet registrů může být v každém procesoru jiný, z ničeho nevyplývá (typicky to bývají malé desítky).

Jak vypadají instrukce?

- Toto je velmi zjednodušený **příklad** (takže ne dogma) architektury, která má 32b (=4B) instrukce i adresy:
  - Procesor má třeba 16 registrů (to z ničeho nevyplývá).
    - Registry jsou pojmenované r0 až r15 (například).
  - Každý registr má délku 32 bitů (kvůli velikosti adres).
  - Prvních 12b instrukce určuje, která instrukce to je.
    - Nemusí to být 12, to je dáno architekturou (příklad).
    - Procesor tedy může mít maximálně 4,096 instrukcí (2^{12}).
  - Dalších 20b instrukce je chápáno jako 5 čtveřic bitů; každá udává registr, se kterým instrukce pracuje.
    - 20b proto, že celkem má instrukce 32b, ale prvních 12b už je využito jako identifikátor instrukce.
    - 4b na identifikaci registrů stačí, protože jich je 16 (2^4).

Příklad instrukce ADD

- Příklad pro architekturu z minulého slajdu.
  - Instrukce ADD sčítá dvě čísla uložená v registrech a výsledek ukládá do dalšího registru.
    - Prvních 12 bitů tvoří číslo, které říká procesoru, že se jedná o instrukci ADD (např. 39 ~ 000000100111).
    - Další 4 bity udávají, ve kterém registru je první sčítanec, kdyby to byl třeba r8, pak by tyto bity byly 1000.
    - Následující 4b označují druhý sčítanec (př. r7 ~ 0111).
    - Pak následují 4b na výsledek, například r0, tedy 0000.
    - Poslední dvě čtveřice bitů instrukce nevyužívá (to je dáno tím, že je to instrukce ADD, která potřebuje jen 3).
    - Celá instrukce: 00000010011110000111100000000000
    - Jsou-li tedy na adrese v PCR právě tyto bity, procesor v dalším kroku sečte čísla v r7 a r8, výsledek uloží do r0.

Příklad instrukce ADDM

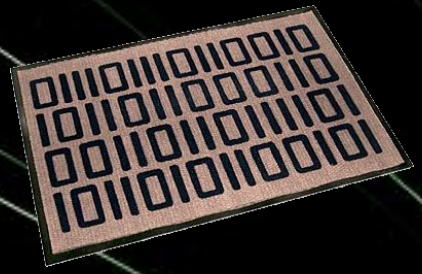
- Příklad pro architekturu z předminulého slajdu.
  - Instrukce ADDM sčítá dvě čísla, která vezme z adresy paměti, která je uložena v registrech a výsledek uloží opět do paměti na adresu označenou registrem.
    - Parametry instrukce jsou stejné (jako na minulém slajdu) jen její číslo musí být jiné než 39, protože to už patří ADD. Řekněme, že identifikátor ADDM je 40.
    - Tato instrukce **nevnímá obsah registrů jako čísla, ale jako adresy**, takže přečte číslo z adresy, která je v registru r8, k tomu přičte číslo, které přečte z adresy, která je v r7 a součet uloží do adresy, která je v r0.
    - Jestli procesor sečte čísla v registrech, nebo je použije jako adresy (nebo s nimi udělá něco jiného), záleží na instrukci (každá instrukce dělá něco jiného).

Příklady instrukcí v paměti

- To, co procesor dělá, záleží na tom, co mu vrátí paměť z adresy, která je uložena v PCR.
- Vrátí-li 00000010011111000011100000000000 (viz. předminulý slajd), udělá právě to, co bylo na předminulém slajdu.
- Vrátí-li 00000010100010000011100000000000, provede procesor ADDM místo ADD (změnili jsme číslo 39 na číslo 40, 39 je ADD, 40 je ADDM).
- 00000010011111010111001000000000 je ADD, ale sčítá r13 (1101) a r7, výsledek ukládá do r2 (0010).
- 00000010011101111011101110000000 vynásobí r7 dvěma. Proč?

Malá rekapitulace

- Posledních šest slajdů může působit zmateně. Pro ujasnění je třeba **POCHOPIT** následující:
  - Proč se do n bitů vejde 2^n stavů, co jsou to vlastně stavy a proč se na ně můžeme dívat jako na čísla.
    - K tomu pomůže představa zámku na kolo, který má 4 pozice – heslo může mít 10^4 stavů (deset proto, že zámek je v desítkové soustavě, tedy čísla 0000 až 9999).
  - Identifikátor čehokoliv (registru, instrukce, adresy, ….) musí být tak dlouhý, aby měl tolik stavů, kolik věcí daného typu chceme identifikovat (rozlišit od sebe).
    - Chcete-li od sebe odlišit 12 krabiček, můžete je očíslovat 0 až 11. V desítkové soustavě stačí dvě cifry ($10^2 \geq 12$), ve dvojkové potřebujeme 4 cifry ($2^4 \geq 12$).



