Metody kompilacji

Laboratorium 9

Piotr Błaszyński

21 kwietnia 2017

Zadania (wyjaśnienie w dalszej części dokumentu):

- dodać kompilację deklaracji i wywołania prostych funkcji,
  - reguły gramatyki,
  - generowanie kodu wynikowego,
- dodać kompilację instrukcji tworzenia dynamicznych tablic jednowymiarowych,
  - reguły gramatyki,
  - generowanie kodu wynikowego,

Funkcje w MIPS realizuje się przy pomocy instrukcji *jal*, *jr* i rejestru $\$ra$.

Przykładowy kod wywołujący prostą funkcję (ostatnia instrukcja *syscall* kończy działanie programu, żeby nie wykonywać jeszcze raz kodu zawartego w funkcji):

```
.text
        li $t0, 42
        jal myfoo
        li $v0, 10
        syscall

myfoo:
        li $t0, 88
        jr $ra
```

Alternatywna wersja z funkcją main (trzeba do niej samodzielnie skoczyć):

```
.text
        b main
myfoo:
        li $t0, 88
        jr $ra

main:
        li $t0, 42
        jal myfoo
        li $t1, 88
```

Reguły gramatyki dla funkcji należy opracować samodzielnie (zgodnie z projektem własnego języka). Możliwe jest wykonywanie na końcu działania kompilatora dodatkowej pętli iterującej po wygenerowanym kodzie i liście funkcji i zamiana tych nazw na właściwe etykiety. Można też etykiety wstawiać od razu w trakcie kompilacji, a na koniec należy tylko zweryfikować, czy wszystkie wywoływane funkcje istnieją. Ewentualne parametry do funkcji (ponad wymagania) można przekazywać poprzez rejestry $\$a0$ i $\$a1$, wartość można zwracać poprzez rejestr $\$v0$. W przypadku kompilatora budowanego zgodnie z wcześniejszymi uproszczonymi założeniami (wartości nie są przechowywane w rejestrach a tylko do nich wstawiane na moment wykonywania obliczeń) nie jest również konieczne przechowywanie wartości rejestrów na stosie. Jeżeli jednak zaszłaby taka konieczność, funkcja na starcie przesuwa wartość w rejestrze $\$sp$ o -4 (rozmiar przechowywanego rejestru) i ładuje pod adres wskazywany przez ten rejestr wartość z rejestru do zapamiętania. Na końcu funkcji jest „lustrzany" kod, który pobiera wartości rejestrów ze stosu i zwiększa wartość w rejestrze $\$sp$.

Dynamiczne tablice w MIPS tworzy się przy pomocy wywołania systemowego sbrk. Reguły gramatyki należy opracować samodzielnie.
Dla przykładowego kodu w C++:

```
int n=20;
int *x = new int[10];
int *y = new int[n];
```

Należy wygenerować kod:

```
.data
        n:      .word 0
        x:      .word 0
        y:      .word 0
.text
        li $t0, 20
        sw $t0, n

        li $v0, 9
        li $a0, 10
        syscall
        sw $v0, x

        li $v0, 9
        lw $a0, n
        syscall
        sw $v0, y
```

Odwoływanie się do komórek tak utworzonych tablic realizuje się identycznie jak odwołania do tablic o statycznym rozmiarze.