Acta Universitatis Carolinae. Mathematica et Physica

Ladislav Koubek

Programování příkazů cyklu v kompilátoru z jazyka Algol 60

Acta Universitatis Carolinae. Mathematica et Physica, Vol. 10 (1969), No. 1-2, 91--96

Persistent URL: http://dml.cz/dmlcz/142237

1969 ACTA UNIVERSITATIS CAROLINAE MATHEMATICA ET PHYSICA VOL.10

PROGRAMOVÁNÍ PŘÍKAZŮ CYKLU V KOMPILÁTORU Z JAZYKA ALGOL 60

L. KOUBEK
Centrum numerické matematiky UK, Praha

ПРОГРАММИРОВАНИЕ ОПЕРАТОРОВ ЦИКЛА В ТРАНСЛЯТОРЕ С ЯЗЫКА АЛГОЛ 60. В работе приводится краткое описание алгоритма компиляции операторов цикла в трансляторе с языка АЛГОЛ 60. Отмечается, что операторы цикла можно вполне правомерно принимать (что следует считать выгодным) за специальный случай условного оператора.

Syntaktická stavba příkazu cyklu je v ALGOLu 60 velmi jednoduchá, umožňuje však programování cyklů s neobyčejně složitým průběhem hodnot parametru cyklu, který nadto může být i proměnnou s indexem.

V našem algoritmu pracujeme s příkazy cyklu jako se speciálním případem podmíněných příkazů, což umožňuje automatické nalezení koncového symbolu příslušného k symbolu for. Druhou složku úrovně v tomto případě zvyšujeme při nalezení symbolu for a snižujeme při nalezení do.

Abychom mohli jednoduše programovat cykly s více prvky seznamů cyklu a s parametrem, který může být proměnnou s indexem, a přitom nemuseli zvětšovat počet údajů chráněných v paměti počítače po celou dobu, po kterou je příkaz cyklu (jako programovací jednotka) otevřen, zařadíme buď přímo do strojového programu nebo do pole konstant dvě buňky, do kterých zapíšeme instrukce U s neurčenými

adresami. Pro stručnost jim budeme v dalším říkat "adresa cyklu" a "adresa příkazu".

Při nalezení symbolu for zvětšíme index m a do m-tých buněk pracovních polí zařadíme údaje:

J_m:= úroveň symbolu for
O_m:= 0
M_m:= adresa první dosud neobsazené buňky programu
F_m:= 0
Z_m:= -1
T_m:= 0
E_m:= adresa cyklu.

Protože cyklus je vždy příkaz, nemůže před symbolem for být v programovací jednotce symbol := a nemůže být také otevřen žádný mikroprogram. Proto můžeme využít údaje v O_m, M_m, Z_m, T_m a E_m jinak než v případě, kdy kompilujeme skutečné podmíněné příkazy.

Údaj v O_m indikuje druh prvku seznamu cyklů, protože u každého druhu je předepsán jiný způsob prověrky, zda prvek seznamu je vyčerpán. Program prověrky musíme zařadit při nalezení symbolu , nebo do. Proto při nalezení oddělovače step zapíšeme do O_m libovolné kladné číslo, při nalezení while záporné číslo a po zařazení programu prověrky opět 0.

Je-li parametr cyklu proměnnou s indexem, musíme celý program výpočtu adresy parametru (tj. výpočet všech hodnot indexových výrazů, linearizaci a dosazení do pracovní buňky f_m) provádět před každou změnou hodnoty parametru cyklu, protože hodnoty některých veličin v indexových výrazech se mohly při provádění cyklu změnit. Nejjednodušší je zařazovat tento úsek programu před program výpočtu každého prvku seznamu cyklů. Do buňky M_m zapíšeme při nalezení sambolu for počáteční adresu tohoto úseku programu a do Z_m dáme počet instrukcí, který je možno určit při nalezení symbolu :=. K tomu je ovšem třeba upravit pravidlo C 2.

