# Úvod

## Schéma počítače

* Harvardská architektura
  + CPU – vykonává instrukce
  + Kódová paměť – je v ní uložen program s instrukcemi pro CPU
  + Datová paměť – proměnné programu
* Von-Neumanovská architektura
  + CPU
  + Paměť
    - Slouží zároveň jako kódová i datová
* Běžné počítače dnes používají spíš Von- Neumanovská
* Harvardská se prosazuje spíše na mikročipech → je jednodušší na implementaci

## Start počítače

* CPU startup vector – odtud procesor začíná vykonávat instrukce
  + Natvrdo v CPU (např. 0xFFFFFFF0)
  + Obvykle JMP někam, kde se instrukcí vejde více
* Firmware ROM
  + Interní paměť obvykle na základové desce
  + Test a konfigurace hw
  + Mapování adres zařízení
  + Hledání užitečného sw
    - Obvykle boot sector na pevném disku, v něm tzv. bootloader → spouštní kernel OS
  + Implementuje funkce nutné pro bootování

# Přenos informace v počítači

## Analogový přenos

* Každé možné hodnotně dat přiřazuji konkrétní napětí (nebo frekvenci, …)
* Problémy s rozsahem, v praxi lze reprezentovat analogově jen omezenou množinu dat
* Např. napětí je ovlivňováno délkou vodiče, teplotou, materiálem, ostatními vodiči

## Digitální přenos

* Samotná informace se kóduje pomocí malé množiny hodnoty, které přenáším analogově
* V počítači 1/0 → bit informace
* Obvykle reprezentovány napětím v rozsahu 0-5 V
  + Mezi nimi tzv. shadow zone, hodnota v tomto intervalu se nijak neinterpretuje
* Příklad na rozsahu 0-5 V
  + 5-3 V odpovídá 1
  + 3-2 V shadow zone
  + 2-0 V odpovídá 0
* Není podmínkou, aby 1 byla vždy vyšší napětí než 0
* Napětí se vždy měří vůči referenčnímu vodiči – GND

## Sériový přenos

* Přenos více bitů v sérii za sebou
* Interpretace dat
  + MSB nebo LSB
* Délka jednoho bitu musí být přesně stanovena
  + Přenosová rychlost – baud (symboly/sekunda)
* Délka jednoho bitu musí být přesně stanovena
  + HW hodiny se po čase rozejdou
* Řešení přenosu
  + Idle stav
    - Na začátku je linka v idle stavu
    - Vysílání začíná tzv. rising edge – snadno se detekuje
    - Dojde k synchronizaci hodin – lze ovšem přenést omezené množství bitů, než se hodiny rozejdou
    - Obvykle start bit, datových 8 bitů, stop bit → přenos 8 b dat na 10 b
    - Používá jej RS232 linka
  + Hodinový signál
    - Další tzv. referenční vodič s digitálním hodinovým signálem
    - „Diktuje“, kdy na datovém vodiči číst
    - Neefektivní, obvykle detekuji jen rising edge → poloviční rychlost
      * Kde je rychlost nutná, lze detekovat i falling edge, ovšem hw náročné → DDR (double data rate)
    - Používá I2C
  + Clock recovery (průběžná korekce hodin)
    - Pokud se v datech dostatečně často střídají 1/0, lze hodiny synchronizovat podle samotného signálu → tzv. clock-recovery
    - Z 8 b udělám 10 b takové, že se pravidelně střídají 1/0
    - Používá jej USB, PCIe

## Typy přenosu

* Half-duplexní linka
  + 1 datový vodič – zařízení se v přenosu střídají
  + Komplikované
  + Nikdy nelze posílat najednou oběma směry
* Full-duplexní linka
  + 2 nezávislé simplexí linky
  + Např. RS-232
    - Navíc má speciální vodič pro out-of-band signály
      * Sdělení nutné informace uprostřed přenosu jiných dat
* Point-to-point linka
  + 1 master a 1 slave
  + Nepraktické, jeden procesor by musel mít mnoho vstupů a výstupů
* Multidrop/bus (sběrnice)
  + 1 master a mnoho slaves
  + Typicky procesor je připojen jednou linkou k sběrnici, na kterou je napojen ostatní hw
  + Jednotlivá zařízení jsou rozlišena adresou
    - Rozsah těchto adres = adresový prostor
  + Linky jsou obvykle Half-duplexní bez floating stavu

## Komunikační protokol

* Dohoda, jak přenos vypadá
* Problém starších zařízení – jsou 7b
* Řešení je tzv. řadič který přijímá data od zařízení a zpracovává je a předává je v jiném formátu dál

