8. GENEROVÁNÍ CÍLOVÉHO PROGRAMU

Generátor cílového programu je poslední částí kompilačních překladačů. Jeho vstupem je program ve vnitřní formě, vytvořený při zpracování sémantiky a jeho výstupem je cílový program.

8.1 Druhy cílových programů

Cílový program kompilátoru může být:

1) Posloupnost strojových instrukcí s absolutními adresami.
2) Posloupnost strojových instrukcí s relativními adresami.
3) Posloupnost příkazů jazyka symbolických adres.

Překlad zdrojového programu do strojových instrukcí s absolutními adresami je nejefektivnějším způsobem kompilace. Vytvořený program lze okamžitě spustit a celý proces překladu a výpočtu lze provést během krátké doby. Hlavní nevýhodou tohoto typu překladu je skutečnost, že musí být překládán celý program - není možné použití předem připravené knihovny. Z tohoto důvodu se generování strojových instrukcí s absolutními adresami používá jen pro malé programy a v případě, že se jedná především o ladění programů (zejména při výuce). Překlad do strojových instrukcí s absolutními adresami provádějí inkrementální překladače.

Posloupnost strojových instrukcí s relativními adresami je nejčastějším a nejběžnějším výstupem kompilátoru. Výstupem kompilátoru jsou tzv. přemístitelné moduly, které je ovšem nutno zpracovat sestavovacím programem. V tomto případě je možné použití předem připravených knihoven, které obsahují jednotlivé programy opět ve formě přemístitelných modulů.

Program v jazyce symbolických adres je z hlediska konstrukce překladače nejjednodušším výstupem. Překladač v tomto případě neprovádí určování adres a v cílovém programu jsou obvykle zachovány jména veličin zdrojového programu. Nevýhodou tohoto přístupu je nutnost použití překladače jazyka symbolických adres k překladu cílového programu kompilátoru do strojového kódu. Tím vlastně přidáváme k celému procesu další fázi, která může být stejně časově náročná jako celá předchozí kompilace.

8.2 Hypotetický počítač

Vytvoření cílového programu závisí ve velké míře na použitém cílovém počítači. Proto budeme definovat hypotetický počítač, jehož vlastnosti budou blízké skutečným počítačům a při tom bude dostatečně jednoduchý, abychom mohli ukázat hlavní myšlenky generování instrukcí dostatečně přehledně.

Hypotetický počítač má paměť s pevnou délkou slova a střadač. Budeme předpokládat, že hypotetický počítač má tyto instrukce:

LOAD	P	ulož obsah paměťového místa P do střadače,
STORE	P	ulož obsah střadače na paměťové místo P,
ADD	P	přičti k obsahu střadače obsah paměťového místa P,
SUB	P	odečti od obsahu střadače obsah paměťového místa P,
MPY	P	vynásob obsah střadače obsahem paměťového místa P,
DIV	P	vyděl obsah střadače obsahem paměťového místa P,
CHS		změň znaménko obsahu střadače.

8.3 Generování instrukcí pro obrácený polský zápis

Předpokládejme, že vstupem generátoru cílového programu bude program v postfixovém zápisu. Úkolem generátoru cílového programu je přeložit tento program do jazyka hypotetického počítače. Takový překlad je jistou obdobou vyhodnocování programu nebo výrezu v postfixovém (obráceném polském zápisu). Rozdíl spočívá pouze v tom, že potřebné operace se neprovádí okamžitě, ale do cílového programu se vkládají instrukce, které zajistí provedení potřebných operací v době výpočtu.

Do zásobníku nebudeme ukládat hodnoty operandů, ale adresy nebo případně jiné informace o umístění operandů. Možnosti pro uložení operandů bývají v počítačích různé. Ve výše definovaném hypotetickém počítači může být operand uložen v paměti (P) nebo ve střadači. Proto si pro operátory, které budeme překládat (+, -, *, /, NEG) sestrojíme tabulky, ve kterých budou uvedeny vytvářené instrukce pro různé možnosti uložení operandů v hypotetickém počítači.

Tabulky pro operátory + a - budou vypadat takto:

+	STR	P (P)
STR		ADD P(P) STR
P (L)	ADD P (L) STR	LOAD P (L) ADD P (P) STR

-	STR	P (P)
STR		SUB P (P) STR
P (L)	STORE Ti LOAD P (L) SUB Ti STR	LOAD P (L) SUB P (P) STR

Poznámka:

V každé položce tabulky je uveden nakonec údaj o tom, kde bude uložen výsledek operace. Písmeno v závorce udává, zda se jedná o pravý (P) nebo levý (L) operand.

Pro násobení a dělení jsou tabulky podobné jako pro sečítání a odčítání. Pouze se změní operační kód ADD na MPY a SUB na DIV.

