Acta Universitatis Carolinae. Mathematica et Physica

Ladislav Koubek

Překlad promenných s indexy a přepínačů v překladači z jazyka Algol 60

Acta Universitatis Carolinae. Mathematica et Physica, Vol. 10 (1969), No. 1-2, 87--90

Persistent URL: http://dml.cz/dmlcz/142236

PŘEKLAD PROMĚNNÝCH S INDEXY A PŘEPÍNAČŮ V PŘEKLADAČI Z JAZYKA ALGOL 60

L. KOUBEK
Centrum numerické matematiky UK, Praha

РАБОТА С ПЕРЕМЕННЫМИ С ИНДЕКСАМИ И С ПЕРЕКЛЮЧИТЕЛЯМИ В ТРАНСЛЯТОРЕ С ЯЗЫКА АЛГОЛ 60. В работе описывается способ, посредством которого в трансляторе с языка АЛГОЛ 60 проводится компиляция индексных выражений и вместе с тем переменным с индексами ставятся в соответствие конкретные адреса. Отмечается, что индексные выражения можно считать некоторым частным случаем условных выражений, вследствие чего алгоритм транслятора существенно упрощается.

Při práci s proměnnými s indexem musí kompilátor sestavit program, kterým vypočteme hodnoty všech indexových výrazů a podle zvoleného linearizačního vzorce určíme konkrétní adresu proměnné.

V našem algoritmu řešíme tento problém tak, že při vstupu do bloku (viz /1/) se vypočtou linearizační konstanty podle vzorců uvedených např. v /2/ nebo /3/. Tyto konstanty uložíme do paměti před první buňku pole a adresu první z nich dosadíme do adresy, kterou kompilátor přiřadil identifikátoru pole.

Při kompilaci indexových výrazů stačí tedy znát jen adresu přiřazenou identifikátoru pole. Překlad indexových výrazů provádíme podobně jako překlad podmíněných výrazů. Při nalezení otevírající indexové závorky zvětšíme nejprve index m a do sedmi polí $J,\ldots$ dáme údaje:
J_m:= úroveň otevírajícího symbolu prvého indexového výrazu,

O_m:= instrukce φ_{r-1} a její stupeň s (φ_{r-1}),
M_m:= index mikroprogramu, do kterého právě zapisujeme,
F_m:= nejmenší index dosud nepoužitých mikroprogramů,
Z_m:= obsah buňky Q (viz /4/) určující typ výrazu,
T_m:= adresa přiřazená identifikátoru pole (se záporným znaménkem),
E_m:= 0
Je vidět, že jen údaje v Z_m a T_m se liší od údajů, které do těchto buněk dosazujeme po nalezení if. Do pracovních buněk A a S dáváme stejné údaje jako při nalezení if, do buňky Q umístíme nulu, která je příznakem typu integer. Pak pokračujeme v práci podle pravidla A 0.

Ukončujícím symbolem každého indexového výrazu může být jen čárka nebo uzavírající závorka.

V prvém případě dokončíme programování indexového výrazu, je-li ⟨Q⟩ = 0 zařadíme program transformační funkce a podle hodnoty E_m instrukci násobení linearizační konstantou. Mezivýsledek uložíme do pracovní buňky f_m. Poté zvětšíme E_m, obnovíme obsahy S, A a Q a pokračujeme podle pravidla A 0.

V druhém případě zařadíme program výpočtu konkrétní adresy proměnné s indexem, výsledek přešleme do f_m, zmenšíme m a obnovíme stav S, Q a indexu mikroprogramu, do kterého zapisujeme. Do A zašleme adresu f_m s příznakem adresy druhého řádu.

Popsaným způsobem dostaneme zřejmě správný program pro libovolnou kombinaci podmíněných výrazů a proměnných s indexem, tj. ať je proměnná s indexem v podmíněném výrazu nebo naopak, podmíněný výraz v indexovém výrazu atd. Nevadí ani případ, kdy v indexovém výraze je použita další proměnná s indexem.

