Part I

miniC kompajler (MICKO)

Chapter 1

Karakteristike kompajlera

MICKO je kompajler koji prevodi sa jezika miniC (izvorni jezik) na hipotetski asemblerski jezik (ciljni jezik). Ime je nastalo kao akronim od **mi**ni**C** **ko**mpajler. MICKO je napravljen isključivo u edukativne svrhe. Implementiran je pomoću generatora skenera i parsera: *flex* i *bison* [**?**]. Sav dodatni kod je napisan na programskom jeziku C. Tokom kompajliranja, kompajler prijavljuje korisniku eventualno postojanje grešaka u izvornom miniC programu.

Figure 1.1: MICKO kompajler

MICKO je vrlo jednostavna implementacija miniC jezika, nastala kao odgovor na potrebe kursa *Programski prevodioci*, Fakulteta tehničkih nauka u Novom Sadu. Osnovni cilj je edukacija, a ne efikasnost koda (ovo se naročito odnosi na implementaciju tabele simbola).

1.1 Faze kompajliranja

Implementacija miniC kompajlera je podeljena na uobičajene faze kompajliranja: leksičku, sintaksnu, semantičku analizu i generisanje koda. Optimizacija koda nije implementirana.

Leksička analiza je početni deo čitanja i analiziranja programskog teksta. Tekst se čita i deli na simbole programskog jezika (npr. ključna reč, ime promenljive ili broj).

Sintaksna analiza preuzima tokene koji su stvoreni tokom leksičke analize i proverava da li su navedeni u ispravnom redosledu. Ova faza se naziva parsiranje (*parsing*). Tokom parsiranja se pravi stablo koje se naziva stablo parsiranja (*parse tree*) koje odražava strukturu programa.

Semantička analiza proverava konzistentnost programa: npr. da li je promenljiva, koja se koristi, prethodno deklarisana i da li se koristi u skladu sa svojim tipom.

Generisanje međukoda prevodi program na jednostavan međujezik (*intermediate language*) koji je nezavisan od ciljne mašine (*machine-independent*), obično asemblerski jezik [**?**], u ovom slučaju hipotetski asemblerski jezik.

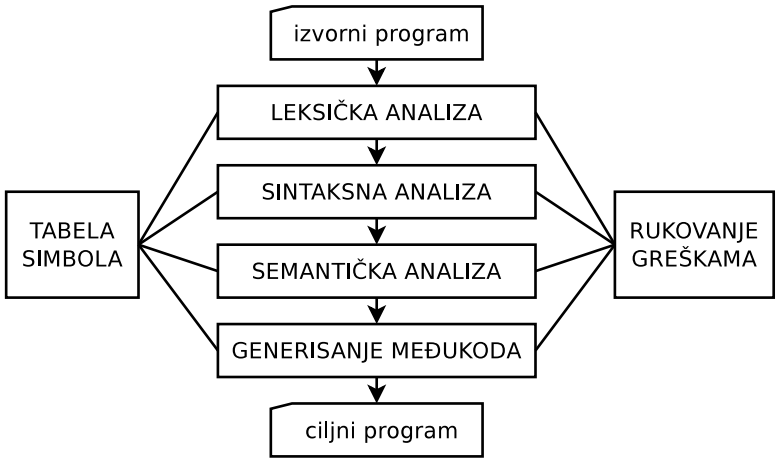

Figure 1.2: Faze kompajliranja

1.2 Tabela simbola

Sve faze kompajliranja koriste tabelu simbola. To je struktura podataka u kojoj se čuvaju sve informacije o svim simbolima koji su prepoznati u toku kompajliranja. Na osnovu ovih informacija moguće je uraditi semantičku analizu i generisanje asemblerskog koda. Na primer, uz ime funkcije, potrebno je čuvati i informaciju o tipu povratne vrednosti te funkcije (da bi se mogla uraditi provera tipova u `return` iskazu), broj parametara i tipovi parametara (da bi se mogla proveriti ispravnost broja argumenata i njihovih tipova prilikom poziva te funkcije).

1.3 Greške u kompajliranju

Tokom svih faza kompajliranja moguća je pojava greške u izvornom kodu, pa kompajler treba da pomoge programeru da ih identifikuje i locira. Programi mogu sadržati greške na različitim nivoima:

- leksičke (npr. pogrešno napisano ime, ključna reč ili operator),
- sintaksne (npr. relacioni izraz sa nepotpunim parom zagrada) ili
- semantičke (npr. operator primenjen na nekompatibilni operand).