Tabulka pro změnu znaménka (NEG) má tvar:

NEG	STR	P
	CHS STR	LOAD P CHS STR

Algoritmus 8.1

Generování cílového programu pro obrácený polský zápis (postfixový zápis).

Vstup: Program v obráceném polském zápisu.

Výstup: Posloupnost instrukcí hypotetického počítače.

Metoda: Algoritmus používá zásobník, do kterého ukládá údaje o operandech.

1. Čti instrukci z vnitřní formy programu. Je-li to popis operandu ulož jej do zásobníku a opakuj 1. Není-li to popis operandu, přejdi na 2
2. Vyber tabulku pro čtený operátor a z této tabulky vyber položku podle druhu operandů (operandu) na vrcholu zásobníku.
3. Je-li některý z operandů, jehož popis je v zásobníku, uložen ve střadači - s výjimkou těch operandů, které vstupují do právě generovaných instrukcí - proveď uložení střadače a v zásobníku nahraď STR označením pomocné proměnné.
   (uložení střadače: generuj (STORE Ti), i:=i+1)
4. Generuj instrukce vybrané v 2.
5. V zásobníku zruš použité operandy a na vrchol přidej označení místa, kam byl uložen výsledek.

Příklad 8.1

Ukážeme si postup generování instrukcí pro výraz

- (A + B) * (C - D), který v obráceném polském zápisu vypadá takto:

(1)	TA adr(A)	(7)	DR
(2)	DR	(8)	TA adr(D)
(3)	TA adr(B)	(9)	DR
(4)	DR	(10)	_
(5)	+	(11)	*
(6)	TA adr(C)	(12)	NEG

Generování cílového programu bude probíhat takto:

Čtené instrukce	Obsah zásobníku	Vybraná položka z tabulky	Generované instrukce
TA adr(A),DR	adr(A)		
TA adr(B),DR	adr(A),adr(B)		
+	STR	+ (P,P)	LOAD adr(A) ADD adr(B)
TA adr(C),DR	STR,adr(C)		
TA adr(D),DR	STR,adr(C),adr(D)		
-	T1, adr(C),adr(D)		STORE T1
	T1, STR	- (P,P)	LOAD adr(C) SUB adr(D)

Čtené instrukce	Obsah zásobníku	Vybraná položka z tabulky	Generované instrukce
*	STR	* (P,STR)	MPY T1
NEG	STR	NEG (STR)	CHS

Příklad 8.2

Pro výraz (A + B) / (C - D) má program v obráceném polském zápisu tvar:

(1) TA adr(A)
(2) DR
(3) TA adr(B)
(4) DR
(5) +
(6) TA adr(C)
(7) DR
(8) TA adr(D)
(9) DR
(10) -
(11) /

Čtené instrukce	Obsah zásobníku	Vybraná položka z tabulky	Generované instrukce
TA adr(A),DR	adr(A)		
TA adr(B),DR	adr(A), adr(B)		
+	STR	+ (P,P)	LOAD adr(A) ADD adr(B)
TA adr(C),DR	STR, adr(C)		
TA adr(D),DR	STR, adr(C),adr(D)		
-	T1,adr(C),adr(D)		STORE T1
	T1,STR	- (P,P)	LOAD adr(C) SUB adr(D)
/	STR	/ (P,STR)	STORE T2 LOAD T1 DIV T2

Poznámka:

Kombineci instrukcí TA, DR chápeme jako popis jednoho operandu.

8.4 Generování instrukcí pro program v jazyce tříadresových instrukcí

Nejdříve si ukážeme, jak generovat cílový program v případě, že program ve vnitřní formě je reprezentován trojicemi. V zásadě lze použít podobného postupu jako při generování cílového programu pro postfixový zápis. Je možné dokonce použít stejných tabulek vytvářených instrukcí pro jednotlivé operátory. Informace o operandech je součástí trojic. Pouze informace o operandech, které jsou výsledkem předchozích trojic je uvedena v zásobníku.

Algoritmus pro překlad trojic do strojového kódu hypotetického počítače můžeme zapsat takto:

Algoritmus 8.2

Generování cílového programu pro trojice.

Vstup: Program ve formě trojic.

Výstup: Posloupnost instrukcí hypotetického počítače.

Metoda: Algoritmus používá zásobník, do kterého ukládá informace o operandech, které jsou výsledkem jednotlivých trojic.

1. Přečti jednu trojici z vnitřní formy programu.
2. Vyber tabulku pro operátor uvedený v trojici a z této tabulky vyber položku podle
   a) operandů uvedených v trojici,
   b) operandů uvedených na vrcholu zásobníku v případě, že ve zpracovávané trojici je použit výsledek některé předchozí trojice.
3. Je-li některý z operandů v zásobníku uložen ve střadači - s výjimkou operandů, které vstupují do právě generovaných instrukcí - proveď uložení střadače a v zásobníku nahraď STR označením pomocné proměnné.
4. Generuj instrukce vybrané v 2.
5. V zásobníku zruš použité operandy a na vrchol přidej označení místa, kam byl uložen výsledek.

