Týden 9

Přednáška

V první části přednášky proběhla druhá zápočtová písemka.

Model RAM

Ve studijním textu je detailně popsán model RAM, který je realističtějším modelem počítače než Turingův stroj. Je tam uveden i konkrétní RAM (tedy konkrétní program = posloupnost instrukcí), který řeší následující problém.

VSTUP: Neprázdná posloupnost kladných celých čísel ukončená nulou.

VÝSTUP: Odchylky jednotlivých čísel od aritmetického průměru zadané posloupnosti zaokrouhleného dolů.

Činnost zmíněného RAMu je také přiblížena jednou z animací.
Každému by ovšem mělo být jasné, že podívat se na definici, příklad a animaci RAMu je něco zcela jiného než konkrétní RAM sestavit. Teprve „ošahání si“ jednotlivých instrukcí RAMu při konkrétním programování nás může přivést ke skutečnému porozumění, pasivní pohled na animaci to sotva může nahradit.
Proto jsme se pustili do společného sestrojení RAMu řešícího výše zmíněný problém. (Uvažovali jsme celá nenulová čísla místo kladných, ale to je nepodstatné.) Uvědomili jsme si, že nejdříve musíme pochopitelně navrhnout algoritmus, který pak budeme programovat. Ten jsme popsali následujícím pseudokódem pascalského typu.

```
(* variables: *)
A: array [1.. ] of integer;
cislo, pocet, soucet, prumer: integer;
pocet:=0; soucet:=0; read(cislo);
while cislo ≠ 0 do
{ pocet:=pocet+1; A[pocet]:=cislo; soucet:=soucet+cislo; read(cislo) };
prumer:= soucet div pocet; (* celociselne deleni *)
for i:=1 to pocet do { write(A[i]-prumer) };
```

Tento pseudokód jsme relativně přímočaře postupně přepsali pomocí instrukcí RAMu.
Pro jednoduché proměnné jsme si vyhradili paměťové buňky s následujícími adresami

cislo ... 2
pocet ... 3
soucet ... 4
prumer ... 5

Poli A jsme přiřadili základní adresu 8 (prvek A[1] jsme ukládali do buňky 9, prvek A[2] do buňky 10, atd.).
Dospěli jsme k programu

```
 1  READ
 2  JZERO 13
 3  STORE 2
 4  LOAD 3
 5  ADD =1
 6  STORE 3
 7  STORE 1
 8  LOAD 2
 9  STORE *8
10  ADD 4
11  STORE 4
12  JUMP 1

13  LOAD 4
14  DIV 3
15  STORE 5

16  LOAD =1

17  STORE 1
18  SUB 3
19  JGTZ 26
20  LOAD *8
21  SUB 5
22  WRITE
23  LOAD 1
24  ADD =1
25  JUMP 17

26  HALT
```

Přitom jsme si detailně promysleli význam všech instrukcí (částečně jsme si je „animovali“ na konkrétním příkladu), využití pracovního registru 0 a indexregistru 1 a rozdíly v argumentech typu “$= i$”, “i” a “$*i$” (kde i je zápis celého čísla).

Samozřejmě je vhodné si program detailně okomentovat, aby bylo jasné, že se opravdu jedná o (jednu možnost) přepsání uvedeného pseudokódu do instrukcí RAMu. (Kdo se neúčastnil tvorby RAMu na přednášce, měl by si nezávisle udělat sám; pohled na hotovou věc moc nepomůže, jak jsme již zmínili dříve.)

Simulace mezi RAMy a Turingovými stroji

Shodli jsme se, že je vcelku jasné, jak lze jakýkoli Turingův stroj s jednostranně nekonečnou páskou přímočaře simulovat RAMem.
Naopak je to technicky komplikovanější, ale myšlenkově to pro programátory zase není tak náročné – simulaci RAMu vícepáskovým Turingovým strojem ilustruje jedna z animací.

Sekce pro hlubší zájemce – důkazy

Podrobně jsme si probrali důkaz tvrzení, že „diagonální problém zastavení“ (Diagonal Halting Problem, DHP) není algoritmicky rozhodnutelný. Použili jsme důkaz sporem — tak, jak je uvedeno ve studijním textu (část 6.5.).

