Ministerul Educaţiei și Cercetării al Republicii MoldovaUniversitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică

COMPUTER ARCHITECTURE

Laboratory work 5:

Practice tasks in Assembly Language

Elaborated:
st. gr. FAF-213 Konjevic Alexandra

Verified:
asist. Univ Vladislav Voitcovschi

Chişinău – 2023

Tasks:

0.

```
random:
    ; Input: rdi = seed, if 0 then use time of day
    ; Output: rax = random number between 1 and 55
    ;         rdi = new seed

    ; Save register values
    push rbx
    push rcx
    push rdx

    ; Get the unix time as the seed
    cmp rdi, 0
    jne _generate_random
    ; Get the time of day
    mov eax, SYS_TIME_OF_DAY
    mov ebx, 0
    int 0x80
    mov rdi, rax

    _generate_random:
    ; Generate a big random number
    mov rax, rdi
    mov rbx, 134775813
    mov rcx, 1
    mul rbx
    add rax, rcx
    mov rdi, rax
    ; Get the remainder
    mov rbx, 55
    xor rdx, rdx
    div rbx
    inc rdx     ; Increment the remainder to get a number between 1 and 55
    mov rax, rdx
```

1.

```
concat_str_prompt:
    ; Write string prompt
    mov rdi, string_prompt_msg
    call print

    ; Read string
    mov eax, SYS_READ
    mov ebx, STDIN
    mov ecx, string       ; Where to store input
    mov edx, 255          ; Max length to read
    int 0x80

    ; Write another string prompt

    mov rdi, string_prompt2_msg
    call print

    ; Read string
    mov eax, SYS_READ
    mov ebx, STDIN
    mov ecx, string2      ; Where to store input
    mov edx, 255          ; Max length to read
    int 0x80

    ; Write result message
    mov rdi, result_msg
    call print

    ; Concat strings in string3
    mov rdi, string       ; Move string pointer into rdi
    mov rsi, string2      ; Move string2 pointer into rsi
    mov rdx, string3      ; Move string3 pointer into rdx
    call concat_str

    mov rdi, string3      ; Move string3 pointer into rdi
    call print            ; Call print procedure
```

24.

```
string_prefix_prompt:
    ; Write string prompt
    mov rdi, string_prompt_msg
    call print

    ; Read string
    mov eax, SYS_READ
    mov ebx, STDIN
    mov ecx, string      ; Where to store input
    mov edx, 255         ; Max length to read
    int 0x80

    ; Write prefix prompt
    mov rdi, prefix_msg
    call print

    ; Read prefix
    mov eax, SYS_READ
    mov ebx, STDIN
    mov ecx, string2     ; Where to store input
    mov edx, 255         ; Max length to read
    int 0x80

    ; Write result message
    mov rdi, result_msg
    call print

    ; prepare arguments for concat_str procedure
    mov rdi, string2      ; Move string2 pointer into rdi
    mov rsi, string       ; Move string pointer into rsi
    mov rdx, string3      ; Move string3 pointer into rdx
    call concat_str       ; Call concat_str procedure

    mov rdi, string3      ; Move string3 pointer into rdi
    call print            ; Call print procedure

    jmp prompt           ; Jump to prompt
```

19.

```
str_to_int_prompt:
    ; Write prompt

    mov rdi, str_to_int_msg
    call print

    ; Read string
    mov eax, SYS_READ
    mov ebx, STDIN
    mov ecx, string      ; Where to store input
    mov edx, 255         ; Max length to read
    int 0x80

    mov rdi, string      ; Move string pointer into rdi
    call str_to_int      ; Call str_to_int procedure

    mov rcx, rax         ; Move result into counter register
    mov rdi, dot         ; Move dot pointer into rdi

    _str_to_int_prompt_loop:
    call print           ; Call print procedure
    dec rcx              ; Decrement counter
    cmp rcx, 0           ; Compare counter to 0
    jne _str_to_int_prompt_loop ; Jump if not equal to 0

    call print_newline
    jmp prompt           ; Jump to prompt
```

10.

```
remove_spaces:
    ; Removes spaces from a string
    ; Input: rdi = address of string to modify

    push rsi        ; Save the address of the original string

    mov rsi, rdi    ; Copy the address of the string to rsi
    xor ecx, ecx    ; Clear ecx (counter for non-space characters)

    _remove_spaces_loop:
    mov al, byte [rsi]  ; Load the current character

    cmp al, 0          ; Check if end of string
    je _remove_spaces_done

    cmp al, byte [space]  ; Compare with space character
    je _remove_spaces_skip

    mov byte [rdi + rcx], al  ; Copy the non-space character to the modified string
    inc rcx                 ; Increment the counter

    _remove_spaces_skip:
    inc rsi              ; Increment the string pointer
    jmp _remove_spaces_loop

    _remove_spaces_done:
    mov byte [rdi + rcx], 0  ; Add null terminator to the modified string

    pop rsi               ; Restore rsi

    ret
```

