Міністерство освіти і науки України
Національний авіаційний університет
Навчально-науковий інститут комп'ютерних інформаційних технологій
Кафедра комп'ютеризованих систем управління

Домашня робота
з дисципліни «Системне програмування»
Варіант №24

Виконав:
студент ННІКІТ СП-225
Клокун В. Д.
Перевірив:
Артамонов Є. Б.

Київ 2018

1 Завдання

Робота складається з 2 завдань, які повинні знаходитись в одному виконуваному файлі, створеного за допомогою мови асемблера. Вибір завдань виконується за допомогою діалогового меню. Для виконання домашньої роботи згідно з номером варіанта були поставлені такі завдання:

1. Дані числа a, b ($0 < a < b$) і набір з 10 елементів. Знайти мінімальний і максимальний з елементів, що містяться в інтервалі (a, b). Якщо необхідні елементи відсутні, то вивести -1.

2. Відображення вмісту директорії (аналог `DIR`). Можливість виводу структури каталогів у вигляді дерева.

2 Хід роботи

Початковий код необхідної програми був розроблений для платформи macOS з 64-бітною розрядністю на діалекті мови асемблера NASM з використанням засобів інфраструктури LLVM.

Початковий код необхідного виконуваного файлу був розділений на 4 файли: `arrload.asm`, `main.asm`, `p01findminmax.asm` та `p02dirtree.asm`. Файл `arrload.asm` (ліст. 2.1) містить початковий код функції, яка завантажує, обробляє та виводить початковий набір даних із заданого файлу в масив.

Лістинг 2.1: Завантаження початкових даних для виконання завдання 1

```
1      section .text
2      global  _loadarr
3      extern  _fopen, _fclose, _fscanf, _printf
4
5  ; loadarr(long *arr, char *filename)
6  _loadarr:
7      ; set up the stack
8      push    rbp
9      mov     rbp, rsp
10
11     ; transfer arguments
12     mov     r15, rdi      ; r15 = arr
13     mov     r14, rsi      ; r14 = filename
14     ; open the file
15     mov     rdi, r14
16     lea     rsi, [rel moderead]
17     call    _fopen
18     mov     [rel fp], rax ; fp = fopen(filename, "r");
19
```

```
20      mov     r13, r15        ; store current array index
21      mov     r12, 10         ; i = 10
22  readarr:
23      ; read number from file
24      mov     rdi, [rel fp]
25      lea     rsi, [rel fmtldn]
26      mov     rdx, r13
27      mov     al, 0
28      call    _fscanf
29      ; fscanf(fp, "%ld\n", &a[i]);
30      add     r13, 8          ; advance to next element
31      dec     r12             ; decrease loop counter
32      cmp     r12, 0
33      jnz     readarr         ; check if loop didnt end
34  ; end readarr
35
36      mov     r13, r15
37      mov     r12, 10
38  printarr:
39      lea     rdi, [rel fmtldn]
40      mov     rsi, [rel r13]
41      mov     al, 0
42      call    _printf
43      add     r13, 8
44      dec     r12
45      cmp     r12, 0
46      jnz     printarr
47
48  fclose:
49      mov     rdi, [rel fp]
50      call    _fclose
51
52      mov     rsp, rbp
53      pop     rbp
54      ret
55  ; end loadarr()
56
57      section .bss
58  fp:     resq 1
59
60      section .data
61  moderead:
62      db "r", 0
63  fmtldn:
64      db "%ld", 10, 0
```

Файл `main.asm` (ліст. 2.2) містить початковий код основної функції програми, яка відповідає за взаємодію з користувачем, виклик функцій для виконання завдань, підготовку та вивід даних, а також завершення роботи програми.