Před vyslovením upraveného pravidla C 2 zavedeme novou pracovní buňku kompilátoru, kterou označíme C.

Pravidlo C 2: Načteme-li symbol := pak obsah buňky Q zapíšeme do Q1 a je-li:
a) $\langle Z_m \rangle \geq 0$, zapíšeme do první volné buňky mikroprogramu M instrukci H ⟨A⟩, do A dáme adre-

su nuly, do S instrukci B a pokračujeme podle A O.
b) $\langle Z_m \rangle < 0$, zapíšeme do Z_m počet instrukcí programu zapsaných po nalezení symbolu for (tj. rozdíl mezi běžnou adresou programu a $\langle M_m \rangle$), do C dosadíme $\langle A \rangle$, do A dáme adresu nuly, do S instrukci B a pokračujeme podle A O.

V dalším budeme ještě potřebovat strojovou instrukci Rt. Tato instrukce do adresové části buňky t zapíše svou vnější adresu zvětšenou o dvě. (Realizujeme-li překladač na počítači, který obdobnou instrukci nemá, můžeme ji modelovat nejvýše třemi strojovými instrukcemi.)

Popišme nyní postup práce pro jednotlivé druhy prvků seznamu cyklů, z nichž každý je ukončen symbolem , nebo do.

Nejjednodušším prvkem seznamu cyklů je aritmetický výraz a. Tento prvek je vyčerpán tím, že jeho hodnotu dosadíme do parametru cyklu a necháme cyklus proběhnout. V našem algoritmu je tento prvek charakterizován tím, že $\langle Q_m \rangle = 0$. Je-li ukončovacím symbolem oddělovač ,, přepíšeme nejprve program výpočtu adresy parametru cyklu (když $\langle Z_m \rangle > 0$ a běžná adresa strojového programu větší než $\langle M_m \rangle + \langle Z_m \rangle$), dokončíme programování výrazu a, zařadíme instrukci H $\langle C \rangle$ a dvě instrukce prověrky R $\langle E_m \rangle$, U $\langle E_m \rangle$ +1. Poté pokračujeme v načítání dalšího symbolu textu podle pravidla A O.

Druhým případem je prvek "krok - do". Ten má tvar a step b until c, kde a, b, c jsou aritmetické výrazy.

Při nalezení oddělovače step dosadíme do O_m kladnou hodnotu, v případě potřeby přepíšeme program výpočtu adresy parametru cyklu, dokončíme programování výrazu, zařadíme instrukci H $\langle C \rangle$ a načítáme další text.

Při nalezení oddělovače until nejprve dokončíme programování výrazu b, zařadíme instrukce H φ, R $\langle E_m \rangle$, U σ, kde φ je pracovní buňka strojového programu. Je-li k počet instrukcí strojového programu výrazu b a φ vnější adresa instrukce U σ, je $\sigma = \varphi + k + 4$. Pak znovu přepíšeme program výrazu b a připojíme instrukce v H φ, + v, Hv. Proč zařazujeme právě tento úsek programu, bude nejlépe vidět z příkladu.

Po nalezení oddělovače , dokončíme program výrazu c, zařadíme instrukci -v, xφ, T φ + 2, U $\langle E_m \rangle$ + 1. φ je vnější adresa instrukce T.

Posledním případem je prvek "dokud", jehož tvar je a <u>while</u> b, kde a je aritmetický a b boolovský výraz.

Při nalezení oddělovače <u>while</u> dosadíme do 0_m záporné číslo, do programu nejprve zařadíme instrukce R $\langle E_m \rangle$, U φ + 1, kde φ je opět vnější adresa. Poté přepíšeme program výpočtu adresy parametru cyklu, dokončíme programování výrazu a a zařadíme instrukci H $\langle C \rangle$. Pak pokračujeme dále v načítání textu.

Při nalezení oddělovače , dokončíme program výrazu b a zařadíme instrukce T φ + 2, U $\langle E_m \rangle$ + 1, kde φ je opět vnější adresa instrukce T.

