Technická univerzita v Košiciach

Dokumentácia k druhému zadaniu z predmetu Asembler

Martin Novysedlák

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. Na vstupe sú zadané čísla (00-99) v desiatkovej sústave. Zotrieďte zadané čísla od najmenšieho po najväčšie a vytlačte ich. Čísla vytlačte v šestnástkovej sústave.

Analýza problému:

De facto problém je vyžiadanie dvojciferných čísel a aplikovanie triediaceho algoritmu nad poľom zadaných čísel a následné zobrazenie v hexadecimálnej sústave.
V zadaní sa píše, že vstupom su dvojciferné čisla, takže vieme, že budeme volať get_char zo súboru asm_io.asm konkrétne dva krát. Prvá číslica z jedneho prvku zo vstupu bude vynásobené konštantou 10 a k nemu prirátané číslo z druhého volania get_char. Toto spracované číslo sa priradí do zoznamu. Ak je pole naplnené, srpaví sa bubble sort nad týmto poľom. Takto zoradené pole bude vypísané v šestnástkovej sústave, ktoré zabezpečí bitový posun a and-ovanie.

Opis riešenia:

V segmente .data su inicializované premenné poľa, dlžky poľa a pomocnej premennej maximalného kontrolovaného indexu. Začína vyžiadaním prvého znaku, ktoré predstavuje rád desiatok v čísle a druhý znak, čo je počet jednotiek v čísle. Spracované číslo sa pushne do stacku pre neskorsie použitie. Postupne sa ingorujú zadaných viacero medzier a proces sa opakuje. Ak skončí, následuje slučka, ktorá iteruje poľom a vyberá zo stacku hodnoty a ukaldá ich na pozície. Takto je vytvorený zoznam zo vstupu, ktorý je obratený zoznam zo vstupu (posledný prvok vstupu je prvým prvkom v zozname). Nasleduje inicializácia registrov a premennej max_index. Následne sa provnáva prvý a druhý prvok. Ak je prvý prvok vačší, tak si vymenia pozície. Potom je do registra edx presunutá hodnota 1, aby sa vedelo, že sme prehodili elementy. Ďalej sa inkrementuje počítadlo a iteruje sa cez dalšiu dvojicu prvkov v poli. Ak sa sa iterator nachadza na konci pola a register edx obasahuje 0 tak vieme, že žiadna dvojica sa neprehodila a z toho vyplýva, že pole je zoradené. Presúvame sa na výpis. Tam prechádzame poľom ešte raz a prvky uložene v registri eax sa vypíšu šesnastkovo pomocou funkcie print_int_hex. Tam sa spraví bitový posun 8 bitového registra o 4 doprava a vypíseme hodnotu. Ak je číslo na 4 bitoch väčšie ako 9, priráta sa hodnota 7, aby sa ASCII hodnota dostala na alfabetické znaky. Druhá číslica je vypísaná zandovaním pôvodnej hodnoty 8 bitového registra s konštantou 00001111b. Takto dostaneme posledné 4 bity a tie vypíšeme analogicky. Za každé číslo je na konci vypísaný znak ‘h’.

Zdrojový kód:

```
%include "asm_io.inc"


segment .data

   array times 100 db 0
```

```
    length db 0
    max_index db 0

segment .text
    global  _asm_main

print_int_hex:
    enter 0,0
    push eax

    shr eax, 4h
    add eax, 48
    cmp eax, '9'
    jbe dont_add_to_alpha
    add eax, 7
dont_add_to_alpha:
    call print_char
    pop eax
    and eax, 00001111b
    add eax, 48
    cmp eax, '9'
    jbe dont_add_to_alpha2
    add eax, 7
dont_add_to_alpha2:
    call print_char
    mov eax, 104
    call print_char

    leave
```

```
    ret

_asm_main:
    enter    0, 0
    pusha

    call read_char
char_loop:
    mov ebx,10
    sub eax, '0'
    mul ebx
    mov ebx, eax
    call read_char
    sub eax, '0'
    add eax, ebx
    push eax
    inc BYTE[length]
    call read_char
    cmp eax, 10
    je fill_array
    cmp eax, 13
    je fill_array
    cmp eax, 32
    je skip_spaces
    jmp char_loop
skip_spaces:
    call read_char
    cmp eax, 32
    je skip_spaces
```

```
    jmp char_loop

fill_array:
    mov ecx, [length]
    mov ebx, array
l1:
    pop eax
    mov [ebx], al
    inc ebx
    loop l1

setup_sort:
    xor eax, eax
    mov al, [length]
    dec al
    mov [max_index], al

initialize_sort:
    xor eax,eax
    xor ecx, ecx
    mov edi,array ; will be indexing left element of bubble, max index value is [length] - 2
    mov esi,array ; will be indexing right element of bubble, value is ecx+1 and max value is [length] - 1 = [max_index]
    inc esi
    xor edx,edx ; will be detecting if swap has occured, if not than array is sorted. 1 if swap was made, otherwise 0
    mov cl, [max_index]

sort: ; bubble sort. EDI has to be less than ESI
    xor eax, eax
```

```
    xor ebx, ebx
    mov al, [edi]
    mov bl, [esi]
    cmp al, bl
    ja swap_bubble
back:
    inc edi
    inc esi
    loop sort
    cmp edx, 0
    je print_array
    jmp initialize_sort

swap_bubble:
    xor eax, eax
    mov al, [esi]
    push eax
    mov al, [edi]
    push eax
    pop eax
    mov [esi], al
    pop eax
    mov [edi], al
    mov edx, 1h
    jmp back

print_array:
    xor ecx, ecx
    mov cl, [length]
```

```
    mov ebx, array
l2:
    xor eax, eax
    mov al, [ebx]
    call print_int_hex
    mov eax, 32
    call print_char
    inc ebx
    loop l2


_asm_end:

    popa
    mov eax, 0

    leave
    ret
```