Acta Universitatis Carolinae. Mathematica et Physica

Ladislav Koubek

Kompilace popisů a příkazů procedur v překladači z jazyka Algol 60

Acta Universitatis Carolinae. Mathematica et Physica, Vol. 10 (1969), No. 1-2, 97--102

Persistent URL: http://dml.cz/dmlcz/142238

KOMPILACE POPISŮ A PŘÍKAZŮ PROCEDUR V PŘEKLADAČI Z JAZYKA ALGOL 60

L. KOUBEK
Centrum numerické matematiky UK, Praha

ТРАНСЛЯЦИЯ И ОПИСАНИЕ ОПЕРАТОРОВ ПРОЦЕДУР В ТРАНСЛЯТОРЕ С ЯЗЫКА АЛГОЛ 60. В работе дается обзор принципов, по которым в трансляторе с языка АЛГОЛ 60 проводится компиляция описаний и операторов процедур так, чтобы полностью соблюдалась возможность рекурсивного вызывания процедур. Для управления работой процедур отводится одна индексная ячейка и, смотря по ее состоянию, вносится в программу рабочая память процедур. В работе детально описывается способ, при помощи которого программируется присваивание значений данной индексной ячейке, а также вся система работы с формальными и фактическими параметрами.

Práce s procedurami je nejsložitější partií v ALGOLu 60.

V této práci podáme jen stručný přehled principů té části kompilátoru, která se týká procedur a to jak kompilace popisů, tak operátorů procedury a zápisů funkcí.

V sémantice příkazu procedury a v jejím zpřesnění v "Dodatku ke zprávě o ALGOLu 60" /1/ se mluví o "smyšleném bloku" operační části procedury. Chceme-li připustit rekursivní vyvolávání procedur, je nejvhodnější tento smyšlený blok opravdu realizovat alespoň pro umístění údajů o formálních parametrech a pro hodnoty veličin lokalizovaných v proceduře. Přesněji řečeno, každé úrovni vyvolání procedury přiřadíme určité pracovní pole procedury, do kterého umístíme hodnotu parametrů, veličin lokalizovaných v ope-

rační části procedury a také všechny mezivýsledky potřebné při výpočtu výrazů. Tím automaticky odstraníme všechny potíže vyplývající z eventuelních kolizí v označeních i z rekursivního vyvolávání procedur. Toto pole zařazujeme do strojového programu při každém vyvolání procedury a rušíme po jejím provedení, tj. při každém výstupu z procedury přes její konec nebo příkazem skoku, jehož cílový výraz vede mimo operační část procedury (přesněji řečeno mimo operační část procedury, formálně přiřazenou úrovni jejího vyvolání).

Při realizaci algoritmu na počítači zařadíme toto pracovní pole procedury nejsnáze tak, že jeden indexový registr vyhradíme pro řízení chodu procedur. Základní hodnotu do něj dosadíme při každém startu výpočtu podle strojového programu a při každém vyvolání procedury hodnotu registru zvětšujeme buď o konstantní veličinu určenou maximálně přípustným počtem všech formálních parametrů, veličin lokalizovaných v proceduře a pracovních buněk, nebo o veličinu závislou na vyvolávané proceduře, určenou skutečným počtem všech uvedených veličin.

V prvém případě zavádíme ovšem při konkrétní realizaci jisté omezení ALGOLu 60, neboť předem určujeme maximální počet formálních parametrů a veličin lokalizovaných v proceduře a omezujeme také "délku" výrazů v operační části procedury. Omezení tohoto druhu jsou však v každé realizaci translátoru nutná i u ostatních veličin, a proto jsme tento způsob zvolili v realizaci PHEN-ALGOL. Jeho technické provedení je dosti jednoduché.

Formálním parametrům a veličinám lokalizovaným v proceduře pak přiřazujeme jen relativní adresy vzhledem k začátku pracovního pole procedury a do programu je zařazujeme s příznakem modifikace zvoleným řídícím registrem.