Příklad 8.3

Výraz (A + B) / (C - D) pomocí trojic zapíšeme takto:

(1) + A, B
(2) - C, D
(3) / (1), (2)

Proces překladu těchto trojic do strojového kódu hypotetického počítače probíhá takto:

Překládaná trojice	Vybraná položka z tabulky	Generované instrukce	Obsah zásobníku
(1) + A, B	+ (P, P)	LOAD A ADD B	STR
uložení střadače		STORE T_1	T1
(2) - C, D	- (P, P)	LOAD C SUB D	STR T1
(3) / (1),(2)	/ (STR, P)	STORE T_2 LOAD T_1 DIV T_2	STR

Výsledek je úplně stejný jako v případě obráceného polského zápisu.

Nyní si ukážeme algoritmus generování instrukcí pro trojice, který se nazývá stromový algoritmus (KOMP). Algoritmus je rekurzivní a zpracovává jednotlivé trojice počínaje poslední. Opět si vytvoříme tabulky, ve kterých budou uvedeny činnosti algoritmu pro různé kombinace operandů v trojicích a různé operátory.

Pro operátor + vypadá tabulka takto:

+	STR	P(P)	číslo (P)
STR	———	ADD P(P) STR	STORE Ti KOMP (číslo (P)) ADD Ti STR
P(L)	ADD P(L) STR	LOAD P(L) ADD P(P) STR	KOMP (číslo (P)) ADD (P (L)) STR
číslo(L)	———	KOMP(číslo(L)) ADD P (P) STR	KOMP (číslo(L)) STR OPAKUJ

Pro operátor * je tabulka podobná, pouze místo ADD je MPY.
Pro operátor - bude tabulka vypadat takto:

-	STR	P(P)	číslo (P)
STR	———	SUB P(P) STR	———
P(L)	STORE Ti Ti OPAKUJ	LOAD P(L) SUB P(P) STR	KOMP (číslo (P)) STR OPAKUJ
číslo(L)	STORE Ti Ti OPAKUJ	KOMP(číslo(L)) SUB P(P) STR	KOMP (číslo(P)) STR OPAKUJ

Pro operátor / je tabulka podobná, pouze místo SUB obsahuje DIV.

Pro operátor NEG má tabulka tvar

NEG	STR	P	číslo
	CHS	LOAD P CHS	KOMP (číslo) CHS

V těchto tabulkách znamená:

KOMP volání procedury KOMP (rekurzivní),
OPAKUJ opakované prohlížení tabulky.

Práci algoritmu KOMP si ukážeme na příkladě.

Příklad 8.4

Je dán výraz

(A + B) / (C - D) .

V jazyce trojic je tento výraz zapsán takto:

(1) + A,B
(2) - C,D
(3) / (1),(2)

Vyvoláme KOMP(3). Parametrem KOMP je číslo poslední trojice.

Generování cílového programu bude probíhat takto:

Zpracovávaná trojice	Vybraná položka z tabulky	Obsah vybrané položky	Generované instrukce
(3)/(1),(2)	/(číslo(L),číslo(P))	KOMP(číslo(P)) STR OPAKUJ	
(2) - C,D	- (P(L), P(P))	LOAD P(L) SUB P(P) STR	LOAD C SUB D
(3)/(1),STR	/(číslo(L),STR)	STORE Ti Ti OPAKUJ	STORE T1
(3)/(1),T1	/(číslo(L),P(P))	KOMP(číslo(L)) DIV P(P) STR	
(1) + A, B	+(P(L),P(P))	LOAD P(L) ADD P(P) STR	LOAD A ADD B
(3)/(1),T1	/(číslo(L),P(P)) pokračování	KOMP(číslo(L)) DIV P(P)	DIV T_1

Tato metoda se hodí zejména v případě, že generování cílového programu se provádí v samostatném průchodu. Důvodem je zpětný chod ve vnitřní formě programu. V případě generování instrukcí pro čtveřice se dá použít všech předchozích metod. Pomocné proměnné ovšem nevytváří kompilační program sám, ale jsou již uvedeny ve čtveřicích.

Také se dá použít přístup takový, že pro každou čtveřici se vytvoří úsek programu bez ohledu na okolí. Toto je ovšem neefektivní, protože vznikne mnoho zbytečných přesunů. Uvedené metody jsou vhodné pro počítač, který má jen jeden registr a operační paměť. V případě, že počítač má více registrů, nebo víceúrovňovou operační paměť je třeba tyto postupy přizpůsobit počítači tak, aby cílový program optimálním způsobem využíval nejrychlejších registrů pro uchování mezivýsledků.