

Лабораторна робота №1

Аналіз програмного коду мов високого рівня

Виконав:

Студент 3 курсу ФТІ

групи ФІ-92

Поночевний Назар Юрійович

Варіант 6

Мета роботи

Отримати навички розпізнавання конструкцій мов високого рівня в машинному коді для архітектур x86/x64 та ARM/ARM64 на прикладі C/C++.

Завдання 1:

Проаналізувати машинний код прикладу hanoi.c для Windows x64, ARM, ARM64 (MSVC), для Linux amd64, arm, arm64 (GCC), для Linux amd64 (LLVM clang, <https://llvm.org/>);

- 1) Спочатку перевіримо реалізацію hanoi.cpp

The screenshot shows the Microsoft Visual Studio interface. On the left is the code editor with the file 'hanoi.cpp' open. The code implements the Hanoi Tower problem using recursion. On the right is the 'Output' window titled '(Global Scope)' which displays the command-line arguments passed to the debugger: 'Move disk 1 from A to C', 'Move disk 2 from A to B', 'Move disk 1 from C to B', 'Move disk 3 from A to C', 'Move disk 1 from B to A', 'Move disk 2 from B to C', and 'Move disk 1 from A to C'. Below the command-line output, it says 'Press any key to close this window.'

```
#include <iostream>
void hanoi(int n, char from, char to, char aux) {
    if (n == 1)
        printf(" Move disk 1 from %c to %c\n", from, to);
    else {
        hanoi(n - 1, from, aux, to);
        printf(" Move disk %d from %c to %c\n ", n, from, to);
        hanoi(n - 1, aux, to, from);
    }
}
int main() {
    hanoi(3, 'A', 'C', 'B');
```

- 2) Переводимо MSVC у режим x64 і компілюємо

The screenshot shows a terminal window running the Microsoft Visual Studio 2019 Developer Command Prompt. It starts by executing 'vcvarsall.bat x86_amd64' to initialize the environment for x64 compilation. Then, it runs the command 'cl /FAcsu hanoi.cpp' to compile the 'hanoi.cpp' file. The output shows the compiler version (Microsoft (R) C/C++ Optimizing Compiler Version 19.29.30133 for x64), copyright information (Copyright (C) Microsoft Corporation. All rights reserved.), the linker version (Microsoft (R) Incremental Linker Version 14.29.30133.0), and the final output files ('/out:hanoi.exe' and 'hanoi.obj').

```
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Auxiliary\Build>vcvarsall.bat x86_amd64
*****
** Visual Studio 2019 Developer Command Prompt v16.11.0
** Copyright (c) 2021 Microsoft Corporation
*****
[vcvarsall.bat] Environment initialized for: 'x86_x64'

C:\Users\User\source\repos\Lab1-reverse\Lab1-reverse>cl /FAcsu hanoi.cpp
Microsoft (R) C/C++ Optimizing Compiler Version 19.29.30133 for x64
Copyright (C) Microsoft Corporation. All rights reserved.

hanoi.cpp
Microsoft (R) Incremental Linker Version 14.29.30133.0
Copyright (C) Microsoft Corporation. All rights reserved.

/out:hanoi.exe
hanoi.obj
```

3) Переводимо MSVC у режим ARM і компілюємо

```
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Auxiliary\Build>vcvarsall.bat x86_arm
*****
** Visual Studio 2019 Developer Command Prompt v16.11.0
** Copyright (c) 2021 Microsoft Corporation
*****
[vcvarsall.bat] Environment initialized for: 'x86_arm'

C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Auxiliary\Build>
C:\Users\User\source\repos\Lab1-reverse\Lab1-reverse>cl /FAcsu hanoi.cpp
Microsoft (R) C/C++ Optimizing Compiler Version 19.29.30133 for x86
Copyright (C) Microsoft Corporation. All rights reserved.

hanoi.cpp
Microsoft (R) Incremental Linker Version 14.29.30133.0
Copyright (C) Microsoft Corporation. All rights reserved.

/out:hanoi.exe
hanoi.obj

C:\Users\User\source\repos\Lab1-reverse\Lab1-reverse>dir
Volume in drive C has no label.
Volume Serial Number is E411-13D6

Directory of C:\Users\User\source\repos\Lab1-reverse\Lab1-reverse

10/13/2021  09:42 AM    <DIR>      .
10/13/2021  09:42 AM    <DIR>      ..
10/13/2021  09:21 AM    <DIR>      Debug
10/13/2021  09:42 AM            7,458 hanoi.cod
10/13/2021  09:20 AM            321 hanoi.cpp
10/13/2021  09:42 AM            102,400 hanoi.exe
10/13/2021  09:42 AM            2,063 hanoi.obj
10/13/2021  09:20 AM            7,195 Lab1-reverse.vcxproj
10/13/2021  09:20 AM            978 Lab1-reverse.vcxproj.filters
10/13/2021  09:18 AM            168 Lab1-reverse.vcxproj.user
                           7 File(s)       120,583 bytes
                           3 Dir(s)   78,482,485,248 bytes free
```

4) Переводимо MSVC у режим ARM64 і компілюємо

```
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Auxiliary\Build>vcvarsall.bat x86_arm64
*****
** Visual Studio 2019 Developer Command Prompt v16.11.0
** Copyright (c) 2021 Microsoft Corporation
*****
[vcvarsall.bat] Environment initialized for: 'x86_arm64'

C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Auxiliary\Build>
```

```
C:\Users\User\source\repos\Lab1-reverse\Lab1-reverse>cl /FAcsu hanoi.cpp
Microsoft (R) C/C++ Optimizing Compiler Version 19.29.30133 for x86
Copyright (C) Microsoft Corporation. All rights reserved.

hanoi.cpp
Microsoft (R) Incremental Linker Version 14.29.30133.0
Copyright (C) Microsoft Corporation. All rights reserved.

/out:hanoi.exe
hanoi.obj
hanoi.obj : error LNK2019: unresolved external symbol __acrt_iob_func referenced in function _printf
hanoi.obj : error LNK2019: unresolved external symbol __stdio_common_vfprintf referenced in function __vfprintf_l
LINK : error LNK2001: unresolved external symbol _mainCRTStartup
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.29.30133\lib\x64\libcpmt.lib : warning LNK4272: library machine type 'x64' conflicts with target machine type 'x86'
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.29.30133\lib\x64\LIBCMT.lib : warning LNK4272: library machine type 'x64' conflicts with target machine type 'x86'
hanoi.exe : fatal error LNK1120: 3 unresolved externals

C:\Users\User\source\repos\Lab1-reverse\Lab1-reverse>
```

- 5) Bci коди додав у архів. Тепер спробуємо зробити те саме для Linux amd64 (gcc)

```
└─(kali㉿kali)-[~/lab1-reverse]
└─$ gcc -g hanoi.c -o hanoi.amd64

└─(kali㉿kali)-[~/lab1-reverse]
└─$ objdump -drwC -Mintel -S hanoi.amd64 > hanoi.amd64.2.lst
```

- 6) Тепер Linux ARM (gcc)

```
└─(kali㉿kali)-[~/lab1-reverse]
└─$ arm-linux-gnueabi-gcc -g hanoi.c -o hanoi.arm

└─(kali㉿kali)-[~/lab1-reverse]
└─$ arm-linux-gnueabi-objdump -drwC -S hanoi.arm > hanoi_gcc_arm.lst
```

- 7) I Linux arm64 (gcc)

```
└─(kali㉿kali)-[~/lab1-reverse]
└─$ aarch64-linux-gnu-gcc -g hanoi.c -o hanoi.aarch64

└─(kali㉿kali)-[~/lab1-reverse]
└─$ aarch64-linux-gnu-objdump -drwC -S hanoi.aarch64 > hanoi_gcc_aarch64.lst
```

- 8) Файли зберігаємо у архів і спробуємо для Linux amd64 (llvm clang)

```
└─(kali㉿kali)-[~/lab1-reverse]
└─$ clang -g hanoi.c -o hanoi.amd64

└─(kali㉿kali)-[~/lab1-reverse]
└─$ llvm-objdump -drwC -S hanoi.amd64 > hanoi_llvm_amd64.lst
llvm-objdump: warning: 'hanoi.amd64': failed to parse debug information for hanoi.amd64
```

