**Kolegium Nauk Przyrodniczych  
Uniwersytet Rzeszowski**

**Przedmiot:**

**Architektura Systemów Komputerowych**

**Temat sprawozdania**

**Dokumentacja projektu**

**Wykonał:**

**Bartłomiej Florek 125115**

**Prowadzący: Mgr inż. Jarosław Szkoła**

**Rzeszów 2024**

**Temat projektu: wyliczanie średniej na podstawie liczb wprowadzonych przez użytkownika z zabezpieczeniami.**

Obraz zawierający Czcionka, tekst, zrzut ekranu, design

Opis wygenerowany automatycznie

Finit – inicjalizuje FPU, czyli koprocesor, dzięki któremu wyliczymy średnia na floatach.

Definicja zmiennych suma oraz licznik.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Na rejestr ESI zostanie odłożony rejestr EBX, po to aby później móc go wykorzystać.

Zostaje wywołana funkcja „liczba\_iteracji”, która przed skokiem odłoży „print” na stos ESP.

W labelu „lilczba\_iteracji” wywoływany jest print oraz robimy miejsce na stosie dla zmiennej przez instrukcje „push ESP”.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Call „skanuj\_liczbe” zanim przeskoczy do tej etykiety, odłoży na stos „print2”. W etykiecie „skanuj\_liczbe” wywołujemy metode scanf z rejestru ESI, a po zeskanowaniu liczby, usuwamy ze stosu print2 oraz wolną przestrzeń na ESP. Zmienna wprowadzona przez użytkownika zostanie przypisana do tego pierwszego printa.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Tak samo jak w poprzednich callach, zanim skoczymy, odkładamy na stos etykietę, która znajduje się zaraz pod tym wywołaniem, w tym przypadku „print3”. W etykiecie „przypisanie” wyświetlamy jaką liczbę wprowadził użytkownik. Odkładamy na rejestr ECX (czyli rejestr licznikowy) naszą zmienną.

Następnie sprawdzamy przez instrukcje CMP, wartość na rejestrze ECX z liczba 0.

Instrukcja JECXZ, sprawdza czy wartość ECX jest równa 0, jeśli tak to skoczy do etykiety „domyślna”, która ustawi wartość na 1 aby wprowadzić chociaż jedną liczbę.

Instrukcja JG, sprawdza czy wartość ECX, jest większa od 0, jeśli tak to skoczy do etykiety „poprawna”.

Natomiast jeśli wartość na ECX jest mniejsza od 0 to wtedy wszystkie instrukcje skoku pominiemy i wykona się instrukcja NEG, która zaneguje tę wartość na wartość dodatnią i wykona się wszystko w etykiecie „poprawna”

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

W etykiecie „poprawna” ustawiamy na rejestry EDI oraz EBX, odpowiednio sumę oraz licznik, które będą sumować wprowadzone liczby oraz inkrementować licznik. Następnie ze stosu usuwamy „print3” oraz „zmienna”, które już wykorzystaliśmy i skaczemy bezwarunkowo przez instrukcje JMP do pętli. W etykiecie „domyslna” ustawiamy tylko wartość ECX na 1.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

W pętli ustawiamy na ECX wprowadzoną przez użytkownika liczbę, czyli liczba iteracji. Uruchamiamy calla, odłożymy na stosie „format”, w „getaddr” wywołujemy printa, i zostawiamy wolną przestrzeń na wprowadzoną liczbę. Uruchamiamy dobrze już nam znaną instrukcje call.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

W getaddr2, wywołujemy scanf, i po wprowadzeniu przez użytkownika liczby usuwamy ze stosu. Nasza zmienna zostanie zapisana w „zmienna”.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

W getaddr3, wywołujemy printa i wypisujemy wprowadzona liczbę. Na rejestr EAX, ustawiamy nasza zmienna, i wykonując instrukcje TEST, która ustawi flagę SF na 1 jeśli EAX jest mniejszy od 0. Instrukcja JGE skoczy do etykiety „negat” jeśli spełnione są warunki, czyli jeśli wartość na rejestrze EAX jest dodatnia, natomiast jeśli nie jest dodatnia, wykona się jej negacja.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

W etykiecie „negat” dodajemy do rejestru EDI, naszą zmienna zapisaną na rejestrze EAX, i usuwamy ze stosu „format” oraz „zmienna”. Ze stosu przez instrukcje POP usuniemy wartość ECX. INC EBX zwiększy nasz licznik do dzielenie za każda wprowadzoną liczbę, oraz LOOP petla, która sprawdzi czy wartość na ECX nie jest równa 0, jeśli nie jest to zmniejszy jej wartość oraz skoczy do etykiety petla. Jeśli ECX jest równy 0 to na stos wypchniemy EDI, czyli nasza sumę wszystkich wprowadzonych liczb.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

W „wypisz\_sume”, wywołujemy printa który wyświetli nasza sumę. FILD qword [esp] sprawi, że na koprocesor zostanie odłożona liczba z rejestru EDI. Następnie stos zostaje wyczyszczony, a wartość z rejestru EBX, czyli naszego licznik zostaje wypchnięta.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Następują te same instrukcje co w przypadku wyświetlania sumy, czyli wywołanie printa, odłożenie na stos koprocesora wartość z rejestru EBX, oraz usuniecie ze stosu printa oraz EBX. FDIV operuje na koprocesorze, a dokładniej dzieli st1(nasza suma) przez st0(licznik). Następnie przygotowujemy miejsce na stosie aby zapisać wynik operacji dzielenia zmiennoprzecinkowej. FSTP qword [esp] zapisuje wartość z wierzchu stosu FPU na stosie ESP.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

W etykiecie „srednia” wypisujemy średnia w zmiennoprzecinkowym formacie, z dokładnością do 2 miejsc po przecinku. W etykiecie koniec, wywołujemy printa, usuwamy ze stosu zbędne już miejsca i printa, a na koniec pushujemy na stos 0, aby móc wyjść z programu oraz wywołujemy funkcje exit z rejestru ESI.

**Program jest również dostarczony w wersji .exe, który może zostać uruchomiony poprzez przejście do cmd, wejście do folderu, w którym się znajduje ten plik, i wpisanie jego nazwy, wtedy program się uruchomi w konsoli.**  
  
Projekt znajduje się na githubie: <https://github.com/flor3kk/Assembly/tree/main/projekt>

Aby uruchomić projekt w programie ConTEXT, musimy utworzyć folder, do którego skopiować program asmloader.exe oraz plik projekt\_ask.asm a następnie wcisnąć F9 (kompilacja) oraz F10 (uruchomienie).