Języki formalne i techniki translacji

Laboratorium - Projekt

Termin oddania: przed 1 czerwca 2014
Wysłanie do wykładowcy: przed 23:59 15 czerwca 2014

Używając BISON-a i LEX-a napisz kompilator prostego języka imperatywnego do kodu maszyny rejestrowej. Specyfikacja języka i maszyny jest zamieszczona poniżej. Kompilator powinien sygnalizować miejsce i rodzaj błędu (np. druga deklaracja zmiennej, użycie niezadeklarowanej zmiennej, ...), a w przypadku braku błędów zwracać kod na maszynę rejestrową. Kod wynikowy powinien wykonywać się jak najszybciej (w miarę optymalnie, mnożenie i dzielenie powinny być wykonywane w czasie logarytmicznym w stosunku do wartości argumentów).

Program powinien być oddany z plikiem `Makefile` kompilującym go oraz z plikiem `README` opisującym dostarczone pliki i sposób użycia kompilatora. (Przy przesyłaniu do wykładowcy powinien być spakowany programem zip i nazwany numerem indeksu studenta.)

Prosty język imperatywny

```
program       -> CONST cdeclarations VAR vdeclarations BEGIN commands END

cdeclarations -> cdeclarations identifier=num
               |

vdeclarations -> vdeclarations identifier
               |

commands    -> commands command
             |

command     -> identifier := expression;
             | IF condition THEN commands ELSE commands END
             | WHILE condition DO commands END
             | READ identifier;
             | WRITE identifier;

expression  -> num
             | identifier
             | identifier + identifier
             | identifier - identifier
             | identifier * identifier
             | identifier / identifier
             | identifier % identifier

condition   -> identifier == identifier
             | identifier != identifier
             | identifier < identifier
             | identifier > identifier
             | identifier <= identifier
             | identifier >= identifier
```

Język powinien być zgodny z powyższą gramatyką i spełniać następujące warunki:

1. `identifier` jest opisany wyrażeniem regularnym `[_a-z]+`;
2. `num` jest liczbą naturalną w zapisie dziesiętnym (nie ma ograniczeń na wielkość liczby);
3. działania arytmetyczne są wykonywane na liczbach naturalnych, w szczególności $a-b = \max\{a-b, 0\}$, dzielenie przez zero daje wynik 0 i resztę także 0;
4. instrukcja `READ` czyta wartość z zewnątrz i podstawia pod zmienną, a `WRITE` wypisuje wartość zmiennej/liczby na zewnątrz,
5. rozróżniamy małe i duże litery;
6. w programie można użyć komentarzy postaci: `(* komentarz *)`, które nie mogą być zagnieżdżone.

Maszyna rejestrowa Maszyna rejestrowa składa się ze specjalnego rejestru a, licznika rozkazów k oraz ciągu komórek pamięci p_i, dla $i = 0, 1, 2, ...$ (komórki o numerach 0, 1 i 2 są szybsze niż pozostałe - patrz czasy wykonania w tabelce). Maszyna pracuje na liczbach naturalnych (wynikiem odejmowania większej liczby od mniejszej jest 0). Program maszyny składa się z ciągu rozkazów, który niejawnie numerujemy od zera. W kolejnych krokach wykonujemy zawsze rozkaz o numerze k aż napotkamy instrukcję `HALT`. Początkowa zawartość rejestrów i komórek pamięci jest nieokreślona, a licznik rozkazów k ma wartość 0. Poniżej jest lista rozkazów wraz z ich interpretacją i czasem wykonania (ilością kroków potrzebnych do jego wykonania):

Rozkaz	Interpretacja	Czas
`SCAN i`	pobraną liczbę zapisuje w komórce p_i oraz $k \leftarrow k+1$	100
`PRINT i`	wyświetla zawartość komórki p_i oraz $k \leftarrow k+1$	100
`LOAD i`	$a \leftarrow p_i$ oraz $k \leftarrow k+1$	jeśli $i < 3$ to 10 wpp. 100
`STORE i`	$p_i \leftarrow a$ oraz $k \leftarrow k+1$	jeśli $i < 3$ to 10 wpp. 100
`ADD i`	$a \leftarrow a + p_i$ oraz $k \leftarrow k+1$	jeśli $i < 3$ to 10 wpp. 100
`SUB i`	$a \leftarrow \max\{a - p_i, 0\}$ oraz $k \leftarrow k+1$	jeśli $i < 3$ to 10 wpp. 100
`SHR`	$a \leftarrow \lfloor a/2 \rfloor$ oraz $k \leftarrow k+1$	1
`SHL`	$a \leftarrow 2 * a$ oraz $k \leftarrow k+1$	1
`INC`	$a \leftarrow a+1$ oraz $k \leftarrow k+1$	1
`DEC`	$a \leftarrow \max\{a-1, 0\}$ oraz $k \leftarrow k+1$	1
`ZERO`	$a \leftarrow 0$ oraz $k \leftarrow k+1$	1
`JUMP i`	$k \leftarrow i$	1
`JZ i`	jeśli $a = 0$ to $k \leftarrow i$, wpp. $k \leftarrow k+1$	1
`JG i`	jeśli $a > 0$ to $k \leftarrow i$, wpp. $k \leftarrow k+1$	1
`JODD i`	jeśli a nieparzyste to $k \leftarrow i$, wpp. $k \leftarrow k+1$	1
`HALT`	zatrzymaj program	0