- 9) Тепер час порівняти як відрізняються машинний код у MSVC, GCC та LLVM

```
GROUP 1
hanoi_llvm_amd64.ilst

GROUP 2
hanoi_llvm_amd64.ilst

GROUP 3
hanoi_llvm_amd64.ilst

GROUP 4
hanoi_llvm_amd64.ilst

hanoi_llvm_amd64.ilst
1 ?hanoi@YAXHDD@Z PROC ; hanoi
2
3 ; 2 : void hanoi(int n, char from,
4 char to, char aux) {
5
6     $LNS5:
7         .0000044 48 8C 24 20 mov    BYTE PTR [rbp+32], r9b
8         .00005 44 88 44 24 18 mov    BYTE PTR [rbp+44], r8b
9         .00008 88 54 24 10 mov    BYTE PTR [rbp+16], d1
10        .00006 89 4C 24 08 mov    DWORD PTR [rbp+8], ecx
11        .00012 48 83 ec 28 sub    rbp, 40
12        ; 0000028H
13
14 ; 3 : if (n == 1)
15     .00016 83 7c 24 30 01 cmp    DWORD PTR [r9b], 1
16     .0001b 75 1d jne    SHORT $LN2@hanoi
17
18 ; 4 : printf(" Move disk 1 from %c
19 to %c\n", from, to);
20
21     .0001d 0F be 44 24 40 movsx  eax, BYTE PTR [r9b]
22     .00022 0F be 4c 24 38 movsx  ecx, BYTE PTR [rbp+r9b]
23     .00027 44 8D 00 00 00 mov    r8d, eax
24     .00028 8B D1 00 edx, ecx
25     .00033 E8 00 00 00 00 call    printf
26     .00038 EB 5C jmp    SHORT $LN3@hanoi
27
28 ; 5 : else {
29
hanoi_llvm_amd64.ilst
1 0000000000001135 ;hanoi:
2 #include <stdio.h>
3
4 void hanoi(int n, char from, char to,
5 char aux) {
6     .1135: 55
7         push   rbp
8         .1136: 48 89 e5
9         mov    rbp, rsp
10        .1139: 48 83 ec 10
11        sub    $16, %rsp
12        sub    rsp, 0x10
13        .113d: 89 7d fc
14        mov    DWORD PTR [rbp-0x4], edi
15        .1140: 89 c8
16        mov    eax, ecx
17        mov    ecx, esi
18        .1142: 89 f1
19        mov    eax, esi
20        .1144: 88 4d f8
21        mov    BYTE PTR [rbp-0x8], cl
22        .1147: 88 55 f4
23        mov    BYTE PTR [rbp-0xc], dl
24        .114a: 88 45 f0
25        mov    BYTE PTR [rbp-0x10], al
26        if (n == 1)
27        .114d: 83 7d fc 01
28        cmp    DWORD PTR [rbp-0x4], 0x1
29        .1151: 75 1d
30        jne    SHORT $LN2@hanoi
31        .1178: ;hanoi+0x3b
32        printf("Move disk 1 from %c to
33 %c\n", from, to);
34        .1153: 0F be 55 f4
35        movsx  edx, BYTE PTR [rbp-0xc]
36        .1157: 0F be 45 f8
37        mov    eax, BYTE PTR [rbp-0x8]
38        .115b: 89 c6
39        mov    esi, eax
40
41        .115d: 48 83 3d 20 00 00 00
42        lea    rdi,[rip+0x9a]
43        .2004: ;IO_stdin.uscd+0x4> #
44
45 ; 6 : if (n > 1)
46 ; 7 : hanoi(n-1, from, aux, to);
47 ; 8 : hanoi(n-1, aux, to, from);
48 ; 9 : aux = aux + 1;
49 ; 10 : IO_stdin.uscd+0x4> #
50
51 ; 11 : else {
52
hanoi_llvm_amd64.ilst
1 ;000000000000401135 ;hanoi:
2 void hanoi(int n, char from, char to,
3 char aux) {
4     .401136: 55
5         pushq  %rbp
6         .401134: 48 83 ec 10
7         sub    $16, %rsp
8         .401138: 89 7d fc
9         movl  %edi, -4(%rbp)
10        .40113b: 48 88 75 fb
11        movb  %sil, -5(%rbp)
12        .40113f: 88 55 fa
13        movb  %dl, -6(%rbp)
14        .401142: 88 4d f9
15        movb  %cl, -7(%rbp)
16        ; if (n == 1)
17        .401145: 83 7d fc 01
18        cmp    $1, -4(%rbp)
19        .401149: 0F 85 0E 00 00 00
20        jne    SHORT $LN1@hanoi+0x3d
21        ; printf("Move disk 1 from %c to
22 %c\n", from, to);
23        .401151: 0F be 75 fb
24        movsb  -5(%rbp), %esi
25        .401153: 0F be 55 fa
26        movsb  -6(%rbp), %edx
27        .401157: 48 bf 24 20 00 00 00 00
28        movabsq $40202500, %rdi
29        .401161: 0F be 00 00
30        mov    $0, %eax
31        .401163: 48 c8 fe ff ff
32        call    0x4041030(<printf@plt>)
33        .401168: 0F 50 00 00 00
34        jmp    .401161->hanoi+0x9a
35
36 ; 11 : else {
37 ; 12 : hanoi(n-1, from, aux, to);
38 ; 13 : hanoi(n-1, aux, to, from);
39 ; 14 : aux = aux + 1;
40 ; 15 : IO_stdin.uscd+0x4> #
41
42 ; 16 : else {
43 ; 17 : aux = aux + 1;
44 ; 18 : IO_stdin.uscd+0x4> #
45
46 ; 19 : else {
47 ; 20 : hanoi(n-1, aux, to, from);
48 ; 21 : aux = aux + 1;
49 ; 22 : IO_stdin.uscd+0x4> #
50
51 ; 23 : else {
52 ; 24 : hanoi(n-1, aux, to, from);
53 ; 25 : aux = aux + 1;
54 ; 26 : IO_stdin.uscd+0x4> #
55
56 ; 27 : else {
57 ; 28 : hanoi(n-1, aux, to, from);
58 ; 29 : aux = aux + 1;
59 ; 30 : IO_stdin.uscd+0x4> #
60
61 ; 31 : else {
62 ; 32 : hanoi(n-1, aux, to, from);
63 ; 33 : aux = aux + 1;
64 ; 34 : IO_stdin.uscd+0x4> #
65
66 ; 35 : else {
67 ; 36 : hanoi(n-1, aux, to, from);
68 ; 37 : aux = aux + 1;
69 ; 38 : IO_stdin.uscd+0x4> #
70
71 ; 39 : else {
72 ; 40 : hanoi(n-1, aux, to, from);
73 ; 41 : aux = aux + 1;
74 ; 42 : IO_stdin.uscd+0x4> #
75
76 ; 43 : else {
77 ; 44 : hanoi(n-1, aux, to, from);
78 ; 45 : aux = aux + 1;
79 ; 46 : IO_stdin.uscd+0x4> #
80
81 ; 47 : else {
82 ; 48 : hanoi(n-1, aux, to, from);
83 ; 49 : aux = aux + 1;
84 ; 50 : IO_stdin.uscd+0x4> #
85
86 ; 51 : else {
87 ; 52 : hanoi(n-1, aux, to, from);
88 ; 53 : aux = aux + 1;
89 ; 54 : IO_stdin.uscd+0x4> #
90
91 ; 55 : else {
92 ; 56 : hanoi(n-1, aux, to, from);
93 ; 57 : aux = aux + 1;
94 ; 58 : IO_stdin.uscd+0x4> #
95
96 ; 59 : else {
97 ; 60 : hanoi(n-1, aux, to, from);
98 ; 61 : aux = aux + 1;
99 ; 62 : IO_stdin.uscd+0x4> #
100
101 ; 63 : else {
102 ; 64 : hanoi(n-1, aux, to, from);
103 ; 65 : aux = aux + 1;
104 ; 66 : IO_stdin.uscd+0x4> #
105
106 ; 67 : else {
107 ; 68 : hanoi(n-1, aux, to, from);
108 ; 69 : aux = aux + 1;
109 ; 70 : IO_stdin.uscd+0x4> #
110
111 ; 71 : else {
112 ; 72 : hanoi(n-1, aux, to, from);
113 ; 73 : aux = aux + 1;
114 ; 74 : IO_stdin.uscd+0x4> #
115
116 ; 75 : else {
117 ; 76 : hanoi(n-1, aux, to, from);
118 ; 77 : aux = aux + 1;
119 ; 78 : IO_stdin.uscd+0x4> #
120
121 ; 79 : else {
122 ; 80 : hanoi(n-1, aux, to, from);
123 ; 81 : aux = aux + 1;
124 ; 82 : IO_stdin.uscd+0x4> #
125
126 ; 83 : else {
127 ; 84 : hanoi(n-1, aux, to, from);
128 ; 85 : aux = aux + 1;
129 ; 86 : IO_stdin.uscd+0x4> #
130
131 ; 87 : else {
132 ; 88 : hanoi(n-1, aux, to, from);
133 ; 89 : aux = aux + 1;
134 ; 90 : IO_stdin.uscd+0x4> #
135
136 ; 91 : else {
137 ; 92 : hanoi(n-1, aux, to, from);
138 ; 93 : aux = aux + 1;
139 ; 94 : IO_stdin.uscd+0x4> #
140
141 ; 95 : else {
142 ; 96 : hanoi(n-1, aux, to, from);
143 ; 97 : aux = aux + 1;
144 ; 98 : IO_stdin.uscd+0x4> #
145
146 ; 99 : else {
147 ; 100 : hanoi(n-1, aux, to, from);
148 ; 101 : aux = aux + 1;
149 ; 102 : IO_stdin.uscd+0x4> #
150
151 ; 103 : else {
152 ; 104 : hanoi(n-1, aux, to, from);
153 ; 105 : aux = aux + 1;
154 ; 106 : IO_stdin.uscd+0x4> #
155
156 ; 107 : else {
157 ; 108 : hanoi(n-1, aux, to, from);
158 ; 109 : aux = aux + 1;
159 ; 110 : IO_stdin.uscd+0x4> #
160
161 ; 111 : else {
162 ; 112 : hanoi(n-1, aux, to, from);
163 ; 113 : aux = aux + 1;
164 ; 114 : IO_stdin.uscd+0x4> #
165
166 ; 115 : else {
167 ; 116 : hanoi(n-1, aux, to, from);
168 ; 117 : aux = aux + 1;
169 ; 118 : IO_stdin.uscd+0x4> #
170
171 ; 119 : else {
172 ; 120 : hanoi(n-1, aux, to, from);
173 ; 121 : aux = aux + 1;
174 ; 122 : IO_stdin.uscd+0x4> #
175
176 ; 123 : else {
177 ; 124 : hanoi(n-1, aux, to, from);
178 ; 125 : aux = aux + 1;
179 ; 126 : IO_stdin.uscd+0x4> #
180
181 ; 127 : else {
182 ; 128 : hanoi(n-1, aux, to, from);
183 ; 129 : aux = aux + 1;
184 ; 130 : IO_stdin.uscd+0x4> #
185
186 ; 131 : else {
187 ; 132 : hanoi(n-1, aux, to, from);
188 ; 133 : aux = aux + 1;
189 ; 134 : IO_stdin.uscd+0x4> #
190
191 ; 135 : else {
192 ; 136 : hanoi(n-1, aux, to, from);
193 ; 137 : aux = aux + 1;
194 ; 138 : IO_stdin.uscd+0x4> #
195
196 ; 139 : else {
197 ; 140 : hanoi(n-1, aux, to, from);
198 ; 141 : aux = aux + 1;
199 ; 142 : IO_stdin.uscd+0x4> #
200
201 ; 143 : else {
202 ; 144 : hanoi(n-1, aux, to, from);
203 ; 145 : aux = aux + 1;
204 ; 146 : IO_stdin.uscd+0x4> #
205
206 ; 147 : else {
207 ; 148 : hanoi(n-1, aux, to, from);
208 ; 149 : aux = aux + 1;
209 ; 150 : IO_stdin.uscd+0x4> #
210
211 ; 151 : else {
212 ; 152 : hanoi(n-1, aux, to, from);
213 ; 153 : aux = aux + 1;
214 ; 154 : IO_stdin.uscd+0x4> #
215
216 ; 155 : else {
217 ; 156 : hanoi(n-1, aux, to, from);
218 ; 157 : aux = aux + 1;
219 ; 158 : IO_stdin.uscd+0x4> #
220
221 ; 159 : else {
222 ; 160 : hanoi(n-1, aux, to, from);
223 ; 161 : aux = aux + 1;
224 ; 162 : IO_stdin.uscd+0x4> #
225
226 ; 163 : else {
227 ; 164 : hanoi(n-1, aux, to, from);
228 ; 165 : aux = aux + 1;
229 ; 166 : IO_stdin.uscd+0x4> #
230
231 ; 167 : else {
232 ; 168 : hanoi(n-1, aux, to, from);
233 ; 169 : aux = aux + 1;
234 ; 170 : IO_stdin.uscd+0x4> #
235
236 ; 171 : else {
237 ; 172 : hanoi(n-1, aux, to, from);
238 ; 173 : aux = aux + 1;
239 ; 174 : IO_stdin.uscd+0x4> #
240
241 ; 175 : else {
242 ; 176 : hanoi(n-1, aux, to, from);
243 ; 177 : aux = aux + 1;
244 ; 178 : IO_stdin.uscd+0x4> #
245
246 ; 179 : else {
247 ; 180 : hanoi(n-1, aux, to, from);
248 ; 181 : aux = aux + 1;
249 ; 182 : IO_stdin.uscd+0x4> #
250
251 ; 183 : else {
252 ; 184 : hanoi(n-1, aux, to, from);
253 ; 185 : aux = aux + 1;
254 ; 186 : IO_stdin.uscd+0x4> #
255
256 ; 187 : else {
257 ; 188 : hanoi(n-1, aux, to, from);
258 ; 189 : aux = aux + 1;
259 ; 190 : IO_stdin.uscd+0x4> #
260
261 ; 191 : else {
262 ; 192 : hanoi(n-1, aux, to, from);
263 ; 193 : aux = aux + 1;
264 ; 194 : IO_stdin.uscd+0x4> #
265
266 ; 195 : else {
267 ; 196 : hanoi(n-1, aux, to, from);
268 ; 197 : aux = aux + 1;
269 ; 198 : IO_stdin.uscd+0x4> #
270
271 ; 199 : else {
272 ; 200 : hanoi(n-1, aux, to, from);
273 ; 201 : aux = aux + 1;
274 ; 202 : IO_stdin.uscd+0x4> #
275
276 ; 203 : else {
277 ; 204 : hanoi(n-1, aux, to, from);
278 ; 205 : aux = aux + 1;
279 ; 206 : IO_stdin.uscd+0x4> #
280
281 ; 207 : else {
282 ; 208 : hanoi(n-1, aux, to, from);
283 ; 209 : aux = aux + 1;
284 ; 210 : IO_stdin.uscd+0x4> #
285
286 ; 211 : else {
287 ; 212 : hanoi(n-1, aux, to, from);
288 ; 213 : aux = aux + 1;
289 ; 214 : IO_stdin.uscd+0x4> #
290
291 ; 215 : else {
292 ; 216 : hanoi(n-1, aux, to, from);
293 ; 217 : aux = aux + 1;
294 ; 218 : IO_stdin.uscd+0x4> #
295
296 ; 219 : else {
297 ; 220 : hanoi(n-1, aux, to, from);
298 ; 221 : aux = aux + 1;
299 ; 222 : IO_stdin.uscd+0x4> #
300
301 ; 223 : else {
302 ; 224 : hanoi(n-1, aux, to, from);
303 ; 225 : aux = aux + 1;
304 ; 226 : IO_stdin.uscd+0x4> #
305
306 ; 227 : else {
307 ; 228 : hanoi(n-1, aux, to, from);
308 ; 229 : aux = aux + 1;
309 ; 230 : IO_stdin.uscd+0x4> #
310
311 ; 231 : else {
312 ; 233 : hanoi(n-1, aux, to, from);
313 ; 234 : aux = aux + 1;
314 ; 235 : IO_stdin.uscd+0x4> #
315
316 ; 236 : else {
317 ; 237 : hanoi(n-1, aux, to, from);
318 ; 238 : aux = aux + 1;
319 ; 239 : IO_stdin.uscd+0x4> #
320
321 ; 240 : else {
322 ; 241 : hanoi(n-1, aux, to, from);
323 ; 242 : aux = aux + 1;