## Sběrnice

* Bus interface
  + Formátuje a předává data sběrnici

### PCIe

* Sériová sběrnice
* Jsou na ni napojena všechna zařízení, na paket reagují jen ta, pro která je určený
* Dedikované druhy paketů
  + Memory write, Memory ready
  + Reaguje na ně pouze memory controller
  + Neobsahují adresu zařízení, ale místa v paměti
* Součástí paketu je i adresa procesoru, pokud na něj očekává odpověď

### USB

* Čtyři piny
* Dva prostřední slouží pro digitální diferenciální přenos
  + Diferenciální přenos – jeden z vodičů je referenční, tj. napětí na druhém měřím vůči prvnímu
* Dva krajní pro napájení (5V, GND)
* Single master

### I2C

* Multi master
* Dva vodiče
  + SDA – seriál data
  + SCL – seriál clock
    - „Tiká“, jen pokud probíhá přenos
* 9 bitů na byte informace
  + 8 bitů data (MSB-first)
  + 1 acknowledgement bit
    - 0 = ack = ano
    - 1 = nak = ne

# Reprezentace čísel

## Záporná čísla

* Nejjednodušším řešením zvolit MSB jako znaménko
  + Ale nefungují klasické operace pro bitové sčítání → nepoužitelné

### Jednotkový doplněk

* U záporných čísel se prohodí 1 a 0
* Pak funguje sčítání atd.
* Problém je, že existují dvě nuly

### Dvojkový doplněk

* Stejný jako jednotkový doplněk, jen se navíc přičte 1
* Ve výsledku je MSB stejně znaménkový
* Řeší problém s nulami, ale vytváří anti symetrii v hodnotách
* Rozsah -2n-1 až 2n-1 – 1

### Algoritmus pro reprezentaci záporného čísla:

* Převrátím hodnoty kladné reprezentace
* Přičtu jedničku

## Čísla v Pythonu

* Reprezentována v poli o 32b hodnotách → v podstatě neomezená přesnost
* K tomu je uloženo další 32b číslo (v C), které říká, kolik b zapírá číslo v Pythonu
  + Tím je omezena přesnost na 232-1
* Příklad pro x=5 v Pythonu
  + 4B/8B na adresu (podle architektury) , protože x je v Pythonu jako objekt ukazatele na hodnotu
  + N \* 4B na pole hodnot samotného čísla
    - Z 32 bitů je ovšem využito jen 30, protože proměnná v Pythonu je signed → 31 bitů a proměnná v C je opět signed → 30 bitů
  + 4B na hodnotu, která určuje, kolik bitů je pro číslo využito
  + 4B určující typ proměnné (číslo)
  + 4B pro tzv. reference-couting – kolik proměnných ukazuje na danou hodnotu v paměti
    - Využívá jej GC
* Pro uložení 32bitového čísla v Pythonu je tedy třeba cca. 24B paměti
* Každou operací na čísle vzniká nový objekt (např. x += 1)

## Operace s čísly

* Truncation
  + Useknutí cifer, které se do menšího datového typu nevejdou
    - U signed typů nemusí fungovat – může se změnit znaménko
* Extension
  + Zero extension – doplnění nul na prázdná místa
  + Signed extension – doplnění o MSB (nutné u signed typů)

## Reálná čísla

### Fixed-point reprezentace

* Pevný počet bitů pro části před a za desetinou čárkou
  + Aritmetika funguje stejně jako u celých čísel
* Problém při operacích s hodně velkými nebo malými čísly

### Floating-point reprezentace

* Čísla v normalizovaném formátu sign\*1.010011…\*2exp
* Základ exponentu = mantisa
  + Na prvním místě vždy 1 → neukládáme (ale musíme s ní počítat!)
* Exponent v bias reprezentaci (bias = hodnota kterou přičítáme)
  + (-n,n) → (0,2n+1)
* Obvykle implementována hw, ovšem stejně pomalejší než celá čísla
* IEEE 754 je standard pro implementaci floating-point čísel
  + 32b – single je 1b/8b/23b (sign/exponent/mantisa)
  + 64b – double je 1b/11b/52b
* Problémy
  + 0.1 nelze reprezentovat ve floating point přesně → zaokrohlouvání
  + Nikdy neporovnávat floating point čísla přesně
  + V bankovnictví nedržím hodnotu v korunách, ale v haléřích
* Speciální hodnoty
  + 0 jsou samé 0 → proto využíváme exponent v bias reprezentaci
  + Samé 1 v exponentu → speciální hodnoty (∞, NaN)
* Sčítání
  + Nutná normalizace, tj. zarovnání desetinných čárek

