Překladač z jazyka daného gramatikou do mezikódu ve formě čtveřic

Individuální semestrální projekt z předmětu Programovací jazyky a překladače, LS 2018/19

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

```
PROG  ->   program ID ; BLOCK .
BLOCK ->   begin LIST end
LIST  ->   STMT
       |   STMT ; LIST
STMT  ->   BLOCK
       |   if EXPR then STMT
       |   ID := EXPR
EXPR  ->   EXPR + EXPR
       |   EXPR and EXPR
       |   ID
       |   NUM
ID    ->   case-sensitive identifikátor, začínající písmenem
NUM   ->   přirozené číslo
```

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

```
label: (OP, ARG_1, ARG_2, ARG_3)
```

a následující syntaxí a sémantikou:

OP	Syntaxe	Sémantika
MOV	`MOV(ARG_1,NULL,ARG_3)`	Přesuň hodnotu `ARG_1` na pozici `ARG_3`
JZ	`JZ(ARG_1,NULL,ARG_3)`	Při `ARG_1=0` přejdi na adresu `ARG_3`
ADD	`ADD(ARG_1,ARG_2,ARG_3)`	`(ARG_1 + ARG_2)`, výsledek ulož do `ARG_3`
AND	`AND(ARG_1,ARG_2,ARG_3)`	`(ARG_1 AND ARG_2)`, výsledek ulož do `ARG_3`
HALT	`HALT(NULL,NULL,NULL)`	Závěrečná instrukce bez argumentů

A 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
```

[3] 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.

[4] Překladač pracuje se standardním vstupem a výstupem *(stdin, stdout)* ,

[5] 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.

[6] Spustitelný soubor, realizující Váš překladač, bude v závislosti na parametrech příkazového řádku pracovat v následujících režimech:

Parametr	**Funkce**	**Charakteristika**	**Příklad výstupu**
-t	Základní trasování	Vypíše pouze sekvenčně řazený seznam aktuálně aplikovaných pravidel.	`Reducing by rule #11`
-v	Úplné trasování	Rozšiřuje možnosti volby *–t* o výpis čísla aktuálního řádku a úplnou textovou reprezentaci použitého pravidla.	`Reducing by rule #2, line #11 (BEGIN statement_list END)`
-d	Syntaktická analýza	Provádí se výhradně syntaktická analýza bez jakýchkoli dalších sémantických akcí.	`Syntax OK`
-h	Nápověda	Vypíše vzorovou syntaxi příkazového řádku	Stručná charakteristika všech parametrů

Vzorové chování překladače (v tomto příkladu se výsledný spustitelný soubor jmenuje *project*) za předpokladu korektní syntaxe zdrojového kódu č. 1 z níže uvedené tabulky příkladů:

```
program test;

begin
 first:=0;
 second:=20;

 begin
  if first then
   third:=15
 end
end.
```

```
./project <test.dl
0: ( MOV, 0, NULL, first )
1: ( MOV, 20, NULL, second )
2: (  JZ, first, NULL, 4 )
3: ( MOV, 15, NULL, third )
4: (HALT, NULL, NULL, NULL)
```

./project –h <test.dl

```
=====================================================
VSPJ, KTS, xPJP, semestralni prace. Autor: JV

Pouziti: ./project [options]

Options:
-h    vypis parametru prikazove radky
-t    sekvencni vypis cisel pouzitych pravidel
-v    sekvencni vypis cisel pouzitych pravidel, cisel radku a prave
      strany gramatickych pravidel
-d    pouze kontrola syntaxe

Vstup:     stdin
Vystup:    stdout
=====================================================
```

./project –t <test.dl

```
Reducing by rule #11
Reducing by rule #7
Reducing by rule #11
Reducing by rule #7
Reducing by rule #10
Reducing by rule #11
Reducing by rule #7
Reducing by rule #6
Reducing by rule #3
Reducing by rule #2
Reducing by rule #3
Reducing by rule #4
Reducing by rule #4
Reducing by rule #2
Reducing by rule #1
```

+ tabulka mezikódu – viz bod B

./project –v <test.dl

