Лабораторна робота 1

Охріменко Анастасія

Варіант 10

Аналіз програмного коду мов високого рівня

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/);**

Кожен раз задаємо цільову архітектуру зміними оточення у скрипті vcvarsall.bat. Далі за допомогою компілятора cl.exe та його опції /FA генеруємо файл лістингу асемблера з аналізом кожної строчки коду. Також присутні опції c - Включає машинний код у списку, s - Включає вихідний код у списку, u - Кодує файл списку у форматі UTF-8 та включає маркер порядку байтів.

Windows x64:

```
**********************************************************************
** Visual Studio 2022 Developer Command Prompt v17.14.13
** Copyright (c) 2025 Microsoft Corporation
**********************************************************************

C:\Users\asanty\source\repos>cd C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Auxiliary\Build

C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Auxiliary\Build>vcvarsall.bat amd64
**********************************************************************
** Visual Studio 2022 Developer Command Prompt v17.14.13
** Copyright (c) 2025 Microsoft Corporation
**********************************************************************
[vcvarsall.bat] Environment initialized for: 'x64'

C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Auxiliary\Build>cd C:\Users\asanty\source\repos

C:\Users\asanty\source\repos>cl /FAcsu hanoi.c
Microsoft (R) C/C++ Optimizing Compiler Version 19.44.35215 for x64
Copyright (C) Microsoft Corporation.  All rights reserved.

hanoi.c
Microsoft (R) Incremental Linker Version 14.44.35215.0
Copyright (C) Microsoft Corporation.  All rights reserved.

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

```
1   ; Listing generated by Microsoft (R) Optimizing Compiler Version 19.44.35215.0
2
3   include listing.inc
4
5   INCLUDELIB LIBCMT
6   INCLUDELIB OLDNAMES
7
8   PUBLIC  __local_stdio_printf_options
9   PUBLIC  _vfprintf_l
10  PUBLIC  printf
11  PUBLIC  hanoi
12  PUBLIC  main
13  EXTRN   __acrt_iob_func:PROC
14  EXTRN   __stdio_common_vfprintf:PROC
15  _DATA   SEGMENT
16  COMM    ?_OptionsStorage@?1??__local_stdio_printf_options@@9@9:QWORD
17  _DATA   ENDS
18  ;   COMDAT pdata
19  pdata   SEGMENT
20  $pdata$_vfprintf_l DD imagerel $LN3
21      DD  imagerel $LN3+67
22      DD  imagerel $unwind$_vfprintf_l
23  pdata   ENDS
24  ;   COMDAT pdata
25  pdata   SEGMENT
26  $pdata$printf DD imagerel $LN3
27      DD  imagerel $LN3+87
28      DD  imagerel $unwind$printf
29  pdata   ENDS
30  pdata   SEGMENT
31  $pdata$hanoi DD imagerel $LN5
32      DD  imagerel $LN5+157
33      DD  imagerel $unwind$hanoi
34  $pdata$main DD  imagerel $LN3
35      DD  imagerel $LN3+30
36      DD  imagerel $unwind$main
37  pdata   ENDS
38  _DATA   SEGMENT
```

Windows ARM:

```
C:\Users\asanty\source\repos>cd C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Auxiliary\Build

C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Auxiliary\Build>vcvarsall.bat x86_arm
**********************************************************************
** Visual Studio 2022 Developer Command Prompt v17.14.13
** Copyright (c) 2025 Microsoft Corporation
**********************************************************************
[vcvarsall.bat] Environment initialized for: 'x86_arm'

C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Auxiliary\Build>cd C:\Users\asanty\source\repos

C:\Users\asanty\source\repos>cl /FAcsu hanoi.c
Microsoft (R) C/C++ Optimizing Compiler Version 19.44.35215 for x86
Copyright (C) Microsoft Corporation.  All rights reserved.

hanoi.c
Microsoft (R) Incremental Linker Version 14.44.35215.0
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\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.44.35207\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\asanty\source\repos>
```

```
; Listing generated by Microsoft (R) Optimizing Compiler Version 19.44.35215.0

    TITLE   C:\Users\asanty\source\repos\hanoi.obj
    .686P
    .XMM
    include listing.inc
    .model  flat

INCLUDELIB LIBCMT
INCLUDELIB OLDNAMES

PUBLIC  ___local_stdio_printf_options
PUBLIC  __vfprintf_l
PUBLIC  _printf
PUBLIC  _hanoi
PUBLIC  _main
EXTRN   ___acrt_iob_func:PROC
EXTRN   ___stdio_common_vfprintf:PROC
_DATA   SEGMENT
COMM    ?_OptionsStorage@?1??__local_stdio_printf_options@@9@9:QWORD
_DATA   ENDS
_DATA   SEGMENT
$SG10021 DB ' Move disk 1 from %c to %c', 0aH, 00H
$SG10022 DB ' Move disk %d from %c to %c', 0aH, ' ', 00H
_DATA   ENDS
; Function compile flags: /Odtp
; File C:\Users\asanty\source\repos\hanoi.c
_TEXT   SEGMENT
_main   PROC

; 11   : int main() {

  00000 55       push    ebp
  00001 8b ec        mov     ebp, esp

; 12   :     hanoi(3, 'A', 'C', 'B');
```

Windows ARM64:

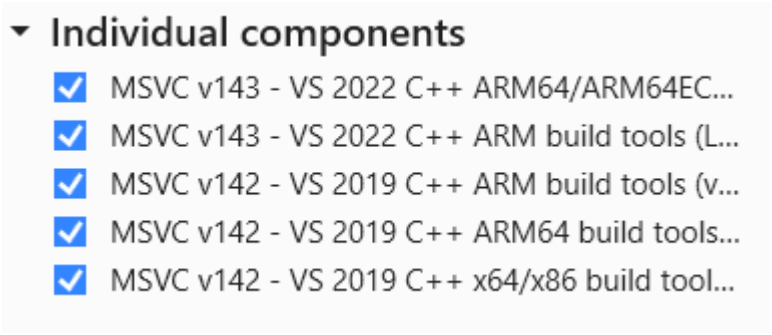
```
C:\Users\asanty\source\repos>cd C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Auxiliary\Build

C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Auxiliary\Build>vcvarsall.bat x86_arm64
**********************************************************************
** Visual Studio 2022 Developer Command Prompt v17.14.13
** Copyright (c) 2025 Microsoft Corporation
**********************************************************************
[vcvarsall.bat] Environment initialized for: 'x86_arm64'

C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Auxiliary\Build>cd C:\Users\asanty\source\repos

C:\Users\asanty\source\repos>cl /FAcsu hanoi.c
Microsoft (R) C/C++ Optimizing Compiler Version 19.44.35215 for x86
Copyright (C) Microsoft Corporation.  All rights reserved.

hanoi.c
Microsoft (R) Incremental Linker Version 14.44.35215.0
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\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.44.35207\lib\x64\LIBCMT.lib : warning LNK4272:
library machine type 'x64' conflicts with target machine type 'x86'
hanoi.exe : fatal error LNK1120: 3 unresolved externals
```

```
; Listing generated by Microsoft (R) Optimizing Compiler Version 19.44.

