Wrocław 13.06.2018

# Zadanie projektowe z przedmiotu Organizacja i Architektura Komputerów

## Temat: Prosty jednoprzebiegowy kompilator własnego języka programowania.

**Autorzy:**Jan Venulet 235542  
Piotr Kołeczek 234940

**Prowadzący projekt:**Mgr inż. Dominik Żelazny

# Wstęp teoretyczny.

Program, który stanowi nasze zadanie projektowe to analizator leksykalny wraz z analizatorem składniowym, tak zwanym „parserem”. Pełnią one funkcję analizatora tekstu o wcześniej określonej składni. Nasz język programowania, który został przez nas zaimplementowany wprowadza nową składnie, a co za tym idzie odgórnie wyznaczoną interpretacje ciągów znaków pliku tekstowego. Zamysłem realizacji projektu było stworzenie programu implementowanego w języku wysokiego poziomu C, gdzie znajduje się rdzeń naszego projektu, natomiast w języku niskopoziomowym, takim który bazuje na podstawowych operacjach procesora, zwanym językiem asemblera, napisane zostałyby wstawki realizujące konkretnie zadane funkcje. Program pozwoli na kompilacje, to znaczy w innym wypadku wyjdzie nie wykonując poleceń, jeśli podany do niego plik nie zawiera konstrukcji

nazwa\_pliku.l

, gdzie oczywiście *nazwa\_pliku* jest dowolnym ciągiem znaków.

Kompilator został stworzony w środowisku systemowym LINUX ze względu na wygodę pracy i możliwości rozwoju naszego programu, a sam język programowania jest językiem interpretowanym, czyli język, programy, które zostały napisane w takim języku są kompilowane „w locie”, ponieważ na żądanie użytkownika uruchamiana jest pewna procedura, odpowiedzialna za pewną wykonywaną czynność.

# Plan projektu

**Etap 1:**

* Implementacja arytmetyki czterodziałaniowej
* Operacja kończąca działanie programu

**Etap 2:**

* Zmiana dużych liter na małe
* Odbicie lustrzane stringu
* Operacje na liczbach przekraczających rozmiar rejestru ( np. 1024 bajtowe)
* Strumienie wejścia/wyjścia

**Etap 3:**

* Pętle oraz instrukcje warunkowe
* Deklaracja zmiennych programu

# Przebieg prac.

W analizatorze leksykalnym skaner działa na zasadzie „następny leksem”. Oznacza to, że każdy wprowadzony znak, na który składa się kod jest analizowany oraz interpretowany. Dla zobrazowania tej idei rozpatrzmy prosty przykład, będący równaniem:

Jest to bardzo popularny wzór, opisujący równoważność masy i energii danego ciała. Został on zapisany w tak zwanej notacji matematycznej, który jest rzecz jasna niezrozumiały dla komputera.

Zatem spróbujmy przedstawić to samo równanie w troszkę inny sposób:

Powyższy zapis może stać się zapisem zrozumiałym dla komputera. Dlaczego? Otóż w tym zapisie równania możemy wyszczególnić każdy znak, któremu przypisujemy już jakąś funkcję.

**- „E”** – jest to znak określający lewą stronę równania;

**- „=”** - znak równości;

**- „m”** – jedna ze zmiennych równania;

**- „\*”** - operator iloczynu;

**- „m”** – druga ze zmiennych;

**- „^”** – operator potęgi;

**- „2”** – liczba.

I w taki właśnie sposób reprezentowany jest każdy kod, który interpretuje kompilator. Każdy kolejny znak jest analizowany, a następnie wyzwalana jego funkcja bądź procedura.

Poniżej znajduje się tabelka z niektórymi znanymi naszemu kompilatorowi znakami:

# Bibliografia

http://edu.pjwstk.edu.pl/wyklady/jfa/scb/frames-jfa-main-node5.html