MAGAZÍN
KATEDRY INFORMATIKY

číslo 6 | prosinec 2016 Univerzita Palackého v Olomouci

Úvodní slovo

Vážení čtenáři,
rádi bychom se s Vámi v tomto vydání našeho magazínu opět podělili o informace z naší katedry a ze světa informatiky vůbec.

Jednou z novinek, kterou tento semestr přinesl, je korespondenční seminář Olinx, který si klade za cíl popularizovat informatiku mezi středoškolskými studenty. V každém semestru lidé z naší katedry zpracují dvě témata formou série krátkých článků. Účastníci semináře, kteří vypracují řešení zadaných úkolů, mají šanci získat hodnotné ceny. V prvním semestru tohoto několikaletého projektu jsme se zaměřili na šifrování zpráv a programovací jazyk C.

Hlavním tématem tohoto vydání magazínu je voxelová grafika. Téma zpracoval náš doktorand Pavel Procházka. V jeho článku se dočtete o způsobech reprezentace trojrozměrných dat a tom, jak vhodná data získat, například lékařským skenováním. V dalším článku se pak Jan Outrata věnuje triatlonovém klubu, který působí při naší katedře. Závěrem máme jako obvykle hádanku i s podrobným vysvětlením.

Za všechny, co tento magazín připravovali, Vám příjemné prožití vánočních svátků přeje

Petr Krajča

redaktor Magazínu Katedry informatiky Univerzity Palackého v Olomouci

Obsah

Voxelová grafika

Pavel Procházka

Voxelová grafika je zobecněním rastrové grafiky do třetí dimenze, které představuje zajímavou alternativu k dominující 3D polygonové grafice. Základním elementem voxelové grafiky je nepřekvapivě voxel – volumetric element, barevný nebo průhledný bod v trojrozměrné mřížce. Voxelová grafika je tudíž schopná zachytit nejen povrch modelu, ale také jeho vnitřek. Tím se otevírají možnosti přirozeného využití volumetrického přístupu. Voxelová grafika nachází využití v reprezentaci a zobrazování výstupů z výpočetní tomografie, umožňuje reprezentaci modelů pro stále rozšířenější 3D tisk, použití nachází také v kinematografii a počítačových hrách. Jak už to ovšem bývá, platí i zde pravidlo „něco za něco“. Oproti polygonové grafice voxelová grafika obsahuje více informace, proto modely bývají mnohem větší. Hlavní výhodou je naopak snadná úprava modelů prostým přidáváním nebo odebíráním voxelů, toho se využívá hlavně v interaktivních hrách pro efektní docílení destrukce, případně k simulaci mraků a mlhy. Do budoucna lze počítat s pokroky ve vývoji takzvaných volumetrických displejů ve stylu sci-fi „holografického“ zobrazení, zde voxelová grafika jistě najde své uplatnění.

Reprezentace voxelové grafiky

Jak už bylo řečeno, voxelová grafika je zobecněním rastrové grafiky do třetí dimenze. Na (diskrétní) obraz se lze tedy dívat jako na zobrazení $D_x \times D_y \times D_z \to C$, kde $D_x = \{0, 1, \ldots, n\}, D_y = \{0, 1, \ldots, m\}, D_y = \{0, 1, \ldots, k\}$ pro nějaká přirozená čísla n, m, k představující velikosti v jednotlivých dimenzích. C je množina barev nějakého barevného modelu, například RGBA. Jednoduchou reprezentací takového obrazu je třídimenzionální mřížka. V běžné rastrové grafice se tento přístup používá celkem běžně – například u formátu `BMP`. Obrázky ve formátu `BMP` nejsou komprimované a jejich načítání je tedy jednoduché. Podstatným rozdílem je fakt, že zatímco počet pixelů obrazu ve 2D roste v závislosti na jeho rozměrech kvadraticky, ve 3D roste počet voxelů kubicky. Takže zatímco obrázek o rozměru 1024×1024 má při 24-bitové reprezentaci pixelu velikost 3MB, pro voxelový obraz rozměru $1024 \times 1024 \times 1024$ potřebujeme 3GB.

