DOKUMENTÁCIA K PROGRAMU ZADANIE.ASM
TITULNÁ STRANA
Názov programu: Počítadlo slov obsahujúcich spoluhlásky prvého slova
Autor: Marek
Predmet: Asembler

ZNENIE ZADANIA
Načítajte z klávesnice reťazec znakov ukončených znakom konca riadku. Slová vo vstupe sú oddelené najmenej jedným znakom medzera. Uvažujte aj prvé, resp. posledné slovo vstupu. Určte počet slov obsahujúcich všetky spoluhlásky 1. slova. Počet vytlačte osmičkovo.
Výsledok musí byť vytlačený podľa stanoveného formátu 000o - 377o.

ANALÝZA PROBLÉMU
Program musí riešiť nasledujúce podúlohy:

1. Načítanie vstupného reťazca z klávesnice
   - Vstupy: znaky zadané používateľom
   - Výstupy: reťazec uložený v pamäti
   - Špeciálne prípady: prázdny vstup, vstup presahujúci veľkosť bufferu
   - Riziká: pretečenie bufferu

2. Identifikácia spoluhlások v prvom slove
   - Vstupy: vstupný reťazec
   - Výstupy: zoznam spoluhlások nachádzajúcich sa v prvom slove
   - Špeciálne prípady: prvé slovo neobsahuje žiadne spoluhlásky
   - Logika: Prechádzame prvé slovo a zaznamenávame všetky spoluhlásky (písmená okrem a, e, i, o, u)

3. Počítanie slov obsahujúcich všetky spoluhlásky prvého slova
   - Vstupy: vstupný reťazec, zoznam spoluhlások z prvého slova
   - Výstupy: počet slov spĺňajúcich podmienku
   - Špeciálne prípady: žiadne slovo nespĺňa podmienku
   - Logika: Pre každé slovo kontrolujeme, či obsahuje všetky spoluhlásky z prvého slova

4. Konverzia počtu do osmičkovej sústavy a výpis
   - Vstupy: počet slov (desiatková sústava)
   - Výstupy: počet slov (osmičková sústava) vo formáte XXXo
   - Špeciálne prípady: výsledok 0
   - Riziká: nesprávny formát výstupu

OPIS RIEŠENIA
Program je rozdelený do niekoľkých logických častí:

1. Inicializácia a načítanie vstupu
```assembly
_asm_main:
    enter 0, 0
    pusha

    xor eax, eax
    mov ecx, 26
    mov edi, consonant_flags
    rep stosb

    mov edi, input_buffer
    xor ecx, ecx

read_loop:
    call read_char
    cmp al, 10
    je read_done
    mov [input_buffer + ecx], al
```

```
    inc ecx
    cmp ecx, BUFFER_SIZE-1
    jl read_loop

read_done:
    mov byte [input_buffer + ecx], 0
```
V tejto časti program inicializuje pole príznakov pre spoluhlásky (consonant_flags) na nuly a načíta vstupný reťazec z klávesnice znak po znaku, až kým nenarazí na znak konca riadku (ASCII 10). Vstup je uložený do input_buffer a ukončený nulovým znakom.

2. Spracovanie prvého slova a identifikácia spoluhlások
```assembly
process_first_word:
    mov esi, input_buffer

first_word_loop:
    mov al, [esi]
    cmp al, 0
    je first_word_done
    cmp al, ' '
    je first_word_done

    cmp al, 'a'
    jl check_uppercase_first
    cmp al, 'z'
    jg skip_char_first

    cmp al, 'a'
    je skip_char_first
    cmp al, 'e'
    je skip_char_first
    cmp al, 'i'
    je skip_char_first
    cmp al, 'o'
    je skip_char_first
    cmp al, 'u'
    je skip_char_first

    mov edx, eax
    sub edx, 'a'
    mov byte [consonant_flags + edx], 1
    jmp skip_char_first
```
Táto funkcia prechádza prvé slovo vstupného reťazca a identifikuje všetky spoluhlásky. Pre každú spoluhlásku nastaví príslušný príznak v poli consonant_flags na hodnotu 1. Funkcia rozlišuje medzi malými a veľkými písmenami, pričom veľké písmená konvertuje na malé pred spracovaním.