    TITLE   C:\Users\asanty\source\repos\hanoi.obj
    .686P
    .XMM
    include listing.inc
    .model  flat

INCLUDELIB LIBCMT
INCLUDELIB OLDNAMES

PUBLIC  ___local_stdio_printf_options
PUBLIC  __vfprintf_l
PUBLIC  _printf
PUBLIC  _hanoi
PUBLIC  _main
EXTRN   ___acrt_iob_func:PROC
EXTRN   ___stdio_common_vfprintf:PROC
_DATA   SEGMENT
COMM    ?_OptionsStorage@?1??__local_stdio_printf_options@@9@9:QWORD
_DATA   ENDS
_DATA   SEGMENT
$SG10021 DB ' Move disk 1 from %c to %c', 0aH, 00H
$SG10022 DB ' Move disk %d from %c to %c', 0aH, ' ', 00H
_DATA   ENDS
; Function compile flags: /Odtp
; File C:\Users\asanty\source\repos\hanoi.c
_TEXT   SEGMENT
_main   PROC

; 11   : int main() {

  00000 55       push    ebp
  00001 8b ec        mov     ebp, esp

; 12   :     hanoi(3, 'A', 'C', 'B');
```

Результат: оскільки це різні архітектури, то і компільований асемблерний код для кожної мав би бути різним, оскільки в кожній архітектурі команди різні. Але в цьому випадку для архітектур arm та arm64 вийшов однаковий асемблерний код. Виникає кожен раз помилка лінкування через відсутність потрібної бібліотеки. Тим не менш

у мене всі компоненти встановлені:

Була встановлена застарівша версія, брала з методички один в один. Тому встановила нові:



Тепер асемблерний код для arm:

```
; File C:\Users\asanty\source\repos\hanoi.c
    .text$mn

  00000       |main|  PROC

; 11    : int main() {

  00000 e92d 4800     push        {r11,lr}
  00004 46eb          mov         r11,sp
  00006 b082          sub         sp,sp,#8
  00008       |$M4|

; 12    :     hanoi(3, 'A', 'C', 'B');

  00008 2342          movs        r3,#0x42
  0000a 2243          movs        r2,#0x43
  0000c 2141          movs        r1,#0x41
  0000e 2003          movs        r0,#3
  00010 f000 f800     bl          hanoi

; 13    : }


  00014 2300          movs        r3,#0
  00016 9300          str         r3,[sp]
  00018 9800          ldr         r0,[sp]
  0001a       |$M3|
  0001a b002          add         sp,sp,#8
  0001c e8bd 8800     pop         {r11,pc}
  00020       |$M5|

              ENDP  ; |main|

; Function compile flags: /Odsp
; File C:\Users\asanty\source\repos\hanoi.c
    .text$mn

  00000       |hanoi| PROC
```

Для arm64:

```
; File C:\Users\asanty\source\repos\hanoi.c

    AREA    |.text$mn|, CODE, ARM64

  00000      |main|  PROC

; 11   : int main() {

  00000      |$LN3|
  00000 a9bf7bfd     stp         fp,lr,[sp,#-0x10]!
  00004 910003fd     mov         fp,sp

; 12   :    hanoi(3, 'A', 'C', 'B');

  00008 52800843     mov         w3,#0x42
  0000c 52800862     mov         w2,#0x43
  00010 52800821     mov         w1,#0x41
  00014 52800060     mov         w0,#3
  00018 94000000     bl          hanoi
  0001c d503201f     nop

; 13   : }


  00020 52800000     mov         w0,#0
  00024 a8c17bfd     ldp         fp,lr,[sp],#0x10
  00028 d65f03c0     ret

            ENDP  ; |main|

; Function compile flags: /Odsp
; File C:\Users\asanty\source\repos\hanoi.c

    AREA    |.text$mn|, CODE, ARM64

  00000      |hanoi| PROC

; 2    : void hanoi(int n, char from, char to, char aux) {
```

Результат2:

Тепер можна розглянути відмінності компілюваного коду різних архітектур:

Ознака	**AMD64**	**ARM**	**ARM64**	**Чому так відбувається**
Регістри	eax, ebx ...	r0,r1,r2...r15	w0,w1...w30 x0,x1...x30	AMD64 8 регістрів спільного призначення (R8 — R15), всі 16 регістрів спільного призначення 64-бітні. ARM існує 30 універсальних 32-бітних регістрів. Перші 16 регістрів доступні в режимі користувацького рівня, додаткові регістри доступні в режимі привілейованого виконання програмного забезпечення. ARM64 Архітектура передбачає 31 регістр загального призначення. Кожен регістр може використовуватися як 64-бітний регістр X (X0..X30) або як 32-бітний регістр W (W0..W30).
Виклик функції	call func	bl func	bl func	x86 має стекову модель викликів, ARM — регістр LR (Link Register) для збереження адреси повернення.

Повернення з функції	ret	push {r11,lr}
		mov r11,sp
		sub sp,sp,#8
			ret	AMD - читає адресу з вершини стеку, ARM замість стека використовує спеціальний регістр для адреси повернення lr (Link Register)

Реєстр стека	rsp	sp або r13	sp	Усі архітектури мають вказівник на стек.

Linux amd64

Для компіляції на лінукс використовуємо gcc -Wa , -adhln -g hanoi.c > hanoi .amd64.lst.
a: turn on listings

d: omit debugging directives (Без d у листинг вставляються спеціальні директиви для налагоджувача)

n: omit forms processing (У звичайному листингу асемблер намагається форматувати вихід)

h: include high-level source

l: include assembly