324 ; 243 : IO_stdin.uscd+0x4> #
325
326 ; 244 : else {
327 ; 245 : hanoi(n-1, aux, to, from);
328 ; 246 : aux = aux + 1;
329 ; 247 : IO_stdin.uscd+0x4> #
330
331 ; 248 : else {
332 ; 249 : hanoi(n-1, aux, to, from);
333 ; 250 : aux = aux + 1;
334 ; 251 : IO_stdin.uscd+0x4> #
335
336 ; 252 : else {
337 ; 253 : hanoi(n-1, aux, to, from);
338 ; 254 : aux = aux + 1;
339 ; 255 : IO_stdin.uscd+0x4> #
340
341 ; 256 : else {
342 ; 257 : hanoi(n-1, aux, to, from);
343 ; 258 : aux = aux + 1;
344 ; 259 : IO_stdin.uscd+0x4> #
345
346 ; 260 : else {
347 ; 261 : hanoi(n-1, aux, to, from);
348 ; 262 : aux = aux + 1;
349 ; 263 : IO_stdin.uscd+0x4> #
350
351 ; 264 : else {
352 ; 265 : hanoi(n-1, aux, to, from);
353 ; 266 : aux = aux + 1;
354 ; 267 : IO_stdin.uscd+0x4> #
355
356 ; 268 : else {
357 ; 269 : hanoi(n-1, aux, to, from);
358 ; 270 : aux = aux + 1;
359 ; 271 : IO_stdin.uscd+0x4> #
360
361 ; 272 : else {
362 ; 273 : hanoi(n-1, aux, to, from);
363 ; 274 : aux = aux + 1;
364 ; 275 : IO_stdin.uscd+0x4> #
365
366 ; 276 : else {
367 ; 277 : hanoi(n-1, aux, to, from);
368 ; 278 : aux = aux + 1;
369 ; 279 : IO_stdin.uscd+0x4> #
370
371 ; 280 : else {
372 ; 281 : hanoi(n-1, aux, to, from);
373 ; 282 : aux = aux + 1;
374 ; 283 : IO_stdin.uscd+0x4> #
375
376 ; 284 : else {
377 ; 285 : hanoi(n-1, aux, to, from);
378 ; 286 : aux = aux + 1;
379 ; 287 : IO_stdin.uscd+0x4> #
380
381 ; 288 : else {
382 ; 289 : hanoi(n-1, aux, to, from);
383 ; 290 : aux = aux + 1;
384 ; 291 : IO_stdin.uscd+0x4> #
385
386 ; 292 : else {
387 ; 293 : hanoi(n-1, aux, to, from);
388 ; 294 : aux = aux + 1;
389 ; 295 : IO_stdin.uscd+0x4> #
390
391 ; 296 : else {
392 ; 297 : hanoi(n-1, aux, to, from);
393 ; 298 : aux = aux + 1;
394 ; 299 : IO_stdin.uscd+0x4> #
395
396 ; 300 : else {
397 ; 301 : hanoi(n-1, aux, to, from);
398 ; 302 : aux = aux + 1;
399 ; 303 : IO_stdin.uscd+0x4> #
400
401 ; 304 : else {
402 ; 305 : hanoi(n-1, aux, to, from);
403 ; 306 : aux = aux + 1;
404 ; 307 : IO_stdin.uscd+0x4> #
405
406 ; 308 : else {
407 ; 309 : hanoi(n-1, aux, to, from);
408 ; 310 : aux = aux + 1;
409 ; 311 : IO_stdin.uscd+0x4> #
410
411 ; 312 : else {
412 ; 313 : hanoi(n-1, aux, to, from);
413 ; 314 : aux = aux + 1;
414 ; 315 : IO_stdin.uscd+0x4> #
415
416 ; 316 : else {
417 ; 317 : hanoi(n-1, aux, to, from);
418 ; 318 : aux = aux + 1;
419 ; 319 : IO_stdin.uscd+0x4> #
420
421 ; 320 : else {
422 ; 321 : hanoi(n-1, aux, to, from);
423 ; 322 : aux = aux + 1;
424 ; 323 : IO_stdin.uscd+0x4> #
425
426 ; 324 : else {
427 ; 325 : hanoi(n-1, aux, to, from);
428 ; 326 : aux = aux + 1;
429 ; 327 : IO_stdin.uscd+0x4> #
430
431 ; 328 : else {
432 ; 329 : hanoi(n-1, aux, to, from);
433 ; 330 : aux = aux + 1;
434 ; 331 : IO_stdin.uscd+0x4> #
435
436 ; 332 : else {
437 ; 333 : hanoi(n-1, aux, to, from);
438 ; 334 : aux = aux + 1;
439 ; 335 : IO_stdin.uscd+0x4> #
440
441 ; 336 : else {
442 ; 337 : hanoi(n-1, aux, to, from);
443 ; 338 : aux = aux + 1;
444 ; 339 : IO_stdin.uscd+0x4> #
445
446 ; 340 : else {
447 ; 341 : hanoi(n-1, aux, to, from);
448 ; 342 : aux = aux + 1;
449 ; 343 : IO_stdin.uscd+0x4> #
450
451 ; 344 : else {
452 ; 345 : hanoi(n-1, aux, to, from);
453 ; 346 : aux = aux + 1;
454 ; 347 : IO_stdin.uscd+0x4> #
455
456 ; 348 : else {
457 ; 349 : hanoi(n-1, aux, to, from);
458 ; 350 : aux = aux + 1;
459 ; 351 : IO_stdin.uscd+0x4> #
460
461 ; 352 : else {
462 ; 353 : hanoi(n-1, aux, to, from);
463 ; 354 : aux = aux + 1;
464 ; 355 : IO_stdin.uscd+0x4> #
465
466 ; 356 : else {
467 ; 357 : hanoi(n-1, aux, to, from);
468 ; 358 : aux = aux + 1;
469 ; 359 : IO_stdin.uscd+0x4> #
470
471 ; 360 : else {
472 ; 361 : hanoi(n-1, aux, to, from);
473 ; 362 : aux = aux + 1;
474 ; 363 : IO_stdin.uscd+0x4> #
475
476 ; 364 : else {
477 ; 365 : hanoi(n-1, aux, to, from);
478 ; 366 : aux = aux + 1;
479 ; 367 : IO_stdin.uscd+0x4> #
480
481 ; 368 : else {
482 ; 369 : hanoi(n-1, aux, to, from);
483 ; 370 : aux = aux + 1;
484 ; 371 : IO_stdin.uscd+0x4> #
485
486 ; 372 : else {
487 ; 373 : hanoi(n-1, aux, to, from);
488 ; 374 : aux = aux + 1;
489 ; 375 : IO_stdin.uscd+0x4> #
490
491 ; 376 : else {
492 ; 377 : hanoi(n-1, aux, to, from);
493 ; 378 : aux = aux + 1;
494 ; 379 : IO_stdin.uscd+0x4> #
495
496 ; 380 : else {
497 ; 381 : hanoi(n-1, aux, to, from);
498 ; 382 : aux = aux + 1;
499 ; 383 : IO_stdin.uscd+0x4> #
500
501 ; 384 : else {
502 ; 385 : hanoi(n-1, aux, to, from);
503 ; 386 : aux = aux + 1;
504 ; 387 : IO_stdin.uscd+0x4> #
505
506 ; 388 : else {
507 ; 389 : hanoi(n-1, aux, to, from);
508 ; 390 : aux = aux + 1;
509 ; 391 : IO_stdin.uscd+0x4> #
510
511 ; 392 : else {
512 ; 393 : hanoi(n-1, aux, to, from);
513 ; 394 : aux = aux + 1;
514 ; 395 : IO_stdin.uscd+0x4> #
515
516 ; 396 : else {
517 ; 397 : hanoi(n-1, aux, to, from);
518 ; 398 : aux = aux + 1;
519 ; 399 : IO_stdin.uscd+0x4> #
520
521 ; 400 : else {
522 ; 401 : hanoi(n-1, aux, to, from);
523 ; 402 : aux = aux + 1;
524 ; 403 : IO_stdin.uscd+0x4> #
525
526 ; 404 : else {
527 ; 405 : hanoi(n-1, aux, to, from);
528 ; 406 : aux = aux + 1;
529 ; 407 : IO_stdin.uscd+0x4> #
530
531 ; 408 : else {
532 ; 409 : hanoi(n-1, aux, to, from);
533 ; 410 : aux = aux + 1;
534 ; 411 : IO_stdin.uscd+0x4> #
535
536 ; 412 : else {
537 ; 413 : hanoi(n-1, aux, to, from);
538 ; 414 : aux = aux + 1;
539 ; 415 : IO_stdin.uscd+0x4> #
540
541 ; 416 : else {
542 ; 417 : hanoi(n-1, aux, to, from);
543 ; 418 : aux = aux + 1;
544 ; 419 : IO_stdin.uscd+0x4> #
545
546 ; 420 : else {
547 ; 421 : hanoi(n-1, aux, to, from);
548 ; 422 : aux = aux + 1;
549 ; 423 : IO_stdin.uscd+0x4> #
550
551 ; 424 : else {
552 ; 425 : hanoi(n-1, aux, to, from);
553 ; 426 : aux = aux + 1;
554 ; 427 : IO_stdin.uscd+0x4> #
555
556 ; 428 : else {
557 ; 429 : hanoi(n-1, aux, to, from);
558 ; 430 : aux = aux + 1;
559 ; 431 : IO_stdin.uscd+0x4> #
560
561 ; 432 : else {
562 ; 433 : hanoi(n-1, aux, to, from);
563 ; 434 : aux = aux + 1;
564 ; 435 : IO_stdin.uscd+0x4> #
565
566 ; 436 : else {
567 ; 437 : hanoi(n-1, aux, to, from);
568 ; 438 : aux = aux + 1;
569 ; 439 : IO_stdin.uscd+0x4> #
570
571 ; 440 : else {
572 ; 441 : hanoi(n-1, aux, to, from);
573 ; 442 : aux = aux + 1;
574 ; 443 : IO_stdin.uscd+0x4> #
575
576 ; 444 : else {
577 ; 445 : hanoi(n-1, aux, to, from);
578 ; 446 : aux = aux + 1;
579 ; 447 : IO_stdin.uscd+0x4> #
580
581 ; 448 : else {
582 ; 449 : hanoi(n-1, aux, to, from);
583 ; 450 : aux = aux + 1;
584 ; 451 : IO_stdin.uscd+0x4> #
585
586 ; 452 : else {
587 ; 453 : hanoi(n-1, aux, to, from);
588 ; 454 : aux = aux + 1;
589 ; 455 : IO_stdin.uscd+0x4> #
590
591 ; 456 : else {
592 ; 457 : hanoi(n-1, aux, to, from);
593 ; 458 : aux = aux + 1;
594 ; 459 : IO_stdin.uscd+0x4> #
595
596 ; 460 : else {
597 ; 461 : hanoi(n-1, aux, to, from);
598 ; 462 : aux = aux + 1;
599 ; 463 : IO_stdin.uscd+0x4> #
600
601 ; 464 : else {
602 ; 465 : hanoi(n-1, aux, to, from);
603 ; 466 : aux = aux + 1;
604 ; 467 : IO_stdin.uscd+0x4> #
605
606 ; 468 : else {
607 ; 469 : hanoi(n-1, aux, to, from);
608 ; 470 : aux = aux + 1;
609 ; 471 : IO_stdin.uscd+0x4> #
610
611 ; 472 : else {
612 ; 473 : hanoi(n-1, aux, to, from);
613 ; 474 : aux = aux + 1;
614 ; 475 : IO_stdin.uscd+0x4> #
615
616 ; 476 : else {
617 ; 477 : hanoi(n-1, aux, to, from);
618 ; 478 : aux = aux + 1;
619 ; 479 : IO_stdin.uscd+0x4> #
620
621 ; 480 : else {
622 ; 481 : hanoi(n-1, aux, to, from);
623 ; 482 : aux = aux + 1;
624 ; 483 : IO_stdin.uscd+0x4> #
625
626 ; 484 : else {
627 ; 485 : hanoi(n-1, aux, to, from);
628 ; 486 : aux = aux + 1;
629 ; 487 : IO_stdin.uscd+0x4> #
630
631 ; 488 : else {
632 ; 489 : hanoi(n-1, aux, to, from);
633 ; 490 : aux = aux + 1;
634 ; 491 : IO_stdin.uscd+0x4> #
635
636 ; 492 : else {
637 ; 493 : hanoi(n-1, aux, to, from);
638 ; 494 : aux = aux + 1;
639 ; 495 : IO_stdin.uscd+0x4> #
640
641 ; 496 : else {
642 ; 497 : hanoi(n-1, aux, to, from);
643 ; 498 : aux = aux + 1;
644 ; 499 : IO_stdin.uscd+0x4> #
645
646 ; 500 : else {
647 ; 501 : hanoi(n-1, aux, to, from);
648 ; 502 : aux = aux + 1;
649 ; 503 : IO_stdin.uscd+0x4> #
650
651 ; 504 : else {
652 ; 505 : hanoi(n-1, aux, to, from);
653 ; 506 : aux = aux + 1;
654 ; 507 : IO_stdin.uscd+0x4> #
655
656 ; 508 : else {
657 ; 509 : hanoi(n-1, aux, to, from);
658 ; 510 : aux = aux + 1;
659 ; 511 : IO_stdin.uscd+0x4> #
660
661 ; 512 : else {
662 ; 513 : hanoi(n-1, aux, to, from);
663 ; 514 : aux = aux + 1;
664 ; 515 : IO_stdin.uscd+0x4> #
665
666 ; 516 : else {
667 ; 517 : hanoi(n-1, aux, to, from);
668 ; 518 : aux = aux + 1;
669 ; 519 : IO_stdin.uscd+0x4> #
670
671 ; 520 : else {
672 ; 521 : hanoi(n-1, aux, to, from);
673 ; 522 : aux = aux + 1;
674 ; 523 : IO_stdin.uscd+0x4> #
675
676 ; 524 : else {
677 ; 525 : hanoi(n-1, aux, to, from);
678 ; 526 : aux = aux + 1;
679 ; 527 : IO_stdin.uscd+0x4> #
680
681 ; 528 : else {
682 ; 529 : hanoi(n-1, aux, to, from);
683 ; 530 : aux = aux + 1;
684 ; 531 : IO_stdin.uscd+0x4> #
685
686 ; 532 : else {
687 ; 533 : hanoi(n-1, aux, to, from);
688 ; 534 : aux = aux + 1;
689 ; 535 : IO_stdin.uscd+0x4> #
690
691 ; 536 : else {
692 ; 537 : hanoi(n-1, aux, to, from);
693 ; 538 : aux = aux + 1;
694 ; 539 : IO_stdin.uscd+0x4> #
695
696 ; 540 : else {
697 ; 541 : hanoi(n-1, aux, to, from);
698 ; 542 : aux = aux + 1;
699 ; 543 : IO_stdin.uscd+0x4> #
700
701 ; 544 : else {
702 ; 545 : hanoi(n-1, aux, to, from);
703 ; 546 : aux = aux + 1;
704 ; 547 : IO_stdin.uscd+0x4> #
705
706 ; 548 : else {
707 ; 549 : hanoi(n-1, aux, to, from);
708 ; 550 : aux = aux + 1;
709 ; 551 : IO_stdin.uscd+0x4> #
710
711 ; 552 : else {
712 ; 553 : hanoi(n-1, aux, to, from);
713 ; 554 : aux = aux + 1;
714 ; 555 : IO_stdin.uscd+0x4> #
715
716 ; 556 : else {
717 ; 557 : hanoi(n-1, aux, to, from);
718 ; 558 : aux = aux + 1;
719 ; 559 : IO_stdin.uscd+0x4> #
720
721 ; 560 : else {
722 ; 561 : hanoi(n-1, aux, to, from);
723 ; 562 : aux = aux + 1;
724 ; 563 : IO_stdin.uscd+0x4> #
725
726 ; 564 : else {
727 ; 565 : hanoi(n-1, aux, to, from);
728 ; 566 : aux = aux + 1;
729 ; 567 : IO_stdin.uscd+0x4> #
730
731 ; 568 : else {
732 ; 569 : hanoi(n-1, aux, to, from);
733 ; 570 : aux = aux + 1;
734 ; 571 : IO_stdin.uscd+0x4> #
735
736 ; 572 : else {
737 ; 573 : hanoi(n-1, aux, to, from);
738 ; 574 : aux = aux + 1;
739 ; 575 : IO_stdin.uscd+0x4> #
740
741 ; 576 : else {
742 ; 577 : hanoi(n-1, aux, to, from);
743 ; 578 : aux = aux + 1;
744 ; 579 : IO_stdin.uscd+0x4> #
745
746 ; 580 : else {
747 ; 581 : hanoi(n-1, aux, to, from);
748 ; 582 : aux = aux + 1;
749 ; 583 : IO_stdin.uscd+0x4> #
750
751 ; 584 : else {
752 ; 585 : hanoi(n-1, aux, to, from);
753 ; 586 :
```