Przejście do nieistniejącego rozkazu jest traktowane jako błąd.

Przykładowe kody programów i odpowiadające im kody maszyny rejestrowej

```
1  CONST
2    dwa=2 zero=0 jeden=1
3  VAR
4      a b
5  BEGIN
6      a:=1000;
7      WHILE a>zero DO
8          b:=a/dwa;
9          b:=dwa*b;
10         IF a>b THEN
11             WRITE jeden;
12         ELSE
13             WRITE zero;
14         END
15         a:=a/dwa;
16     END
17 END
```

```
0  ZERO
1  INC
2  SHL
3  INC
4  SHL
5  INC
6  SHL
7  INC
8  SHL
9  INC
10 SHL
11 SHL
12 INC
13 SHL
14 SHL
15 SHL
16 STORE 0
17 JZ 28
18 SHR
19 SHL
20 STORE 1
21 LOAD 0
22 SUB 1
23 STORE 2
24 PRINT 2
25 LOAD 0
26 SHR
27 JUMP 16
28 HALT
```

```
1  (* NWD(a,b) *)
2  CONST
3  VAR
4      a b
5  BEGIN
6      READ a;
7      READ b;
8      WHILE a!=b DO
9          IF a<b THEN (* a <-> b *)
10             a:=a+b;
11             b:=a-b;
12             a:=a-b;
13         ELSE
14         END
15         a:=a-b;
16     END
17     WRITE a;
18 END
```

```
0  SCAN 0
1  SCAN 1
2  LOAD 0
3  SUB 1
4  STORE 2
5  LOAD 1
6  SUB 0
7  ADD 2
8  JZ 24
9  LOAD 1
10 SUB 0
11 JZ 20
12 LOAD 0
13 ADD 1
14 STORE 0
15 SUB 1
16 STORE 1
17 LOAD 0
18 SUB 1
19 STORE 0
20 LOAD 0
21 SUB 1
22 STORE 0
23 JUMP 2
24 PRINT 0
25 HALT
```

Optymalność wykonywania mnożenia i dzielenia Dla następującego programu

```
1  (* Rozklad liczby na czynniki pierwsze *)
2  CONST zero=0 jeden=1
3  VAR n m reszta potega dzielnik
4  BEGIN
5      READ n;
6      dzielnik:=2;
7      m:=dzielnik*dzielnik;
8      WHILE n>=m DO
9          potega:=0;
10         reszta:=n%dzielnik;
11         WHILE reszta==zero DO
12             n:=n/dzielnik;
13             potega:=potega+jeden;
14             reszta:=n%dzielnik;
15         END
16         IF potega>zero THEN (* czy znaleziono dzielnik *)
17             WRITE dzielnik;
18             WRITE potega;
19         ELSE
20             dzielnik:=dzielnik+jeden;
21             m:=dzielnik*dzielnik;
22         END
23     END
24     IF n!=jeden THEN (* ostatni dzielnik *)
25         WRITE n;
26         WRITE jeden;
27     ELSE
28     END
29 END
```

kod wynikowy na załączonej maszynie powinien działać w czasie porównywalnym do poniższych wyników

```
Uruchamianie programu.
? 1234567890
> 2
> 1
> 3
> 2
> 5
> 1
> 3607
> 1
> 3803
> 1
Skończono program (wykonano kroków: 4******).
----------
Uruchamianie programu.
? 12345678901
> 857
> 1
> 14405693
> 1
Skończono program (wykonano kroków: 5******).
----------
Uruchamianie programu.
? 12345678903
> 3
> 1
> 4115226301
> 1
Skończono program (wykonano kroków: 12*******).
```