Przykładowe wykorzystanie procedur z formatowaniem

Procedury w Assemblerze to prekursory funkcji/procedur z języków C/C++ i innych języków współcznesnych. Pozwalają one na wydzielenie fragmentów kodu, które mogą zostać użyte wiele razy. Pozwala to na zmniejszenie objętości kodu, a także większą kontrolę nad pisanymi programami. Każdą procedurę wywołuje instrukcja call <nazwa procedury> - nazwę stanowi zwyczajna etykieta (ang.label), która czasem posiada dodany przedrostek, oznaczający, iż ta etykieta jest początkiem procedury, np proc_<nazwa>. Ponadto każda procedura kończy się słowem kluczowym ret, czyli return - powrót do miejsca w kodzie, z którego dana procedura została wywołana.

Poniższy program pokazuje podstawowe użycie procedur z poprawnym formatowaniem, co sprawia, że stał się on znacznie bardziej czytelny od pozostałych w tym kursie. Program pozwala na wybranie rodzaju operacji: dodawanie lub odejmowanie oraz wykonaniu określonego działania na dwóch liczbach.

```
.Model small // Inicjalizacja wielkości modelu - small.

.stack 100h     // Inicjalizacja stosu o wielkości 100h. W przeciwieństwie do rejestru ds - segment stosu od razu zapisuje swój adres w rejestrze ss (tak samo kod w rejestrze cs)


.data // Inicjalizacja segmentu danych.

message db 13, 10, "Input a and b $"    // Definicja zmiennej message typu db. Inicjalizacja wartościami: 13, 10 - > kod nowego znaku, oraz łańcuch znaków zakończony $.

a_message db 13, 10, "First number: $"      // Definicja zmiennej a_message typu db. Inicjalizacja wartościami: 13, 10 - > kod nowego znaku, oraz łańcuch znaków zakończony $.

b_message db 13, 10, "Second number: $"     // Definicja zmiennej b_message typu db. Inicjalizacja wartościami: 13, 10 - > kod nowego znaku, oraz łańcuch znaków zakończony $.

result_message db 13, 10, "Result is: $"    // Definicja zmiennej result_message typu db. Inicjalizacja wartościami: 13, 10 - > kod nowego znaku, oraz łańcuch znaków zakończony $.

q_message db 13, 10, "1 - Addition, 2 - Subtraction $"  // Definicja zmiennej q_message typu db. Inicjalizacja wartościami: 13, 10 - > kod nowego znaku, oraz łańcuch znaków zakończony $.

a db 0 //Definicja zmiennej a i inicjalizacja wartością 0. Tutaj przechowana zostanie wartość pobrana od użytkownika.

b db 0  //Definicja zmiennej b i inicjalizacja wartością 0. Tutaj przechowana zostanie wartość pobrana od użytkownika.

result db 0     //Definicja zmiennej result i inicjalizacja wartością 0. Tutaj przechowany zostanie wynik dodawania lub odejmowania.


.code // Inicjalizacja segmentu kodu.

MAIN PROC // Inicjalizacja głównej procedury.

mov ax,@data  // Skopiowanie adresu segmentu danych w pamięci (przydzielonym przez procesor w lini .data) do rejestru ax. Rejestr ds (wykorzystany w następnej lini) przyjmuje tylko wartości z innych rejestrów ogólnego przeznaczenia, zatem wykorzytano rejestr AX.
```

```
mov ds,ax       // Skopiowanie wartości rejestru ax do rejestru ds. Teraz rejestr ds wskazuje na początek segmentu danych w pamięci, co umożliwia i ułatwia programiście wiele operacji na pamięci.

call proc_typing_numbers      //wywołanie procedury proc_typing_numbers - procedura pozwalająca na pobranie liczb z konsoli.

call proc_choose_operation    //wywołanie procedury proc_choose_operation - procedura pozwalająca wybór operacji: 1-dodawanie, 2-odejmowanie.

call proc_operate             //wywołanie procedury proc_operate - procedura pozwalająca na wykoanie operacji wg wcześniejszego wyboru.

call proc_show_result         //wywołanie procedury proc_show_result - procedura pozwalająca na wyświetlenie wyniku operacji.

call proc_exit //wywołanie procedury proc_exit - procedura pozwalająca na wyjście z programu.


proc_typing_numbers: //etykieta proc_typing_numbers - początek procedury

  mov ah, 09h //Inicjalizacja przerwania int 21h - instrukcja 09 - wypisanie łańcucha znaków z pamięci.

  mov dx, offset message //Skopiowanie wartości offsetu zmiennej message do rejestru dx.

  int 21h // Wykonananie przerwania int 21h - intrukcji 09

  mov ah, 09h //Inicjalizacja przerwania int 21h - instrukcja 09 - wypisanie łańcucha znaków z pamięci.

  mov dx, offset a_message //Skopiowanie wartości offsetu zmiennej a_message do rejestru dx.

  int 21h // Wykonananie przerwania int 21h - intrukcji 09

  mov ah, 01h //Inicjalizacja przerwania int 21h - instrukcja 01

  int 21h // Wykonananie przerwania int 21h - intrukcji 01 - oczekiwanie i pobranie znaku z klawiatury.