Завдання 2:

Реалізувати мовою C/C++, проаналізувати результати компіляції (за варіантом), для платформ i686, amd64, arm, aarch64:

- Комбінаторні алгоритми [25], будь-який на Ваш вибір з вказаного класу:
 - 6. на графах – розфарбування графів;
 - Криптографічні алгоритми, алгоритми кодування та контролю цілісності [26, 27]:
 - 6. ARCfour / RC4;

10) Реалізуємо розфарбування графу жадібним алгоритмом

```

(kali㉿kali)-[~/lab1-reverse]
└─$ cat graph-coloring.cpp
// A C++ program to implement greedy algorithm for graph coloring
#include <iostream>
#include <list>
using namespace std;

// A class that represents an undirected graph
class Graph
{
    int V; // No. of vertices
    list<int> *adj; // A dynamic array of adjacency lists
public:
    // Constructor and destructor
    Graph(int V) { this->V = V; adj = new list<int>[V]; }
    ~Graph() { delete [] adj; }

    // function to add an edge to graph
    void addEdge(int v, int w);

    // Prints greedy coloring of the vertices
    void greedyColoring();
};

void Graph::addEdge(int v, int w)
{
    adj[v].push_back(w);
    adj[w].push_back(v); // Note: the graph is undirected
}

// Assigns colors (starting from 0) to all vertices and prints
// the assignment of colors
void Graph::greedyColoring()
{
    int result[V];

    // Assign the first color to first vertex
    result[0] = 0;

    // Initialize remaining V-1 vertices as unassigned
    for (int u = 1; u < V; u++)
        result[u] = -1; // no color is assigned to u

    // A temporary array to store the available colors. True
    // value of available[cr] would mean that the color cr is
    // assigned to one of its adjacent vertices
    bool available[V];
    for (int cr = 0; cr < V; cr++)
        available[cr] = false;

    // Assign colors to remaining V-1 vertices
    // Assign colors to remaining V-1 vertices
    for (int u = 1; u < V; u++)
    {
        // Process all adjacent vertices and flag their colors
        // as unavailable
        list<int>::iterator i;
        for (i = adj[u].begin(); i != adj[u].end(); ++i)
            if (result[*i] != -1)
                available[result[*i]] = true;

        // Find the first available color
        int cr;
        for (cr = 0; cr < V; cr++)
            if (available[cr] == false)
                break;

        result[u] = cr; // Assign the found color

        // Reset the values back to false for the next iteration
        for (i = adj[u].begin(); i != adj[u].end(); ++i)
            if (result[*i] != -1)
                available[result[*i]] = false;
    }

    // print the result
    for (int u = 0; u < V; u++)
        cout << "Vertex " << u << " --- Color "
            << result[u] << endl;
}

// Driver program to test above function
int main()
{
    Graph g1(5);
    g1.addEdge(0, 1);
    g1.addEdge(0, 2);
    g1.addEdge(1, 2);
    g1.addEdge(1, 3);
    g1.addEdge(2, 3);
    g1.addEdge(3, 4);
    cout << "Coloring of graph 1 \n";
    g1.greedyColoring();

    Graph g2(5);
    g2.addEdge(0, 1);
    g2.addEdge(0, 2);
    g2.addEdge(1, 2);
    g2.addEdge(1, 4);
    g2.addEdge(2, 4);
    g2.addEdge(4, 3);
    cout << "\nColoring of graph 2 \n";
    g2.greedyColoring();
}

```

```
(kali㉿kali)-[~/lab1-reverse]
└─$ g++ -g graph-coloring.cpp -o graph-coloring.amd64

(kali㉿kali)-[~/lab1-reverse]
└─$ ./graph-coloring.amd64
Coloring of graph 1
Vertex 0 → Color 0
Vertex 1 → Color 1
Vertex 2 → Color 2
Vertex 3 → Color 0
Vertex 4 → Color 1

Coloring of graph 2
Vertex 0 → Color 0
Vertex 1 → Color 1
Vertex 2 → Color 2
Vertex 3 → Color 0
Vertex 4 → Color 3
```

11) Компілюємо для Linux i686 (gcc)

```
(kali㉿kali)-[~/lab1-reverse]
└─$ g++ -m32 -g graph-coloring.cpp -o graph-coloring.i686 -march=i686

(kali㉿kali)-[~/lab1-reverse]
└─$ i686-linux-gnu-objdump -drwC -Mintel -S graph-coloring.i686 > graph-coloring_gcc_i686.lst
```

12) Компілюємо для Linux amd64 (gcc)

```
(kali㉿kali)-[~/lab1-reverse]
└─$ g++ -g graph-coloring.cpp -o graph-coloring.amd64

(kali㉿kali)-[~/lab1-reverse]
└─$ objdump -drwC -Mintel -S graph-coloring.amd64 > graph-coloring_gcc_amd64.lst
```

13) Компілюємо для Linux arm (gcc)

```
(kali㉿kali)-[~/lab1-reverse]
└─$ arm-linux-gnueabi-g++ -g graph-coloring.cpp -o graph-coloring.arm

(kali㉿kali)-[~/lab1-reverse]
└─$ arm-linux-gnueabi-objdump -drwC -S graph-coloring.arm > graph-coloring_gcc_arm.lst
```

14) Компілюємо для Linux aarch64 (gcc)

```
(kali㉿kali)-[~/lab1-reverse]
└─$ aarch64-linux-gnu-g++ -g graph-coloring.cpp -o graph-coloring.aarch64

(kali㉿kali)-[~/lab1-reverse]
└─$ aarch64-linux-gnu-objdump -drwC -S graph-coloring.aarch64 > graph-coloring_gcc_aarch64.lst
```