3. Počítanie slov obsahujúcich všetky spoluhlásky prvého slova
```assembly
process_input:
    mov esi, input_buffer
    xor ebx, ebx

    call process_first_word

    mov esi, input_buffer
skip_first:
    mov al, [esi]
    cmp al, 0
    je done_processing
    cmp al, ' '
```

```
    je found_space
    inc esi
    jmp skip_first

found_space:
process_next_word:
    inc esi
    mov al, [esi]
    cmp al, 0
    je done_processing
    cmp al, ' '
    je process_next_word

    push esi

    sub esp, 26
    mov ecx, 26
    mov edi, esp
    xor eax, eax
    rep stosb
```
Táto časť programu najprv spracuje prvé slovo, aby identifikovala spoluhlásky, a potom prechádza zvyšné slová vo vstupnom reťazci. Pre každé slovo vytvorí dočasné pole príznakov na zásobníku, ktoré označuje, ktoré spoluhlásky sa v slove nachádzajú.

```assembly
check_word:
    mov ecx, 26
    mov edx, 1

check_consonants:
    dec ecx
    cmp byte [consonant_flags + ecx], 1
    jne skip_consonant

    cmp byte [esp + ecx], 1
    je skip_consonant

    xor edx, edx
    jmp check_done

skip_consonant:
    cmp ecx, 0
    jne check_consonants

check_done:
    cmp edx, 1
    jne word_done
    inc ebx
```
Po spracovaní slova program kontroluje, či slovo obsahuje všetky spoluhlásky z prvého slova. Ak áno, zvýši počítadlo (ebx). Kontrola prebieha tak, že program prechádza všetky pozície v poli consonant_flags, a ak je na danej pozícii hodnota 1 (spoluhláska z prvého slova), kontroluje, či je táto spoluhláska prítomná aj v aktuálnom slove.

4. Konverzia počtu do osmičkovej sústavy a výpis
```assembly
    mov eax, ebx
    mov ecx, 3
    xor edi, edi

print_octal:
```

```
    xor edx, edx
    mov ebx, 8
    div ebx
    push edx
    inc edi
    cmp edi, ecx
    jl print_octal

print_digits:
    pop eax
    add eax, '0'
    call print_char
    dec edi
    jnz print_digits

    mov eax, 'o'
    call print_char
    call print_nl
```
Táto časť programu konvertuje počet slov (uložený v ebx) do osmičkovej sústavy a vypisuje ho vo formáte XXXo. Program používa delenie číslom 8 a ukladá zvyšky na zásobník. Potom vypisuje číslice v správnom poradí a pridáva znak 'o' na koniec.

Použitie registrov:
- EAX: Používa sa na rôzne účely - načítanie znakov, dočasné uloženie hodnôt, výsledok delenia pri konverzii do osmičkovej sústavy
- EBX: Počítadlo slov spĺňajúcich podmienku, deliteľ pri konverzii do osmičkovej sústavy
- ECX: Počítadlo cyklov, index pri prechádzaní poľa príznakov
- EDX: Dočasné uloženie hodnôt, zvyšok po delení pri konverzii do osmičkovej sústavy
- ESI: Ukazovateľ do vstupného reťazca
- EDI: Ukazovateľ do poľa príznakov, počítadlo číslic pri výpise
- ESP: Ukazovateľ na zásobník, používa sa pri vytváraní dočasného poľa príznakov pre každé slovo

Program efektívne využíva zásobník na vytvorenie dočasných polí príznakov pre každé slovo, čím šetrí pamäť. Taktiež správne ošetruje rôzne špeciálne prípady, ako napríklad prázdny vstup alebo slová bez spoluhlások.