Volumetrická data se vyznačují vlastností, které se říká řídkost. Kdyby byl totiž obraz plný tak, jako je převážná většina 2D obrázků, tj. většina voxelů by byla neprůhledná, viděli bychom jenom barevný kvádr. Obsah objektů reprezentovaných ve volumetrických datech je navíc homogení, tj. voxely uvnitř objektů mají stejnou barvu. To dává spolu s řídkostí prostor pro kompresi. Ke kompresi volumetrických dat existují různé přístupy. První z nich pochází z herního Build enginu (Duke Nukem 3D, Blood, Shadow Warrior), kde se využívá tzv. run-length principu. Pokud se v obrazu za sebou nachází sekvence po sobě jdoucích voxelů stejné barvy, zakódujeme je tak, že uvedeme barvu a délku této sekvence. Run-length kódování nemusí vést ke kompresi, pokud data neobsahují dostatečně dlouhé sekvence stejných voxelů. Lze vymyslet příklad, ve kterém jsou komprimovaná data větší než původní, například `WBWB` je nutné zakódovat pomocí `1W1B1W1B`. V reálných implementacích to vypadá tak, že základna kvádru obalujícího model je 2D obrázek, který obsahuje na jednotlivých pozicích odkazy na sloupce zakódované pomocí run-length principu.

Oktalový strom

Oktalový strom (octree) je stromová datová struktura založená na myšlence dělení prostoru skrze středový bod. Každý uzel představuje krychli v prostoru a má 8 potomků. Každý potomek představuje jednu osminu krychle svého rodiče a to tak, že sjednocením všech potomků dostaneme právě krychli rodiče. Listové uzly nesou pouze informaci o barvě voxelu. Pokud známe rozměry a polohu krychle představované kořenovým uzlem, není nutné explicitně udržovat informace o poloze krychlí potomků. Oktalový strom lze jednoduše zkomprimovat. Projdeme celý strom a ověříme, zda mají všichni potomci nějakého uzlu stejnou barvu. Pokud je odpověď ano, sloučíme je do rodiče. Toto opakujeme, dokud se ve stromě takový uzel nachází.

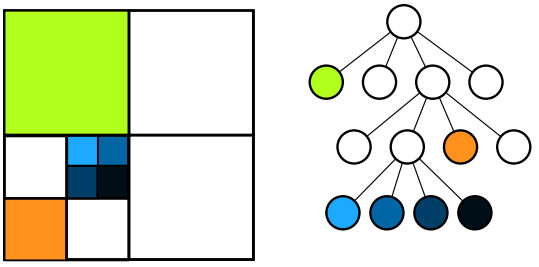
Označíme-li velikost hrany krychle kořene oktalového stromu jako N, snadno vidíme, že výška oktalového stromu je vždy menší nebo rovna $\log_2 N$. V nejhorším případě má oktalový strom kubický počet uzlů (v závislosti na velikosti hrany krychle kořene stromu). Vysvětlení je jednoduché: v každé úrovni $l \in \{0, 1, \ldots, \log_2 N\}$ má oktalový strom 8^l prvků. Celkový počet uzlů takového stromu je pak

$$\sum_{l=0}^{\log_2 N} 8^l = \frac{8^{\log_2 N+1} - 1}{8 - 1} = \frac{8N^3 - 1}{7},$$

protože $8^{\log_2 N} = N^3$.

Výhodou oktalového stromu je snadné odstraňování aliasingu. V každém uzlu totiž lze dopředu spočítat průměrnou barvu krychlí jeho potomků. Takto zakódovaný oktalový strom pak reprezentuje analogii k tzv. mipmapové pyramidě používané k filtraci textur. Další pozoruhodnou výhodou oktalového stromu je potenciálně nekonečná úroveň detailů, které se dá docílit pomocí techniky zvané instancing (odkazy na existující oktalové podstromy) a zpětnými odkazy směřujícími zpátky do stromu. Pomocí zpětných odkazů lze implementovat například fraktály nebo definovat strukturu materiálů s významnou úsporou paměti a potenciálně nekonečnými detaily.