Лістинг 2.2: Основна частина програми

```
    section .text
    global  _main
    extern  _printf, _loadarr, _scanf, _findmin, _findmax, _dir,\
    _tree

_main:
    push    rbx ; prepare the stack

taskselection:
    ; print selection prompt
    lea     rdi, [rel selprompt]
    mov     al, 0
    call    _printf

    ; scan the choice
    lea     rdi, [rel fmtsel]
    lea     rsi, [rel sel]
    mov     al, 0
    call    _scanf

    ; '1' (49) = task1, '2' (50) = task2
    ; any other char = exit
    cmp     byte [rel sel], 49
    je      task1
    cmp     byte  [rel sel], 50
    je      task2
    jmp     exit

task1:
    ; prompt for lower and upper bounds (a, b)
    lea     rdi, [rel abprompt]
    mov     al, 0
    call    _printf

    ; read the bounds
    lea     rdi, [rel fmtab]
    lea     rsi, [rel a]
    lea     rdx, [rel b]
    mov     al, 0
    call    _scanf
```

```
41
42      ; load array from file
43      lea     rdi, [rel arr]
44      lea     rsi, [rel filename]
45      call    _loadarr
46
47      ; findmin(arr, lowerbound, upperbound);
48      lea     rdi, [rel arr]
49      mov     rsi, [rel a]
50      mov     rdx, [rel b]
51      call    _findmin
52      mov     [rel resmin], rax
53
54      ; findmax(arr, lowerbound, upperbound);
55      lea     rdi, [rel arr]
56      mov     rsi, [rel a]
57      mov     rdx, [rel b]
58      call    _findmax
59      mov     [rel resmax], rax
60
61      ; print min and max values
62      lea     rdi, [rel fmtld]
63      mov     rsi, [rel resmin]
64      mov     rdx, [rel resmax]
65      mov     al, 0
66      call    _printf
67
68      ; return to task selection
69      jmp     taskselection
70
71  task2:
72      call    _dir            ; emulate DIR
73      call    _tree           ; print directory tree
74      jmp     taskselection ; return to task selection
75
76  exit:
77      pop     rbx             ; clean up the stack
78      xor     rax, rax        ; EXIT_SUCCESS
79      ret
80
81      section .bss
82  sel:    resb 256
83  a:      resq 1
84  b:      resq 1
85  resmin: resq 1
```

```
86  resmax: resq 1
87  arr:    resq 10
88
89      section .data
90  selprompt:
91      db "Welcome!", 10,\
92         "Please select the task (1, 2).", 10,\
93         "If you want to quit, type anything else.", 10,\
94         "Your choice: ", 0
95  fmtsel:
96      db "%s", 0
97  abprompt:
98      db "Enter min range bound A and max range bound B:", 10, 0
99  fmtld:
100     db "Min_ab: %ld", 9, "Max_ab: %ld", 10, 0
101 fmtab:
102     db "%ld %ld", 0
103 filename:
104     db "arrtestfile", 0
```

Файл `p01findminmax.asm` (ліст. 2.3) містить початковий код, що описує роботу функцій `findmin()` та `findmax()`. Ці функції відповідають за пошук у початковому наборі даних мінімального та максимального елементів, які містяться у заданому інтервалі.

