**Vysoká škola polytechnická Jihlava**

Katedra technických studií   
Programovací jazyky a překladače

**Překladač z jazyka daného gramatikou do mezikódu ve formě čtveřic**semestrální práce

Autor práce: Tomáš Cink  
Jihlava 2019

Obsah

[Zadání 3](#_Toc9174949)

[1 Analýza problému 5](#_Toc9174950)

[2 Teoretický a technický návrh řešení 6](#_Toc9174951)

[3 Testování 7](#_Toc9174952)

[4 Ukázka vstupů a výstupů 8](#_Toc9174953)

[5 Zhodnocení projektu 9](#_Toc9174954)

Seznam obrázků

**Nenalezena položka seznamu obrázků.**

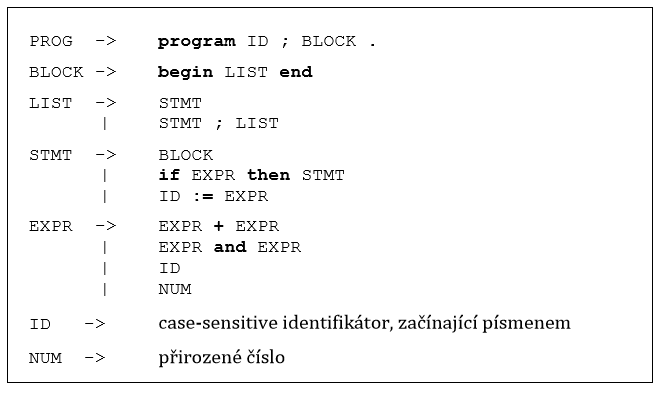
Seznam použitých zkratek

LEX Lexikální analýza

YACC Yet another compiler compiler

# Zadání

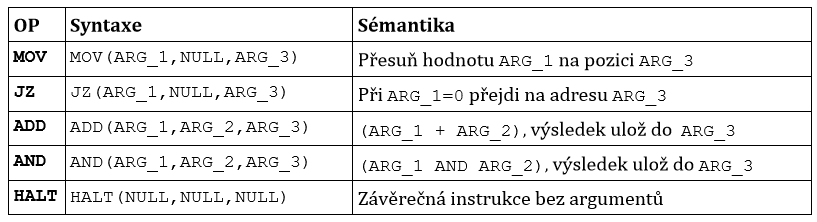
Navrhněte, vytvořte, otestujte a zdokumentujte překladač programovacího jazyka, generovaného následující gramatikou:



a generujícího intermediální kód (mezikód) ve formě čtveřic (quadruples) s tímto obecným formátem:



a následující syntaxí a sémantikou:



**Poznámky ke specifikaci jazyka a způsobu implementace:**

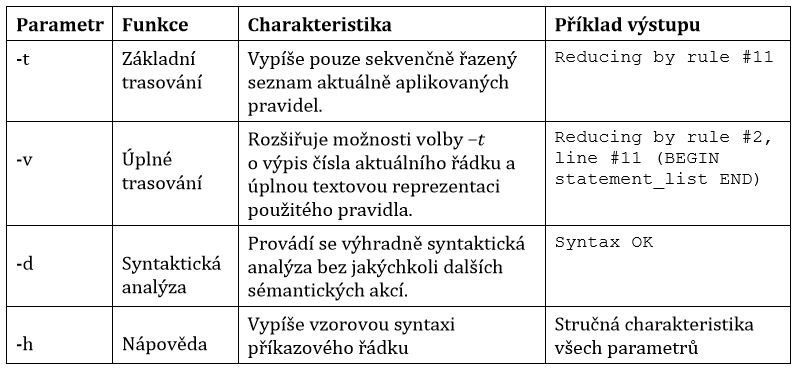
1. Všechny „neviditelné“ mezery (white spaces) jsou ignorovány
2. Jazyk povoluje samostatně stojící celořádkové komentáře, předznamenané středníkem:

Kód 1

; Toto je příklad komentáře

Kód 2

1. Výsledný překladač bude realizován pomocí standardních generátorů (např. typu lex a yacc) ve formě dvou souborů, lexikálního a syntaktického analyzátoru. Překlad projektu zajistěte prostřednictvím utility make. Ke zdrojovým kódům dodejte nejméně 3 testovací příklady.
2. Překladač pracuje se standardním vstupem a výstupem (stdin, stdout)
3. Zajistěte, aby překladač po nalezení jakékoli syntaktické chyby nepřerušil svoji práci, ale zpracoval i zbytek vstupního kódu. Na konci vypíše, pokud možno co nevýstižnější informaci o chybách, které se v průběhu překladu vyskytly.
4. Spustitelný soubor, realizující Váš překladač, bude v závislosti na parametrech příkazového řádku pracovat v následujících režimech:



# Analýza problému

Hlavním krokem k analýze je rozvrhnout tokeny potřebné v gramatice. V zadání jsou zvýrazněné a jsou to slova *program, begin, end, if, then, and* a symboly *„;“* pro zaznačení komentáře, *„.“* pro označení konce programu, *„+“* ke sčítání dvou expression a přiřazení pomocí *„:=“*. Dále lze zadat libovolnou proměnnou značenou v ID anebo numerické hodnoty značené NUM. Tyto tokeny jsou vytvořeny v lexikální analýze.

Pro syntaktickou analýzu lze využít zadanou gramatiku, kde jsou zvolené neterminály *PROG*, *BLOCK*, *LIST*, *STMT*, *EXPR* ve kterých se dále využívá terminálů ID a NUM a též tokenů z lexikální analýzy.

# Teoretický a technický návrh řešení

# Testování

Pro testování jsou vytvořeny tři testovací soubory

# Ukázka vstupů a výstupů

# Zhodnocení projektu