```
anastasiia@anastasiia:~/Desktop/lab1_rev$ gcc -Wa,-adhln -g hanoi.c > hanoi.amd6
4.lst
anastasiia@anastasiia:~/Desktop/lab1_rev$ ls
a.out  hanoi  hanoi.amd64.lst  hanoi.c
anastasiia@anastasiia:~/Desktop/lab1_rev$ cat hanoi.amd64.lst
   1                            .file   "hanoi.c"
   2                            .text
   3                    .Ltext0:
   4                            .file 0 "/home/anastasiia/Desktop/lab1_rev" "han
oi.c"
   5                            .section        .rodata
   6                    .LC0:
   7 0000 204D6F76              .string " Move disk 1 from %c to %c\n"
   7      65206469
   7      736B2031
   7      2066726F
   7      6D202563
   8                    .LC1:
   9 001c 204D6F76              .string " Move disk %d from %c to %c\n "
   9      65206469
   9      736B2025
   9      64206672
   9      6F6D2025
```

Linux arm

```
anastasiia@anastasiia:~/Desktop/lab1_rev$ arm-linux-gnueabi-gcc -Wa,-adhln -g h
anoi.c > hanoi.arm.lst
anastasiia@anastasiia:~/Desktop/lab1_rev$ ls
a.out  hanoi  hanoi.amd64.lst  hanoi.arm.lst  hanoi.c
anastasiia@anastasiia:~/Desktop/lab1_rev$ cat hanoi.arm.lst
   1                            .arch armv5t
   2                            .fpu softvfp
   3                            .eabi_attribute 20, 1
   4                            .eabi_attribute 21, 1
   5                            .eabi_attribute 23, 3
   6                            .eabi_attribute 24, 1
   7                            .eabi_attribute 25, 1
   8                            .eabi_attribute 26, 2
   9                            .eabi_attribute 30, 6
  10                            .eabi_attribute 34, 0
  11                            .eabi_attribute 18, 4
  12                            .file   "hanoi.c"
  13                            .text
  14                    .Ltext0:
  15                            .cfi_sections   .debug_frame
  16                            .file 1 "hanoi.c"
  17                            .section        .rodata
  18                            .align  2
  19                    .LC0:
```

Linux arm64

```
anastasiia@anastasiia:~/Desktop/lab1_rev$ aarch64-linux-gnu-gcc -Wa,-adhln -g h
anoi.c > hanoi.aarch64.lst
anastasiia@anastasiia:~/Desktop/lab1_rev$ cat hanoi.aarch64.lst
   1                            .arch armv8-a
   2                            .file   "hanoi.c"
   3                            .text
   4                    .Ltext0:
   5                            .file 0 "/home/anastasiia/Desktop/lab1_rev" "ha
noi.c"
   6                            .section        .rodata
   7                            .align  3
   8                    .LC0:
   9 0000 204D6F76              .string " Move disk 1 from %c to %c\n"
   9      65206469
   9      736B2031
   9      2066726F
   9      6D202563
  10 001c 00000000              .align  3
  11                    .LC1:
  12 0020 204D6F76              .string " Move disk %d from %c to %c\n "
  12      65206469
  12      736B2025
  12      64206672
  12      6F6D2025
```

Linux amd64 (LLVM clang, https://llvm.org/) :

```
anastasiia@anastasiia:~/Desktop/lab1_rev$ clang hanoi.c -S -O3 -o hanoi
anastasiia@anastasiia:~/Desktop/lab1_rev$ ls
a.out  hanoi.aarch64.lst  hanoi.arm.lst  llvm-project
hanoi  hanoi.amd64.lst    hanoi.c
anastasiia@anastasiia:~/Desktop/lab1_rev$ cat hanoi
        .text
        .file   "hanoi.c"
        .globl  hanoi                           # -- Begin function hanoi
        .p2align        4, 0x90
        .type   hanoi,@function
hanoi:                                  # @hanoi
        .cfi_startproc
# %bb.0:
        pushq   %rbp
        .cfi_def_cfa_offset 16
        pushq   %r15
        .cfi_def_cfa_offset 24
        pushq   %r14
        .cfi_def_cfa_offset 32
        pushq   %r13
```

Різниця між компіляцією на Windows та Linux полягає у системних викликах (syscall, WinAPI DLL)

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: на графах – цикли;**

I686 :

```
anastasiia@anastasiia:~/Desktop/lab1_rev$ i686-linux-gnu-g++ -Wa,-adhln -g graph
_cycle.cpp -o graph.i686 > graph.i686.lst
```

```
175:/usr/i686-linux-gnu/include/c++/13/new **** { return __p; }
 11                             .loc 1 175 1
 12                             .cfi_startproc
 13 0000 F30F1EFB               endbr32
 14 0004 55                     pushl   %ebp
 15                             .cfi_def_cfa_offset 8
 16                             .cfi_offset 5, -8
 17 0005 89E5                   movl    %esp, %ebp
 18                             .cfi_def_cfa_register 5
 19 0007 E8FCFFFF               call    __x86.get_pc_thunk.ax
 19      FF
 20 000c 05010000               addl    $_GLOBAL_OFFSET_TABLE_, %eax
 20      00
 21                             .loc 1 175 10
 22 0011 8B450C                 movl    12(%ebp), %eax
 23                             .loc 1 175 15
 24 0014 5D                     popl    %ebp
 25                             .cfi_restore 5
 26                             .cfi_def_cfa 4, 4
 27 0015 C3                     ret
 28                             .cfi_endproc
 29                     .LFE468:
 31                             .section        .text._ZdlPvS_,"axG",@progbits,_ZdlPvS_,comdat
 32                             .weak   _ZdlPvS_
 34                     _ZdlPvS_:
 35                     .LFB470:
```

Amd64 :

```
anastasiia@anastasiia:~/Desktop/lab1_rev$ g++ -Wa,-adhln -g graph_cycle.cpp > graph.amd64.lst
```