Partie textu k prostudování

Model RAM (část 6.3.), simulace mezi výpočetními modely, Church-Turingova teze (6.4.), rozhodnutelnost a nerozhodnutelnost problémů (6.5.).
(Máte si udělat přinejmenším dobrou první představu a zamyslet se nad příklady, speciálně těmi plánovanými na cvičení, ať se můžete na cvičení aktivně účastnit a případné problémy si tam objasnit.)

Cvičení

Na začátku diskuse opravené druhé zápočtové písemky.

Prezentace referátů

Referát č. 18 (Další otázka týkající se Turingových strojů)
Představme si, že navrhujeme Turingův stroj M, přičemž chceme používat již hotový M_1 jako proceduru, které lze předat vstup (parametr), tj. řetězec symbolů, a obdržet od ní příslušný výstup.
K tomu se hodí chápat M jako dvoupáskový (víme, že vícepáskový stroj lze simulovat jednopáskovým, je-li potřeba). Když M potřebuje výstup M_1 odpovídající vstupu w, neboli potřebuje zjistit řetězec $M_1(w)$, napíše w na druhou pásku, na ní pak nechá běžet M_1 a až M_1 skončí, překopíruje si M výsledný řetězec z druhé pásky na „základní“ pásku a udělá si s ním, co potřebuje.
Popište teď, jak byste řešili případ, kdy M potřebuje takto „volat“ stroje $M_1, M_2, \ldots, M_k$, které se ovšem mohou také *rekurzivně volat navzájem*.
Referát č. 19
V definici modelu RAM v základním studijním materiálu je uvedena hodnota operandu $*i$ jako číslo uložené na adrese, jež je dána součtem čísla i a čísla uloženého v indexregistru.

Jiná užívaná možnost nepřímé adresace je, že hodnota $*i$ je chápána jako číslo uložené na adrese, která je uložena v buňce s adresou i.
Ukažte, jak lze RAM-program v jedné variantě simulovat RAM-programem v druhé variantě a naopak. (Ilustrujte na jednoduchém příkladu.)

Příklady

Příklad 9.1

Zjistěte, co dělají dva níže uvedené fragmenty programů pro stroje RAM. Připomeňme, že paměťová buňka s adresou 0 je pracovní registr a buňka s adresou 1 je indexregistr. Hodnota operandu $*i$ (i je zápis celého čísla, např. 281) je číslo uložené v buňce s adresou $i + j$, kde j je aktuální obsah indexregistru. Oproti základní definici zde užíváme také symbolické názvy paměťových buněk (vyhrazených pro příslušné proměnné) a symbolická návěští.

```
        READ
        STORE   N
        LOAD    =2
cykl:   STORE   temp
        LOAD    N
        JGTZ    body
        LOAD    temp
        WRITE
        HALT
body:   SUB     =1
        STORE   N
        LOAD    temp
        MUL     temp
        JUMP    cykl
```

```
        LOAD    N
        STORE   1
zmena   LOAD    *A
        STORE   X
cykl    LOAD    1
        SUB     =1
        JZERO   konec
        STORE   1
        LOAD    *A
        SUB     X
        JGTZ    zmena
        JUMP    cykl
konec   HALT
```

Příklad 9.2

Navrhněte Turingův stroj M s jednostranně nekonečnou páskou, který pro dané vstupní slovo $w \in \{a, b\}^*$ sestrojí (výstupní slovo) $w(w)^R$. Stroj M tedy realizuje příslušné (vstupně/výstupní) zobrazení $f_M : \{a, b\}^* \to \{a, b\}^*$ (např. $f_M(abb) = abbbba$).
Pak zvolte vhodné kódování slov v abecedě $\{a, b\}$ tak, aby analogické vstupně/výstupní zobrazení mohl realizovat RAM. Navrhněte konkrétní RAM M', který toto zobrazení realizuje; přitom postupujte tak, že RAM M' přímočaře simuluje Turingův stroj M. (Nejde o co nejjednodušší RAM pro daný úkol, ale o to, abyste aplikovali obecný postup prokazující, že každý TS je možné simulovat RAMem.)