# Dokumentacja Projektu

**Tytuł**: Program odwracający kolejność wyrazów we wprowadzonym tekście

**Autor**: Adrianna Dziewulska

**Numer indeksu**: 91261

**Przedmiot**: Architektura Systemów Komputerowych

**Prowadzący**: Wojciech Nabiałek

**Data**: 13.12.2024

# Spis treści:

# 1. Cel projektu

Celem projektu jest stworzenie programu w Symulatorze Procesora 8086, który:

- Odwraca kolejność wyrazów we wprowadzonym z klawiatury zdaniu, które kończy się kropką.
- Jeżeli wyrazy zostaną odwrócone, zapali się zielone światło na sygnalizatorze.
- Jeżeli wyrazy nie mogą zostać odwrócone z powodu braku separatorów, zapali się czerwone światło na sygnalizatorze.

Program ma na celu praktyczne zastosowanie możliwości środowiska Symulatora Procesora 8086 w analizie tekstu oraz obsłudze urządzeń zewnętrznych.

# 2. Opis działania programu

Program rezerwuje przestrzeń w pamięci na bufor, w którym będą przechowywane dane. Wielkość bufora odpowiada rozmiarowi wyświetlacza VDU. Tekst wprowadzany przez użytkownika za pomocą klawiatury jest odczytywany dzięki funkcjom symulatora, które przetwarzają wpisane znaki.

Program kontroluje wprowadzone znaki pod kątem obecności separatorów (spacji i kropek) oraz sprawdza, czy ich liczba mieści się w przydzielonym buforze. Każde zdanie musi kończyć się kropką. Każdy wprowadzony znak jest tymczasowo przechowywany w buforze pamięci. Wyrazy są rozpoznawane na podstawie spacji poprzedzającej ich początek lub, w przypadku pierwszego wyrazu, braku wcześniejszych znaków.

Całe wyrazy są następnie zapisywane na stos, co umożliwia ich odwrócenie podczas odczytu. Odwrócony tekst jest wyświetlany na wyświetlaczu VDU. W przypadku pomyślnego odwrócenia całego zdania, zapala się zielone światło na sygnalizatorze, a odwrócone zdanie zostaje wyświetlone na wyświetlaczu. Jeśli w tekście brakuje separatorów lub jego długość przekracza rozmiar bufora, na sygnalizatorze zapala się czerwone światło.

# 3. Wykorzystane narzędzia i mechanizmy

1. Symulator Procesora 8086: Środowisko programu.
2. Instrukcje wejścia/wyjścia: Do wczytania tekstu i jego wyświetlania.
3. Bufor pamięci: Do przechowywania danych wejściowych.
4. Stos: Do przechowywania wyrazów w trakcie odwracania kolejności.
5. Urządzenia zewnętrzne: Sygnalizator (zielone/czerwone światło).

# 4. Lista kroków działania programu

1. Inicjalizacja:
    - `OUT 01`: Włączenie sygnalizacji stanu początkowego.
    - `MOV BL, 80`: Ustawienie wskaźnika `BL` na początkowy adres bufora (80).
2. Pętla główna (Start) :
    - `IN 00`: Odczytanie znaku wprowadzonego przez użytkownika i zapisanie go do rejestru `AL`.
    - `MOV [BL], AL`: Przechowanie znaku w buforze pod adresem wskazywanym przez `BL`.
    - `INC BL`: Zwiększenie wskaźnika bufora `BL`.
    - Sprawdzanie znaków:
        - `CMP AL, 2E`: Jeśli wprowadzony znak to kropka (`.`), przejdź do etykiety `Cout`.
        - `CMP AL, 20`: Jeśli wprowadzony znak to spacja, przejdź do etykiety `Inc`.
    - Sprawdzanie zakresu bufora:
        - `CMP BL, B0`: Jeśli wskaźnik `BL` osiągnie wartość `B0`, przejdź do etykiety `Error`.

- `JMP Start`: Powrót do początku pętli.

3. Zwiększenie liczby wyrazów (Inc):
   - `INC DL`: Zwiększenie licznika wyrazów (`DL`).
   - `JMP Start`: Powrót do pętli głównej.
4. Odwracanie tekstu (Cout):
   - `CMP DL, 0`: Jeśli licznik wyrazów wynosi zero, przejdź do etykiety `Error`.
   - `DEC BL`: Przesunięcie wskaźnika `BL` wstecz.
   - `MOV DL, 0`: Wyzerowanie licznika wyrazów.
   - `MOV CL, C0`: Ustawienie wskaźnika `CL` na początek miejsca przeznaczonego na odwrócony tekst (C0).
   - Pętla odczytu znaków (CoutIn):
     - `MOV AL, [BL]`: Pobranie znaku z bufora do rejestru `AL`.
     - `PUSH AL`: Umieszczenie znaku na stosie.
     - `INC DL`: Zwiększenie licznika znaków.
     - `DEC BL`: Przesunięcie wskaźnika `BL` wstecz.
     - Warunki zakończenia odczytu:
       - `CMP BL, 7E`: Jeśli wskaźnik `BL` osiągnie wartość `7E`, przejdź do `CoutPopE`.
       - `CMP AL, 20`: Jeśli znak to spacja, przejdź do `CoutPop`.
       - `CMP AL, 00`: Jeśli znak to NULL, przejdź do `CoutPopE`.
     - `JMP CoutIn`: Kontynuacja odczytu.
5. Przenoszenie znaków ze stosu na wyświetlacz:
   - Przenoszenie całych wyrazów (CoutPop):
     - `POP AL`: Pobranie znaku ze stosu.
     - `MOV [CL], AL`: Zapisanie znaku w miejscu wskazywanym przez `CL`.
     - `INC CL`: Przesunięcie wskaźnika `CL` w przód.