Zdrojový kód:

```
%include "asm_io.inc"

segment .data
    BUFFER_SIZE     equ 1024

segment .bss
    input_buffer    resb BUFFER_SIZE
    consonant_flags resb 26

segment .text
    global _asm_main

_asm_main:
    enter 0, 0
    pusha

    xor eax, eax
    mov ecx, 26
    mov edi, consonant_flags
    rep stosb
```

```
    mov edi, input_buffer
    xor ecx, ecx

read_loop:
    call read_char
    cmp al, 10
    je read_done
    mov [input_buffer + ecx], al
    inc ecx
    cmp ecx, BUFFER_SIZE-1
    jl read_loop

read_done:
    mov byte [input_buffer + ecx], 0

    call process_input

    mov eax, ebx
    mov ecx, 3
    xor edi, edi

print_octal:
    xor edx, edx
    mov ebx, 8
    div ebx
    push edx
    inc edi
    cmp edi, ecx
    jl print_octal

print_digits:
    pop eax
    add eax, '0'
    call print_char
    dec edi
    jnz print_digits

    mov eax, 'o'
    call print_char
    call print_nl

    popa
    xor eax, eax
    leave
    ret

; Funkcia na spracovanie vstupného reťazca a počítanie slov
process_input:
    mov esi, input_buffer
    xor ebx, ebx

    call process_first_word

    mov esi, input_buffer
skip_first:
    mov al, [esi]
    cmp al, 0
    je done_processing
    cmp al, ' '
    je found_space
    inc esi
    jmp skip_first

found_space:
```

```
process_next_word:
    inc esi
    mov al, [esi]
    cmp al, 0
    je done_processing
    cmp al, ' '
    je process_next_word

    push esi

    sub esp, 26
    mov ecx, 26
    mov edi, esp
    xor eax, eax
    rep stosb

word_loop:
    mov al, [esi]
    cmp al, 0
    je check_word
    cmp al, ' '
    je check_word

    cmp al, 'a'
    jl check_uppercase
    cmp al, 'z'
    jg next_char

    cmp al, 'a'
    je next_char
    cmp al, 'e'
    je next_char
    cmp al, 'i'
    je next_char
    cmp al, 'o'
    je next_char
    cmp al, 'u'
    je next_char

    mov edx, eax
    sub edx, 'a'
    mov byte [esp + edx], 1
    jmp next_char

check_uppercase:
    cmp al, 'A'
    jl next_char
    cmp al, 'Z'
    jg next_char

    add al, 32

    cmp al, 'a'
    je next_char
    cmp al, 'e'
    je next_char
    cmp al, 'i'
    je next_char
    cmp al, 'o'
    je next_char
    cmp al, 'u'
    je next_char

    mov edx, eax
```

```
    sub edx, 'a'
    mov byte [esp + edx], 1

next_char:
    inc esi
    jmp word_loop

check_word:
    mov ecx, 26
    mov edx, 1

check_consonants:
    dec ecx
    cmp byte [consonant_flags + ecx], 1
    jne skip_consonant

    cmp byte [esp + ecx], 1
    je skip_consonant

    xor edx, edx
    jmp check_done

skip_consonant:
    cmp ecx, 0
    jne check_consonants

check_done:
    cmp edx, 1
    jne word_done
    inc ebx

word_done:
    add esp, 26
    pop esi

find_word_end:
    mov al, [esi]
    cmp al, 0
    je done_processing
    cmp al, ' '
    je found_space
    inc esi
    jmp find_word_end

done_processing:
    ret

; Funkcia na spracovanie prvého slova a identifikáciu jeho spoluhlások
process_first_word:
    mov esi, input_buffer

first_word_loop:
    mov al, [esi]
    cmp al, 0
    je first_word_done
    cmp al, ' '
    je first_word_done

    cmp al, 'a'
    jl check_uppercase_first
    cmp al, 'z'
    jg skip_char_first

    cmp al, 'a'
```

```
    je skip_char_first
    cmp al, 'e'
    je skip_char_first
    cmp al, 'i'
    je skip_char_first
    cmp al, 'o'
    je skip_char_first
    cmp al, 'u'
    je skip_char_first

    mov edx, eax
    sub edx, 'a'
    mov byte [consonant_flags + edx], 1
    jmp skip_char_first

check_uppercase_first:
    cmp al, 'A'
    jl skip_char_first
    cmp al, 'Z'
    jg skip_char_first

    add al, 32

    cmp al, 'a'
    je skip_char_first
    cmp al, 'e'
    je skip_char_first
    cmp al, 'i'
    je skip_char_first
    cmp al, 'o'
    je skip_char_first
    cmp al, 'u'
    je skip_char_first

    mov edx, eax
    sub edx, 'a'
    mov byte [consonant_flags + edx], 1

skip_char_first:
    inc esi
    jmp first_word_loop

first_word_done:
    ret
```