Porovnejme velikosti reprezentace pomocí rastru a pomocí oktalového stromu. Zaručuje reprezentace oktalovým stromem menší velikost než reprezentace rastrovou mřížkou? Odpověď na tuto otázku lze nalézt v předchozím odstavci. Zcela naplněný a nezkomprimovatelný oktalový strom má $\frac{8N^3-1}{7}$ uzlů. Obraz zakódovaný mřížkovou metodou má N^3 voxelů. Protože platí $\frac{8N^3-1}{7} > N^3$ (pro $N > 1$), víme, že i u oktalového stromu může dojít k nevýhodné kompresi. Dále poznamenejme, že nelistový uzel musí být mnohem větší než jednotlivý voxel v mřížce, aby umožňoval pamatovat si odkazy na následníky. Zcela naplněný oktalový strom není samozřejmě jediným příkladem, kdy dojde k nevýhodné kompresi, zkuste vymyslet jiný příklad.

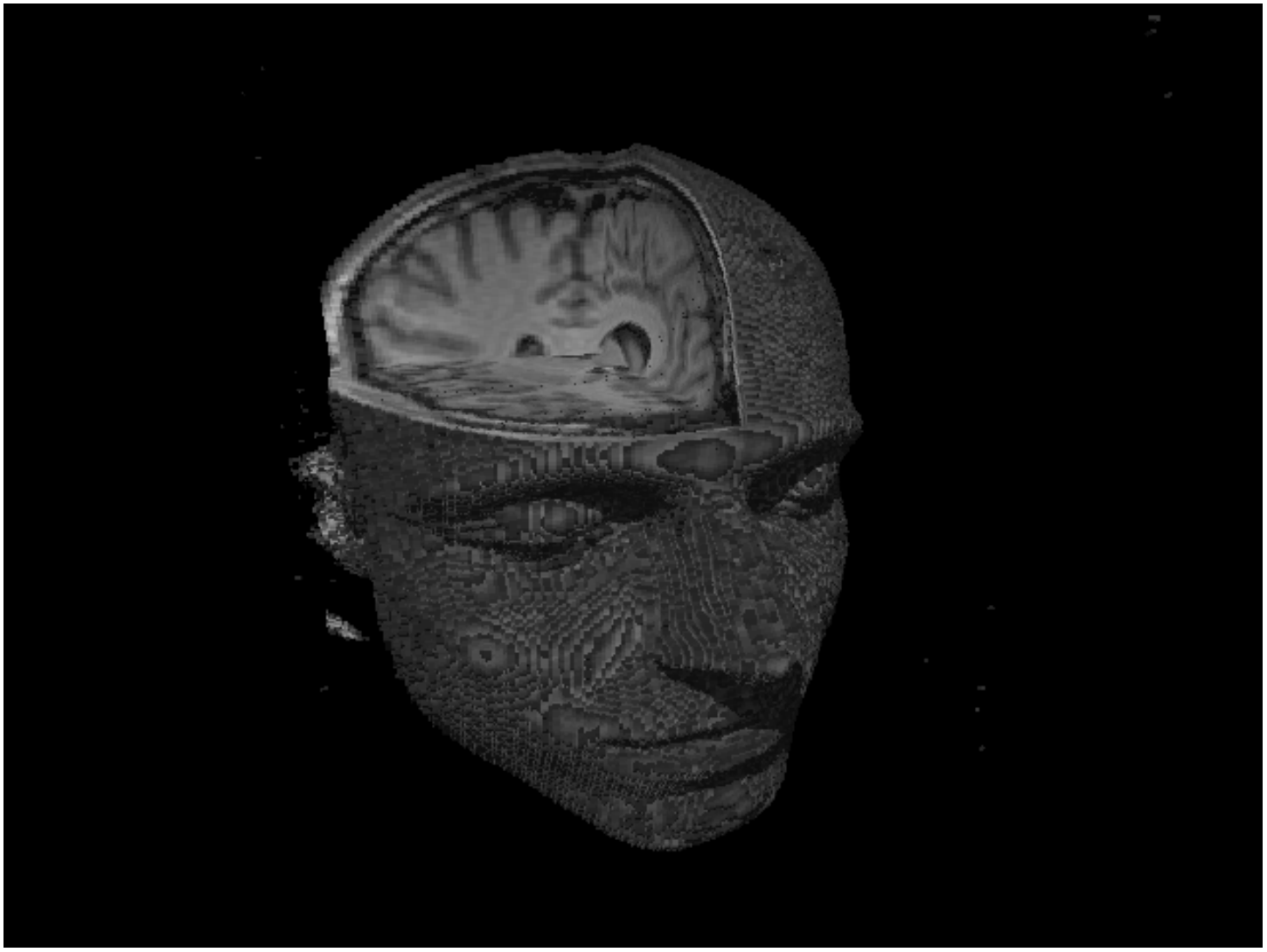


Struktura kvadrantového stromu – analogie oktalového stromu ve 2D.

Dosud nepublikovaným rozšířením oktalového stromu je jeho varianta, ve které může být uzlem stromu buď uzel oktalového stromu nebo mřížkový model příslušné velikosti. Této struktuře pracovně říkám adaptivní oktalový strom. V každém uzlu stromu se při kompresi ověří, zda je podstrom výhodnější reprezentovat pomocí oktalového podstromu nebo pomocí mřížky. Poté se příslušný podstrom převede na výhodnější reprezentaci, tak aby nedocházelo v žádném místě stromu k nevýhodné kompresi. Indukcí lze snadno ukázat, že každý model zakódovaný pomocí adaptivního oktalového stromu je menší nebo roven (co do množství spotřebované paměti) stejnému modelu zakódovanému pomocí mřížky nebo pomocí běžného oktalového stromu. Díky lokální optimalitě v každém uzlu také v praxi vykazuje adaptivní strom zhruba dvoutřetinovou úsporu paměti oproti běžnému oktalovému stromu a velký potenciál v rychlosti vykreslování.

Jak přijít k voxelovým datům?