Kompajler treba da:

- saopšti prisustvo grešaka jasno i ispravno,
- da se oporavi od greške dovoljno brzo da bi mogao da detektuje naredne greške i
- da ne usporava bitno obradu ispravnih programa.

1.4 Primer prevođenja

Kada se miniC kompajleru prosledi datoteka `abs.mc` (listing 1.1) sa funkcijom `abs()` koja računa apsolutnu vrednost broja, kompajler izgeneriše datoteku `abs.asm` koja sadrži ekvivalentan kod, napisan na hipotetskom asemblerskom jeziku (listing 1.2):

```
$./micko <abs.mc
$
```

Listing 1.1: `abs.mc`

```
1  int abs(int i) {
2    int res;
3    if(i < 0)
4      res = 0 - i;
5    else
6      res = i;
7    return res;
8  }
9
10 int main() {
11   return abs(-5);
12 }
```

Listing 1.2: `abs.asm`

```
abs:
        PUSH  %14
        MOV   %15,%14
        SUBS  %15,$4,%15
@abs_body:
@if1:
        CMPS  8(%14),$0
        JGES  @false1
@true1:
        SUBS  $0,8(%14),%0
        MOV   %0,-4(%14)
        JMP   @exit1
@false1:
        MOV   8(%14),-4(%14)
@exit1:
        MOV   -4(%14),%13
        JMP   @abs_exit
@abs_exit:
        MOV   %14,%15
        POP   %14
        RET
main:
        PUSH  %14
        MOV   %15,%14
@main_body:
        PUSH  $-5
        CALL  abs
        ADDS  %15,$4,%15
        MOV   %13,%0
        MOV   %0,%13
        JMP   @main_exit
@main_exit:
        MOV   %14,%15
        POP   %14
        RET
```

Pošto datoteka `abs.mc` sadrži leksički, sintaksno i semantički ispravan kod, kompajler je usprešno preveo program i izgenerisao datoteku sa ekvivalentnim asemblerskim programom. Ukoliko bi se kompajleru prosledila datoteka sa neispravanim kodom, on bi korisniku prijavio grešku. Za greške koje je kompajler predvideo, prijaviće tačnu lokaciju i detaljan opis greške i nastaviće parsiranje (npr. ulaz 2 i ulaz 4). Za greške koje nije predvideo, kompajler će ispisati samo poruku da postoji greška i završiće kompajliranje (npr. ulaz 1 i ulaz 3).

1.4.1 Neispravan ulaz 1

Na primer, ako bi se miniC kompajleru prosledio prethodni program, ali sa izmenjenom relacijom u `if` iskazu (linija 3, operator < zamenjen operatorom >):

```
if(i > 0)
  ...
```

kompajler bi prijavio leksičku grešku (a nakon toga i sintaksnu), jer miniC jezik ne podržava relacioni operator >:

```
$./micko <abs.mc
line 3: LEXICAL ERROR on char >
line 3: ERROR: syntax error
$
```

1.4.2 Neispravan ulaz 2

Ukoliko bi se miniC kompajleru prosledio `return` iskaz kojem nedostaje separator "`;`" na kraju iskaza (linija 7):

```
return res
```

kompajler bi prijavio sintaksnu grešku, ali bi se i oporavio od nje i nastavio parsiranje:

```
$./micko <abs.mc
line 8: ERROR: Missing ';' in return statement
$
```

1.4.3 Neispravan ulaz 3

Međutim, ukoliko bi mu se prosledilo zaglavlje funkcije u kojem iza definicije parametra nedostaje zatvorena mala zagrada (linija 1):

```
int abs(int i {
```

kompajler bi prijavio sintaksnu grešku i završio parsiranje, jer nema oporavak od ovakve sintaksne greške:

```
$./micko <abs.mc
line 1: ERROR: syntax error
$
```

1.4.4 Neispravan ulaz 4

Ako bi se u `abs` programu, u liniji 6, izmenilo ime promenljive iz `res` u `re`:

```
re = i;
```

kompajler bi prijavio semantičku grešku na poziciji sa leve strane znaka jednako (linija 6):

```
$./micko <abs.mc
line 6: ERROR: invalid lvalue 're' in assignment
$
```