```
 GNU nano 7.2                                                graph.amd64.lst
172:/usr/include/c++/13/new ****
173:/usr/include/c++/13/new **** // Default placement versions of operator new.
174:/usr/include/c++/13/new **** _GLIBCXX_NODISCARD inline void* operator new(std::size_t, void* __p) _GLIBCXX_USE_NOEXCEPT
175:/usr/include/c++/13/new **** { return __p; }
 11                             .loc 1 175 1
 12                             .cfi_startproc
 13 0000 F30F1EFA               endbr64
 14 0004 55                     pushq   %rbp
 15                             .cfi_def_cfa_offset 16
 16                             .cfi_offset 6, -16
 17 0005 4889E5                 movq    %rsp, %rbp
 18                             .cfi_def_cfa_register 6
 19 0008 48897DF8               movq    %rdi, -8(%rbp)
 20 000c 488975F0               movq    %rsi, -16(%rbp)
 21                             .loc 1 175 10
 22 0010 488B45F0               movq    -16(%rbp), %rax
 23                             .loc 1 175 15
 24 0014 5D                     popq    %rbp
 25                             .cfi_def_cfa 7, 8
 26 0015 C3                     ret
 27                             .cfi_endproc
 28                         .LFE471:
 30                             .section        .text._ZdlPvS_,"axG",@progbits,_ZdlPvS_,comdat
 31                             .weak   _ZdlPvS_
 33                         _ZdlPvS_:
 34                         .LFB473:
176:/usr/include/c++/13/new ****  GLIBCXX_NODISCARD inline void* operator new[](std::size_t, void*   p)  GLIBCXX_USE_NOEXCEPT
```

Arm :

```
anastasiia@anastasiia:~/Desktop/lab1_rev$ arm-linux-gnueabi-g++ -Wa,-adhln -g graph_cycle.cpp -o graph.arm > graph.arm.lst
```

```
  1                             .arch armv5t
  2                             .fpu softvfp
  3                             .eabi_attribute 20, 1
  4                             .eabi_attribute 21, 1
  5                             .eabi_attribute 23, 3
  6                             .eabi_attribute 24, 1
  7                             .eabi_attribute 25, 1
  8                             .eabi_attribute 26, 2
  9                             .eabi_attribute 30, 6
 10                             .eabi_attribute 34, 0
 11                             .eabi_attribute 18, 4
 12                             .file   "graph_cycle.cpp"
 13                             .text
 14                         .Ltext0:
 15                             .cfi_sections   .debug_frame
 16                             .file 1 "graph_cycle.cpp"
 17                             .section        .text._ZnwjPv,"axG",%progbits,_ZnwjPv,comdat
 18                             .align  2
 19                             .weak   _ZnwjPv
 20                             .syntax unified
 21                             .arm
 23                         _ZnwjPv:
 24                             .fnstart
 25                         .LFB466:
 26                             .file 2 "/usr/arm-linux-gnueabi/include/c++/13/new"
```

Aarch64 :

```
anastasiia@anastasiia:~/Desktop/lab1_rev$ aarch64-linux-gnu-g++ -Wa,-adhln -g graph_cycle.cpp -o graph.aarch64 > graph.aarch64.lst
```

```
 GNU nano 7.2                                          graph.aarch64.lst
172:/usr/aarch64-linux-gnu/include/c++/13/new ****
173:/usr/aarch64-linux-gnu/include/c++/13/new **** // Default placement versions of operator new.
174:/usr/aarch64-linux-gnu/include/c++/13/new **** _GLIBCXX_NODISCARD inline void* operator new(std::size_t, void* __p) _GLIBCXX_US
175:/usr/aarch64-linux-gnu/include/c++/13/new **** { return __p; }
 13                             .loc 1 175 1
 14                             .cfi_startproc
 15 0000 FF4300D1               sub     sp, sp, #16
 16                             .cfi_def_cfa_offset 16
 17 0004 E00700F9               str     x0, [sp, 8]
 18 0008 E10300F9               str     x1, [sp]
 19                             .loc 1 175 10
 20 000c E00340F9               ldr     x0, [sp]
 21                             .loc 1 175 15
 22 0010 FF430091               add     sp, sp, 16
 23                             .cfi_def_cfa_offset 0
 24 0014 C0035FD6               ret
 25                             .cfi_endproc
 26                     .LFE469:
 28                             .section        .text._ZdlPvS_,"axG",@progbits,_ZdlPvS_,comdat
 29                             .align  2
 30                             .weak   _ZdlPvS_
 32                     _ZdlPvS_:
 33                     .LFB471:
176:/usr/aarch64-linux-gnu/include/c++/13/new **** _GLIBCXX_NODISCARD inline void* operator new[](std::size_t, void* __p) _GLIBCXX_
```

Знайдемо 4 основі функції (isCyclicUtil, constructadj, isCyclic, main) в graph.amd64.lst:

```
  5:graph_cycle.cpp **** bool isCyclicUtil(vector<vector<int>> &adj, int u, vector<bool> &visited, vector<bool> &recStack)
  6:graph_cycle.cpp **** {
667                     .loc 4 6 1
668                     .cfi_startproc
669 0000 F30F1EFA       endbr64
670 0004 55             pushq   %rbp
671                     .cfi_def_cfa_offset 16
672                     .cfi_offset 6, -16
673 0005 4889E5         movq    %rsp, %rbp
674                     .cfi_def_cfa_register 6
675 0008 4883EC60       subq    $96, %rsp
676 000c 48897DB8       movq    %rdi, -72(%rbp)
677 0010 8975B4         movl    %esi, -76(%rbp)
678 0013 488955A8       movq    %rdx, -88(%rbp)
679 0017 48894DA0       movq    %rcx, -96(%rbp)
680                     .loc 4 6 1
681 001b 64488B04       movq    %fs:40, %rax
681      25280000
681      00
682 0024 488945F8       movq    %rax, -8(%rbp)
683 0028 31C0           xorl    %eax, %eax
```

