Zadanie domowe 5

Generowanie kodu.

W tym zadaniu zajmiemy się generowaniem kodu w asemblerze procesora MIPS 2000. Zacznijcie od przeczytania wprowadzenia do MIPS i Spim autorstwa Larusa (jest to Appendix A z podręcznika Pattersona i Hennessy'ego *Computer Organization and Design*):

`www.math.us.edu.pl/~pgladki/teaching/2012-2013/tk2_spim.pdf`

Następnie zainstalujcie Spima, którego możecie ściągnąć stąd:

`http://spimsimulator.sourceforge.net`

Załóżcie nowy katalog **lab5** i skopiujcie do niego całą zawartość katalogu **lab4**. Następnie ściągnijcie plik

`www.math.us.edu.pl/~pgladki/teaching/2012-2013/tk2_lab5.zip`

i rozpakujcie go; w szczególności znajdziecie tam pliki codegen.sml, compile.sml, mips.sig i mips.sml. Przeczytajcie ze zrozumieniem pliki mips.sig i mips.sml. Następnie updatujcie sources.cm przez dodanie mips.sig, mips.sml i codegen.sml. Przeczytajcie ze zrozumieniem plik codegen.sml.

Zaimplementujcie funkcję gen_func.

- skopiujcie rejestry calee-save i ra do nowych tymczasowych (można je utworzyć za pomocą Mips.newReg());
- skopiujcie rejestry argumentów do nowych tymczasowych (w Funie jest tylko jeden);
- utwórzcie nowe środowisko przez dodanie parametrów funkcji na szczyt środowiska globalnych nazw funkcji (w Fun jest tylko jeden);
- wygenerujcie kod dla wyrażeń objętych funkcją dla tego nowego środowiska;
- skopiujcie rezultat do rejestru zwracającego wartość;
- skopiujcie tymczasowe rejestry calee-save i ra z powrotem do rejestrów calee-save;
- wydajcie etykietę kończącą.

Na tym etapie nie powinniście martwić się dodatkowymi instrukcjami Move, które się wygenerują – prawie wszystkie zostaną usunięte po właściwej alokacji rejestrów.

Zaimplementujcie funkcję gen_exp.

- dobrym pomysłem jest pozbawienie argumentów przekazywanych do gen_exp ze wszystkich konstruktorów Pos i Con (dlaczego?);
- dla każdego rodzaju wyrażenia wygenerujcie odpowiadający kod do rejestru wyjściowego;
- możecie dość swobodnie używać Mips.newReg() dla tworzenia tymczasowych rejestrów;
- dla wyrażeń Ref i Tuple wywołajcie alloc dla zarezerwowania odpowiedniej liczby słów, a następnie przechowajcie wartości w pamięci;
- dla wyrażenia <>zamiast powyższego wystarczy zastosować 0.

Przetestujcie rezultat wywołując Compile.compile "test.fun". W rezultacie powinniście wygenerować test.fun.noregalloc.s, czyli język asemblera, który nie będzie działał :) ponieważ:

- zawiera rejestry tymczasowe postaci $x1, $x2, które nie są rejestrami MIPSa;
- części prologów i epilogów procedur nie zostały jeszcze uwzględnione (stanie się to dopiero po ukończeniu alokacji rejestrów).

Gdy uda Wam się już doprowadzić wszystko do działania, przeczytajcie uważnie .noregalloc.s i upewnijcie się, że wygląda sensownie – na tym etapie jest to jedyny sposób debuggowania zadania, dopóki nie mamy do dyspozycji alokatora rejestrów.

Prześlijcie mailem pliki README oraz codegen.sml. Termin zadania mija w **piątek, 28 czerwca.** Proszę pamiętać o oznaczeniu maila tagiem `[aghtk2]`.