Adrian Nurkowski, 272859

Cel zadania

Celem zadania było stworzenie kalkulatora dużych liczb (maksymalnie 100 bajtowych)   
w pozycyjnym zapisie szesnastkowym w języku Assembler x86 (32bit) w składni Intel lub AT&T. Kalkulator miał wykonywać jedną z trzech operacji wybranych przez użytkownika: dodawanie, odejmowanie, mnożenie. Liczby oraz operacja powinny być wprowadzane z linii wiersza poleceń w formacie ”./kalkulator LiczbaA Operacja LiczbaB”. Wynik działania miał być wypisany w jednej linii wiersza poleceń w formacie liczby szesnastkowej bez znaku zakończonej znakiem nowej linii.

Możliwie szczegółowy opis algorytmów dodawania, odejmowania i mnożenie dużych liczb za pomocą podstawowych instrukcji analogicznych działań

Dodawanie:

1. Zacznij od ostatniej cyfry LiczbyA i ostatniej cyfry LiczbyB
2. Dodaj cyfrę LiczbyB do odpowiadającej jej cyfrze LiczbyA
3. Sprawdź czy suma tych cyfr jest większa bądź równa ich systemowi
   1. Jest większa bądź równa: odejmij system od ich sumy i dodaj 1 w następnym dodawaniu LiczbyB do LiczbyA
   2. Jest mniejsza: przejdź do kroku 4
4. Zapisz wynik ich sumy na odpowiadające im miejsce.
5. Przejdź do kolejnych cyfr
6. Powtarzaj kroki 2-5 do momentu, kiedy jednej z liczb skończą się cyfry
7. Dodaj pozostałe cyfry liczby z większą ilością cyfr do wyniku
8. Sprawdź czy ostatnim krokiem 3 był krok 3.1
   1. Ostatnim krokiem był 3.1: Przejdź do kroku 8
   2. Ostatnim krokiem był 3.2: Przejdź do kroku 10
9. Dodaj 1 do następnej cyfry
10. Sprawdź czy suma tych cyfr jest większa bądź równa ich systemowi
    1. Jest większa bądź równa: odejmij system od ich sumy, zapisz wynik ich sumy na odpowiadające im miejsce i wróć do kroku 8
    2. Jest mniejsza: Zapisz wynik ich sumy na odpowiadające im miejsce
11. Wypisz wynik

Odejmowanie:

1. Zacznij od pierwszej cyfry LiczbyA i pierwszej cyfry LiczbyB
2. Sprawdź czy LiczbaA oraz LiczbaB mają cyfrę na tym samym miejscu
   1. Mają cyfrę: przejdź do kroku 3
   2. Nie mają cyfry: przejdź do następnej cyfry Liczby z mniejszą ilością cyfr i przejdź do kroku 2
3. Sprawdź czy LiczbaA jest mniejsza od LiczbyB
   1. Jest mniejsza: Sprawdź czy poprzednia cyfra LiczbyA ‘istnieje’ i jest większa niż 0
      1. Jest większa: odejmij 1 od poprzedniej cyfry LiczbyA i dodaj wartość systemu do cyfry liczbyA
      2. Nie jest większa: Przejdź do kroku 3.1
      3. Nie istnieje: Przejdź do kroku 8
   2. Jest większa: przejdź do kroku 4
4. Odejmij cyfrę LiczbyB od odpowiadającej jej cyfrze LiczbyA
5. Zapisz wynik ich różnicy na odpowiadające im miejsce
6. Przejdź do kolejnych cyfr
7. Powtarzaj kroki 2-6 do momentu, kiedy skończą się cyfry
8. Wypisz wynik

Mnożenie

1. Zacznij od ostatniej cyfry LiczbyA i ostatniej cyfry LiczbyB
2. Pomnóż cyfrę LiczbyB przez cyfrę LiczbyA
3. Dodaj wynik ich iloczynu na odpowiadające im miejsce
4. Przejdź do kolejnej cyfry LiczbyA
5. Powtarzaj kroki 2-4 do momentu, kiedy LiczbieA skończą się cyfry
6. Powtarzaj kroki 2-5 do momentu, kiedy LiczbieB skończą się cyfry
7. Zacznij od ostatniej cyfry wyniku
8. Sprawdź czy cyfra wyniku jest większa bądź równa ich systemowi
   1. Jest większa bądź równa ich systemowi: odejmuj liczbę systemu od cyfry wyniku, dopóki cyfra wyniku nie będzie mniejsza od systemu, cyfrę wyniku zastąp wynikiem odejmowania a ilość powtórzeń odejmowania dodaj do kolejnej cyfry wyniku
   2. Jest mniejsza: Przejdź do kroku 9
9. Przejdź do następnej liczby
10. Powtarzaj kroki 7-9 do momentu, kiedy LiczbieB skończą się cyfry
11. Wypisz wynik

Opis implementacji powyższych algorytmów

Nie potrafiłem zaimplementować powyższych algorytmów. Podjąłem próbę napisania mojego algorytmu dodawania, ale nie potrafiłem go zaimplementować. Wynikało to z braku moich umiejętności kodowania w języku Assembler. Korzystałem z pomocy sztucznej inteligencji, ponieważ nie mogłem znaleźć żadnych kursów tego języka na platformę x86 (32bit). Chciałem wykorzystać Bibliotekę C a dokładniej jedne z jej funkcji takich jak ‘printf’ oraz ‘scanf’ ale nie udało mi się. Wynikało to z powodu, iż system jaki używałem był oparty na architekturze x86\_64 GNU/Linux, a mój kod był dla architektury x86\_32. Próbowałem również zainstalować tę bibliotekę dla wersji 32bitowej, ale po jej instalacji system nie mógł uruchomić pliku ‘kalkulator’ (nie wyszukiwał go). Mój kod również nie działa na liczbach w zapisie szesnastkowym tylko w zapisie dziesiątkowym, , ponieważ chciałem spróbować najpierw napisać ten program dla liczb w systemie dziesiątkowym który jest naturalnym systemem dla człowieka.

Możliwe metody optymalizacji zaimplementowych algorytmów

Sprawienie by aktualnie działały, dodanie obsługi liczb ujemnych oraz jego optymalizacja pod względem prędkości.