```
 32:graph_cycle.cpp **** vector<vector<int>> constructadj(int V, vector<vector<int>> &edges)
 33:graph_cycle.cpp **** {
885                     .loc 4 33 1
886                     .cfi_startproc
887                     .cfi_personality 0x9b,DW.ref.__gxx_personality_v0
888                     .cfi_lsda 0x1b,.LLSDA9771
889 01ce F30F1EFA       endbr64
890 01d2 55             pushq   %rbp
891                     .cfi_def_cfa_offset 16
892                     .cfi_offset 6, -16
893 01d3 4889E5         movq    %rsp, %rbp
894                     .cfi_def_cfa_register 6
895 01d6 53             pushq   %rbx
896 01d7 4883EC58       subq    $88, %rsp
897                     .cfi_offset 3, -24
898 01db 48897DB8       movq    %rdi, -72(%rbp)
899 01df 8975B4         movl    %esi, -76(%rbp)
900 01e2 488955A8       movq    %rdx, -88(%rbp)
901                     .loc 4 33 1
902 01e6 64488B04       movq    %fs:40, %rax
902      25280000
902      00
903 01ef 488945E8       movq    %rax, -24(%rbp)
904 01f3 31C0           xorl    %eax, %eax
905 01f5 488D45C8       leaq    -56(%rbp), %rax
906 01f9 488945E0       movq    %rax, -32(%rbp)
```

```
 43:graph_cycle.cpp **** bool isCyclic(int V, vector<vector<int>> &edges)
 44:graph_cycle.cpp **** {
1083                    .loc 4 44 1
1084                    .cfi_startproc
1085                    .cfi_personality 0x9b,DW.ref.__gxx_personality_v0
1086                    .cfi_lsda 0x1b,.LLSDA9781
1087 034b F30F1EFA          endbr64
1088 034f 55               pushq   %rbp
1089                    .cfi_def_cfa_offset 16
1090                    .cfi_offset 6, -16
1091 0350 4889E5           movq    %rsp, %rbp
1092                    .cfi_def_cfa_register 6
1093 0353 53               pushq   %rbx
1094 0354 4881ECC8         subq    $200, %rsp
1094      000000
1095                    .cfi_offset 3, -24
1096 035b 89BD3CFF         movl    %edi, -196(%rbp)
1096      FFFF
1097 0361 4889B530         movq    %rsi, -208(%rbp)
1097      FFFFFF
1098                    .loc 4 44 1
1099 0368 64488B04         movq    %fs:40, %rax
1099      25280000
1099      00
1100 0371 488945E8         movq    %rax, -24(%rbp)
1101 0375 31C0             xorl    %eax, %eax
```

```
 63:graph_cycle.cpp **** int main()
 64:graph_cycle.cpp **** {
1370                    .loc 4 64 1
1371                    .cfi_startproc
1372                    .cfi_personality 0x9b,DW.ref.__gxx_personality_v0
1373                    .cfi_lsda 0x1b,.LLSDA9782
1374 0583 F30F1EFA          endbr64
1375 0587 55               pushq   %rbp
1376                    .cfi_def_cfa_offset 16
1377                    .cfi_offset 6, -16
1378 0588 4889E5           movq    %rsp, %rbp
1379                    .cfi_def_cfa_register 6
1380 058b 4157             pushq   %r15
1381 058d 4156             pushq   %r14
1382 058f 4155             pushq   %r13
1383 0591 4154             pushq   %r12
1384 0593 53               pushq   %rbx
1385 0594 4881EC08         subq    $264, %rsp
1385      010000
1386                    .cfi_offset 15, -24
1387                    .cfi_offset 14, -32
1388                    .cfi_offset 13, -40
1389                    .cfi_offset 12, -48
1390                    .cfi_offset 3, -56
1391                    .loc 4 64 1
1392 059b 64488B0C         movq    %fs:40, %rcx
1392      25280000
1392      00
1393 05a4 48894DC8         movq    %rcx, -56(%rbp)
1394 05a8 31C9             xorl    %ecx, %ecx
```

- **Криптографічні алгоритми, алгоритми кодування та контролю цілісності [26, 27]: Salsa20;**

I686 :

```
anastasiia@anastasiia:~/Desktop/lab1_rev/Salsa20$ i686-linux-gnu-g++ -Wa,-adhln -g salsa20.cpp -o salsa20.i686 > salsa20.i686.lst
/usr/lib/gcc-cross/i686-linux-gnu/13/../../../../i686-linux-gnu/bin/ld: /usr/lib/gcc-cross/i686-linux-gnu/13/../../../../i686-linux-gnu/
lib/../lib/Scrt1.o: in function `_start':
(.text+0x1e): undefined reference to `main'
collect2: error: ld returned 1 exit status
```

```
11:salsa20.cpp   **** {
10                              .loc 1 11 1
11                              .cfi_startproc
12 0000 F30F1EFB                endbr32
13 0004 55                      pushl   %ebp
14                              .cfi_def_cfa_offset 8
15                              .cfi_offset 5, -8
16 0005 89E5                    movl    %esp, %ebp
17                              .cfi_def_cfa_register 5
18 0007 53                      pushl   %ebx
19 0008 83EC64                  subl    $100, %esp
20                              .cfi_offset 3, -12
21 000b E8FCFFFF                call    __x86.get_pc_thunk.ax
21      FF
22 0010 05010000                addl    $_GLOBAL_OFFSET_TABLE_, %eax
22      00
23 0015 8B4508                  movl    8(%ebp), %eax
24 0018 8945A4                  movl    %eax, -92(%ebp)
25 001b 8B450C                  movl    12(%ebp), %eax
26 001e 8945A0                  movl    %eax, -96(%ebp)
27                              .loc 1 11 1
28 0021 65A11400                movl    %gs:20, %eax
28      0000
29 0027 8945F4                  movl    %eax, -12(%ebp)
30 002a 31C0                    xorl    %eax, %eax
12:salsa20.cpp   ****           int i;
13:salsa20.cpp   ****           uint32_t x[16];
```

Amd64 :

```
anastasiia@anastasiia:~/Desktop/lab1_rev/Salsa20$ g++ -Wa,-adhln -g salsa20.cpp -o salsa20.amd64 > salsa20.amd64.lst
/usr/bin/ld: /usr/lib/gcc/x86_64-linux-gnu/13/../../../x86_64-linux-gnu/Scrt1.o: in function `_start':
(.text+0x1b): undefined reference to `main'
collect2: error: ld returned 1 exit status
```