Společná paměť na instrukce i data

- **Jedním z nejzákladnějších rysů von Neumannovy architektury je to, že instrukce i data se ukládají do společné (operační) paměti!!**
  - To nebylo na začátku 20. století vůbec samozřejmé, paměti na instrukce a data byly fyzicky oddělené.
- Jak se rozeznají instrukce od ostatních dat?
  - Jen interpretací. Je-li do PCR vložena adresa, na které instrukce nejsou, procesor začne data zpracovávat tak, jako kdyby to instrukce byly, nepozná rozdíl.
  - To může být nebezpečné, protože někdo (útočník) může taková data procesoru „podstrčit“ schválně.
    - Procesor nemá šanci to poznat (protože je to jen hloupý čip), ochranu musí zajistit operační systém.

Řadič

- Hlavní úlohou řadiče je synchronizace komunikace ostatních zařízení po sběrnicích.
- Sběrnice jsou dráty, po kterých si zařízení posílají informace zakódované do binární soustavy (jedniček a nul), přičemž jednička je reprezentována proudem, nula pak jeho absencí.
  - Z toho logicky plyne, že komunikace nemůže probíhat oběma směry současně a také to, že více komunikací probíhajících na stejných drátech se vzájemně ruší.
- Od řadiče vedou dráty ke všem zařízením, kterými dává signály, jak a kdy mohou po sběrnicích komunikovat (tím řadič celou komunikaci řídí).

Aritmeticko-logická jednotka (ALU)

- ALU je část procesoru, ve které probíhají veškeré výpočty, kterých procesor umí mnoho druhů:
  - Logické operace (vyhodnocování logických výrazů).
  - Aritmetické operace nad celými čísly.
  - Aritmetické operace nad desetinnými čísly.
  - Mnohé další (každý procesor umí něco jiného).
- Některé operace trvají déle než jeden tik hodin.
  - To mimo jiné znamená, že skutečná výpočetní síla procesoru nezávisí jen na jeho taktovací frekvenci.
    - 2GHz procesor může být rychlejší než 3Ghz procesor.
- Operace jsou prováděny nad daty, která se načtou z paměti a výsledky se případně zase do paměti uloží.

Vstup a výstup

- Pro vstup(ní) a výstup(ní zařízení) se používá zkratka I/O, což jsou první písmena slov input/output.
- Ze vstupních zařízení je možné číst nějaká data.
  - Klávesnice, myš, scanner, mikrofon, CD-ROM, ….
- Na výstupní zařízení lze data ukládat/zapisovat.
  - Monitor, tiskárna, reproduktory, data projektor, ….
- Existují i zařízení vstupně-výstupní.
  - Síťová karta, pevný disk, CD-RW, dotykový display, ….
- K I/O zařízením procesor přistupuje stejně jako k paměti (každé zařízení má nějaký rozsah adres).
  - Ze vstupních zařízení lze číst (předhodí se jim adresa a vypadnou data), na výstupní ukládat, na v-v obojí.

Buffery zařízení

- Jakým způsobem funguje adresování u zařízení?
- Každé zařízení v sobě má tzv. buffer, což je (většinou malá) paměť. Z té lze číst a/nebo do ní zapisovat.
- Tyto paměti (v zařízeních) neslouží k uchovávání dat, ale ke komunikaci se zařízeními, například:
  - Grafická karta v sobě obsahuje paměť, kam když se zapíší data, přímo se zobrazí na monitoru. Monitor se skládá ze čtverečků (pixelů), každý má svou barvu a adresu, takže zápis čísla na adresu mění barvu pixelu.
  - Klávesnice má v sobě (výrazně menší) paměť, do které ukládá několik naposledy stisknutých kláves. Chce-li procesor zjistit, které klávesy byly stisknuty, přečte data z odpovídajících adres.

Hardwarová neměnnost počítače

- **Důležitým rysem von Neumannovy architektury je, že se struktura počítače nemění pro různé typy zadaných úloh, které má počítač řešit!!**
- Dříve bylo běžné, že pokud měl počítač řešit nějakou úlohu, bylo jej třeba fyzicky upravit (programovalo se přímo propojováním nějakých kabelů).
- Podle von Neumannovy architektury se programuje pouze na úrovni software, počítač zůstává stejný.
- Výhodou von Neumannovských počítačů je hlavně jejich univerzalita (ta je dána programovatelností).
- Na druhou stranu hardwarové řešení problému bývá výrazně (řádově) rychlejší než softwarové.

Rekapitulace

- Měli byste chápat, jak funguje paměť a adresování, což je základ pro pochopení čehokoliv dalšího.
- Minimální představa fungování procesoru zahrnuje alespoň porozumění toku instrukcí (PCR, ….).
- Pokročilejší představa fungování procesoru znamená pochopení, že pokud k architektuře z příkladu dostanete seznam instrukcí (s popisem, co dělají), obsah registrů procesoru (včetně PCR), budete schopni simulovat jeho běh krok za krokem.
- Fungování řadiče je složité, nad rámec přednášky, ale měli byste alespoň chápat, k čemu tam vůbec je.
- Na ALU, vstupu a výstupu není nic až tak složitého.