Metody kompilacji

Laboratorium 5

Piotr Błaszyński

18 marca 2019

Zadania (wyjaśnienie w dalszej części dokumentu):

- jeżeli jeszcze nie ma, to dodać opcję kompilacji -std=c++11 do Makefile (w miejscach gdzie używamy g++),
- przygotować regułę dla symbolu nieterminalnego przedstawiającego przypisanie (jeżeli wcześniej tego nie zrobiliśmy),
- utworzyć tablicę symboli (zapisywać w niej wszystkie identyfikatory),
- generować zmienne tymczasowe do przechowywania wyników trójek,
- dla każdej trójki generować 4 linie kodu asemblera i przechowywać je w wektorze,
- dopisać kod zapisujący linie z vectora do pliku *yyout*, wywołać ten kod po *yyparse*,
- po *yyparse* zapisać tablicę symboli do pliku *symbols.txt*,
- zapisać symbole z tablicy symboli przed kodem w bloku danych.

Tablica symboli zawiera wszystkie identyfikatory, również identyfikatory zmiennych tymczasowych generowanych przez kompilator. Powinna to być tablica haszująca (w C++ *std* :: *map*). Tablica symboli pozwala przechować i wyszukiwać na podstawie identyfikatora informacje o zmiennych: typ zmiennej, miejsce przechowywania, ew. rozmiary tablicy. Należy ją na koniec działania kompilatora zapisać do pliku *symbols.txt*.

Zmienne tymczasowe będą służyć do przechowywania w pamięci wyników pośrednich obliczeń. Kolejne zmienne powinny być numerowane. Nie powinno byc możliwości użycia zmiennej o takiej samej nazwie jak zmienna tymczasowa w normalnym kodzie.

Przykładowy kod generowany dla trójki ma postać:

```
li $t0, 27
lw $t1, x
sub $t0, $t0, $t1
sw $t0, result15
```

Rejestry $\$t0 - \$t7$ to tak zwane rejestry tymczasowe. Wartość do takiego rejestru można wstawić przy pomocy instrukcji (na razie) *li* lub *lw*. Instrukcja *li* wstawia wartość liczbową (bezpośrednią). Instrukcja *lw* wstawia wartość spod adresu wskzywanego nazwą zmiennej. Instrukcja *sub* wykonuje odejmowanie, instrukcja *add* dodawanie, *mul* mnożenie, *div* dzielenie a *sw* wstawia wartość z rejestru do komórki pamięci. W przypadku operacji arytmetycznych operacja jest wykonywana na dwóch ostatnich rejestrach a wynik jest przechowywany w pierwszym rejestrze. Uogólniony kod generowany dla trójki ma postać (podkreślenia wskazują miejsca do wypełnienia):

```
l_ $t0, __
l_ $t1, __
___ $t0, $t0, $t1
sw $t0, ____
```

Kod dla przypisania można skrócić (formę skróconą pozostawiam do wymyślenia).

Część nagłówkowa (blok danych) służyć będzie do zarezerwowania miejsca na zmienne (w przyszłości również stałe). Blok danych zaczyna się od dyrektywy *.data* Format bloku danych:

```
nazwa: typ wartosc
```

Przykłady:

```
x:          .word   0
arr:        .space  40
napis:      .asciiz  "Napis na ekran"
f:          .float 3.14
```

Kod zaczyna się od dyrektywy *.text*. Komentarz w kodzie asemblera jest oznaczany znakiem '#'.

Przykładowy kod dla wyrażenia $(x = 3; z = 5 + x * 2;)$:

```
.data
        x:      .word   0
        z:      .word   0
  result1:      .word   0
  result2:      .word   0
.text
        li $t0, 3
        sw $t0, x
        lw $t0, x
        li $t1, 2
        mul $t0, $t0, $t1
        sw $t0, result1

        li $t0, 5
        lw $t1, result1
        add $t0, $t0, $t1
        sw $t0, result2

        lw $t0, result2
        sw $t0, z
```