15) Всі файли кладемо в архів і порівнюємо

```
GROUP 1          graph-coloring_gcc_i686.lst    graph-coloring_gcc_amd64.lst    graph-coloring_gcc_arm.lst    graph-coloring_gcc_aarch64.lst
graph-coloring_gcc_i686.lst
1 00000000000000000000000000000000 <
Graph::greedyColoring():>
2 // Assigns colors (starting
3 // from 0) to all vertices and
4 // prints
5 void Graph::greedyColoring()
6 {
7     12ac:                                1258:                                10000000000000000000000000000000 <
8         push    ebp                                push    rbp                                Graph::greedyColoring():>
9         mov     ebp,esp                            mov     rbp,rsp                            2 // Assigns colors (starting
10        push   esi                                push   r15                                3 // from 0) to all vertices and
11        push   ebx                                push   r14                                4 // prints
12        mov     esp,0x50                            push   r13                                5 void Graph::greedyColoring()
13        call   _i150                                push   r12                                6 {
14        12b6:                                1260:                                00000000000000000000000000000000 <
15        12b7:                                1261:                                ab0: e92d4ff0 push
16        12b8:                                1262:                                {r4, r5, r6, r7, r8,
17        12b9:                                1263:                                r9, s1, fp, lr}
18        12ba:                                1264:                                ab4: e28db020 add
19        12bb:                                1265:                                fp, sp, #32
20        12bc:                                1266:                                ab8: e24dd044 sub
21        12bd:                                1267:                                sp, sp, #68 ; 0x44
22        12be:                                1268:                                abc: e500b058 str
23        12bf:                                1269:                                r0, [fp, #-88] ;
24        12c0:                                1270:                                0xfffffff8
25        12c1:                                1271:                                ac0: [pc, #1220] ldr
26        12c2:                                1272:                                s1, [pc, #1220] ;
27        12c3:                                1273:                                f8c <graph::greedyColor-
28        12c4:                                1274:                                ing>(.+)
29        12c5:                                1275:                                11 // print the result
30        12c6:                                1276:                                for (int u = 0; u < V;
31        12c7:                                1277:                                u++)
32        12c8:                                1278:                                cout << "Vertex "
33        12c9:                                1279:                                << u << " ---"
34        12ca:                                1280:                                << result[u] <<
35        12cb:                                1281:                                endl;
36        12cc:                                1282:                                // print the result
37        12cd:                                1283:                                for (int u = 0; u < V;
38        12ce:                                1284:                                u++)
39        12cf:                                1285:                                cout << "Color "
40        12d0:                                1286:                                << color[u] <<
41        12d1:                                1287:                                endl;
42        12d2:                                1288:                                // print the result
43        12d3:                                1289:                                for (int u = 0; u < V;
44        12d4:                                1290:                                u++)
45        12d5:                                1291:                                cout << "Color "
46        12d6:                                1292:                                << color[u] <<
47        12d7:                                1293:                                endl;
48        12d8:                                1294:                                // print the result
49        12d9:                                1295:                                for (int u = 0; u < V;
50        12da:                                1296:                                u++)
51        12db:                                1297:                                cout << "Color "
52        12dc:                                1298:                                << color[u] <<
53        12dd:                                1299:                                endl;
54        12de:                                1300:                                // print the result
55        12df:                                1301:                                for (int u = 0; u < V;
56        12e0:                                1302:                                u++)
57        12e1:                                1303:                                cout << "Color "
58        12e2:                                1304:                                << color[u] <<
59        12e3:                                1305:                                endl;
60        12e4:                                1306:                                // print the result
61        12e5:                                1307:                                for (int u = 0; u < V;
62        12e6:                                1308:                                u++)
63        12e7:                                1309:                                cout << "Color "
64        12e8:                                1310:                                << color[u] <<
65        12e9:                                1311:                                endl;
66        12ea:                                1312:                                // print the result
67        12eb:                                1313:                                for (int u = 0; u < V;
68        12ec:                                1314:                                u++)
69        12ed:                                1315:                                cout << "Color "
70        12ee:                                1316:                                << color[u] <<
71        12ef:                                1317:                                endl;
72        12f0:                                1318:                                // print the result
73        12f1:                                1319:                                for (int u = 0; u < V;
74        12f2:                                1320:                                u++)
75        12f3:                                1321:                                cout << "Color "
76        12f4:                                1322:                                << color[u] <<
77        12f5:                                1323:                                endl;
78        12f6:                                1324:                                // print the result
79        12f7:                                1325:                                for (int u = 0; u < V;
80        12f8:                                1326:                                u++)
81        12f9:                                1327:                                cout << "Color "
82        12fa:                                1328:                                << color[u] <<
83        12fb:                                1329:                                endl;
84        12fc:                                1330:                                // print the result
85        12fd:                                1331:                                for (int u = 0; u < V;
86        12fe:                                1332:                                u++)
87        12ff:                                1333:                                cout << "Color "
88        12f0:                                1334:                                << color[u] <<
89        12f1:                                1335:                                endl;
90        12f2:                                1336:                                // print the result
91        12f3:                                1337:                                for (int u = 0; u < V;
92        12f4:                                1338:                                u++)
93        12f5:                                1339:                                cout << "Color "
94        12f6:                                1340:                                << color[u] <<
95        12f7:                                1341:                                endl;
96        12f8:                                1342:                                // print the result
97        12f9:                                1343:                                for (int u = 0; u < V;
98        12fa:                                1344:                                u++)
99        12fb:                                1345:                                cout << "Color "
100       12fc:                                1346:                                << color[u] <<
101       12fd:                                1347:                                endl;
102       12fe:                                1348:                                // print the result
103       12ff:                                1349:                                for (int u = 0; u < V;
104       12f0:                                1350:                                u++)
105       12f1:                                1351:                                cout << "Color "
106       12f2:                                1352:                                << color[u] <<
107       12f3:                                1353:                                endl;
108       12f4:                                1354:                                // print the result
109       12f5:                                1355:                                for (int u = 0; u < V;
110       12f6:                                1356:                                u++)
111       12f7:                                1357:                                cout << "Color "
112       12f8:                                1358:                                << color[u] <<
113       12f9:                                1359:                                endl;
114       12fa:                                1360:                                // print the result
115       12fb:                                1361:                                for (int u = 0; u < V;
116       12fc:                                1362:                                u++)
117       12fd:                                1363:                                cout << "Color "
118       12fe:                                1364:                                << color[u] <<
119       12ff:                                1365:                                endl;
120       12f0:                                1366:                                // print the result
121       12f1:                                1367:                                for (int u = 0; u < V;
122       12f2:                                1368:                                u++)
123       12f3:                                1369:                                cout << "Color "
124       12f4:                                1370:                                << color[u] <<
125       12f5:                                1371:                                endl;
126       12f6:                                1372:                                // print the result
127       12f7:                                1373:                                for (int u = 0; u < V;
128       12f8:                                1374:                                u++)
129       12f9:                                1375:                                cout << "Color "
130       12fa:                                1376:                                << color[u] <<
131       12fb:                                1377:                                endl;
132       12fc:                                1378:                                // print the result
133       12fd:                                1379:                                for (int u = 0; u < V;
134       12fe:                                1380:                                u++)
135       12ff:                                1381:                                cout << "Color "
136       12f0:                                1382:                                << color[u] <<
137       12f1:                                1383:                                endl;
138       12f2:                                1384:                                // print the result
139       12f3:                                1385:                                for (int u = 0; u < V;
140       12f4:                                1386:                                u++)
141       12f5:                                1387:                                cout << "Color "
142       12f6:                                1388:                                << color[u] <<
143       12f7:                                1389:                                endl;
144       12f8:                                1390:                                // print the result
145       12f9:                                1391:                                for (int u = 0; u < V;
146       12fa:                                1392:                                u++)
147       12fb:                                1393:                                cout << "Color "
148       12fc:                                1394:                                << color[u] <<
149       12fd:                                1395:                                endl;
150       12fe:                                1396:                                // print the result
151       12ff:                                1397:                                for (int u = 0; u < V;
152       12f0:                                1398:                                u++)
153       12f1:                                1399:                                cout << "Color "
154       12f2:                                1400:                                << color[u] <<
155       12f3:                                1401:                                endl;
156       12f4:                                1402:                                // print the result
157       12f5:                                1403:                                for (int u = 0; u < V;
158       12f6:                                1404:                                u++)
159       12f7:                                1405:                                cout << "Color "
160       12f8:                                1406:                                << color[u] <<
161       12f9:                                1407:                                endl;
162       12fa:                                1408:                                // print the result
163       12fb:                                1409:                                for (int u = 0; u < V;
164       12fc:                                1410:                                u++)
165       12fd:                                1411:                                cout << "Color "
166       12fe:                                1412:                                << color[u] <<
167       12ff:                                1413:                                endl;
168       12f0:                                1414:                                // print the result
169       12f1:                                1415:                                for (int u = 0; u < V;
170       12f2:                                1416:                                u++)
171       12f3:                                1417:                                cout << "Color "
172       12f4:                                1418:                                << color[u] <<
173       12f5:                                1419:                                endl;
174       12f6:                                1420:                                // print the result
175       12f7:                                1421:                                for (int u = 0; u < V;
176       12f8:                                1422:                                u++)
177       12f9:                                1423:                                cout << "Color "
178       12fa:                                1424:                                << color[u] <<
179       12fb:                                1425:                                endl;
180       12fc:                                1426:                                // print the result
181       12fd:                                1427:                                for (int u = 0; u < V;
182       12fe:                                1428:                                u++)
183       12ff:                                1429:                                cout << "Color "
184       12f0:                                1430:                                << color[u] <<
185       12f1:                                1431:                                endl;
186       12f2:                                1432:                                // print the result
187       12f3:                                1433:                                for (int u = 0; u < V;
188       12f4:                                1434:                                u++)
189       12f5:                                1435:                                cout << "Color "
190       12f6:                                1436:                                << color[u] <<
191       12f7:                                1437:                                endl;
192       12f8:                                1438:                                // print the result
193       12f9:                                1439:                                for (int u = 0; u < V;
194       12fa:                                1440:                                u++)
195       12fb:                                1441:                                cout << "Color "
196       12fc:                                1442:                                << color[u] <<
197       12fd:                                1443:                                endl;
198       12fe:                                1444:                                // print the result
199       12ff:                                1445:                                for (int u = 0; u < V;
200       12f0:                                1446:                                u++)
201       12f1:                                1447:                                cout << "Color "
202       12f2:                                1448:                                << color[u] <<
203       12f3:                                1449:                                endl;
204       12f4:                                1450:                                // print the result
205       12f5:                                1451:                                for (int u = 0; u < V;
206       12f6:                                1452:                                u++)
207       12f7:                                1453:                                cout << "Color "
208       12f8:                                1454:                                << color[u] <<
209       12f9:                                1455:                                endl;
210       12fa:                                1456:                                // print the result
211       12fb:                                1457:                                for (int u = 0; u < V;
212       12fc:                                1458:                                u++)
213       12fd:                                1459:                                cout << "Color "
214       12fe:                                1460:                                << color[u] <<
215       12ff:                                1461:                                endl;
216       12f0:                                1462:                                // print the result
217       12f1:                                1463:                                for (int u = 0; u < V;
218       12f2:                                1464:                                u++)
219       12f3:                                1465:                                cout << "Color "
220       12f4:                                1466:                                << color[u] <<
221       12f5:                                1467:                                endl;
222       12f6:                                1468:                                // print the result
223       12f7:                                1469:                                for (int u = 0; u < V;
224       12f8:                                1470:                                u++)
225       12f9:                                1471:                                cout << "Color "
226       12fa:                                1472:                                << color[u] <<
227       12fb:                                1473:                                endl;
228       12fc:                                1474:                                // print the result
229       12fd:                                1475:                                for (int u = 0; u < V;
230       12fe:                                1476:                                u++)
231       12ff:                                1477:                                cout << "Color "
232       12f0:                                1478:                                << color[u] <<
233       12f1:                                1479:                                endl;
234       12f2:                                1480:                                // print the result
235       12f3:                                1481:                                for (int u = 0; u < V;
236       12f4:                                1482:                                u++)
237       12f5:                                1483:                                cout << "Color "
238       12f6:                                1484:                                << color[u] <<
239       12f7:                                1485:                                endl;
240       12f8:                                1486:                                // print the result
241       12f9:                                1487:                                for (int u = 0; u < V;
242       12fa:                                1488:                                u++)
243       12fb:                                1489:                                cout << "Color "
244       12fc:                                1490:                                << color[u] <<
245       12fd:                                1491:                                endl;
246       12fe:                                1492:                                // print the result
