Sprawozdanie

Grzegorz Szczepanek
Nr indeksu: 280678

18.04.2025

Wprowadzenie

W niniejszym sprawozdaniu przedstawiono wyniki realizacji ćwiczenia z programowania w asemblerze dla architektury x86 w systemie Linux. Ćwiczenie polegało na implementacji podstawowych operacji arytmetycznych: dodawania, odejmowania oraz mnożenia dwóch liczb całkowitych, przy użyciu rejestrów procesora oraz wywołań systemowych do wyświetlania wyników.

Cel ćwiczenia

Celem ćwiczenia było:

- Poznanie sposobu reprezentacji i przetwarzania liczb całkowitych w asemblerze.
- Zastosowanie instrukcji arytmetycznych.
- Zamiana liczby całkowitej na jej reprezentacje tekstowa i wyświetlenie na standardowym wyjściu.

Treść ćwiczenia

Zadanie polegało na wykonaniu trzech operacji:

- Dodanie dwóch liczb.
- Odjecie jednej liczby od drugiej.
- Mnożenie dwóch liczb z użyciem instrukcji `imull`.

Przebieg ćwiczenia i omówienie kodu

Dodawanie

Do wykonania dodawania użyto instrukcji `addl`, która dodaje wartość z pamieci (`num2`) do zawartości rejestru `%eax`, w którym wcześniej została załadowana pierwsza liczba. Po

obliczeniu wyniku wykonywana jest petla konwertujaca liczbe dziesietna na ciag znaków ASCII poprzez dzielenie przez 10 i odkładanie reszty.

Wartość rejestrowa jest przekształcana na ciag cyfr zapisanych w odwrotnej kolejności, dlatego konieczne było odwrócenie ciagu w buforze przed jego wyświetleniem. Nastepnie, za pomoca wywołań systemowych, wynik został wypisany na ekran.

Odejmowanie

Operacja odejmowania została wykonana przy użyciu instrukcji `subl`, która od wartości w rejestrze `%eax` odejmuje druga liczbe z pamieci. W przypadku uzyskania wyniku ujemnego, została zastosowana instrukcja `negl`, która zmienia znak rejestru `%eax`, a na końcu do bufora został dopisany znak minus, jeśli wynik był pierwotnie ujemny.

Kod wykorzystuje dzielenie przez 10 w celu ekstrakcji kolejnych cyfr liczby, zapisujac je od końca bufora i ostatecznie wypisujac w prawidłowej kolejności.

Mnożenie

Mnożenie dwóch liczb zostało zrealizowane za pomoca instrukcji `imull`, która wykonuje mnożenie ze znakiem. Do rejestru `%eax` ładowana jest pierwsza liczba, a nastepnie jest ona mnożona przez druga (`num2`). Wynik zostaje zapisany bezpośrednio w `%eax`, co eliminuje konieczność korzystania z dodatkowych rejestrów.

Instrukcja `imull` różni sie od `mull` tym, że obsługuje liczby ze znakiem, co oznacza, że działa poprawnie również dla wartości ujemnych, zachowujac poprawność arytmetyki całkowitej. W przypadku operacji na liczbach dodatnich wynik jest taki sam, ale wykorzystanie `imull` daje wieksza elastyczność przy rozszerzaniu programu.

Wynik mnożenia, zapisany w `%eax`, jest nastepnie przekształcany na postać tekstowa. W tym celu zastosowano dzielenie przez 10 i odkładanie cyfr w odwrotnej kolejności do bufora. Po zakończeniu petli wskaźnik przesuwany jest do poczatku liczby, a nastepnie wartość wypisywana jest na standardowe wyjście.

Podsumowanie

Ćwiczenie pozwoliło na praktyczne zapoznanie sie z podstawowymi operacjami arytmetycznymi w asemblerze. Zrealizowane zadania uwidoczniły mechanike pracy z rejestrami oraz sposób reprezentacji danych w systemie.

W szczególności cenne było zastosowanie instrukcji `imull`, która pozwala na prosta realizacje mnożenia liczb całkowitych bez konieczności implementowania algorytmu wielokrotnego dodawania.

Wnioski

Ćwiczenie pokazało, jak istotna jest znajomość architektury i rejestrów procesora podczas implementacji nawet najprostszych operacji. Optymalizacja przy użyciu wbudowanych instrukcji, takich jak `imull`, upraszcza kod i zwieksza jego czytelność. Przekształcenie wyników do postaci tekstowej okazało sie powtarzalnym i uniwersalnym schematem, który można zastosować w wielu innych programach asemblerowych.