```
Reducing by rule #11, line #4   (INTEGER)
Reducing by rule #7, line #4    (ASSIGNMENT)
Reducing by rule #11, line #5   (INTEGER)
Reducing by rule #7, line #5    (ASSIGNMENT)
Reducing by rule #10, line #8   (NAME)
Reducing by rule #11, line #9   (INTEGER)
Reducing by rule #7, line #10   (ASSIGNMENT)
Reducing by rule #6, line #10   (IF expression THEN statement)
Reducing by rule #3, line #10   (STATEMENT)
Reducing by rule #2, line #10   (BEGIN statement_list END)
Reducing by rule #3, line #11   (STATEMENT)
Reducing by rule #4, line #11   (STATEMENT_LIST)
Reducing by rule #4, line #11   (STATEMENT_LIST)
Reducing by rule #2, line #11   (BEGIN statement_list END)
Reducing by rule #1, line #11   (PROGRAM)
```

+ tabulka mezikódu – viz bod B

B Příklady chování překladače

Zdrojový program 1	Mezikód

`program test;`
`begin`
`  first:=0;`
`  second:=20;`
`  begin`
`   if first then third:=15`
`  end`
`end.`
	`0: ( MOV, 0, NULL, first )`
	`1: ( MOV, 20, NULL, second )`
	`2: (  JZ, first, NULL, 4 )`
	`3: ( MOV, 15, NULL, third )`
	`4: (HALT, NULL, NULL, NULL)`

Zdrojový program 2	Mezikód

`program test;`
`begin`
`; single if statement`
`if x then y:=1;`
 	
`; recursive if statements`
`if x then if y then z:=3;`
 	
`if x then if y then if z then k:=1;`
 	
`; expression as a condition in if statement`
`if x+2 and x then p:=1;`
 	
`; block as a branch of if statement`
`if x+2 then`
`  begin`
`   p:=1`
`  end`
`end.`
	`0: (  JZ, x, NULL, 2 )`
	`1: ( MOV, 1, NULL, y )`
	`2: (  JZ, x, NULL, 5 )`
	`3: (  JZ, y, NULL, 5 )`
	`4: ( MOV, 3, NULL, z )`
	`5: (  JZ, x, NULL, 9 )`
	`6: (  JZ, y, NULL, 9 )`
	`7: (  JZ, z, NULL, 9 )`
	`8: ( MOV, 1, NULL, k )`
	`9: ( AND, 2, x, t_0 )`
	`10: ( ADD, x, t_0, t_1 )`
	`11: (  JZ, t_1, NULL, 13 )`
	`12: ( MOV, 1, NULL, p )`
	`13: ( ADD, x, 2, t_2 )`
	`14: (  JZ, t_2, NULL, 16 )`
	`15: ( MOV, 1, NULL, p )`
	`16: (HALT, NULL, NULL, NULL)`

Zdrojový program 3	Mezikód

`; test for DL assignment statement`
`program test;`
 	
`begin`
`; simple assignment`
`  y:=1;`
`; variable assignment`
`  x:=y;`
`; more complex expression assignment`
`  x:= x + y and x`
`end.`
	`0: ( MOV, 1, NULL, y )`
	`1: ( MOV, y, NULL, x )`
	`2: ( AND, y, x, t_0 )`
	`3: ( ADD, x, t_0, t_1 )`
	`4: ( MOV, t_1, NULL, x )`
	`5: (HALT, NULL, NULL, NULL)`

Zdrojový program 4	Mezikód

`; DL program declaration test`
`program test;`
`begin`
 	
`  x:=0`
`end.`
	`0: ( MOV, 0, NULL, x )`
	`1: (HALT, NULL, NULL, NULL)`

Zdrojový program 5	Mezikód

`; testing statement list`
`program test;`
 	
`begin`
` x:= 1;`
` y:= x;`
` s:= 1 + 2 and 3;`
 	
` begin`
`  c:= 1`
` end;`
 	
` begin`
`  c:= 1;`
`  b:= 2`
` end;`
`if x then y:=2`
`end.`
	`0: ( MOV, 1, NULL, x )`
	`1: ( MOV, x, NULL, y )`
	`2: ( AND, 2, 3, t_0 )`
	`3: ( ADD, 1, t_0, t_1 )`
	`4: ( MOV, t_1, NULL, s )`
	`5: ( MOV, 1, NULL, c )`
	`6: ( MOV, 1, NULL, c )`
	`7: ( MOV, 2, NULL, b )`
	`8: (  JZ, x, NULL, 10 )`
	`9: ( MOV, 2, NULL, y )`
	`10: (HALT, NULL, NULL, NULL)`

Zdrojový program 6	Mezikód

`; testing DL expressions`
`program test;`
`begin`
 	
`; expressions with numbers only`
`x:=1;`
`x:=1+2;`
`x:=1 and 2;`
`x:=1 + 2 and 3;`
`x:=1 and 2 + 3;`
 	
`; expressions with vars`
`x:= a;`
`x:= a+b;`
`x:= a and b;`
`x:= a + b and c;`
`x:= a and b + c;`
 	
`; mixed case`
`x:= a + 4;`
`x:= 0 and b;`
`x:= 1 + b and 3;`
`x:= a and 2 + c`
 	
`end.`
	`0: ( MOV, 1, NULL, x )`