247       12ff:                                1493:                                for (int u = 0; u < V;
248       12f0:                                1494:                                u++)
249       12f1:                                1495:                                cout << "Color "
250       12f2:                                1496:                                << color[u] <<
251       12f3:                                1497:                                endl;
252       12f4:                                1498:                                // print the result
253       12f5:                                1499:                                for (int u = 0; u < V;
254       12f6:                                1500:                                u++)
255       12f7:                                1501:                                cout << "Color "
256       12f8:                                1502:                                << color[u] <<
257       12f9:                                1503:                                endl;
258       12fa:                                1504:                                // print the result
259       12fb:                                1505:                                for (int u = 0; u < V;
260       12fc:                                1506:                                u++)
261       12fd:                                1507:                                cout << "Color "
262       12fe:                                1508:                                << color[u] <<
263       12ff:                                1509:                                endl;
264       12f0:                                1510:                                // print the result
265       12f1:                                1511:                                for (int u = 0; u < V;
266       12f2:                                1512:                                u++)
267       12f3:                                1513:                                cout << "Color "
268       12f4:                                1514:                                << color[u] <<
269       12f5:                                1515:                                endl;
270       12f6:                                1516:                                // print the result
271       12f7:                                1517:                                for (int u = 0; u < V;
272       12f8:                                1518:                                u++)
273       12f9:                                1519:                                cout << "Color "
274       12fa:                                1520:                                << color[u] <<
275       12fb:                                1521:                                endl;
276       12fc:                                1522:                                // print the result
277       12fd:                                1523:                                for (int u = 0; u < V;
278       12fe:                                1524:                                u++)
279       12ff:                                1525:                                cout << "Color "
280       12f0:                                1526:                                << color[u] <<
281       12f1:                                1527:                                endl;
282       12f2:                                1528:                                // print the result
283       12f3:                                1529:                                for (int u = 0; u < V;
284       12f4:                                1530:                                u++)
285       12f5:                                1531:                                cout << "Color "
286       12f6:                                1532:                                << color[u] <<
287       12f7:                                1533:                                endl;
288       12f8:                                1534:                                // print the result
289       12f9:                                1535:                                for (int u = 0; u < V;
290       12fa:                                1536:                                u++)
291       12fb:                                1537:                                cout << "Color "
292       12fc:                                1538:                                << color[u] <<
293       12fd:                                1539:                                endl;
294       12fe:                                1540:                                // print the result
295       12ff:                                1541:                                for (int u = 0; u < V;
296       12f0:                                1542:                                u++)
297       12f1:                                1543:                                cout << "Color "
298       12f2:                                1544:                                << color[u] <<
299       12f3:                                1545:                                endl;
300       12f4:                                1546:                                // print the result
301       12f5:                                1547:                                for (int u = 0; u < V;
302       12f6:                                1548:                                u++)
303       12f7:                                1549:                                cout << "Color "
304       12f8:                                1550:                                << color[u] <<
305       12f9:                                1551:                                endl;
306       12fa:                                1552:                                // print the result
307       12fb:                                1553:                                for (int u = 0; u < V;
308       12fc:                                1554:                                u++)
309       12fd:                                1555:                                cout << "Color "
310       12fe:                                1556:                                << color[u] <<
311       12ff:                                1557:                                endl;
312       12f0:                                1558:                                // print the result
313       12f1:                                1559:                                for (int u = 0; u < V;
314       12f2:                                1560:                                u++)
315       12f3:                                1561:                                cout << "Color "
316       12f4:                                1562:                                << color[u] <<
317       12f5:                                1563:                                endl;
318       12f6:                                1564:                                // print the result
319       12f7:                                1565:                                for (int u = 0; u < V;
320       12f8:                                1566:                                u++)
321       12f9:                                1567:                                cout << "Color "
322       12fa:                                1568:                                << color[u] <<
323       12fb:                                1569:                                endl;
324       12fc:                                1570:                                // print the result
325       12fd:                                1571:                                for (int u = 0; u < V;
326       12fe:                                1572:                                u++)
327       12ff:                                1573:                                cout << "Color "
328       12f0:                                1574:                                << color[u] <<
329       12f1:                                1575:                                endl;
330       12f2:                                1576:                                // print the result
331       12f3:                                1577:                                for (int u = 0; u < V;
332       12f4:                                1578:                                u++)
333       12f5:                                1579:                                cout << "Color "
334       12f6:                                1580:                                << color[u] <<
335       12f7:                                1581:                                endl;
336       12f8:                                1582:                                // print the result
337       12f9:                                1583:                                for (int u = 0; u < V;
338       12fa:                                1584:                                u++)
339       12fb:                                1585:                                cout << "Color "
340       12fc:                                1586:                                << color[u] <<
341       12fd:                                1587:                                endl;
342       12fe:                                1588:                                // print the result
343       12ff:                                1589:                                for (int u = 0; u < V;
344       12f0:                                1590:                                u++)
345       12f1:                                1591:                                cout << "Color "
346       12f2:                                1592:                                << color[u] <<
347       12f3:                                1593:                                endl;
348       12f4:                                1594:                                // print the result
349       12f5:                                1595:                                for (int u = 0; u < V;
350       12f6:                                1596:                                u++)
351       12f7:                                1597:                                cout << "Color "
352       12f8:                                1598:                                << color[u] <<
353       12f9:                                1599:                                endl;
354       12fa:                                1600:                                // print the result
355       12fb:                                1601:                                for (int u = 0; u < V;
356       12fc:                                1602:                                u++)
357       12fd:                                1603:                                cout << "Color "
358       12fe:                                1604:                                << color[u] <<
359       12ff:                                1605:                                endl;
360       12f0:                                1606:                                // print the result
361       12f1:                                1607:                                for (int u = 0; u < V;
362       12f2:                                1608:                                u++)
363       12f3:                                1609:                                cout << "Color "
364       12f4:                                1610:                                << color[u] <<
365       12f5:                                1611:                                endl;
366       12f6:                                1612:                                // print the result
367       12f7:                                1613:                                for (int u = 0; u < V;
368       12f8:                                1614:                                u++)
369       12f9:                                1615:                                cout << "Color "
370       12fa:                                1616:                                << color[u] <<
371       12fb:                                1617:                                endl;
372       12fc:                                1618:                                // print the result
373       12fd:                                1619:                                for (int u = 0; u < V;
374       12fe:                                1620:                                u++)
375       12ff:                                1621:                                cout << "Color "
376       12f0:                                1622:                                << color[u] <<
377       12f1:                                1623:                                endl;
378       12f2:                                1624:                                // print the result
379       12f3:                                1625:                                for (int u = 0; u < V;
380       12f4:                                1626:                                u++)
381       12f5:                                1627:                                cout << "Color "
382       12f6:                                1628:                                << color[u] <<
383       12f7:                                1629:                                endl;
384       12f8:                                1630:                                // print the result
385       12f9:                                1631:                                for (int u = 0; u < V;
386       12fa:                                1632:                                u++)
387       12fb:                                1633:                                cout << "Color "
388       12fc:                                1634:                                << color[u] <<
389       12fd:                                1635:                                endl;
390       12fe:                                1636:                                // print the result
391       12ff:                                1637:                                for (int u = 0; u < V;
392       12f0:                                1638:                                u++)
393       12f1:                                1639:                                cout << "Color "
394       12f2:                                1640:                                << color[u] <<
395       12f3:                                1641:                                endl;
396       12f4:                                1642:                                // print the result
397       12f5:                                1643:                                for (int u = 0; u < V;
398       12f6:                                1644:                                u++)
399       12f7:                                1645:                                cout << "Color "
400       12f8:                                1646:                                << color[u] <<
401       12f9:                                1647:                                endl;
402       12fa:                                1648:                                // print the result
403       12fb:                                1649:                                for (int u = 0; u < V;
404       12fc:                                1650:                                u++)
405       12fd:                                1651:                                cout << "Color "
406       12fe:                                1652:                                << color[u] <<
407       12ff:                                1653:                                endl;
408       12f0:                                1654:                                // print the result
409       12f1:                                1655:                                for (int u = 0; u < V;
410       12f2:                                1656:                                u++)
411       12f3:                                1657:                                cout << "Color "
412       12f4:                                1658:                                << color[u] <<
413       12f5:                                1659:                                endl;
414       12f6:                                1660:                                // print the result
415       12f7:                                1661:                                for (int u = 0; u < V;
416       12f8:                                1662:                                u++)
417       12f9:                                1663:                                cout << "Color "
418       12fa:                                1664:                                << color[u] <<
419       12fb:                                1665:                                endl;
420       12fc:                                1666:                                // print the result
421       12fd:                                1667:                                for (int u = 0; u < V;
422       12fe:                                1668:                                u++)
423       12ff:                                1669:                                cout << "Color "
424       12f0:                                1670:                                << color[u] <<
425       12f1:                                1671:                                endl;
426       12f2:                                1672:                                // print the result
427       12f3:                                1673:                                for (int u = 0; u < V;
428       12f4:                                1674:                                u++)
429       12f5:                                1675:                                cout << "Color "
430       12f6:                                1676:                                << color[u] <<
431       12f7:                                1677:                                endl;
432       12f8:                                1678:                                // print the result
433       12f9:                                1679:                                for (int u = 0; u < V;
434       12fa:                                1680:                                u++)
435       12fb:                                1681:                                cout << "Color "
436       12fc:                                1682:                                << color[u] <<
437       12fd:                                1683:                                endl;
438       12fe:                                1684:                                // print the result
439       12ff:                                1685:                                for (int u = 0; u < V;
440       12f0:                                1686:                                u++)
441       12f1:                                1687:                                cout << "Color "
442       12f2:                                1688:                                << color[u] <<
443       12f3:                                1689:                                endl;
444       12f4:                                1690:                                // print the result
445       12f5:                                1691:                                for (int u = 0; u < V;
446       12f6:                                1692:                                u++)
447       12f7:                                1693:                                cout << "Color "
448       12f8:                                1694:                                << color[u] <<
449       12f9:                                1695:                                endl;
450       12fa:                                1696:                                // print the result
451       12fb:                                1697:                                for (int u = 0; u < V;
452       12fc:                                1698:                                u++)
453       12fd:                                1699:                                cout << "Color "
454       12fe:                                1700:                                << color[u] <<
455       12ff:                                1701:                                endl;
456       12f0:                                1702:                                // print the result
457       12f1:                                1703:                                for (int u = 0; u < V;
458       12f2:                                1704:                                u++)
459       12f3:                                1705:                                cout << "Color "
460       12f4:                                1706:                                << color[u] <<
461       12f5:                                1707:                                endl;
462       12f6:                                1708:                                // print the result
463       12f7:                                1709:                                for (int u = 0; u < V;
464       12f8:                                1710:                                u++)
465       12f9:                                1711:                                cout << "Color "
466       12fa:                                1712:                                << color[u] <<
467       12fb:                                1713:                                endl;
468       12fc:                                1714:                                // print the result
469       12fd:                                1715:                                for (int u = 0; u < V;
470       12fe:                                1716:                                u++)
471       12ff:                                1717:                                cout << "Color "
472       12f0:                                1718:                                << color[u] <<
473       12f1:                                1719:                                endl;
474       12f2:                                1720:                                // print the result
475       12f3:                                1721:                                for (int u = 0; u < V;
476       12f4:                                1722:                                u++)
477       12f5:                                1723:                                cout << "Color "
478       12f6:                                1724:                                << color[u] <<
479       12f7:                                1725:                                endl;
480       12f8:                                1726:                                // print the result
481       12f9:                                1727:                                for (int u = 0; u < V;
482       12fa:                                1728:                                u++)
483       12fb:                                1729:                                cout << "Color "
484       12fc:                                1730:                                << color[u] <<
485       12fd:                                1731:                                endl;
486       12fe:                                1732:                                // print the result
487       12ff:                                1733:                                for (int u = 0; u < V;
488       12f0:                                1734:                                u++)
489       12f1:                                1735:                                cout << "Color "
490       12f2:                                1736:                                << color[u] <<
491       12f3:                                1737:                                endl;
492       12f4:                                173
```