# Paměť

* Paměťový prostor – rozsah adres v paměti
  + Obvykle naddimenzován pro případný upgrade hw
    - Např. pro 200B paměti může být 8b, ale klidně i 16b
  + Např. 32bitový adresový prostor zvládne max. 4GB paměti
* Slovo
  + Správná definice – jednotka přenosu
    - Pro n-bitové zařízení má slovo n bitů
  + Špatná obvyklá definice – 16 b

### Jednotky

* 1(N)B = 1024(N-1)B

## Paměti RAM

* Random Access memory
  + Teoreticky lze přistupovat k libovolné hodnotě ve stjném čase
  + Prakticky ovšem ne
    - Sekvenční přístupy jsou rychlejší
    - Obrácené sekvenční jsou něco mezi
    - Náhodné přístupy jsou pomalejší
* Obvykle je R/W a volatile (tj. po odpojení napájení ztratí data)

### SRAM (Static RAM)

* 1b – 4 až 6 tranzistorů
* Nízká kapacita (řádově MB)
* Rychlost přístupu obvykle 10-100GB/s
* Obvykle využívána pro registry zařízení

### DRAM (Dynamic RAM)

* 1b – 1 tranzistor a 1 kondenzátor
* Levnější a větší než SRAM, ovšem pomalejší (kvůli kondenzátoru)
* Hodnoty si pamatuje krátce (kvůli kondenzátoru)
  + Nutné paměť průběžně obnovovat
    - Tj. s určitou frekvencí se paměť neustále přepisuje
* Rychlost přístupu obvykle 1-10GB/s

## Endianita dat

* Little-endian (LSB first)
  + Tj. na místo v paměti s nejnižší adresou se uloží LSB
  + Mnemotechnická pomůcka – LLL (Little-Last-Lowest)
* Big-endian (MSB first)
  + Tj. na místo v paměti s nejnižší adresou se uloží MSB
* Každé zařízení má určenou endianitu → při přenosu dat nutné řešit kompatibilitu
* Dnes obvykle Little-endian
* Pozor – endianita bitů a bytů může být rozdílná!

## Paměti ROM

* Jsou non-volatile – hodnoty zůstanou po vypnutí počítače
* ROM
  + Pouze jeden zápis (u výrobce) a neomezené čtení
* PROM
  + Pouze jeden zápis, ovšem doma (diody), neomezené čtení
* EPROM
  + Možné mazání, ovšem UV zářením → nepraktické
* EEPROM
  + Mazání elektrickým proudem
  + Pomalé
  + Praktický je počat zápisů omezen
  + Adresují se po jednotlivých bytech
  + Flash paměti
    - Zápis a čtení po velkých blocích
    - Relativně rychlí přístup při sekvenčním čtení
    - SSD disky, flashky
  + HDD
    - Data uložená v magnetické orientaci kovu
    - Plotna rozděleně na sektory – dnes 4kB
    - Jednotlivé plotny čtou čtecí hlavy – ty se pohybují všechny stejně
      * Plotny se otáčí
    - Adresace hodnot je CHS (cylinder/head/sector)
    - Levné a velkokapacitní úložiště
    - Relativně pomalé, vhodné pouze pro sekvenční přístup
  + CD/DVD/BluRay
    - Optické
    - Nejsou vhodné pro archivační účely → po čase se vrátí do původního stavu
    - Data ukládána do spirály
    - Adresace hodnot LBA (linear block adressing)
      * Není trojic, jednodušší pro implementaci v programovacím jazyce
      * Pomalé

### Řadiče

* Registry
  + Adresový – zapisuji adresu, kterou chci zapisovat nebo číst
  + Příkazový – zapisují příkaz (čtení, zápis, mazání)
  + Buffer – vkládám nebo čtu data
  + Info – počet sektorů, velikost sektoru, …
* V praxi se používá pouze LBA, HDD mají vlastní řadič, který převádí z LBA na CHS

### Adresování souborů

* Offset (v B) od začátku paměti
  + Na začátku uložena metadata, data OS apod.
* Base adress – kam můžeme ukládat data
* Meta data – data o datech
  + Ukládá se info o souboru a čísla sektorů, kde se soubor nachází
    - Velikost souboru
  + Fragmentace – uložení souboru do více sektorů
* OS vytvoří abstrakci na disky a poskytuje jednotné API pro zápis, čtení a práci s metadaty

## Memory/mapped I/O

* Využívání jednoho adresového prostoru jak pro paměť, tak pro I/O zařízení
  + Pro I/O jsou mapovány nějaké části paměti, které jsou volné
  + Je potřeba, aby adresy byly unikátní