Dostupnost voxelových dat je velmi omezená. Důvodem je jednak náročný způsob získání reálných obrazů a jednak malé rozšíření příslušného softwaru. Několik pěkných skenů reálných předmětů poskytuje Stanfordská univerzita (`http://graphics.stanford.edu/data/voldata/`). Druhou možností je kreslení modelů



Snímek hlavy autora článku pořízený magnetickou rezonancí.

vlastními silami pomocí řezů, tady postačí běžný editor na zpracování rastrové grafiky, řezy lze také získat z polygonových modelů.

Přirozeným zdrojem by mohla být medicínská data. Na ta se ale bohužel vztahují přísná právní omezení. Z toho důvodu je ideální volbou sken vlastního těla, i když je to volba poněkud dobrodružná. Skenování mi bylo laskavě umožněno v nemocnici U svaté Anny v Brně v roce 2013, kde jsem se zapojil jako dobrovolník do výzkumného projektu zaměřeného na sledování aktivity mozku. Samotný sken byl proveden pomocí magnetické rezonance a s nasazenou EEG čepicí, která měří aktivitu mozku. Příprava na skenování zahrnovala připojení senzorů sledujících činnost srdce (EKG), následné nasazení čepice s elektrodami, které byly nemalým množstvím odolného gelu vodivě spojeny s pokožkou hlavy. Odolnost gelu se projevila hlavně během sundávání čepice. Následné vyšetření bylo provedeno v „tunelu“. Zde byl nejprve proveden NMR snímek, následoval klidový stav, kdy byla sledována aktivita mozku při zavřených očích. NMR je zkratka pro nukleární magnetickou rezonanci pracující na principu interakce atomových jader s magnetickým polem. Poté byl proveden příčný sken a následovala série jednoduchých otázek s otevřenýma očima. Otázky s jistou pravděpodobností obsahovaly gramatické nebo sémantické chyby, případně se jednalo pouze o sekvence znaků `X`. Toto celé trvalo bezmála hodinu za celkové nehybnosti nezbytné pro korektní výstupy. Jinými slovy také informatika může skýtat nečekané příležitosti ke zvýšení hladiny adrenalinu v krvi a nemusí jít jen o neošetřenou výjimku zjištěnou hodinu před odevzdáním zápočtové úlohy.

