VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Formální jazyky a překladače
Překladač imperativního jazyka IFJ19

Tým 052, varianta II

Vedoucí:

Jaroslav Hort	**(xhortj04)**	**25%**
Filip Dráber	(xdrabe09)	25%
Iveta Strnadová	(xstrna14)	25%
Norbert Pócs	(xpocsn00)	25%

8. prosince 2019

1 Generátor výsledného kódu

Po úspěšné analýze je inicializován generátor výsledného kódu. Součástí inicializace je vyvtoření tabulek s rozptýlenými položkami pro ukládání názvu vytvořených proměnných a labelů pro funkce. Vlastní spuštění generátoru spočívá ve volání funkce `generate_code`.

1.1 Tří adresný kód

Tří adresný kód je generován na úrovni parseru a precedenční analýzy. Jeho struktura obsahuje v typ kódu, `ac_type`, a tři operandy typu `Token`, `op1`, `op2`, `res`. Výsledná struktura kódu je uložena do seznamu kódu, který je využit v generátoru. Hlavními funkcemi seznamu jsou funkce `setACAct`, která nastaví aktivitu seznamu na první položku, `actAC`, pro nastavení aktivity na následující položku a `isACActive` pro zjištení, zda je v seznamu aktivní prvek.

1.2 Průběh generování

Ze seznamu tři adresných kódu jsou postupně brány položky a podle typu kódu je rozhodnuto jaká instrukce bude zapsána do výsledného kódu. Výsledný kód je v průběhu generování rozdělen na dva bloky, kód hlavního programu (main) a kód funkcí a ukládán do dynamicého řetězce pro tyto bloky.

Tento způsob implementace byl zvolen z důvodu, že volání funkcí je realizováno pomocí skoku na návěští, pokud by byly funkce zapsány uvnitř hlavního kódu, bylo by nutné je přeskakovat, dokud nebudou volány, což by bylo horší na implementaci a výsledný kód by byl méně přehledný.

Po přečtení všech kódu a vygenerován instrukcí je výsledek vypsán na standardní výstup, nejprve kód funkci a poté kód hlavního bloku programu.

1.3 Vestavěné funkce

Vestavěné funkce jsou definovány v hlavičkovém souboru generátoru. V případě, že program poprvé volá jednu z těchto funkcí, jsou zapsnány do bloku funkcí a funkce je volána jako obyčejné uživatelské funkce. Výsledný kód tedy obsahuje pouze ty funkce, které jsou využívány, tímto rešením udržíme kód relativně čistý.

1.4 Generování identifikátorů

Při generování instrukcí pracujících s identifikátory je nejprve prohledána tabulka, zda nebyl identifikátor generován již dříve. Identifikátory z hlavního bloku programu, takzvané globání identifikátory, mají přednost před identifikátory uvnitř funkcí, nebo-li lokálními identifikátory. Pokud nebyl nalezen, je zapsána insturkce do kódu a do příslušné tabulky je vloženo jméno identifikátoru

1.5 Problém generování cyklu

U generování cyklu `while` byl zjištěn problém možné definice identifikátoru. Tuto možnost jsme ošetřili tak, že parser vygeneruje tří adresný kód `WHILE_START` pro označení začátku cyklu a `WHILE_END` pro konec cyklu. Když generátor narazí na začátek cyklu, projde všechny kódy uvnitř cyklu, provede kontrolu definice identifikátorů a případně identifikátory nadefinuje předem, než cyklus začne.

2 Pomocné knihovny

2.1 Knihovna chyb `errors`

Knihovna obsahuje výčet kódu všech možných chyb, které mohou nastat při překladu programu, globální proměnnou `global_error_code`, kterou jednotlivé moduly nastavují na jeden z typu chyby v případě, že někde nastala. Na této proměnné zavisí spuštění generátoru. Dále máme dvě funkce pro výpis chyby na standardní chybový výstup `stderr`, první pro výpis interní chyby a druhá pro kompilační chyby, která tiskne navíc název souboru a číslo řádku zdrojového programu na které se chyba vyskytuje.

2.2 Dynamický řetězec

Pro jednodušší práci s řetězci v programovacím jazyce C jsme vytvořili knihovnu `dynamic_string`. Knihovna nám umožnuje vytvořit nový řetězec, rozšířit ho připsáním nového řetězce na konec stávajícího, nebo ho skrátit na požadovanou délku.