Pravidla C a D pro symboly , a] přesně formulovat nebudeme. Jejich formulace je zcela zřejmá a úseky programu, které je nutno zařadit, jsou do značné míry závislé na volbě počítače, na kterém algoritmus realizujeme. Podotkněme jen, že je třeba jisté opatrnosti při formulaci obou pravidel pro symbol , . Tento oddělovač se totiž používá nejen k oddělení indexových výrazů, ale také k oddělení parametrů procedur a prvků seznamu cyklů. Z tohoto důvodu do buňky T_m dáváme adresu se záporným znaménkem jako příznak, o který případ fiktivního if se jedná. V dalším totiž uvidíme, že také operátory cyklu a procedur zpracováváme obdobným

způsobem.

V tomto stručném popisu kompilace indexových výrazů jsme předpokládali, že můžeme pracovat s adresami druhého řádu. Chceme-li realizovat překladač na počítači, který adresu druhého řádu nemá, musíme ji modelovat. Např. počítač ODRA 1003, na kterém jsme algoritmus realizovali, adresy druhého řádu nemá. Obsah každé buňky paměti můžeme však přenášet do indexregistrů bez změny obsahu střádače. Proto před každou instrukci, která má být opatřena adresou druhého řádu, zařazujeme instrukci pro přenos obsahu buňky f_m do prvního registru a vlastní instrukci programujeme s nulovou adresou a s modifikací tímto registrem.

V důsledku blokové struktury ALGOLu 60 nemá ovšem valný smysl snažit se o zjednodušení práce u proměnných s konstantními indexy, protože při kompilaci ještě neznáme umístění pole v paměti počítače při výpočtu a nemůžeme proto provádět výpočet adresy už v tomto okamžiku. Jisté zjednodušení by bylo možné jen u vícerozměrných polí, avšak komplikaci algoritmu by pravděpodobně tento zisk nevyvážil.

Zjednodušíme-li poněkud syntaxi přepínače:
⟨seznam přepínačů⟩ ::= ⟨návěští⟩|⟨seznam přepínačů⟩ ⟨návěští⟩,
tj. v seznamu přepínačů připouštíme jen návěští, lze přepínač pokládat za speciální případ lineárního pole a není nutno vytvářet zvláštní algoritmus pro programování příkazů skoku, jejichž cílovým výrazem je přepínač.

Toto omezení se zdá být autorovi oprávněné, protože v žádném z algoritmů publikovaných v literatuře (viz např. /5/) nenašel přepínač, v jehož seznamu by byly obecné cílové výrazy.

Na závěr podotkněme, že je zcela zřejmé, že symbol indexových závorek byl do ALGOLu 60 zaveden především pro názornost zápisu. Při vlastní kompilaci by jej bylo možno bez potíží nahradit závorkami obyčejnými, tak, jak je tomu např. ve FORTRANu. Zcela triviální změnou algoritmu by bylo možné pracovat s uzavírající závorkou jakožto ukončujícím symbolem nejen v případě podmíněných příkazů, ale i v případě seznamu indexů.

Literatura

/1/ Koubek L. - Vytváření seznamů v translátoru z jazyka ALGOL 60 AUC, Math.-Phys., Vol. 10, 103-104 (1969)

/2/ Randell B. a Russel L. J. - ALGOL 60 Implementation, Ac.Press - London 1964

/3/ Koubek L. - Programující program PHEN-ALGOL pro počítače ODRA 1003 a 1013, Závěrečná zpráva č. 10/69 VÚZORT Praha

/4/ Koubek L. - Zobrazení veličin a specifikace parametrů procedur v jedné realizaci překladače z jazyka ALGOL 60 AUC, Math.-Phys., Vol. 10, 71-76 (1969)

/5/ Collected Algorithms from CACM, New York 1966