Лістинг 2.3: Функції для пошуку мінімального та максимального елементів, що містяться в інтервалі (a, b)

```
1       section .text
2       global  _findmin, _findmax
3
4   %assign ARRAYELCOUNT 10
5
6   ; returns min value N that satisfies:
7   ; lowerbound < N < upperbound
8   ; if no such value exists, returns -1
9   ; findmin(long int *arr, long lowerbound, long upperbound);
10  _findmin:
11      ; set up the stack
12      push    rbp
13      mov     rbp, rsp
14
15      ; load inital values
16      mov     r8, rsi             ; r8 = lowerbound
17      mov     r9, rdx             ; r9 = upperbound
```

```
18      mov     r10, 0            ; minchanged = FALSE
19      mov     rax, r9           ; current min value (a[0])
20      ; rsi holds current array pointer
21      lea     rsi, [rdi]        ; rsi = arr
22      mov     rcx, ARRAYELCOUNT ; set loop counter
23      ; rax = currmin = a[0];
24      ; rsi = &a[0]
25
26  findminloop:
27      ; check if a[i] <= currmin
28       cmp     [rsi], rax
29      jg      .enditer
30      ; finalize iteration if it's not
31
32      ; check if a[i] > lowerbound
33      cmp     [rsi], r8
34      jle     .enditer
35      ; finalize iteration if it's not
36
37      ; check if a[i] < upperbound
38      cmp     [rsi], r9
39      jge     .enditer
40      ; finalize iteration if it's not
41
42      ; since lowerbound < a[i] < upperbound
43      ; set currmin = a[i]
44      mov     rax, [rsi]        ; currmin = a[i]
45      mov     r10, 1            ; minchanged = TRUE
46
47  .enditer:
48      add     rsi, 8            ; select next item in array
49      loop    findminloop
50
51      ; if min has not been changed, return rax = -1
52      cmp     r10, 1
53      je      endfindmin
54      mov     rax, -1
55
56  endfindmin:
57      ; clean up the stack
58      mov     rsp, rbp
59      pop     rbp
60      ret
61  ; end _findmin()
62
```

```
63  ; returns max value N that satisfies:
64  ; lowerbound < N < upperbound
65  ; if no such value exists, returns -1
66  ; findmax(long int *arr, long lowerbound, long upperbound);
67  _findmax:
68      ; set up the stack
69      push    rbp
70      mov     rbp, rsp
71
72      ; load inital values
73      mov     r8, rsi           ; r8 = lowerbound
74      mov     r9, rdx           ; r9 = upperbound
75      mov     r10, 0            ; maxchanged = FALSE
76      mov     rax, r8           ; current max value = lowerbound
77      ; rsi holds current array pointer
78      lea     rsi, [rdi]        ; rsi = arr
79      mov     rcx, ARRAYELCOUNT ; set loop counter
80      ; rax = currmax = a[0];
81      ; rsi = &a[0]
82
83  findmaxloop:
84      ; check if a[i] >= currmax
85       cmp     [rsi], rax
86      jl      .enditer
87      ; finalize iteration if it's not
88
89      ; check if a[i] > lowerbound
90      cmp     [rsi], r8
91      jle     .enditer
92      ; finalize iteration if it's not
93
94      ; check if a[i] < upperbound
95      cmp     [rsi], r9
96      jge     .enditer
97      ; finalize iteration if it's not
98
99      ; since lowerbound < a[i] < upperbound
100     ; set currmax = a[i]
101     mov     rax, [rsi]        ; currmax = a[i]
102     mov     r10, 1            ; maxchanged = TRUE
103
104 .enditer:
105     add     rsi, 8            ; select next item in array
106     loop    findmaxloop
107
```

```
108     ; if max has not been changed, return rax = -1
109     cmp     r10, 1
110     je      endfindmax
111     mov     rax, -1
112
113 endfindmax:
114     ; clean up the stack
115     mov     rsp, rbp
116     pop     rbp
117     ret
118 ; end _findmax()
```

Файл `p02dirtree.asm` (ліст. 2.4) містить початковий код функцій `dir()` та `tree()`, які відповідають за вивід вмісту директорії та структури каталогу у вигляді дерева.

Лістинг 2.4: Функції для виводу змісту директорії та структури каталогів у вигляді дерева

```
1     section .text
2     global  _dir, _tree
3     extern  _system
4
5  ; emulates windows DIR behavior
6  ; dir(void)
7  _dir:
8     push    rbp
9     mov     rbp, rsp
10
11    lea     rdi, [rel ls]
12    call    _system
13
14    mov     rsp, rbp
15    pop     rbp
16
17    ret
18 ; end dir()
19
20 ; prints directory contents as a tree
21 ; tree(void)
22 _tree:
23    push    rbp
24    mov     rbp, rsp
25
26    lea     rdi, [rel tree]
```

```
27      call    _system
28
29      mov     rsp, rbp
30      pop     rbp
31
32      ret
33  ; end tree()
34
35      section .data
36  ls:
37      db "ls", 0
38  tree:
39      db "find . -print | sed -e 's;[^/]*/;|____;g;s;____|; |;g'",\
40      0
```

3 Висновок

Під час виконання домашньої роботи ми закріпили розуміння написання простих програм на мові асемблера та застосування простих і складних алгоритмів, а також використання спеціальних команд управління.