- `DEC DL`: Zmniejszenie licznika znaków.
- `CMP DL, 0`: Jeśli licznik wynosi zero, powróć do `CoutIn`.
- `JMP CoutPop`: Kontynuacja przenoszenia znaków.

o Przenoszenie końcowego znaku (CoutPopE):

- `POP AL`: Pobranie znaku ze stosu.
- `MOV [CL], AL`: Zapisanie znaku w miejscu wskazywanym przez `CL`.
- `INC CL`: Przesunięcie wskaźnika `CL`.
- `DEC DL`: Zmniejszenie licznika znaków.
- `CMP DL, 0`: Jeśli licznik wynosi zero, przejdź do `Done`.
- `JMP CoutPop`: Kontynuacja przenoszenia znaków.

6. Obsługa błędów (Error):
   - `MOV AL, 90`: Ustawienie kodu błędu.
   - `OUT 01`: Wyświetlenie sygnału błędu.
   - `JMP Endl`: Przejście do zakończenia programu.
7. Zakończenie programu (Done):
   - `MOV AL, 24`: Ustawienie kodu zakończenia.
   - `OUT 01`: Wyświetlenie sygnału sukcesu.
   - `JMP Endl`: Przejście do zakończenia programu.
8. Koniec programu (Endl):
   - `END`: Zatrzymanie programu.

# 5. Kod programu

```
OUT 01
MOV BL, 80
Start:
   IN 00
   MOV [BL], AL
   INC BL
   CMP AL, 2E
   JZ Cout
   CMP AL, 20
   JZ Inc
   CMP BL, B0
   JZ Error
   JMP Start

Inc:
   INC DL
   JMP Start

Cout:
   CMP DL, 0
   JZ Error
   DEC BL
   MOV DL, 0
   MOV CL, C0
   CoutIn:
         MOV AL, [BL]
         PUSH AL
         INC DL
         DEC BL
         CMP BL, 7E
         JZ CoutPopE
         CMP AL, 20
         JZ CoutPop
```

```
        CMP AL, 00
        JZ CoutPopE
        JMP CoutIn
        CoutPop:
              POP AL
              MOV [CL], AL
              INC CL
              DEC DL
              CMP DL, 0
              JZ CoutIn
              JMP CoutPop
        CoutPopE:
              POP AL
              MOV [CL], AL
              INC CL
              DEC DL
              CMP DL, 0
              JZ Done
              JMP CoutPop

Error:
   MOV AL, 90
   OUT 01
   JMP Endl

Done:
   MOV AL, 24
   OUT 01
   JMP Endl

Endl:
   END
```

# 6. Wyniki działania

1. Przykład poprawnego działania:
    - Wejście: Programowanie w assemblerze jest ciekawe.
    - Wyjście: ciekawe. Jest assemblerze Programowanie
    - Sygnalizator: Zielone światło.
2. Przykład błędnego działania:
    - Wejście: Test.
    - Sygnalizator: Czerwone światło.

# 7. Wnioski

Projekt zrealizowany w środowisku Symulatora Procesora 8086 pozwolił na:

- Praktyczne zastosowanie funkcji wejścia/wyjścia oraz stosu.
- Zrozumienie mechanizmów obsługi tekstu w środowisku symulującym mikroprocesory.
- Efektywne wykorzystanie sygnalizatora jako urządzenia zewnętrznego.

Program spełnił założenia projektowe, umożliwiając obsługę zarówno poprawnych, jak i błędnych danych wejściowych. Dodatkowe wyzwanie stanowiło sterowanie sygnalizatorem, które zostało poprawnie zaimplementowane.

# 8. Bibliografia


1. Dokumentacja Symulatora procesora 8086 na stronie GitHub - Microprocessor Simulator: https://github.com/dwhinham/Microprocessor-Simulator/tree/master Dostęp: 13.12.2024.
2. Materiały dostępne na stronie uniwersytetu: https://strefa.ii.uws.edu.pl/ Dostęp: 13.12.2024.