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 bufora**:
    - Instrukcja `MOV BL, 70` ustawia wskaźnik `BL` na początkowy adres bufora w pamięci (70h).
2. **Start programu (pętla główna)**:
    - `IN 00`: Odczytuje znak wprowadzony przez użytkownika i zapisuje go do rejestru `AL`.
    - `MOV [BL], AL`: Zapisuje odczytany znak z `AL` do pamięci pod adresem wskazywanym przez `BL`.
    - `INC BL`: Zwiększa wskaźnik `BL`, przygotowując miejsce na kolejny znak w buforze.
3. **Sprawdzanie znaków kończących zdanie**:
    - `CMP AL, 2E`: Porównuje odczytany znak z kropką (`.`).
        - Jeśli znak to kropka (`2E`), przechodzi do etykiety `Cout` (odwracanie tekstu i wyświetlanie).
    - `CMP AL, 20`: Porównuje odczytany znak ze spacją ( ).
        - Jeśli znak to spacja (`20`), przechodzi do etykiety `Inc` (zwiększenie licznika wyrazów).
4. **Sprawdzanie końca bufora**:

- `CMP BL, B0`: Sprawdza, czy wskaźnik `BL` osiągnął wartość `B0` (koniec bufora).
    - Jeśli tak, przechodzi do etykiety `Error` (błąd bufora).

5. **Powrót do pętli głównej**:
    - `JMP Start`: Powraca do początku pętli, aby odczytać kolejny znak.
6. **Zwiększenie licznika wyrazów (Inc)**:
    - `INC DL`: Zwiększa licznik wyrazów (`DL`).
    - `JMP Start`: Powraca do początku pętli.
7. **Odwracanie tekstu (Cout)**:
    - `CMP DL, 0`: Sprawdza, czy licznik wyrazów wynosi zero.
        - Jeśli tak, przechodzi do etykiety `Error` (brak wyrazów).
    - `DEC BL`: Cofnięcie wskaźnika `BL` o jedno miejsce.
    - `MOV DL, 0`: Zeruje licznik wyrazów (`DL`).
    - `MOV CL, C0`: Ustawia wskaźnik `CL` na początek miejsca docelowego dla odwróconego tekstu (C0h).
8. **Pętla odczytu znaków (CoutIn)**:
    - `MOV AL, [BL]`: Pobiera znak z bufora wskazywanego przez `BL` do `AL`.
    - `PUSH AL`: Umieszcza znak na stosie.
    - `INC DL`: Zwiększa licznik znaków.
    - `DEC BL`: Przesuwa wskaźnik `BL` wstecz.
    - `CMP AL, 20`: Sprawdza, czy znak to spacja.
        - Jeśli tak, przechodzi do etykiety `CoutPop` (przenoszenie wyrazu z bufora na wyświetlacz).
    - `CMP AL, 00`: Sprawdza, czy znak to NULL (koniec danych).
        - Jeśli tak, przechodzi do etykiety `CoutPopE`.
    - `JMP CoutIn`: Kontynuuje odczytywanie znaków.
9. **Przenoszenie znaków ze stosu na wyświetlacz (CoutPop)**:

- `POP AL`: Pobiera znak ze stosu.
- `MOV [CL], AL`: Zapisuje znak w pamięci wskazywanej przez `CL`.
- `INC CL`: Przesuwa wskaźnik `CL` do następnej pozycji.
- `DEC DL`: Zmniejsza licznik znaków.
- `CMP DL, 0`: Sprawdza, czy licznik znaków wynosi zero.
  - Jeśli tak, powraca do `CoutIn`.
- `JMP CoutPop`: Kontynuuje przenoszenie znaków.

10. **Przenoszenie ostatniego znaku ze stosu (CoutPopE)**:
    - `POP AL`: Pobiera znak ze stosu.
    - `MOV [CL], AL`: Zapisuje znak w pamięci wskazywanej przez `CL`.
    - `INC CL`: Przesuwa wskaźnik `CL`.
    - `DEC DL`: Zmniejsza licznik znaków.
    - `CMP DL, 0`: Sprawdza, czy licznik znaków wynosi zero.
      - Jeśli tak, przechodzi do `Done` (zakończenie programu).
    - `JMP CoutPop`: Kontynuuje przenoszenie znaków.
11. **Obsługa błędów (Error)**:
    - `MOV AL, 90`: Ustawia kod błędu.
    - `OUT 01`: Wyświetla sygnał błędu (czerwone światło).
    - `JMP Endl`: Przechodzi do końca programu
12. **Zakończenie programu (Done)**:
    - `MOV AL, 24`: Ustawia kod zakończenia.
    - `OUT 01`: Wyświetla sygnał sukcesu (zielone światło).
    - `JMP Endl`: Przechodzi do końca programu
13. **Koniec programu (Endl)**:
    - `END`: Zatrzymuje program.

5. Kod programu

```
1. MOV BL, 70
2. Start:
3. IN 00
4. MOV [BL], AL
5. INC BL
6. CMP AL, 2E
7. JZ Cout
8. CMP AL, 20
9. JZ Inc
10.      CMP BL, B0
11.      JZ Error
12.      JMP Start
13.
14.      Inc:
15.      INC DL
16.      JMP Start
17.
18.      Cout:
19.      CMP DL, 0
20.      JZ Error
21.      DEC BL
22.      MOV DL, 0
23.      MOV CL, C0
24.      CoutIn:
25.            MOV AL, [BL]
26.            PUSH AL
27.            INC DL
28.            DEC BL
29.            CMP AL, 20
30.            JZ CoutPop
31.            CMP AL, 00
32.            JZ CoutPopE
33.            JMP CoutIn
```

```
34.            CoutPop:
35.                  POP AL
36.                  MOV [CL], AL
37.                  INC CL
38.                  DEC DL
39.                  CMP DL, 0
40.                  JZ CoutIn
41.                  JMP CoutPop
42.            CoutPopE:
43.                  POP AL
44.                  MOV [CL], AL
45.                  INC CL
46.                  DEC DL
47.                  CMP DL, 0
48.                  JZ Done
49.                  JMP CoutPop
50.
51.      Error:
52.      MOV AL, 90
53.      OUT 01
54.      JMP Endl
55.
56.      Done:
57.      MOV AL, 24
58.      OUT 01
59.      JMP Endl
60.
61.      Endl:
62.      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.