16) Тепер так само з RC4. Спочатку реалізуємо алгоритм

```
$ cat rc4.cpp
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <stdio.h>
#include <string>

class State
{
    FILE *System;
    unsigned char s[256];
    int i, j;

    void swap(int a, int b);

    public:
        unsigned char getbyte(void);
        State(unsigned char key[], int length );
};

State::State(unsigned char key[], int length)
{
    for(int k=0; k<256; k++)
    {
        s[k]=k;
    }

    j=0;

    for(i=0; i<256 ; i++)
    {
        j=(j + s[i] + key[i % length]) % 256;
        swap(i, j);
    }

    i=j=0;
}

void State::swap(int a, int b)
{
    unsigned char temp= s[i];
    s[i]=s[j];
    s[j]=temp;
}

unsigned char State::getbyte(void)
{
    i=(i+1)%256;
    j=(j+s[i])%256;
    swap(i, j);
    int index=(s[i]+s[j])%256;
    return s[index];
}
```

17) Компілюємо для Linux i686 (gcc)

```
[kali㉿kali)-[~/lab1-reverse]
└─$ g++ -m32 -g rc4.cpp -o rc4.i686 -march=i686

[kali㉿kali)-[~/lab1-reverse]
└─$ i686-linux-gnu-objdump -drwC -Mintel -S rc4.i686 > rc4_gcc_i686.lst
```

18) Компілюємо для Linux amd64 (gcc)

```

└─(kali㉿kali)-[~/lab1-reverse]
$ g++ -g rc4.cpp -o rc4.amd64

└─(kali㉿kali)-[~/lab1-reverse]
$ objdump -drwC -Mintel -S rc4.amd64 > rc4_gcc_amd64.lst

```

19) Компілюємо для Linux arm (gcc)

```

└─(kali㉿kali)-[~/lab1-reverse]
$ arm-linux-gnueabi-g++ -g rc4.cpp -o rc4.arm

└─(kali㉿kali)-[~/lab1-reverse]
$ arm-linux-gnueabi-objdump -drwC -S rc4.arm > rc4_gcc_arm.lst

```

20) Компілюємо для Linux aarch64 (gcc)

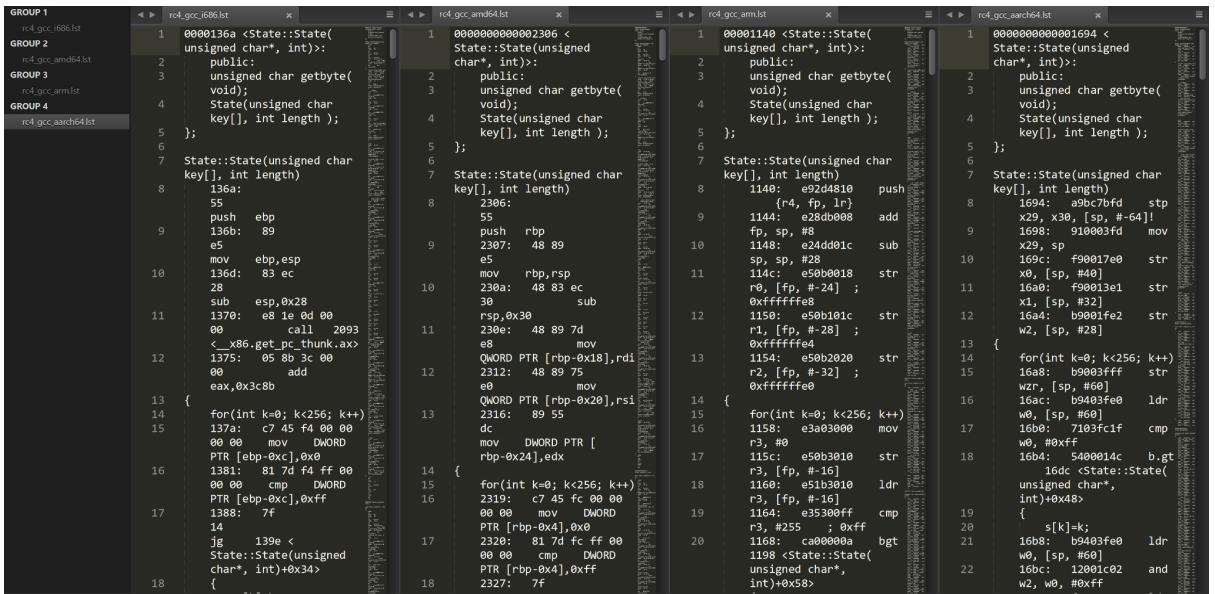
```

└─(kali㉿kali)-[~/lab1-reverse]
$ aarch64-linux-gnu-g++ -g rc4.cpp -o rc4.aarch64

└─(kali㉿kali)-[~/lab1-reverse]
$ aarch64-linux-gnu-objdump -drwC -S rc4.aarch64 > rc4_gcc_aarch64.lst

```

21) Всі файли кладемо в архів і порівнюємо



Завдання 3:

Реалізація функцій стандартної бібліотеки С. Бібліотека за варіантами, функції всі зазначені (за наявності реалізації), версія бібліотеки остання стабільна на момент початку курсу:

6. klibc [33];

Функції:

- стандартного ввод-виводу printf, puts;
- роботи з файлами fopen, fread, fwrite, feof, fclose;
- виконання команд операційної системи system.

Зверніть увагу на відмінності системних викликів у різних ОС Linux та Windows для різних архітектур (x86, x86_64, ARM) [35].

22) Візьмемо реалізацію функції printf у бібліотеці klibc

```
[kali㉿kali]-[~/lab1-reverse]
└─$ cat vsnprintf.c
/*
 * vsnprintf.c
 *
 * vsnprintf(), from which the rest of the printf()
 * family is built
 */
File System

#include <stdarg.h>
#include <stddef.h>
#include <inttypes.h>
#include <string.h>
#include <limits.h>
#include <stdio.h>

enum flags {
    FL_ZERO          = 0x01, /* Zero modifier */
    FL_MINUS         = 0x02, /* Minus modifier */
    FL_PLUS          = 0x04, /* Plus modifier */
    FL_TICK          = 0x08, /* ' modifier */
    FL_SPACE         = 0x10, /* Space modifier */
    FL_HASH          = 0x20, /* # modifier */
    FL_SIGNED        = 0x40, /* Number is signed */
    FL_UPPER         = 0x80 /* Upper case digits */
};

/* These may have to be adjusted on certain implementations */
enum ranks {
    rank_char        = -2,
    rank_short       = -1,
    rank_int         = 0,
    rank_long        = 1,
    rank_longlong    = 2
};

#define MIN_RANK        rank_char
#define MAX_RANK       rank_longlong

#define INTMAX_RANK     rank_longlong
#define SIZE_T_RANK      rank_long
#define PTRDIFF_T_RANK   rank_long

#define EMIT(x) ({ if (o<n){*q++ = (x);} o++; })

static size_t
format_int(char *q, size_t n, uintmax_t val, enum flags flags,
           int base, int width, int prec)
{
    char *qq;
    size_t o = 0, oo;
```

23) Компілюємо для Linux i686 (gcc)

```
[kali㉿kali)-[~/lab1-reverse]
$ i686-linux-gnu-gcc vsnprintf.c -c -o vsnprintf.i686

[kali㉿kali)-[~/lab1-reverse]
$ i686-linux-gnu-objdump -drwC -Mintel -S vsnprintf.i686 > vsnprintf_gcc_i686.lst
```

24) Компілюємо для Linux amd64 (gcc)

```
[kali㉿kali)-[~/lab1-reverse]
$ gcc vsnprintf.c -c -o vsnprintf.amd64

[kali㉿kali)-[~/lab1-reverse]
$ objdump -drwC -Mintel -S vsnprintf.amd64 > vsnprintf_gcc_amd64.lst
```

25) Компілюємо для Linux arm (gcc)

```
[kali㉿kali)-[~/lab1-reverse]
$ arm-linux-gnueabi-gcc vsnprintf.c -c -o vsnprintf.arm

[kali㉿kali)-[~/lab1-reverse]
$ arm-linux-gnueabi-objdump -drwC -S vsnprintf.arm > vsnprintf_gcc_arm.lst
```

26) Компілюємо для Linux aarch64 (gcc)

```
(kali㉿kali)-[~/lab1-reverse]
└─$ aarch64-linux-gnu-gcc vsnprintf.c -c -o vsnprintf.aarch64

(kali㉿kali)-[~/lab1-reverse]
└─$ aarch64-linux-gnu-objdump -drwC -S vsnprintf.aarch64 > vsnprintf_gcc_aarch64.lst
```

27) Всі файли кладемо в архів і порівнюємо

28) Візьмемо реалізацію функції fopen у бібліотеці klibc

```
(kali㉿kali)-[~/lab1-reverse]
└─$ cat fxopen.c
/*
 * fxopen.c
 *
 * Common code between fopen(), fdopen() and the standard descriptors.
 */
#include "stdioioint.h"

FILE *_stdin, *_stdout, *_stderr;

/* This depends on O_RDONLY == 0, O_WRONLY == 1, O_RDWR == 2 */
int __parse_open_mode(const char *mode)
{
    int plus = 0;
    int flags = O_RDONLY;

    while (*mode) {
        switch (*mode++) {
        case 'r':
            flags = O_RDONLY;
            break;
        case 'w':
            flags = O_WRONLY | O_CREAT | O_TRUNC;
            break;
        case 'a':
            flags = O_WRONLY | O_CREAT | O_APPEND;
            break;
        case '+':
            plus = 1;
            break;
        }
    }

    if (plus)
        flags = (flags & ~(O_RDONLY | O_WRONLY)) | O_RDWR;
}

FILE *_fxopen(int fd, int flags, int close_on_err)
{
    FILE *f = NULL;

    f = malloc(sizeof *f);
    if (!f)
        goto err;
    f->fd = fd;
    f->filepos = lseek(fd, 0, SEEK_CUR);
```

29) Компілюємо для Linux i686 (gcc)

```
(kali㉿kali)-[~/lab1-reverse]
└─$ i686-linux-gnu-gcc fxopen.c -c -o fxopen.i686

(kali㉿kali)-[~/lab1-reverse]
└─$ i686-linux-gnu-objdump -drwC -Mintel -S fxopen.i686 > fxopen_gcc_i686.lst
```

30) Компілюємо для Linux amd64 (gcc)

```

└─(kali㉿kali)-[~/lab1-reverse]
$ gcc fxopen.c -c -o fxopen.amd64

└─(kali㉿kali)-[~/lab1-reverse]
$ objdump -drwC -Mintel -S fxopen.amd64 > fxopen_gcc_amd64.lst

```

31) Компілюємо для Linux arm (gcc)

```

└─(kali㉿kali)-[~/lab1-reverse]
$ arm-linux-gnueabi-gcc fxopen.c -c -o fxopen.arm

└─(kali㉿kali)-[~/lab1-reverse]
$ arm-linux-gnueabi-objdump -drwC -S fxopen.arm > fxopen_gcc_arm.lst