Triatlonový klub katedry informatiky

Jan Outrata

Ano, Katedra informatiky Univerzity Palackého v Olomouci má triatlonový klub. A jeho členy nejsou pouze pracovníci a studenti katedry. Členové klubu se účastní závodů v triatlonu i jiných sportech a dosahují na nich poměrně slušných výsledků. Klub, za podpory katedry, také každoročně na Sportovní den UP pořádá běh olomouckými parky – Memoriál Jiřího Hronka. Hlavním cílem klubu je propagovat triatlon a sport obecně jako vhodnou doplňkovou aktivitu a inspirovat ke sportu studenty i širokou veřejnost.

Kolem roku 2010 se několik členů katedry informatiky začalo více zajímat o triatlon. Někteří z nich se i již dříve zúčastnili nějakého závodu, triatlonu nebo běžeckého, a inspirovali tím ty ostatní. A tak se na jaře roku 2011 zrodila myšlenka založit na katedře triatlonový klub. A jejím prvním společným, stvrzujícím, závodem nebyl žádný menší než „ikonický" Moraviaman – závod v dlouhém triatlonu distance Ironman (3,8 km plavání, 180 km cyklistika a 42,2 km běh, maraton).

Po této, pro některé první, štafetové ochutnávce závodu v triatlonu, která pro mě osobně byla obrovským sportovním zážitkem, absolvovali závod v následujících letech někteří účastníci i individuálně a stali se tak skutečnými „ironmany" (ve skutečnosti jeden ze zakládajících členů klubu už v té době trojnásobným ironmanem byl). A mezitím spousta dalších závodů, kromě triatlonových hlavně běžeckých, noví členové klubu s dalšími sporty a mimořádnými výsledky a také uspořádání vlastní běžecké akce. Tolik stručně historie klubu.

V současnosti klub sestává z devíti členů, k původním šesti pracovníkům a studentům katedry (čtyřem mužům a dvěma ženám) přibyli za poslední tři roky tři noví mimo katedru (dvě ženy a jeden muž). Ročníky členů 1971 až 1996. Klub je otevřený a jeho členem se může stát každý zájemce o triatlon nebo některou jeho disciplínu, stačí bez přestávky za sebou uplavat 200 metrů, ujet na kole 20 km a uběhnout 5 km – dobře, tak ne, to není podmínka nutná, pouze postačující.

Většina členů klubu se aktivně věnuje triatlonu, někteří jen některým jeho disciplínám – tj. plavání, jízdě na kole nebo běhu – nebo i jinému sportu (horský běh, kolo, turistika). Pod jmenovkou klubu se účastníme závodů (na amatérské úrovni), pořádáme sportovní akce a společné tréninky a výlety (cyklistické, turistické, obvykle v Beskydech nebo např. výšlap na Praděd). Triatlon a sport vůbec je pro nás různě velký koníček, kterému se věnujeme podle volného času a chuti, protože nás to baví, klub nemá žádné sponzory. Hlavním cílem klubu jako takového je potom propagovat triatlon a sport obecně jako vhodnou doplňkovou aktivitu a inspirovat ke sportu studenty katedry a univerzity i širokou veřejnost.

Když už jsme u té inspirace, členové klubu mají „na kontě", za dobu existence klubu, již několik pozoruhodných (a také pár zcela mimořádných) sportovních výsledků – když uvážíme, že se sportu věnují jako doplňkové činnosti vedle hlavní pracovní náplně vyučujících a vědeckých pracovníků na vysoké škole nebo vysokoškolských studentů.