Registr o stejnou konstantu zmenšujeme při každém výstupu z procedury. Pokud bychom připouštěli výstup z procedury jen přes její konec, nevznikaly by při tomto způsobu žádné komplikace.

Výstup z procedury můžeme však programovat i příkazem skoku, jehož cílový výraz vede z operační části procedury. Proto přiřadíme každé proceduře zvláštní pracovní buňku strojového programu. Do této buňky zapíšeme při každém vyvolání procedury stav řídícího registru. Podobnou buňku přiřadíme také celému programu

a zapíšeme do ní počáteční stav řídícího registru, tj. stav, který má, není-li právě vyvolána žádná procedura.

Příkazy skoku programujeme nepřímo. Do programu dáváme instrukci Us, kde s je buňka přiřazená návěští s. Při programování příkazu s návěštím s známe první adresu jeho programu a víme, v operační části které procedury (programu) je tento příkaz zapsán. Do buňky s umístíme úsek programu, který, z buňky přiřazené proceduře, obnoví hodnotu řídícího registru na stav, který byl při vstupu do procedury (na počáteční hodnotu, je-li příkaz užit v programu mimo procedury) a pak teprve provedeme vlastní skok na první instrukci příkazu s návěštím s.

Je zřejmo, že takto zajistíme správnou úpravu hodnoty řídícího registru ve všech případech s vyjímkou případu, kdy cílovým výrazem příkazu skoku je formální parametr. Při rekursivním vyvolávání procedury může totiž být skutečným parametrem návěští, které je užito v operační části této rekursivně vyvolávané procedury. Skok pak musí nutně vést do nejblíže nižší úrovně vyvolání procedury. Proto v tomto případě zařazujeme do programu příkazu skoku instrukce, kterými nejprve zmenšíme obsah buňky přiřazené proceduře, a pak teprve instrukci Us. Je-li skutečným parametrem návěští zapsané mimo operační část rekursivně vyvolávané procedury, opraví se hodnota řídícího registru správně bez ohledu na to, že jsme snížili hodnotu přiřazené buňky, protože dosazujeme hodnotu z buňky přiřazené jiné proceduře (programu). Je-li však skutečným parametrem návěští z operační části rekursivně vyvolávané procedury, dosazujeme do řídícího registru hodnotu odpovídající předchozímu vyvolání.

Při programování operační části procedury pracujeme s formálními parametry obdobně jako s proměnnými s indexem, tj. užíváme adresu druhého řádu.

Každému formálnímu parametru přiřadíme v pracovním poli procedury tři po sobě následující relativní adresy. První z nich je buňkou návratu podprogramu, který se vytvoří při programování výrazu (přiřazeného) skutečného parametru, druhá je první buňkou tohoto podprogramu (nebo alespoň instrukcí skoku na začátek podprogramu), třetí vlastní pracovní adresou přiřazenou formálnímu parametru. Nalezneme-li v textu identifikátor formálního parametru, zařadíme do programu instrukce Rα, Uα + 1, Hα + 2 s pří-

znakem modifikace řídícím registrem. (α je první z trojice adres přiřazených parametru). Do buňky A (viz /2/) dáme adresu $\alpha + 2$ s příznakem adresy druhého řádu a pracujeme s ní stejně jako s adresou identifikátoru.

Je-li levou stranou dosazovacího příkazu nespecifikovaný formální parametr, musíme ovšem do strojového programu zařadit instrukce, které vytvoří příznak určující, zda má být proveden nebo přeskočen program modifikované transformační funkce (viz /3/) a to při nalezení symbolu :=.

Je-li formální parametr uveden v souhrnu hodnot, zařadíme tři uvedené instrukce do programu již při jeho nalezení v tomto seznamu a identifikátor přeneseme ze seznamu formálních parametrů do seznamu určeného specifikací.

Podobně postupujeme, není-li formální parametr specifikován a je poprvé použit jako identifikátor pole, přepínače nebo procedury, neboť v celé operační části procedury musí být používán jen stejným způsobem a není tedy nutné zjišťovat jeho význam při každém použití. Nesmíme ovšem zapomenout na to, že je nutno přenášet nejen označení parametru, ale také příznak, podle kterého rozhodujeme o provedení programu modifikované transformační funkce.

