Autor: Artur Sobolewski 16.05.2020r.

# Sprawozdanie z laboratorium Architektury Komputerów

Prowadzący:

Dr inż. Jędrzej Ułasiewicz

Sprawozdanie zawiera końcowy kod programu z zadań 12.10.3, 12.10.4 i 12.10.5.

Podczas zajęć wykonałem dwa pierwsze zadania.

## 1. Kod pliku: mylib.s zawierającego definicję funkcji: write, read, open, close oraz exit.

```
#Definicje stałych urzytych w programie
.data
SYSEXIT  = 1
SYSREAD  = 3
SYSWRITE = 4
OPEN  = 5
CLOSE = 6
STDIN = 0
STDOUT = 1
EXIT_SUCCESS = 0
READ_ONLY = 0


.text
.global write
.type write, @function      #Definicja funkcji wypisującej znaki
write:
push %ebp
movl  %esp, %ebp
mov $SYSWRITE, %eax         #Ustawienie wartości %eax na 4 (wywołanie
wypisywania)
movl  8(%ebp), %ebx         #Pobranie drugiego parametru (uchwyt pliku)
movl  12(%ebp), %ecx        #Pobranie trzeciego parametru (adres bufora)
movl  16(%ebp), %edx        #Pobranie czwartego parametru (długość
bufora)
int $0x80                   #Wywołanie funkcji systemowej
movl %ebp, %esp
popl %ebp
ret

.global read
.type read, @function       #Definicja funkcji wczytującej znaki z
konsoli
read:
pushl %ebp
movl  %esp, %ebp
```

```
mov $SYSREAD, %eax          #Ustawienie pierwszego parametru (wywołanie
wczytywania)
mov $STDIN, %ebx            #Ustawienie drugiego parametru (standardowe
wejście-konsola)
movl  8(%ebp), %ecx         #Pobranie trzeciego parametru (uchwyt pliku)
movl  12(%ebp), %edx        #Pobranie czwartego parametru (długość
bufora)
int $0x80                   #Wywołanie funkcji systemowej
movl %eax, %edi             #Przesłanie długości wczytanego ciągu do
%edi
sbbl $1, %edi               #Pomnijszenie liczby o wczytany znak \n
movl %ebp, %esp
popl %ebp
ret

.global exit
.type exit, @function       #Definicja funkcji kończącej działanie
programu
exit:
pushl %ebp
movl  %esp, %ebp
movl  $1, %eax              #Ustawienie pierwszego parametru (wywołanie
zakończenia prog.)
movl  8(%ebp), %ebx         #Pobranie drugiego argumanetu (wartość
zwracana)
int   $0x80                 #Wywołanie funkcji systemowej
movl %ebp, %esp
popl %ebp
ret

#-------------------------------------------------------------
#  Zadanie 12.10.5
#-------------------------------------------------------------

.global open
.type open, @function       #Definicja funkcji otwierającej plik
open:
push %ebp
movl %esp, %ebp
movl $OPEN, %eax            #Ustawienie pierwszego parametru (wywołanie
funkc. open)
movl 8(%ebp), %ebx          #Pobranie drugiego parametru (nazwa pliku)
movl 12(%ebp), %ecx         #Pobranie trzeciego parametru (flagi)
movl 16(%ebp), %edx         #Pobranie czwartego parametru (prawa
dostępu)
int $0x80                   #Wywołanie funkcji systemowej
movl %eax, %esi             #Zapisanie uchwytu do pliku w %esi
movl %ebp, %esp
popl %ebp
ret


.global close
.type open, @function       #Definicja funkcji zamykającej plik
close:
push %ebp
movl %esp, %ebp
movl $CLOSE, %eax           #Ustawienie pierwszego parametru (wywołanie
funkc. close)
movl 8(%ebp), %ebx          #Pobranie drugiego parametru (uchwyt do
pliku)
```

```
    int $0x80                   #Wywołanie funkcji systemowej
    movl %ebp, %esp
    popl %ebp
    ret
```

## 2. Kod pliku: read_write_callmylib.s wywołującego funkcje zdefiniowane w pliku: mylib.s funkcje

```
.data
msg_echo: .ascii "                                          "
msg_echo_len = . -msg_echo

newline: .ascii "\n"
newline_len = . -newline

msg_hello: .ascii "Wprowadz napis\n"
msg_hello_len = . -msg_hello

file_name: .ascii "wyniki.txt"      #Definicja nazwy tworzonego pliku
                                    #tworzonego przez funkcję open
.section .text
.global main
main:

leal msg_hello_len, %eax
pushl %eax                          #Przesłanie długości bufora na stos
leal msg_hello, %eax
push %eax                           #Przesłanie adresu bufora na stos
pushl $1                            #Przesłanie 1 na stos (ustawinie
wyjścia na STDOUT)
call write                          #Wywołanie funkcji write

leal msg_echo_len, %eax
pushl %eax                          #Przesłanie długości bufora na stos
leal msg_echo, %eax
pushl %eax                          #Przesłanie adresu bufora na stos
call read                           #Wywołanie funkcji read

leal msg_echo_len, %eax
pushl %eax                          #Przesłanie długości bufora na stos
leal msg_echo, %eax
pushl %eax                          #Przesłanie adresu bufora na stos
pushl $1                            #Przesłanie 1 na stos (ustawinie
wyjścia na STDOUT)
call write                          #Wywołanie funkcji write

leal newline_len, %eax
pushl %eax                          #Przesłanie długości bufora na stos
leal newline, %eax
pushl %eax                          #Przesłanie adresu bufora na stos
pushl $1                            #Przesłanie 1 na stos (ustawinie
wyjścia na STDOUT)
call write                          #Wywołanie funkcji write

# Przykładowe urzycie funkcji otawarcia i zamknięcia pliku
```

```
pushl $0666                     #Przesłanie wartości określającej prawa
dostępu na stos
pushl $03101                    #Przesłanie wartości określającej flagi na
stos
pushl $file_name                #Przesłanie adresu pierwszego symbolu nazwy
pliku na stos
call open                       #Wywołanie funkcji open

#Zapisanie do wprowadzonego ciągu znaków do pliku
#o nazwie określonej na początku programu
leal msg_echo_len, %eax
pushl %eax                      #Przesłanie długości bufora na stos
leal msg_echo, %eax
pushl %eax                      #Przesłanie adresu bufora na stos
pushl %esi                      #Przesłanie uchwytu do pliku na stos
call write                      #Wywołanie funkcji write

pushl %esi                      #Przesłanie uchwytu do pliku na stos
call close                      #Wywołanie funkcji close

pushl %edi                      #Przesłanie długości wprowadzonego ciągu
znaków na stos
call exit                       #Wywołanie funkcji exit
```