11. Reprezentace čísel v paměti

1. Datové typy

- binární soustava je poziční soustva o základu 2, využívaná v počítačích
- hexadecimální soustava, základ 16 (1-9 + A-F), rychlá konverze mezi dvojkovou a šestnáctkovou soustavou zapříčinila, že je velice užívaná v inf, adresování buněk v paměti
  - *nibbel* - jeden znak z šestnáctkové soustavy; každý z šestnácti znaků převedeme na čtyřbitové číslo a potom jednoduše složíme
- reprezentace celých čísel:
  - **Unsigned**: klasika, 8 bitů, čísla od 0 do 255
  - **Signed**:
    - *bias*: posun, neboli budeme mít rozsah od -128 do 127, takže unsigned 128 bude 0
    - *sign bit*: první bit z osmi se obětuje na určení, zda jde o kladné - 0, či záporné - 1 číslo, problém je v tom, že máme dvě čísla pro nulu, nejednoznačnost, 00000000 = 10000000
    - *one's complement*: jedničkový doplněk řeší problém sign bitu, a to, že respektuje uspořádanost binárních čísel, neboli -127 < -126, ale i v binárce, jednoduše zneguji kladnou část čísla, abych obdržel číslo záporné (5 - 00000101 a -5 - 11111010)
    - *two's complement*: dvojkový doplněk, v podstatě vezmeme jednočkový komplement a přičteme 1, zrušíme nejednoznačnost nuly
  - pakliže potřebujeme reprezentovat větší čísla, tak si prostě alokujeme více paměti, takže vznikají proměnné typu *long, char, …*
- reprezentace desetinných čísel:
  - nás bude zajímat reprezentace typu float, konkrétně binary32:
  - princip:
    1. mějme desitinné čílo, třeba 12.375
    2. rosekneme jej na celoo a desetinnou část
    3. desetinnou konvertujeme do dvojkové soustavy (násobením dvěma a zapisováním overflow bitu), takže 0.011
    4. převede celou část do binárky, takže 1100
    5. sečteme a posuneme o exponent, tak, abychom měli pouze jeden bit před floating pointem, $1100.011 \rightarrow 1.100011 \times 2^3$
    6. k exponentu přičteme 127 a konvertujeme do binarky, takže 130 $\rightarrow$ 10000010
    7. nyní první bit je *sign*, takže ten je u nás 0, potom je *exponent* - 8 bitů, nakonec je *mantissa*, ta je složena z desetinné části čísla z 5. kroku (100011) a zbytek napravo je doplněn 0, tak, aby celkové číslo bylo 32 bitů dlouhé

2. Alokace paměti

- jedná se o proces rezervace operační paměti, kterou daný program ve svém běhu bude potřebovat
- za alokaci je zodpovědný *memory management*
- většina operačních systémů umí nejen alokovat pamět za běhu (výhody multijaderných platforem - thread), ale i realokovat a vracet zpět systému
- rozlišujeme staticky a dynamicky alokovanou paměť, statická se na za-

čátku alokuje a potom už je její velikost až do konce stejná, zatímco dynamická se může přizpůsobit paměťovým nárokům, když paměť není potřebná, tak ji uvolní pro systém a tak zvýší rychlost programu
    - má to ale nevýhody, zakládání dynamického pole je časově náročnější než vytváření statického
    - statické je rychlejší v přístupu k uloženým datům
- *dvojstupňová paměť*: dynamické alokování probíhá za pomocí pointerů, jelikož není jisté, kolik paměti bude zapotřebí, nemůže být datový prostor adresován přímo, pointery uchovávají adresu v paměti a můžou třeba označovat začátek alokované paměti
- zatímco paměť alokovaná staticky je uvolněná po přesně stanovené době, dynamicky alokovanou paměť musí uvolnit buď program, nebo příslušný garbage collector, algoritmus pro správu paměti

3. Endianita

- neboli byte order
- způsob uložení čísel v operační paměti počítače, definuje, v jakém pořadí se uloží jednotlivé bajty číselného datového typu