TECHNICKÁ UNIVERZITA V KOŠICIACH
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
KATEDRA POČÍTAČOV A INFORMATIKY

Dokumentácia ku druhému zadaniu z predmetu Assembler

Denys Kukhar

2019

1 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.

2 *Popis riešenia*

Stratégia

Pomocou cyklu postupne načítam input do nums (konvergujem po symbolovo a ukládam). Po načítaní si uložím rozmer vstupu ako count. Ďalej nasleduje samotný algoritmus usporiadania elementov. Rozhodol som sa použiť bubble sort. Pomocou procedúry hex si vypíšem obsah nums v hexadecimalnej podobe.

Hexový výpis

Pre hexový výpis vykonávam ANDovanie s korešpodnými hodnotami pre získanie potrebných bitov. Tak, najprv beriem horné 4 bity (**AND** operácia s číslom 11110000) a potom pre výpis shiftujem vpravo na 4 miesta. Podobne spravím s bitmi 5-8.

3 *Kód v jazyku Assembler nasm (8086)*

```
; Bubble sort 10 numbers in place
%include "asm_io.inc"

segment .data
hlaska db "im here",0
dlzka db "Dlzka inputu je: ",0
last_el db "Posledny element je: ",0
count dw 0        ; One less than count of the array.

segment .bss
nums resb 20

segment .text
    global _asm_main
```

```
_asm_main:
    enter 0,0
    pusha

    mov esi, nums
    @input_loop:
    call read_char
    cmp eax, ' '
    jz @input_loop
    cmp eax, 10
    jz @end_input_loop
    cmp eax, 13
    jz @end_input_loop
    ; greater digit to convert and store in edx
    sub eax, '0'
    mov ecx, 10
    mul ecx
    mov edx, eax
    ; second digit to conver, add and store

    call read_char
    sub eax, '0'
    add edx, eax
    ; store normal number in memory
    mov [esi], edx
    inc esi
    jmp @input_loop

    @end_input_loop:
    ; Store array size in memory
    sub esi, nums
    mov [count], esi

    call sort_bubble

    ; Print sorted result
    call print_as_hex
    popa
    mov EAX, 0
    leave
    ret

print_arr_length:
  pusha

      mov edx, nums
      mov ebx, count

      call print_nl
      mov eax, dlzka
      call print_string
      mov eax, [ebx]
      add eax, '0'
      call print_char
  popa
  ret
```

```
print_as_hex:
    pusha
    mov esi, nums
    xor ecx, ecx ; counter = 0

    digit_loop:
        mov eax, [esi]

        and al, 0xF0
        shr al, 4
        call hex
        call print_char

        mov eax, [esi]
        and al, 0x0F
        call hex
        call print_char

        mov al, 'h'
        call print_char
        mov al, ' '
        call print_char


        inc esi
        inc ecx
        cmp ecx, [count]
        jl digit_loop

    popa
    ret

hex:
  ; assume the value is in al
  cmp al, 10d
  jl no_need
  sub al, 10d
  add al, 'A'
  ret

  no_need:
  add al, '0'
  ret
```

```
sort_bubble:
    pusha

    mov edx,0
    mov eax, 4
sort:
    mov bl,byte[nums+eax]
    mov cl,byte[nums+eax+1]
    cmp bl,cl
    jl dontswap
    mov byte[nums+eax+1],bl
    mov byte[nums+eax],cl
    dontswap:
    add eax,1
    mov bl,byte[count]
    sub bl,1
    cmp al,bl
    jne sort
    mov eax,0
    add edx,1
    mov bl,byte[count]
    add bl,1
    cmp dl,bl
    jne sort


    popa
    ret
```

4 *Makefile a spúšťanie*

Pre pohodlnosť som si vytvoril Makefile. Nasleduje jeho jednoduchý obsah:

```
all: asm_io.obj zadanie2.obj
    gcc -o zadanie2.exe zadanie2.obj driver.c asm_io.obj
    ./zadanie2
asm_io.obj:
    nasm -f win32 -d COFF_TYPE asm_io.asm
zadanie2.obj:
    nasm -f win32 zadanie2.asm
clean:
    del *.obj zadanie2.exe
```

5 Dodatočná implementácia

Pre vlastné účely som si taktiež vytvoril implementáciu na Python, čo mi velmi pomohla pri napísaní samotného zadania na Assembler`i. Nasleduje jej jednoduchý obsah:

```
def bubble_sort(sequence):
    for passnum in range(len(sequence)-1, 0, -1):
        for i in range(passnum):
            if sequence[i] > sequence[i+1]:
                temp = sequence[i]
                sequence[i] = sequence[i+1]
                sequence[i+1] = temp
    return sequence


def main():
    usr_inpt = input('Enter the sequence of numbers 00-99: ')
    list_of_num = usr_inpt.split(sep=' ')
    # list_of_num = list(map(lambda character: int(character),
list_of_num))
    list_of_num = [int(character) for character in list_of_num]

    sorted = bubble_sort(list_of_num)
    print(sorted)
    # sorted_hex = list(map(lambda num: hex(num).replace('0x',
'')+'h', sorted))
    sorted_hex = [hex(num).replace('0x', '')+'h' for num in sorted]
    print(sorted_hex)


if __name__ == "__main__":
    main()
```