Vyskytne-li se v příkazu formální parametr, který není v seznamu formálních parametrů procedury (jejíž operační část programujeme), musí být formálním parametrem nadřazené procedury. V tomto případě jest ovšem nutno zařazovat instrukce Rα, U$\alpha + 1$, H$\alpha + 2$ modifikované příslušně sníženou hodnotou řídícího registru. Totéž ovšem platí i pro veličiny lokalizované v nadřazené proceduře.

Operační část procedury vytváříme ve tvaru podprogramu. Hodnotu instrukce návratu tohoto podprogramu musíme při vstupu do procedury přenést do nulté buňky pracovního pole přiřazeného proceduře, aby byl zajištěn správný návrat do programu v případě rekursivního vyvolávání procedury. Jinak postupujeme opět tak, že identifikátor procedury pokládáme za speciální případ operátoru postupu if a konec popisu procedury je opět určován srovnáváním úrovně koncových symbolů s úrovní prvního symbolu programu operační části procedury. Do sedmi pracovních polí J,... zapisujeme v tomto případě hodnotu úrovně a údaje, které používáme k částeč-

né kontrole syntaktické správnosti textu.

Při programování příkazu procedury nebo zápisů funkce postupujeme obdobně, jako při programování podmíněného příkazu resp. výrazu. Předem však do programu zařazujeme instrukce, které zvětšují obsah řídícího registru. Před program výpočtu hodnoty (určení adresy) každého skutečného parametru dáváme instrukce, kterými upravíme obsah buňky s relativní adresou $\alpha + 1$ a instrukci nepodmíněného skoku přes program skutečného parametru. Po nalezení ukončujícího symbolu) zařadíme instrukci vyvolání podprogramu procedury.

Vlastní programy výrazů skutečných parametrů, které sestavujeme při nalezení oddělovače , nebo ukončujícího symbolu), mají různé tvary podle toho, o jaký skutečný parametr se jedná.

Je-li skutečným parametrem identifikátor (což poznáme podle toho, že není otevřen žádný mikroprogram), zařadíme do programu instrukci, kterou adresu přiřazenou identifikátoru zapíšeme do střadače. Další instrukcí nastavíme příznak typu identifikátoru a zařadíme instrukci skoku na buňku s relativní adresou $\alpha + 1$ (modifikovanou řídícím registrem).

Je-li skutečným parametrem výraz, zapíšeme nejprve jeho hodnotu do buňky pracovního pole p a její adresu zapíšeme do střadače. Další program je stejný jako v případě, kdy skutečným parametrem je identifikátor, instrukci nastavující příznak typu však zařazovat nemusíme, protože výraz nemůže být levou částí dosazovacího příkazu.

S proměnnou s indexy pracujeme obdobně jako s výrazem.

Je-li ve výraze, který tvoří skutečný parametr, použit formální parametr nadřazené procedury, musíme nejprve zmenšit hodnotu řídícího registru a po nalezení významu skutečného parametru musíme hodnotu řídícího registru opět zvýšit.

Pravidla programování popisů a příkazů procedur vyslovovat nebudeme, neboť jejich znění je z předchozího popisu zcela zřejmé.

Literatura

/1/ Koubek L. - Programující program PHEN-ALGOL pro počítače

ODRA 1003 a 1013
Závěrečná zpráva č. 10/69 VÚZORT Praha

/2/ Koubek L. - Algoritmus kompilace nepodmíněných výrazů, dosazovacích příkazů a příkazu skoku v translátoru z jazyka ALGOL 60, AUC, Math.-Phys., Vol. 10, 57-69 (1969)

/3/ Koubek L. - Zobrazení veličin a specifikace parametrů procedur v jedné realizaci překladače z jazyka ALGOL 60 AUC, Math.-Phys., Vol. 10,71-76 (1969)