	`1: ( ADD, 1, 2, t_0 )`
	`2: ( MOV, t_0, NULL, x )`
	`3: ( AND, 1, 2, t_1 )`
	`4: ( MOV, t_1, NULL, x )`
	`5: ( AND, 2, 3, t_2 )`
	`6: ( ADD, 1, t_2, t_3 )`
	`7: ( MOV, t_3, NULL, x )`
	`8: ( AND, 1, 2, t_4 )`
	`9: ( ADD, t_4, 3, t_5 )`
	`10: ( MOV, t_5, NULL, x )`
	`11: ( MOV, a, NULL, x )`
	`12: ( ADD, a, b, t_6 )`
	`13: ( MOV, t_6, NULL, x )`
	`14: ( AND, a, b, t_7 )`
	`15: ( MOV, t_7, NULL, x )`
	`16: ( AND, b, c, t_8 )`
	`17: ( ADD, a, t_8, t_9 )`
	`18: ( MOV, t_9, NULL, x )`
	`19: ( AND, a, b, t_10 )`
	`20: ( ADD, t_10, c, t_11 )`
	`21: ( MOV, t_11, NULL, x )`
	`22: ( ADD, a, 4, t_12 )`
	`23: ( MOV, t_12, NULL, x )`
	`24: ( AND, 0, b, t_13 )`
	`25: ( MOV, t_13, NULL, x )`
	`26: ( AND, b, 3, t_14 )`
	`27: ( ADD, 1, t_14, t_15 )`
	`28: ( MOV, t_15, NULL, x )`
	`29: ( AND, a, 2, t_16 )`
	`30: ( ADD, t_16, c, t_17 )`
	`31: ( MOV, t_17, NULL, x )`
	`32: (HALT, NULL, NULL, NULL)`

Zdrojový program 7	Mezikód

`; testing BLOCK DL statement`
`program test;`
`begin`
 	
`; just a block`
` begin`
`  x:=0`
` end;`
 	
`; a block with statement_list`
` begin`
`  x:=0;`
`  y:=0;`
`  if x then c:=1;`
`  begin`
`   x:=2`
`  end`
` end;`
 	
`; multiple included blocks`
` begin`
`  begin`
`   begin`
`    y:=2`
`   end`
`  end`
` end`
 	
`end.`
	`0: ( MOV, 0, NULL, x )`
	`1: ( MOV, 0, NULL, x )`
	`2: ( MOV, 0, NULL, y )`
	`3: (  JZ, x, NULL, 5 )`
	`4: ( MOV, 1, NULL, c )`
	`5: ( MOV, 2, NULL, x )`
	`6: ( MOV, 2, NULL, y )`
	`7: (HALT, NULL, NULL, NULL)`

Zdrojový program 8	Mezikód

`program test;`
 	
`begin`
` x:=1;`
` y:=1 + x;`
` if x then`
 	
`  begin`
`   if y then x:=23`
`  end;`
` x:=y and 0`
`end.`
	`0: ( MOV, 1, NULL, x )`
	`1: ( ADD, 1, x, t_0 )`
	`2: ( MOV, t_0, NULL, y )`
	`3: (  JZ, x, NULL, 6 )`
	`4: (  JZ, y, NULL, 6 )`
	`5: ( MOV, 23, NULL, x )`
	`6: ( AND, y, 0, t_1 )`
	`7: ( MOV, t_1, NULL, x )`
	`8: (HALT, NULL, NULL, NULL)`

C Organizační poznámky:

Kompletní projekty odevzdáte v komprimované podobě do školního e-Learning_u:

https://elearning.vspj.cz/mod/assign/view.php?id=70512

Archivní soubor bude obsahovat:

1) Lexikální analyzátor (*.l),
2) Syntaktický analyzátor (*.y),
3) Funkční *makefile*,
4) Tři **vlastní unikátní testovací zdrojové kódy**, každý o rozsahu nejméně 15 řádků, pokrývající dohromady všechna gramatická pravidla,
5) Projektovou dokumentaci, obsahující:
   a) Zadání
   b) Analýzu problému
   c) Teoretický i technický návrh řešení včetně zdůvodnění
   d) Otestování a zdůvodnění realizované funkčnosti
   e) Ukázku vstupů a výstupů překladače (výpis, screenshot apod.)
   f) Celkové zhodnocení projektu

Podmínky pro akceptování semestrálního projektu:

a) Utilitou *make* musí být možno zkompilovat překladač do spustitelné podoby v prostředí *cygwin*.

b) Překladač musí pracovat alespoň v režimu syntaktického analyzátoru - přepínač –d).

c) Řešitel dodá vlastní testovací zdrojové kódy, odlišné od svých spolužáků. V případě shodných testovacích příkladů bude za jejich autora pokládán student s dřívějším datem odevzdání. Dalším předkladatelům shodných testů nebude projekt uznán.

V případě řádného naplnění bodů a) – c) dále hodnocena a bodována:

- Funkčnost a technická úroveň návrhu překladače.
- Obsah a forma technické zprávy.

Projekty, které se nepodaří z jakéhokoli důvodu sestavit pomocí *makefile*, budou z hodnocení vyřazeny!

Nejpozdější termín odevzdání:

neděle, 26. 5. 2019, 23:50.