Nalezneme-li operátor <u>do</u>, zařazujeme stejné úseky programu jako při oddělovači , , ale navíc zapíšeme do adresy operátoru (tj. do instrukce z vnější adresou $\langle E_m \rangle$+1) instrukci Uβ+2, kde β je poslední instrukce programu. Do programu zařadíme další prázdnou instrukci a její vnější adresu zapíšeme do Z_m. Opravíme údaje v A a S, zmenšíme druhou souřadnici úrovně a pokračujeme v načítání textu.

Nalezneme-li symbol ; nebo <u>end</u>, jehož úroveň je rovna úrovni <u>for</u>, zapíšeme (podle upraveného pravidla D O) do programu instrukci U $\langle E_m \rangle$ a do instrukce s vnější adresou Z_m zapíšeme Uτ , kde τ je první odsud neobsazená instrukce programu.

Uveďme příklad programu, který bude sestaven naším algoritmem pro příkaz cyklu:

<u>for</u> v := a <u>step</u> b <u>until</u> c , d , e <u>while</u> f <u>do</u> S ; ,

kde S je příkaz.

vnější adresa:	instrukce:	poznámka:
α -1:	Uα+2	tato instrukce je nutná jen v případě, že adresu cyklu zařazujeme do programu
α :	U	adresa cyklu
α +1:	Uα+29	adresa příkazu
α +2:	Ba	ukončující symbol <u>step</u>
α +3:	Hv	

α+4:	Bb	
α+5:	Hφ	
α+6:	Rα	
α+7:	Uα+12	ukončující symbol until
α+8:	Bb	
α+9:	Hφ	
α+10:	+v	
α+11:	Hv	
α+12:	Bc	
α+13:	-v	
α+14:	xφ	ukončující symbol ,
α+15:	Tα+17	
α+16:	Uα+1	
α+17:	Bd	
α+18:	Hv	ukončující symbol ,
α+19:	Rα	
α+20:	Uα+1	
α+21:	Rα	
α+22:	Uα+23	ukončující symbol while
α+23:	Be	
α+24:	Hv	
α+25:	Bf	
α+26:	Tα+28	ukončující symbol do
α+27:	Uα+1	
α+28:	Uσ	doplněno po nalezení ;
α+29:		program operátoru S
σ-1:		
σ:		pokračování programu
σ:		za cyklem.

Jednoduchou úpravou je možno vynechat v programu některé zbytečné instrukce, např. přesílání do pomocné buňky φ, je-li výraz b jen proměnnou, nebo instrukce v α+21, α+22, jestliže prvním prvkem seznamu cyklů není prvek "dokud". V kompilátoru, který pracuje jen na jediný průchod načítání textu, je ovšem nutno programovat všechny druhy cyklů jednotně, a proto neoptimálně. Nelze využít speciálních instrukcí, které jsou velmi výhodné, je-li krok cyklu konstantní apod.

Z popsaného postupu je zcela zřejmo jak je třeba formulovat

pravidla C a D pro operátory for a do i pro oddělovače step, until, while, , . Také doplněk pravidla D 0 pro případ $T_m = 0$ je zcela zřejmý. Proto tato pravidla už nebudeme explicitně vyslovovat.

Podotkněme jen, že algoritmus automaticky odliší oddělovač , , který je užit pro oddělení prvků seznamu cyklů od , použité k oddělení indexových výrazů a skutečných parametrů procedury.

Literatura

/1/ Koubek L. - Algoritmus kompilace nepodmíněných výrazů, dosazovacích příkazů a příkazů skoku v translátoru z jazyka ALGOL 60, AUC, Math.-Phys., Vol. 10, 57-69 (1969)

/2/ Koubek L. - Zobrazení veličin a specifikace parametrů v jedné realizaci překladače z jazyka ALGOL 60, AUC, Math.-Phys., Vol. 10, 71-76 (1969)