```

32) Компілюємо для Linux aarch64 (gcc)

```

└─(kali㉿kali)-[~/lab1-reverse]
$ aarch64-linux-gnu-gcc fxopen.c -c -o fxopen.aarch64

└─(kali㉿kali)-[~/lab1-reverse]
$ aarch64-linux-gnu-objdump -drwC -S fxopen.aarch64 > fxopen_gcc_aarch64.lst

```

33) Всі файли кладемо в архів і порівнюємо

GROUP 1	fxopen_gcc_i686.lst	fxopen_gcc_amd64.lst	fxopen_gcc_arm.lst	fxopen_gcc_aarch64.lst
1	00000093 <_fxopen>:	1 000000000000000f <_fxopen>:	1 00000000 <_fxopen>:	1 00000000000000bc <_fxopen>:
2	93: 55	2 8F: 55	2 e0: e92d4800 push {fp,	2 bc: a9bd7bfd stp x29,
3	push ebp	3 90: 89 e5	3 lr}	3 x30, [sp, #-48]!
4	94: 89 e5	4 93: 83 ec 20	4 e28db004 add fp, sp,	4 c0: 910000fd mov x29, sp
5	mov ebp,esp	5 sub rsp,0x20	5 #4	5 c4: b9001fe0 str w0, [sp,
6	96: 53	6 97: 89 7d ec	6 #24	5 #28]
7	push ebx	7 97: 89 7d ec	7 ec: e24dd018 sub sp, sp,	5 c8: b9001be1 str w1, [sp,
8	97: 83 ec 14	8 mov DWORD PTR [8 #16]	6 #24]
9	sub esp,0x14	9 rbp+0x14],edi	9 ec: e50b01010 str r0, [fp,	6 cc: b90017e2 str w2, [sp,
10	9a: e8 fc ff ff	10 mov DWORD PTR [10 #20]	7 #20]
11	call 9b <_fxopen+0x8>	11 rbp+0x14],edi	11 f0: e50b01014 str r1, [fp,	7 d0: f90017ff str xzr, [
12	9b: R_386_PC32	12 mov DWORD PTR [12 #24] ; 0xfffffffec	8 sp, #40]
13	_x86_get_pc_thunk.bx	13 rbp+0x14],edx	13 f4: e50b01018 str r2, [fp,	8 d4: d2801b00 mov x0,
14	9f: 81 c3 02 00 00 00	14 a0: 48 c7 45 f8 00 00 00	14 #24] ; 0xffffffe8	9 #0xd8
15	add ebx,0x2 a1:	15 00: mov QWORD PTR [15 f8: e3a03000 mov r3, #0	9 #216
16	R_386_GOTPC	16 rbp+0x8],0x0	16 fc: e50b03008 str r3, [fp,	10 dc: f90017e0 str x0, [sp,
17	_GLOBAL_OFFSET_TABLE_	17 a8: 83 e0 0c	17 #8]	11 #40]
18	a5: c7 45 f4 00 00 00 00	18 ad: 83 e0 00 00 00 00	18 100: e3a00098 mov r0, #152	11 e0: f94017e0 ldr x0, [sp,
19	mov DWORD PTR [19 call b2 <_fxopen+0x23>	19 ; 0x98	12 #40]
20	ebp-0xc],0x0	20 ae: R_X86_64_PLT32	20 malloc: 104: R_ARM_CALL	12 e4: f100001f cmp x0, #0x0
21	sub esp,0xc	21 malloc: 104: R_ARM_CALL	21 malloc	13 e8: 540001c0 b.eq 120
22	af: 68 94 00 00 00	22 bb: 74 2d	22 <_fxopen+0x7c>	14 <_fxopen+0x64> // b.none
23	push 0x94	23 je ea <_fxopen+0x5b>	23 108: e1a03000 mov r3, r0	14 ec: f94017e0 ldr x0, [sp,
24	b4: e8 ff ff ff	24 bd: 48 b8 45 f8	24 108: e1a03000 mov r3, r0	15 #40]
25	call b5 <_fxopen+0x22>	25 mov rax,QWORD PTR [25 108: 0a00000f beq 15c	15 f0: d2800401 mov x1,
26	b5: R_386_PLT32 malloc	26 rbp-0x8],rax	26 11c: e51b1008 ldr r1, [fp,	16 #0x20
27	b9: 83 c4 10	27 b6: 48 83 7d f8 00	27 #8]	16 f4: f9004801 str x1, [x0,
28	add esp,0x10	28 cmp QWORD PTR [28 11c: e51b1008 ldr r0, [fp,	17 #144]
29	bc: 89 45 f4	29 rbp-0x8],0x0	29 #8]	17 f8: b9401fe0 ldr w0, [sp,
30	mov DWORD PTR [30 bb: 74 2d	30 128: e1c125f0 strd r2, [18 #28]
31	ebp-0xc],eax	31 je ea <_fxopen+0x5b>	31 r1, #80] ; 0x50	18 fc: 94000000 b1 0 <
32	b7: 83 d4 00	32 bd: 48 b8 45 f8	32 12c: e51b1008 ldr r0, [fp,	19 isatty: fc: R_AARCH64_CALL26 isatty
33	cmp DWORD PTR [33 mov rax,QWORD PTR [33 #16]	19 100: 7100001f cmp w0, #0x0
34	ebp-0xc],0x0	34 rbp-0x8]	34 12c: ebfffffe bl 0 <	20 104: 19f07e0 cset w0,
35	c3: 74 31	35 c1: 48 c7 80 90 00 00 00	35 isatty	20 ne // ne = any
36	je f6 <_fxopen+0x63>	36 20 00 00 00 00 00 00 00	36 isatty	
37	c5: 8b 45 f4	37 mov eax,DWORD PTR [rax+0x90],0x20	37 isatty	
38	mov eax,DWORD PTR [38 cc: 8b 45 ec	38 isatty	
39	ebp-0xc]			

34) Візьмемо реалізацію функції виконання команд ОС system у бібліотеці klibc

```
(kali㉿kali)-[~/lab1-reverse]
└─$ cat system.c
/*
 * system.c
 *
 * The system() function. If this turns out to actually be used,
 * we may want to try to detect the very simple cases (no shell magic)
 * and handle them internally, instead of requiring that /bin/sh be
 * present.
 */
#include <stdlib.h>
#include <unistd.h>
#include <csignal.h>
#include <sys/wait.h>

int system(const char *string)
{
    pid_t pid;
    struct sigaction ignore, old_int, old_quit;
    sigset(SIGCHLD, ignore);
    static const char *argv[] = { "/bin/sh", "-c", NULL, NULL };
    int status;
    extern char** environ;

    /* Block SIGCHLD and ignore SIGINT and SIGQUIT */
    /* Do this before the fork() to avoid races */

    ignore.sa_handler = SIG_IGN;
    sigemptyset(&ignore.sa_mask);
    ignore.sa_flags = 0;
    sigaction(SIGINT, &ignore, &old_int);
    sigaction(SIGQUIT, &ignore, &old_quit);

    sigemptyset(&masked);
    sigaddset(&masked, SIGCHLD);
    sigprocmask(SIG_BLOCK, &masked, &oldmask);

    pid = fork();

    if (pid < 0)
        return -1;
    else if (pid == 0) {
        sigaction(SIGINT, &old_int, NULL);
        sigaction(SIGQUIT, &old_quit, NULL);
        sigprocmask(SIG_SETMASK, &oldmask, NULL);

        argv[2] = string;
        execve(argv[0], (char *const *)argv, (char *const *)environ);
        _exit(127);
    }
}
```

35) Компілюємо для Linux i686 (gcc)

```
(kali㉿kali)-[~/lab1-reverse]
└─$ i686-linux-gnu-gcc system.c -c -o system.i686

(kali㉿kali)-[~/lab1-reverse]
└─$ i686-linux-gnu-objdump -drwC -Mintel -S system.i686 > system_gcc_i686.lst
```

36) Компілюємо для Linux amd64 (gcc)

```
(kali㉿kali)-[~/lab1-reverse]
└─$ gcc system.c -c -o system.amd64

(kali㉿kali)-[~/lab1-reverse]
└─$ objdump -drwC -Mintel -S system.amd64 > system_gcc_amd64.lst
```

37) Компілюємо для Linux arm (gcc)

```
(kali㉿kali)-[~/lab1-reverse]
└─$ arm-linux-gnueabi-gcc system.c -c -o system.arm

(kali㉿kali)-[~/lab1-reverse]
└─$ arm-linux-gnueabi-objdump -drwC -S system.arm > system_gcc_arm.lst
```

38) Компілюємо для Linux aarch64 (gcc)

```
(kali㉿kali)-[~/lab1-reverse]
└─$ aarch64-linux-gnu-gcc system.c -c -o system.aarch64

(kali㉿kali)-[~/lab1-reverse]
└─$ aarch64-linux-gnu-objdump -drwC -S system.aarch64 > system_gcc_aarch64.lst
```

39) Всі файли кладемо в архів і порівнюємо

```

GROUP 1
system_gcc_i686.lst
111    push    0x0
112    lea     eax,[ebp-0xb0]
113    push    eax
114    push    0x3
115    add    esp,0x10
116    sub    esp,0x4
117    push    0x0
118    add    esp,0x10
119    mov    eax,[ebp-0xb0]
120    push    eax
121    push    0x2
122    add    esp,0x10
123    mov    eax,DWORD PTR [ebp-0xb4]
124    mov    ebx,DWORD PTR [ebp-0x4]
125    c9
126    leave
127    ret
128

system_gcc_amd64.lst
75    mov    rsi,rax
76    mov    edi,0x2
77    call   17f <system+0x17f>
78    mov    ba 00 00 00
79    mov    edx,0x0
80    mov    eax,0xc6
81    mov    rax,[rbp-0x1e0]
82    mov    eax,0x0
83    mov    rax,[rbp-0x2e0]
84    mov    edx,0x0
85    mov    rsi,rax
86    mov    edi,0x2
87    call   1b1 <system+0x1b1>
88    c9
89    leave
90    ret
91

system_gcc_arm.lst
94    e3a00003  mov r0, #3
95    ebfffffe b1 0 <
96    sigaction> 174: R_ARM_CALL
97    e3a02000  mov r2, #0
98    e1a1003  mov r1, r3
99    e3a00002  mov r0, #2
100   ebfffffe b1 0 <
101   sigprocmask> 188:
102   e1a00003  mov r0, r3
103   e24bd008  sub sp, fp,
104   198 <system+0x198>
105   e5b32bb8 ldr r3, [
106   0x0000180 .word
107   19c: 0x0000180 19c:
108   R_ARM_GOTPC
109   _GLOBAL_OFFSET_TABLE_
110   0x000009c .word
111   0x000009c 1a0:
112   R_ARM_REL32
113   .data.rel.local
114   0x0000098 1a4:
115   0x0000090 1a4:
116   R_ARM_REL32
117   .data.rel.local
118   0x0000000 1a8:
119   0x0000000 1a8:
120   R_ARM_GOT32
121   environ
122   0x000007c .word
123   0x000007c 1ac:
124   R_ARM_REL32
125   .data.rel.local
126

system_gcc_aarch64.lst
148: aa0003e1  mov x1, x0
149: 52800040  mov w0,
#0x2
#2
150: 94000000  bl 0 <
sigaction> 150:
R_AARCH64 CALL26 sigaction
151: 9100c3e0  add x0, sp,
#0x130
152: d2800002  mov x2,
#0x0
#0
153: aa0003e1  mov x1, x0
154: 52800000  mov w0,
#0x3
#3
155: 94000000  bl 0 <
sigaction> 164:
R_AARCH64 CALL26 sigaction
156: 9100c3e0  add x0, sp,
#0x30
157: d2800002  mov x2,
#0x0
#0
158: 94000000  bl 0 <
sigprocmask> 178:
R_AARCH64 CALL26 sigprocmask
159: b9402fe0  ldr w0, [
sp, #44]
160: 94000000  bl 0 <
sigprocmask> 178:
R_AARCH64 CALL26 sigprocmask
161: 9100c3e0  add x0, sp,
#0x30
162: d65f03c0  ret
163

```