  mov a, al // Skopiowanie wartości rejestru al do pamięci wewnętrznej w miejscu, na które wskazuje zmienna a.

  mov ah, 09h //Inicjalizacja przerwania int 21h - instrukcja 09 - wypisanie łańcucha znaków z pamięci.

  mov dx, offset b_message //Skopiowanie wartości offsetu zmiennej b_message do rejestru dx.

  int 21h // Wykonananie przerwania int 21h - intrukcji 09

  mov ah, 01h //Inicjalizacja przerwania int 21h - instrukcja 01

  int 21h // Wykonananie przerwania int 21h - intrukcji 01 - oczekiwanie i pobranie znaku z klawiatury.

  mov b, al // Skopiowanie wartości rejestru al do pamięci wewnętrznej w miejscu, na które wskazuje zmienna b.

ret //koniec procedury - powrót do miejsca wywołania


proc_choose_operation:  //etykieta proc_choose_operation - początek procedury
```

```
    mov ah, 09h //Inicjalizacja przerwania int 21h - instrukcja 09 - wypisanie łańcucha znaków z pamięci.
    mov dx, offset q_message //Skopiowanie wartości offsetu zmiennej q_message do rejestru dx.
    int 21h  // Wykonananie przerwania int 21h - intrukcji 09
    mov ah, 01h //Inicjalizacja przerwania int 21h - instrukcja 01
    int 21h  // Wykonananie przerwania int 21h - intrukcji 01 - oczekiwanie i pobranie znaku z klawiatury.
    mov bl, al //Skopiowanie wartości rejestru al do rejestru bl.
ret //koniec procedury - powrót do miejsca wywołania

proc_operate: //etykieta proc_operate - poczgtek procedury
    cmp bl, 31h //Porównanie liczby 31h do wartości rejestru bl (Wybór operacji przez użytkownika nr 1).
                  Jeśli wartości są równe, to wynik tej instrukcji równy jest 0 - ustawia wtedy ona flagę ZF na 1.
    je addition // Jump if equal - jeśli wartość flagi ZF równa jest 1 - skocz do etykiety additionm, jeśli nie - idź dalej
    cmp bl, 32h //Porównanie liczby 32h do wartości rejestru bl (Wybór operacji przez użytkownika nr 2).
                  Jeśli wartości są równe, to wynik tej instrukcji równy jest 0 - ustawia wtedy ona flagę ZF na 1.
    je subtraction // Jump if equal - jeśli wartość flagi ZF równa jest 1 - skocz do etykiety subtraction

    addition: //etykieta
        mov al,a // Skopiowanie wartości z komórki pamięci na którą wskazuje zmienna a do rejestru al.
        add al,b // Dodanie wartości z komórki pamięci na którą wskazuje zmienna b do rejestru al.
        sub al,30h //Odjęcie od rejestru al liczby 30h - konwersja do poprawnego znaku ASCII
        mov result, al      //Skopiowanie znaku ASCII wyniku dodawania z rejestru al do pamięci wewnętrznej, w miejscu
                            na które wskazuje zmienna result.
        ret //koniec procedury - powrót do miejsca wywołania
    subtraction: //etykieta
        mov al,a // Skopiowanie wartości z komórki pamięci na którą wskazuje zmienna a do rejestru al.
        sub al,b // Odjęcie wartości z komórki pamięci na którą wskazuje zmienna b do rejestru al.
        add al,30h //Dodanie od rejestru al liczby 30h - konwersja do poprawnego znaku ASCII
        mov result, al      //Skopiowanie znaku ASCII wyniku odejmowania z rejestru al do pamięci wewnętrznej, w miejscu
                            na które wskazuje zmienna result.
        ret //koniec procedury - powrót do miejsca wywołania
ret //koniec procedury - powrót do miejsca wywołania
```

```
proc_show_result: //etykieta proc_show_result - początek procedury
   mov ah, 09h //Inicjalizacja przerwania int 21h - instrukcja 09 - wypisanie łańcucha znaków z pamięci.
   mov dx, offset result_message //Skopiowanie wartości offsetu zmiennej result_message do rejestru dx.
   int 21h  // Wykonananie przerwania int 21h - intrukcji 09
   mov ah,02 //Inicjalizacja przerwania int 21h - instrukcja 02.
   mov dl,result //Skopiowanie kodu ASCII z pamięci (z miejsca na które wskazuje zmienna result) do rejestru dl.
   int 21h  // Wykonananie przerwania int 21h - intrukcji 02 - wyświetlenie wartości rejestru dl w konsoli.
ret //koniec procedury - powrót do miejsca wywołania

proc_exit: //etykieta proc_show_result - początek procedury
   mov ah, 4Ch //Inicjalizacja przerwania int 21h - instrukcja 4Ch - zakończenie programu
   int 21h // Wykonananie przerwania int 21h - instrukcja 4Ch - zakończenie programu
ret //koniec procedury - powrót do miejsca wywołania

main endp
end main
```