Laboratorium MRJP - projekt zaliczeniowy

Drugi projekt zaliczeniowy obejmuje implementację kompilatora dla prostego jezyka imperatywnego Latte.

Etapy

Projekt składa się z trzech etapów (daty orientacyjne, wiążące terminy w moodle)

1. Frontend - termin 26.11
2. Generator kodu dla LLVM albo x86. Termin 9 stycznia.
3. (opcjonalnie) Ewentualne rozszerzenia. Termin 21 stycznia.

Prowadzący grupę może wyznaczyć inne terminy, w szczególności wymagać osobistej prezentacji projektu. Niezależnie od dodatkowych wymagań prowadzącego, projekt musi być oddany za pośrednictwem moodle.

Punktacja

Łącznie do zdobycia jest maksymalnie 28 punktów, które liczą się do ostatecznej oceny z przedmiotu. Dla zaliczenia laboratorium (i dopuszczenia do egzaminu) należy oddać zadowalające implementacje wszystkich obowiązkowych elementów i uzyskać łącznie z Zadaniem 1 co najmniej 17p.

Punkty mozna uzyskać za:

1. front-end (4)
2. back-end LLVM (8) **albo** x86 (10)
3. użycie rejestrów i phi zamiast alloc w LLVM - dodatkowo 2p
4. wykorzystanie rejestrów dla x86 - dodatkowo do 5p (na 5p pełna alokacja rejestrów)
5. optymalizacje - do 10p
    - LCSE/GCSE - 3/5p
    - optymalizacja pętli (zmienne indukcyjne i redukcja mocy) 2p
    - function inlining 1-3p (proszę zawczasu przedyskutować z prowadzącym)
6. Rozszerzenia (notacja x/y oznacza punkty za backend w wersji odpowiednio LLVM/x86); szczegółowy opis rozszerzeń zawarty jest

w opisie języka.

1. tablice (1)
2. struktury (2)
3. obiekty (atrybuty, metody, dziedziczenie bez zastępowania metod) - dodatkowo (3/4)
4. metody wirtualne - dodatkowo (3/4), czyli za obiekty z metodami wirtualnymi mozna uzyskać 8/10 punktów.
5. odśmiecanie (2)
6. zarządzanie pamięcią a la Rust (2) (wyklucza się z odśmiecaniem)

Struktury/obiekty oznaczają też zagnieżdżone struktury/obiekty oraz dostęp do pól/metod postaci `foo.bar.baz`.

Żeby uzyskać punkty za tablice i struktury/obiekty wymagane są tablice struktur i tablice jako elementy struktur.

Punkty za frontend są przyznawane pod warunkiem oddania (niezerowego) backendu.

Uwaga: podane liczby punktów są wartościami maksymalnymi, sprawdzający może przydzielić mniejsza liczbę punktów w zależności od jakości rozwiązania i generowanego kodu. Oczekujemy, że kod będzie generowany zgodnie z regułami sztuki poznanymi na zajęciach (albo lepiej).

Spóźnienia, termin poprawkowy

Programy oddane po terminie będą obciążane karą 2p za każdy (rozpoczęty) tydzień opóźnienia. Ostatecznym terminem, po którym programy nie będą przyjmowane (“termin poprawkowy”) jest 7 lutego.

Zasady

Projekt zaliczeniowy ma być pisany samodzielnie. Wszelkie przejawy niesamodzielności będą karane. W szczególności:

- nie wolno oglądać kodu innych studentów, pokazywać, ani w jakikolwiek sposób udostępniać swojego kodu
- wszelkie zapożyczenia powinny być opisane z podaniem źródła; dotyczy to także kodu wygenerowanego/zasugerowanego przez narzędzia AI i pokrewne (VS Code, Copilot, Claude, ChatGPT itp.).

Wymagania techniczne

1. Projekt powinien być oddany w postaci spakowanego archiwum TAR (.tar.gz lub .tgz)
2. W korzeniu projektu muszą się znajdować co najmniej:
   - Plik tekstowy **README** opisujący szczegóły kompilacji i uruchamiania programu, używane narzędzia i biblioteki, zaimplementowane rozszerzenia, strukturę katalogów projektu, ewentualnie odnośniki do bardziej szczegółowej dokumentacji.
   - Plik `Makefile` (lub skrypt `build`) pozwalający na zbudowanie programu.
   - katalog **src** zawierający wyłącznie pliki źródłowe projektu (plus ewentualnie dostarczony przez nas plik Latte.cf); pliki pomocnicze takie jak biblioteki itp powinny być umieszczone w inych katalogach.
3. Program musi się kompilować na students poleceniem `make` (ewentualnie `./build`), uruchamianym w katalogu, w którym rozpakowano archiwum (czyli np. `tar xf foo.tgz; make`).
4. Wszelkie używane biblioteki (poza biblioteką standardową używanego jezyka programowania) muszą być opisane w README
5. Po zbudowaniu kompilatora, w korzeniu musi się znajdować plik wykonywalny o nazwie **latc** (może być skryptem uruchamiającym inne programy)
6. Kompilator musi akceptować wszystkie programy testowe z katalogu **good** i odrzucać ze stosownym komunikatem wszystkie programy z katalogu **bad**.Komunikaty o błędach muszą umożliwiać lokalizację błędu (przez numer linii lub kontekst). Dla rozszerzeń musi akceptować wszystkie programy z odpowiednich podkatalogów **extension**. Uruchomienie poprawnego programu testowego ma dawać wyjście takie jak w odpowiednim pliku **.output** (dla wejścia zawartego w odpowiednim pliku **.input**, o ile istnieje)
7. Gdy kompilator akceptuje program, musi wypisać w pierwszej linii stderr napis OK (`"OK\n"`). Dalsze linie i stdout - dowolne. Kompilator musi się w takiej sytuacji zakończyć z kodem powrotu 0.
8. Gdy kompilator odrzuca program musi wypisać w pierwszej linii stderr napis ERROR (`"ERROR\n"`). Dalsze linie powinny zawierać stosowne informacje o błędach. Kompilator musi się w takiej sytuacji zakończyć z kodem powrotu różnym od 0.
9. Rowiązania wymagające stosowania niestandardowego oprogramowania proszę uzgadniać ze sprawdzającymi.

Generacja kodu LLVM

1. Po zbudowaniu, w korzeniu projektu ma się znajdować program wykonywalny **latc_llvm**
2. Wykonanie **latc_llvm foo/bar/baz.lat** dla poprawnego programu wejściowego **baz.lat** ma stworzyć pliki **baz.ll** (kod LLVM) oraz **baz.bc** (bitkod LLVM wykonywalny przy uzyciu **lli**) w katalogu **foo/bar**.
3. Ewentualne funkcje biblioteczne (**printInt** etc.) należy umieścić w pliku **runtime.bc** w katalogu **lib** (dodatkowo proszę zamieścić jego źródło)

Generacja kodu asemblera

1. Po zbudowaniu, w korzeniu projektu ma się znajdować program wykonywalny **latc_ARCH** gdzie ARCH to x86 lub x86_64
2. Wykonanie **latc_ARCH foo/bar/baz.lat** dla poprawnego programu wejściowego **baz.lat** ma stworzyć pliki **baz.s** (kod asemblera) oraz wykonywalny **baz** w katalogu **foo/bar**.
3. Ewentualne funkcje biblioteczne (**printInt** etc.) należy umieścić w pliku **runtime.o** w katalogu **lib** (dodatkowo proszę zamieścić jego źródło)

Testy

Archiwum z programami testowymi:

lattests240919.tgz

Dodatkowe testy (ułożone przez studentów, całkowicie nieoficjalne).