# CPU

* Instrukce – posloupnost n bitů
* Instrukční sada – množina instrukcí podporovaných konkrétním CPU
* Strojový kód – posloupnost instrukcí
* Instruction pointer (IP) – ukazatel do paměti na právě prováděnou instrukci
  + U vícebitové instrukce ukazuje na první bit
  + Jeho velikost odpovídá architektuře (tj. 32bitový nebo 64bitový)
* Operation code (opcode) – typ instrukce
  + Podle toho se interpretují argumenty
  + Obvykle 1B-2B

## Assembler

* Abstrakce na strojovým kódem
* Namísto instrukcí ve formě bytů se využívají klíčová slova
  + Např. JMP, LDA, STA

## Registry CPU

* PC – program counter
  + Ukazuje na místo v paměti, odkud se aktuálně čtou instrukce

### Příznakové registry

* Příznak = flag = 1 bit informace
* Zero – jestli poslední výsledek byla 0
* Sign – záporný výsledek operace
* Carry (přenos) – z nějakých operací (např. sčítání)
* Operace to definují různě (podle toho, co potřebují)

### Akumulátorové registry

* V akumulátorovém registru se provádějí aritmetické operace
* Před začátkem sčítání je nutné nastavit carry příznak na 0
* Při sčítání čísel větších, než které podporuje architektura využití carry příznaku

## Rychlost operací

* Rychlost se měří v operacích za sekundu → hz
* Základní bitové a aritmetické operace se zvládají za 1 instrukci
* Čtení a ukládání do paměti je pomalejší

## Násobení a dělení

* Nejobtížnější operace na implementaci
* Násobení obvykle 10x pomalejší než sčítání, dělení až 100x
* Některé architektury nemají tyto operace implementované hw → nutné sw – ještě pomalejší
* Dělení a násobení mocninami dvojky lze nahradit SHL a SHR
  + Nefunguje u signed čísel

# Reprezentace dat

## Reprezentace obrazu

* Bitmapy
* Indexováno (x,y), kde (0,0) je obvykle vlevo nahoře
* Jednotka obrazu – pixel
  + Bit depth (bitová hloubka)
    - Kolik bitů reprezentuje jeden pixel (barvu)
    - Čípky v oku → RGB
    - U barevného obrázku standartně 4B na pixel
      * 3B na barvy → 24bitová hloubka
      * 1B na alpha kanál – průhlednost
* Ukládání do paměti
  + Nutné krom pixelů uložit i bitovou hloubku, výšku, šířku, pořadí barev (endianita), offset dat
  + Obvykle určeno fomátem

## Reprezentace textu

* String – posloupnost znaků
* Grafém – nejmenší jednotka psaného textu
* Kódování
  + Převod znaku na kód (číslo) a zpět
  + Převod kódu do binární reprezentace a zpět
    - Tj. např. počet bytů jednoho znaku
  + Ukládání do paměti po směru čtení
  + Z historických důvodů nemá metadata – problém určení kódování

### ASCII

* 7b kódování (0-127)
* Znaky abecedy a číslice jsou blízko sebe – praktické
* Pouze anglické znaky
* Rozšířeno pro různé části světa
  + Win1250 – Windowsové kódování pro východní Evropu

### Unicode

* Standardizace – všechny jazyky, všechny symboly, ~~žádné problémy~~
* 0-127 odpovídá ASCII – kompatibilní
* 128-$FFFF – běžně znaky
* Problém – určuje pouze převod textu na kódy, nikoliv kódů na binární data → vznik různých
  + UTF-32 – každý znak je 4B
    - 2 verze UTF32 LE a UTF32 BE
    - Neúsporné
  + UTF-16 - proměnlivá délka znaku (2B/4B)
    - 4B… surrogates (náhradníci): pro určité hodnoty prvních 2B musí být přečteny druhé 2B
    - Interpretace speciálním algoritmem, který zahrnuje pravidla kódování
    - Nelze přesně říct, kolik znaků je v souboru s tímhle kódováním uloženo
    - Opět varianty LE a BE
  + UTF-8 – 1B, 2B, 3B, 4B znaky
    - 1B – první bit je 00
    - nB – prvních n bitů je 1, ten za tím 0 (pro n = 2 je to 110)
    - Každý další byt začíná 10 – lehce lze zjistit, že jsou součástí nějakého znaku
    - Neřeší endianitu
    - Populární na internetu

### Rasterizace textu

* + Převod stringu na bitmapu
  + Potřeba odřádkování (problémy)
    - Původně znaky CR+LF
    - MacOS používá pouze CR
    - Unix pouze LF
    - Unicode přináší nové znaky – LS a PS