

Projekt 5 – część druga: generowanie kodu, dokończenie

1. WSTĘP.

Celem niniejszego projektu jest generowanie kodu dla języka C-. Jest to dokończenie projektu z zeszłego tygodnia, w którym rozpatrywaliśmy tylko pewien fragment tego języka. Wygenerowany kod powinien być zgodny z operacyjną semantyką języka C- opisaną tutaj:

<http://www.math.us.edu.pl/~pgladki/teaching/2011-2012/tk-cminusminus.html#opsem>

1.1. Docelowa architektura. Docelową architekturą dla tego projektu będzie procesor MIPS R2000. Wasz kompilator wygeneruje kod dla tego procesora; kod będzie wykonywany za pomocą symulatora SPIM

<http://spimsimulator.sourceforge.net/>

dostępnego w wersji na Windows, Maca i Linuksa.

1.2. Operacje wejścia i wyjścia. Programy testujące będą wykorzystywały następujące dwie funkcje celem wydrukowania rezultatów obliczeń:

```
extern void print_int(int x);
```

```
extern void print_string(char x[]);
```

Na przykład, program testujący może wyglądać tak:

```
extern void print_int(int x);
```

```
extern void print_string(char x[]);
```

```
void main( void ) { print_int(12345); print_string("\n"); }
```

Poza wygenerowanym kodem powinniście też wygenerować sekwencję instrukcji dla funkcji *print_int()*, *print_string()* tak, jak jest to opisane w notatkach o tłumaczeniu kodu na assembler MIPSa:

<http://www.math.us.edu.pl/~pgladki/teaching/2011-2012/tk-3addr2spim.pdf>

1.3. Testowanie programów. Możecie zobaczyć “spodziewane wyjście” dla Waszych programów testujących najpierw kompilując je w *gcc* z definicjami dla *print_int()*, *print_string()* dostarczonymi w osobnym pliku i podlinkowanymi w oczywisty sposób.

1.4. Dodatkowa dokumentacja. Następujące dokumenty mogą okazać się przydatne w pracy:

(1) Instrukcja dla kodu pośredniego w C-:

<http://www.math.us.edu.pl/~pgladki/teaching/2011-2012/tk-intcode.html>

(2) notatki o tłumaczeniu kodu na assembler MIPSa:

<http://www.math.us.edu.pl/~pgladki/teaching/2011-2012/tk-3addr2spim.pdf>

(3) dokumentacja do SIMPa:

<http://spimsimulator.sourceforge.net/>

2. WYWOŁYWANIE PROGRAMU.

Plik wykonywalny powinien się nazywać *compile* i powinien czytać z pliku *stdin* i zapisywać do pliku *stdout*. Komunikaty o błędach powinny być zapisane do pliku *stderr*. Razem z zadaniem dostarczcie wszystkie źródłówki.