Architektura Komputerowa 2

Sprawozdanie z zadania projektowego

Daniel Gaida

Mateusz Gawłowski

Bartosz Szymański

Kamil Wojcieszak

Temat:

Virtual Machine

1. Wprowadzenie

Celem tego projektu było stworzenie prostego wirtualnego stosowego systemu maszyny (Stack-VM) oraz interpretera, który jest w stanie wykonywać programy zapisane w formacie binarnym. Stack-VM jest abstrakcyjną maszyną wirtualną, która operuje na stosie i wykonuje zestaw prostych instrukcji. Głównym celem projektu było zaprojektowanie, implementacja i uruchomienie interpretera Stack-VM, który umożliwiałby wykonywanie programów zapisanych w tym formacie.

2. Sposób wykonania zadania

Projekt został wykonany w języku programowania C++. Język ten został wybrany ze względu na jego popularność, wszechstronność i efektywność w obszarze programowania systemowego oraz niskopoziomowego. Program tworzy prosty interpreter wirtualnej maszyny stosowej. Interpreter ma za zadanie odczytywać i wykonywać programy zapisane w formacie binarnym.

Działanie interpretera można podzielić na kilka kroków:

1 Wczytanie programu: Interpreter odczytuje plik binarny zawierający program. Program składa się z instrukcji i danych, zapisanych w formacie specyficznym dla wirtualnej maszyny stosowej.

2 Inicjalizacja maszyny wirtualnej: Tworzona jest instancja klasy VirtualMachine, która reprezentuje maszynę wirtualną. Inicjalizowane są odpowiednie zmienne i rezerwowana jest pamięć dla programu.

3 Wykonywanie programu: Interpreter przechodzi przez program instrukcję po instrukcji. Na podstawie kodu instrukcji interpreter rozpoznaje odpowiednią operację do wykonania. Mogą to być operacje arytmetyczne (dodawanie, odejmowanie, mnożenie, dzielenie) lub specjalne operacje (np. zatrzymanie programu).

4 Obsługa danych i pamięci: Interpreter zarządza pamięcią wirtualnej maszyny i wykonuje odpowiednie operacje na stosie. Może odczytywać dane z pamięci, zapisywać wyniki obliczeń oraz przenosić dane między różnymi obszarami pamięci.

5 Zakończenie działania: Interpreter wykonuje program do momentu, gdy napotka instrukcję zatrzymania. Wtedy zakończa działanie i program kończy się.

3. Opis instrukcji

Mnemonik STOP

- Instrukcja STOP służy do zatrzymania działania programu.
- Kod bajtowy instrukcji STOP to 0.

Mnemonik ADD

- Instrukcja ADD wykonuje dodawanie dwóch wartości ze stosu.
- Kod bajtowy instrukcji ADD to 1.

Mnemonik SUB

- Instrukcja SUB wykonuje odejmowanie dwóch wartości ze stosu.
- Kod bajtowy instrukcji SUB to 2.

Mnemonik DIV

- Instrukcja DIV wykonuje dzielenie dwóch wartości ze stosu.
- Kod bajtowy instrukcji DIV to 3.

Mnemonik MUL

- Instrukcja MUL wykonuje mnożenie dwóch wartości ze stosu.
- Kod bajtowy instrukcji MUL to 4.

3. Architektura

Architektura wirtualnej maszyny Stack-VM opiera się na stosie i rejestrze.

Rejestry

Stack-VM posiada dwa rejestry:

- Program Counter (PC): Przechowuje adres bieżącej instrukcji w pamięci.
- Stack Pointer (SP): Wskaźnik na wierzchołek stosu.

Pamięć

- Pamięć wirtualnej maszyny Stack-VM jest zorganizowana jako tablica pamięci o rozmiarze 1000000.

Stos

- Stos w Stack-VM jest wykorzystywany do przechowywania danych i wykonywania operacji. Stos rośnie w kierunku malejących adresów pamięci.

Instrukcje

Instrukcje w Stack-VM mają format 32-bitowy:

- Pierwsze 2 bity określają typ instrukcji.
- Pozostałe 30 bitów zawierają dane lub argumenty instrukcji.

4. Mechanizm działania interpretera

Interpreter wirtualnej maszyny Stack-VM wykonuje program w trzech etapach: pobieranie instrukcji, dekodowanie instrukcji oraz wykonanie instrukcji.

Etap 1: Pobieranie instrukcji

W tym etapie interpreter pobiera następną instrukcję z pamięci na podstawie wartości przechowywanej w Program Counter (PC). PC jest inkrementowany po pobraniu instrukcji.

Etap 2: Dekodowanie instrukcji

W tym etapie interpreter dekoduje instrukcję i odczytuje jej typ oraz ewentualne argumenty na podstawie wartości pobranej z pamięci.

Etap 3: Wykonanie instrukcji

W tym etapie interpreter wykonuje zdekodowaną instrukcję. W przypadku instrukcji typu 0 lub 2, interpreter aktualizuje wartość Stack Pointer (SP) i zapisuje dane na stosie. W przypadku innych typów instrukcji, interpreter wykonuje odpowiednią operację na danych na stosie.

Etap 4: Zakończenie wykonania

Wykonanie programu kończy się, gdy interpreter napotka instrukcję STOP lub warunek zakończenia programu zostanie spełniony.

5. Podsumowanie

W raporcie przedstawiono projekt Stack-VM, który jest prostym wirtualnym procesorem stosowym. Opisano dostępne instrukcje, architekturę maszyny, mechanizm działania interpretera oraz role poszczególnych komponentów. Projekt ten pozwala na wykonywanie programów zapisanych w postaci binarnej, co umożliwia realizację różnych zadań w kontekście wirtualnej maszyny Stack-VM.