Jak již bylo naznačeno výše, čtyři z nich (muži) jsou ironmany, tedy „železnými muži" neboli „železňáky". Jeden navíc dvojnásobným, jeden trojnásobným a je-

den dokonce pětinásobným. Pět (jedna žena) zaběhlo maraton nebo delší běh (i) samostatně, mimo ironmana, někteří vícekrát a tři dokonce horský s plus minus 3 000 nastoupanými a sestoupanými metry (ona jedna členka dokonce více než desetkrát, včetně „legendárních“ LH24 – opakované výběhy na Lysou horu v mrazivých teplotách po 24 hodin nebo dokud „to jde“ – a B7 – Beskydská sedmička, přeběh sedmi beskydských vrcholů, přes 90 km na délku a 5 000 nastoupaných a sestoupaných metrů!). Tři členové (jedna žena, jeden z nich dvakrát) absolvovali Drásala – jeden z nejnáročnějších závodů na horských kolech po hostýnských vrších, s délkou cca 120 km (bikemaraton) a kolem 3 500 nastoupaných metrů. Dva členové klubu (jedna žena) již absolvovali za klub celkem přes 30 závodů a u jednoho to dělá téměř 30 triatlonů. Za celý klub pak evidujeme asi 130 úspěšně dokončených závodů, z toho kolem 40 triatlonů (většinou sprint se zkráceným plaváním a delším během nebo olympijské distance, tj. 0,4/20/6, resp. 1,5/40/10 km). Umístění? Aktuálně 11 třetích, 6 druhých a 8 prvních v pořadí v kategorii (pro všechny závody, nejen triatlony).

Můžeme tedy kvalifikovaně poskytnout pár zkušeností a rad nejen k triatlonu. Zůstáváme stále klubem amatérským vhodným pro začátečníky, nabídnout můžeme společné sportovní aktivity s členy klubu, včetně účastí na závodech.

Nejvýznamnější sportovní akcí pořádanou klubem pro širokou veřejnost je Memoriál Jiřího Hronka – běh olomouckými parky. Tato sportovně-zábavní akce, která primárně není závodem, se koná, za podpory katedry informatiky, od roku 2014 každý rok v polovině května na Sportovní den UP, na počest našeho bývalého kolegy a dlouholetého učitele, vášnivého amatérského sportovce Jirky Hronka. O třetím ročníku Memoriálu jste si mohli přečíst krátký příspěvek v minulém, pátém, čísle Magazínu. Trasy od pohodových 5 km až po maraton (jediný v Olomouci!), účast zdarma, v minulých ročnících účast kolem 80 běžců, jedna z nejpopulárnějších akcí Sportovního dne. Přidáte se? Již čtvrtý ročník se koná 10. května 2017.

A samozřejmě, zvláště když většina členů klubu jsou informatici, máme i webové stránky. A na nich info o členech, zejména jejich závodní výsledky včetně těch výše zmíněných, kalendář plánovaných závodů, galerii fotek a blogové zápisky ze závodů, kontakt na členy a… ovšem, stránku Memoriálu, na které nechybí datum a čas dalšího ročníku, popis trati, seznam přihlášených účastníků či výsledky minulých ročníků. Že jste nepostřehli adresu stránek a současně i „informatické“ jméno našeho klubu?

www.inf.upol.cz/tri

Těšíme se na shledanou na nějakém závodě nebo na Memoriálu JH!

Zapletené operace

Petr Krajča

Dnešní hádanka patří trochu do dávných časů, kdy se programátoři museli potýkat s problémy, na které dnes narazí jen zřídka. Většina dnešních programátorů se s nimi pravděpodobně nesetká vůbec. Proto jsme se rozhodli, že hádanku rozebereme rovnou a trochu podrobněji.

```
void foo(unsigned char *from, unsigned char *to, size_t count)
{
   size_t n = (count + 7) / 8;
   switch (count % 8)
   {
   case 0: do { *to++ = *from++;
   case 7:      *to++ = *from++;
   case 6:      *to++ = *from++;
   case 5:      *to++ = *from++;
   case 4:      *to++ = *from++;
   case 3:      *to++ = *from++;
   case 2:      *to++ = *from++;
   case 1:      *to++ = *from++;
          } while (--n > 0);
   }
}
```

Tentokrát bychom Vám mohli položit vlastně hned několik otázek:

1. Půjde procedura „foo“ vůbec přeložit?
2. Co daný kód vlastně dělá?
3. Proč by někdo takový kód psal?
4. K čemu je to dobré?