13.

```
random_string_prompt:
    ; Write prompt
    mov rdi, number_msg
    call print

    ; Read number
    mov eax, SYS_READ
    mov ebx, STDIN
    mov ecx, string      ; Where to store input
    mov edx, 255         ; Max length to read
    int 0x80

    ; Convert string to int
    mov rdi, string      ; Move string pointer into rdi
    call str_to_int      ; Call str_to_int procedure

    ; Set arguments for the random_string procedure
    xor rdi, rdi         ; Clear rdi
    mov rsi, string      ; Move string pointer into rsi
    mov rdx, rax         ; Move number into rdx

    call random_string   ; Call random_string procedure

    mov rdi, string      ; Move string pointer into rdi
    call print           ; Call print procedure
    call print_newline

    jmp prompt           ; Jump to prompt
```

25.

```
add_suffix:
    ; Input: rdi = address of input string
    ;        rsi = address of suffix string
    ; Output: rdi = address of string containing result

    push rdx        ; Save the value of rdx on the stack

    ; Find the end of the input string
    xor rax, rax    ; Clear rax
    mov rcx, rdi    ; Copy the address of input string to rcx
    call str_len    ; Get the length of the input string
    add rdi, rax    ; Move rdi to the end of the input string

    ; Copy the suffix string to the end of the input string
    _add_suffix_copy:
    mov dl, [rsi]    ; Copy the byte from the suffix string to dl
    mov [rdi], dl    ; Copy dl to the end of the input string
    inc rsi         ; Increment the suffix string pointer
    inc rdi         ; Increment the input string pointer
    cmp byte [rsi], 0    ; Compare the byte in rsi to 0 (end of suffix string)
    jne _add_suffix_copy

    ; Add a null terminator to the end of the result string
    mov byte [rdi], 0

    ; Restore the value of rdx
    pop rdx

    ret
```

9.

```
reverse_string_prompt:
    ; Write string prompt
    mov rdi, string_prompt_msg
    call print

    ; Read string
    mov eax, SYS_READ
    mov ebx, STDIN
    mov ecx, string       ; Where to store input
    mov edx, 255          ; Max length to read
    int 0x80

    ; Write reverse string message
    mov rdi, reverse_msg
    call print

    mov rdi, string       ; Move string pointer into rdi
    mov rsi, string2      ; Move string2 pointer into rsi
    call reverse_str

    mov rdi, string2      ; Move string2 pointer into rdi
    call print
    call print_newline

    jmp prompt            ; Jump to prompt
```

23.

```
array_sum_prompt:
    mov rdi, array_length_msg
    call print
    ; Reset counter
    mov qword [array_sum], 0

    ; Read array length
    mov eax, SYS_READ
    mov ebx, STDIN
    mov ecx, string      ; Where to store input
    mov edx, 255         ; Max length to read
    int 0x80

    ; Convert string to int
    mov rdi, string      ; Move string pointer into rdi
    call str_to_int      ; Call str_to_int procedure

    ; Move array length to counter register
    mov rcx, rax

    ; Loop N times to read N numbers and sum them
    _array_sum_prompt_loop:
    mov rdi, number_msg
    call print

    push rcx             ; Save counter

    ; Read number
    mov eax, SYS_READ
    mov ebx, STDIN
    mov ecx, string      ; Where to store input
    mov edx, 255         ; Max length to read
    int 0x80

    pop rcx              ; Restore counter
```

```
    ; Convert string to int
    mov rdi, string      ; Move string pointer into rdi
    call str_to_int      ; Call str_to_int procedure

    ; Add number to accumulator
    add qword [array_sum], rax

    ; Decrement counter
    dec rcx

    ; Loop if counter is not 0
    cmp rcx, 0
    jne _array_sum_prompt_loop

    ; Write result message
    mov rdi, result_msg
    call print

    ; Convert accumulator to string
    mov rdi, [array_sum]  ; Move accumulator into rdi
    mov rsi, string
    call int_to_str

    ; Print result
    mov rdi, string
    call print
    call print_newline

    jmp prompt           ; Jump to prompt
```

9.

```
print_random_number:
    ; Call the random function to generate a random number
    call random
    ; Move the random number from rax to rdi for printing
    mov rdi, rax
    ; Print the random number
    mov eax, SYS_WRITE
    mov ebx, STDOUT
    mov ecx, random_msg_2
    mov edx, random_msg_len_2
    int 0x80

    ; Call the print function (assuming it's defined elsewhere)
    call print

    ; Jump to the prompt
    jmp prompt

invalid:
    mov rdi, invalid_msg

    pop rax; Restore rax
    mov rdi, choice
    call print

    jmp prompt
```

Conclusion:

In conclusion, this report has provided an overview of assembly language programming, including its syntax, structure, and application in computer systems. Through practical exercises, I have gained hands-on experience in writing programs using assembly language, providing a foundation for further exploration in this field.

By working with assembly language, I have gained insight into the underlying operations of a computer system, and the role of low-level programming in controlling hardware. This knowledge is critical in the development of software and applications for various domains, including embedded systems, operating systems, and game development.

Assembly language programming requires a thorough understanding of computer architecture and hardware, as well as a keen attention to detail. However, with practice and dedication, it is a powerful tool for developers to optimize performance and implement functionality that may not be possible using higher-level languages.

In conclusion, this laboratory work has provided a solid foundation for further exploration of assembly language programming.

Git: https://github.com/alya1007/Labs-semester-4/tree/master/AC