1. Půjde procedura „foo“ vůbec přeložit?

Ano, i přesto, že se konstrukce `switch-case` docela podivně prolíná s `do-while`. Jazyk C, na rozdíl od dalších jazyků, poskytuje poměrně velkou volnost v tom, kde se může `case` vyskytnout, a tak je možné jej umístit i do jiných konstrukcí. Že tak vznikne poněkud nehezký a špatně srozumitelný příbuzný operace `goto` ponechme pro tento okamžik stranou.

2. Co daný kód vlastně dělá?

Jak naznačují názvy jednotlivých argumentů, procedura kopíruje `count` bytů z adresy určené ukazatelem `from` do paměti dané ukazatelem `to`. Kód by se dal tedy přepsat jako:

```
while (count) {
   *to = *from;
   to++;
   from++;
   count--;
}
```

3. Proč by někdo takový kód psal?

Pro odpověď se nejdříve podíváme na to, jak by vypadal kód výše zmíněné jednoduché smyčky, kdyby jej přeložil ne úplně dobrý překladač.

```
  test   rdx,rdx   ;; pokud je `count` == 0,
  je     return    ;; pak koncime

loop:
  mov    al, [rdi] ;; al := *from
  inc    rdi       ;; from++
  mov    [rsi],al  ;; *to := al
  inc    rsi       ;; to++
  dec    rdx       ;; count--
  jne    loop      ;; pokud `count != 0`,
                   ;; skocime na `loop`,
return:            ;; jinak ukoncime proceduru
  ret
```

Klíčové je pro nás tělo smyčky kopírující data. Všimněte si, že se skládá ze šesti operací, přičemž jen první čtyři se starají o vlastní přenos dat, zbylé dvě operace realizují smyčku `while`. Kdybychom to zkusili kvantifikovat, jednu třetinu času z celkového kopírování stráví procesor testováním, zda již došel na konec. Ve skutečnosti, by to dopadlo ještě hůř, jelikož instrukce skoku patří k těm pomalejším.

Co tedy kód z naší hádanky dělá?

Cílem kódu je přenášet bloky dat po osmi bytech. Nejdříve se do proměnné `n` uloží počet cyklů, které se mají provést. Situaci, kdy počet bytů určených k přenosu není dělitelný osmi, vyřešíme v prvním cyklu, kdy se díky konstrukci `switch-case` provede `(count mod 8)` přenosů a v každém dalším cyklu právě osm přenosů. Zde už se `switch` neuplatňuje a používá se jen cyklus `do-while`. Tím se výrazně eliminuje počet operací skoku, jelikož jednotlivé návěští `case` nepředstavují žádný kód a prvotní rozhodnutí `switch-case` lze realizovat jednoduchým skokem na adresu uloženou v tabulce. Kód tedy opravdu může běžet efektivně.

V podstatě je to ručně naprogramovaná optimalizace *rozbalení smyček*, která umožňuje zrychlit běh programu za cenu zvětšení výsledného binárního souboru, a kterou dnešní překladače podle potřeby běžně provádí.

4. K čemu je to dobré?

Kód z hádanky je variantou takzvaného *Duff's device*, který navrhl Tom Duff při práci pro Lucasfilm, když potřebovat implementovat efektivní přenos dat mezi pamětí počítače a do paměti namapovanými registry zařízení. Jargon File Duffův kód označuje jako[1]:

The most dramatic use yet seen of fall through in C.

Z dnešního pohledu optimalizace tohoto typu již příliš smysl nemají a na první pohled čitelný kód by měl být prioritou každého programátora. Pokud byste vzali výše zmíněnou jednoduchou smyčku a nechali překladači volné ruce v optimalizacích, mohli byste pozorovat, jak překladač provede nejen rozbalení smyčky, ale jak například použije vhodné registry pro přenos dat, tj. místo jednotlivých bytů bude přenášet celá 128 bitová nebo 256 bitová slova pomocí příslušných registrů.

1. viz http://foldoc.org/Duff's%20device.

Redakce:
Petr Krajča, Petr Osička
Katedra informatiky PřF UP
17. listopadu 12
771 46 Olomouc

web: www.inf.upol.cz/magazin
email: magazin@inf.upol.cz
atom: http://www.inf.upol.cz/atom/magazin
telefon: 585634701
GPS: 49.591870, 17.262336