```
11:salsa20.cpp   **** {
10                              .loc 1 11 1
11                              .cfi_startproc
12 0000 F30F1EFA                endbr64
13 0004 55                      pushq   %rbp
14                              .cfi_def_cfa_offset 16
15                              .cfi_offset 6, -16
16 0005 4889E5                  movq    %rsp, %rbp
17                              .cfi_def_cfa_register 6
18 0008 4883EC70                subq    $112, %rsp
19 000c 48897D98                movq    %rdi, -104(%rbp)
20 0010 48897590                movq    %rsi, -112(%rbp)
21                              .loc 1 11 1
22 0014 64488B04                movq    %fs:40, %rax
22      25280000
22      00
23 001d 488945F8                movq    %rax, -8(%rbp)
24 0021 31C0                    xorl    %eax, %eax
12:salsa20.cpp   ****           int i;
13:salsa20.cpp   ****           uint32_t x[16];
14:salsa20.cpp   ****
15:salsa20.cpp   ****           for (i = 0; i < 16; ++i)
25                              .loc 1 15 9
26 0023 C745AC00                movl    $0, -84(%rbp)
26      000000
27                              .loc 1 15 2
28 002a EB23                    jmp     .L2
```

Arm :

```
anastasiia@anastasiia:~/Desktop/lab1_rev/Salsa20$ arm-linux-gnueabi-g++ -Wa,-adhln -g salsa20.cpp -o salsa20.arm > salsa20.arm.lst
/usr/lib/gcc-cross/arm-linux-gnueabi/13/../../../../arm-linux-gnueabi/bin/ld: /usr/lib/gcc-cross/arm-linux-gnueabi/13/../../../../arm-li
nux-gnueabi/lib/crt1.o: in function `_start':
(.text+0x40): undefined reference to `main'
collect2: error: ld returned 1 exit status
```

```
11:salsa20.cpp   **** {
25                            .loc 1 11 1
26                            .cfi_startproc
27                            @ args = 0, pretend = 0, frame = 80
28                            @ frame_needed = 1, uses_anonymous_args = 0
29 0000 00482DE9              push    {fp, lr}
30                            .cfi_def_cfa_offset 8
31                            .cfi_offset 11, -8
32                            .cfi_offset 14, -4
33 0004 04B08DE2              add     fp, sp, #4
34                            .cfi_def_cfa 11, 4
35 0008 50D04DE2              sub     sp, sp, #80
36 000c 50000BE5              str     r0, [fp, #-80]
37 0010 54100BE5              str     r1, [fp, #-84]
38                            .loc 1 11 1
39 0014 84349FE5              ldr     r3, .L9
40 0018 003093E5              ldr     r3, [r3]
41 001c 08300BE5              str     r3, [fp, #-8]
42 0020 0030A0E3              mov     r3, #0
12:salsa20.cpp   ****         int i;
13:salsa20.cpp   ****         uint32_t x[16];
14:salsa20.cpp   ****
15:salsa20.cpp   ****         for (i = 0; i < 16; ++i)
43                            .loc 1 15 9
44 0024 0030A0E3              mov     r3, #0
45 0028 4C300BE5              str     r3, [fp, #-76]
```

Aarch64 :

```
anastasiia@anastasiia:~/Desktop/lab1_rev/Salsa20$ aarch64-linux-gnu-g++ -Wa,-adhln -g salsa20.cpp -o salsa20.aarch > salsa20.aarch.lst
/usr/lib/gcc-cross/aarch64-linux-gnu/13/../../../../aarch64-linux-gnu/bin/ld: /usr/lib/gcc-cross/aarch64-linux-gnu/13/../../../../aarch6
4-linux-gnu/lib/../lib/Scrt1.o: in function `_start':
(.text+0x1c): undefined reference to `main'
/usr/lib/gcc-cross/aarch64-linux-gnu/13/../../../../aarch64-linux-gnu/bin/ld: (.text+0x20): undefined reference to `main'
collect2: error: ld returned 1 exit status
```

```
11:salsa20.cpp   **** {
12                            .loc 1 11 1
13                            .cfi_startproc
14 0000 FFC301D1              sub     sp, sp, #112
15                            .cfi_def_cfa_offset 112
16 0004 FD7B06A9              stp     x29, x30, [sp, 96]
17                            .cfi_offset 29, -16
18                            .cfi_offset 30, -8
19 0008 FD830191              add     x29, sp, 96
20 000c E00700F9              str     x0, [sp, 8]
21 0010 E10300F9              str     x1, [sp]
22                            .loc 1 11 1
23 0014 00000090              adrp    x0, :got:__stack_chk_guard
24 0018 000040F9              ldr     x0, [x0, :got_lo12:__stack_chk_guard]
25 001c 010040F9              ldr     x1, [x0]
26 0020 E12F00F9              str     x1, [sp, 88]
27 0024 010080D2              mov     x1, 0
12:salsa20.cpp   ****         int i;
13:salsa20.cpp   ****         uint32_t x[16];
14:salsa20.cpp   ****
15:salsa20.cpp   ****         for (i = 0; i < 16; ++i)
28                            .loc 1 15 9
29 0028 FF1700B9              str     wzr, [sp, 20]
30                            .loc 1 15 2
31 002c 0D000014              b       .L2
```

Трохи розглянемо асемблерний код для aarch:

Опис salsa20_block:

```
10:salsa20.cpp   **** void salsa20_block(uint32_t out[16], uint32_t const in[16])
11:salsa20.cpp   **** {
12                       .loc 1 11 1
13                       .cfi_startproc
14 0000 FFC301D1         sub sp, sp, #112
15                       .cfi_def_cfa_offset 112
16 0004 FD7B06A9         stp x29, x30, [sp, 96]
17                       .cfi_offset 29, -16
18                       .cfi_offset 30, -8
19 0008 FD830191         add x29, sp, 96
20 000c E00700F9         str x0, [sp, 8]
21 0010 E10300F9         str x1, [sp]
22                       .loc 1 11 1
23 0014 00000090         adrp    x0, :got:__stack_chk_guard
24 0018 000040F9         ldr x0, [x0, :got_lo12:__stack_chk_guard]
25 001c 010040F9         ldr x1, [x0]
26 0020 E12F00F9         str x1, [sp, 88]
27 0024 010080D2         mov x1, 0
12:salsa20.cpp   ****     int i;
13:salsa20.cpp   ****     uint32_t x[16];
```

Зміщення:

```
20:salsa20.cpp   ****         QR(x[ 0], x[ 4], x[ 8], x[12]); // column 1
59                       .loc 1 20 3
60 0074 E12B40B9         ldr w1, [sp, 40]
61 0078 E21B40B9         ldr w2, [sp, 24]
62 007c E04B40B9         ldr w0, [sp, 72]
63 0080 4000000B         add w0, w2, w0
64 0084 00648013         ror w0, w0, 25
65 0088 2000004A         eor w0, w1, w0
66 008c E02B00B9         str w0, [sp, 40]
67 0090 E13B40B9         ldr w1, [sp, 56]
68 0094 E22B40B9         ldr w2, [sp, 40]
69 0098 E01B40B9         ldr w0, [sp, 24]
70 009c 4000000B         add w0, w2, w0
71 00a0 005C8013         ror w0, w0, 23
72 00a4 2000004A         eor w0, w1, w0
73 00a8 E03B00B9         str w0, [sp, 56]
74 00ac E14B40B9         ldr w1, [sp, 72]
75 00b0 E23B40B9         ldr w2, [sp, 56]
76 00b4 E02B40B9         ldr w0, [sp, 40]
77 00b8 4000000B         add w0, w2, w0
78 00bc 004C8013         ror w0, w0, 19
79 00c0 2000004A         eor w0, w1, w0
80 00c4 E04B00B9         str w0, [sp, 72]
81 00c8 E11B40B9         ldr w1, [sp, 24]
82 00cc E24B40B9         ldr w2, [sp, 72]
83 00d0 E03B40B9         ldr w0, [sp, 56]
84 00d4 4000000B         add w0, w2, w0
85 00d8 00388013         ror w0, w0, 14
86 00dc 2000004A         eor w0, w1, w0
87 00e0 E01B00B9         str w0, [sp, 24]
```

3. **Реалізація функцій стандартної бібліотеки C. Бібліотека за варіантами, функції всі зазначені (за наявності реалізації), версія бібліотеки остання стабільна на момент початку курсу: uClibc-ng**

   Функції:

– стандартного ввод-виводу printf, puts;

– роботи з файлами fopen, fread, fwrite, feof, fclose;

– виконання команд операційної системи system.

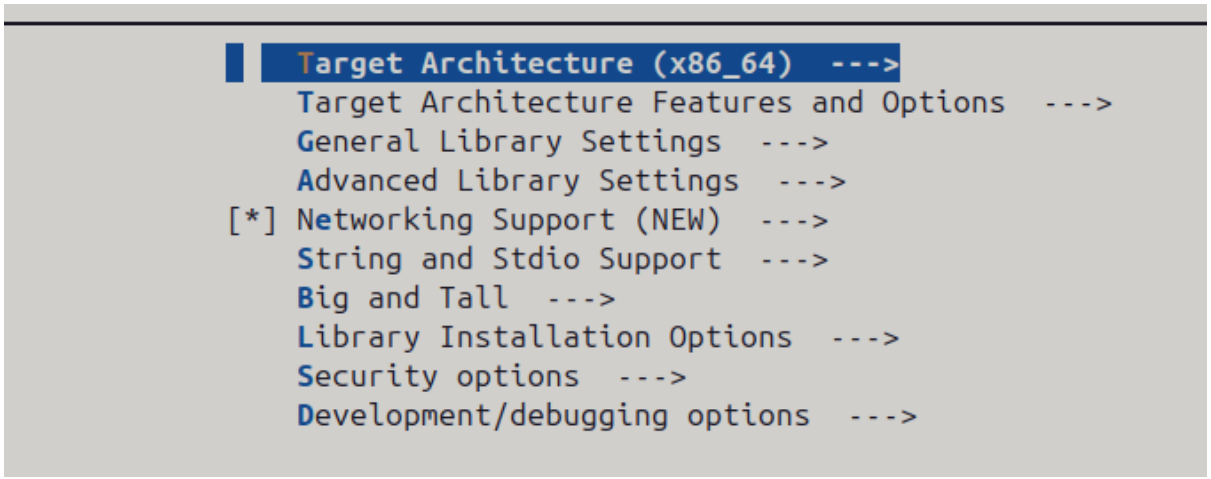
Зверніть увагу на відмінності системних викликів у різних ОС Linux та Windows для різних архітектур (x86, x86_64, ARM)

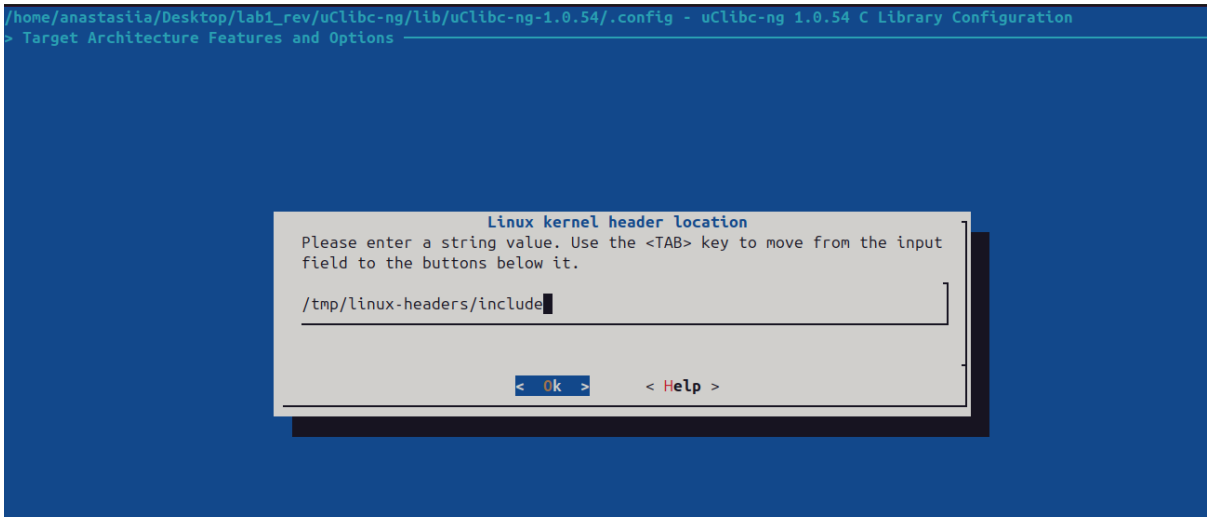
Встановила останю версію uClibc-ng:

```
anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib$ wget https://downloads.uclibc-ng.org/releases/1.0.54/uClibc-ng-1.0.54.tar.xz
--2025-09-14 16:37:58--  https://downloads.uclibc-ng.org/releases/1.0.54/uClibc-ng-1.0.54.tar.xz
Resolving downloads.uclibc-ng.org (downloads.uclibc-ng.org)... 89.238.66.15, 2a00:1828:2000:679::23
Connecting to downloads.uclibc-ng.org (downloads.uclibc-ng.org)|89.238.66.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1986084 (1.9M) [application/x-xz]
Saving to: ‘uClibc-ng-1.0.54.tar.xz’

uClibc-ng-1.0.54.tar.xz           100%[=============================================================>]   1.89M  6.03MB/s    in 0.3s

2025-09-14 16:37:59 (6.03 MB/s) - ‘uClibc-ng-1.0.54.tar.xz’ saved [1986084/1986084]

anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib$ ls
uClibc-ng-1.0.54.tar.xz
```

Налаштовую компіляцію бібліотеки для архітектури x86_64 (відповідно вказую архітектуру та шлях до заголовків). Зробила лише для однієї архітектури, бо для всіх інших не знайшла способу, вказувало на відсутність limits.h, хоча насправді він був:

```
anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib/uClibc-ng-1.0.54$ make menuconfig
```





printf (https://github.com/wbx-github/uclibc-ng/blob/master/libc/stdio/printf.c):

```
anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib/uClibc-ng-1.0.54$ objdump -d libc/stdio/printf.os > printf.lst
```

```
libc/stdio/printf.os:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <__GI_printf>:
   0:   f3 0f 1e fa             endbr64
   4:   48 81 ec d8 00 00 00    sub    $0xd8,%rsp
   b:   48 89 74 24 28          mov    %rsi,0x28(%rsp)
  10:   48 89 54 24 30          mov    %rdx,0x30(%rsp)
  15:   48 89 4c 24 38          mov    %rcx,0x38(%rsp)
  1a:   4c 89 44 24 40          mov    %r8,0x40(%rsp)
  1f:   4c 89 4c 24 48          mov    %r9,0x48(%rsp)
  24:   84 c0                   test   %al,%al
  26:   74 37                   je     5f <__GI_printf+0x5f>
  28:   0f 29 44 24 50          movaps %xmm0,0x50(%rsp)
  2d:   0f 29 4c 24 60          movaps %xmm1,0x60(%rsp)
  32:   0f 29 54 24 70          movaps %xmm2,0x70(%rsp)
  37:   0f 29 9c 24 80 00 00    movaps %xmm3,0x80(%rsp)
  3e:   00
  3f:   0f 29 a4 24 90 00 00    movaps %xmm4,0x90(%rsp)
  46:   00
  47:   0f 29 ac 24 a0 00 00    movaps %xmm5,0xa0(%rsp)
  4e:   00
  4f:   0f 29 b4 24 b0 00 00    movaps %xmm6,0xb0(%rsp)
  56:   00
```

puts (https://github.com/wbx-github/uclibc-ng/blob/master/libc/stdio/puts.c):

```
anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib/uClibc-ng-1.0.54$ find . -name "puts.os"
./libc/stdio/puts.os
anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib/uClibc-ng-1.0.54$ objdump -d libc/stdio/puts.os  > puts.x86_64.lst
```

```
libc/stdio/puts.os:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <puts>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   %rbp
   5:   53                      push   %rbx
   6:   51                      push   %rcx
   7:   48 8b 05 00 00 00 00    mov    0x0(%rip),%rax        # e <puts+0xe>
   e:   48 8b 28                mov    (%rax),%rbp
  11:   48 89 ee                mov    %rbp,%rsi
  14:   e8 00 00 00 00          call   19 <puts+0x19>
  19:   83 f8 ff                cmp    $0xffffffff,%eax
  1c:   75 05                   jne    23 <puts+0x23>
  1e:   83 cb ff                or     $0xffffffff,%ebx
  21:   eb 14                   jmp    37 <puts+0x37>
  23:   48 89 ee                mov    %rbp,%rsi
  26:   bf 0a 00 00 00          mov    $0xa,%edi
  2b:   8d 58 01                lea    0x1(%rax),%ebx
  2e:   e8 00 00 00 00          call   33 <puts+0x33>
  33:   ff c0                   inc    %eax
  35:   74 e7                   je     1e <puts+0x1e>
  37:   89 d8                   mov    %ebx,%eax
  39:   5a                      pop    %rdx
```

fopen (https://github.com/wbx-github/uclibc-ng/blob/master/libc/stdio/fopen.c):

```
anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib/uClibc-ng-1.0.54$ objdump -d libc/stdio/fopen.os  > fopen.os.x86_64.lst
anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib/uClibc-ng-1.0.54$ mv fopen.os.x86_64.lst ~/Desktop/lab1_rev/uClibc-ng/
```

```
 GNU nano 7.2                        /home/anastasiia/Desktop/lab1_rev/uClibc-ng/fopen.os.x86_64.lst

libc/stdio/fopen.os:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <__GI_fopen>:
   0:   f3 0f 1e fa             endbr64
   4:   83 c9 ff                or     $0xffffffff,%ecx
   7:   31 d2                   xor    %edx,%edx
   9:   e9 00 00 00 00          jmp    e <__GI_fopen+0xe>


Trash
```

fread (https://github.com/wbx-github/uclibc-ng/blob/master/libc/stdio/fread.c):

```
anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib/uClibc-ng-1.0.54$ find . -name "fread.os"
./libc/stdio/fread.os
anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib/uClibc-ng-1.0.54$ objdump -d libc/stdio/fread.os > fread.os.x86_64.lst
```

```
anastasiia@anastasiia: ~/Desktop/lab1_rev/uClibc-ng/lib/uClibc-ng-1.0.54
 GNU nano 7.2                                                fread.os.x86_64.lst

libc/stdio/fread.os:     file format elf64-x86-64
```

fwrite (https://github.com/wbx-github/uclibc-ng/blob/master/libc/stdio/fwrite.c):

```
anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib/uClibc-ng-1.0.54$ find . -name "fwrite.os"
./libc/stdio/fwrite.os
anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib/uClibc-ng-1.0.54$ objdump -d libc/stdio/fwrite.os > fwrite.os.x86_64.lst
```

```
 GNU nano 7.2                                              fwrite.os.x86_64.lst

libc/stdio/fwrite.os:     file format elf64-x86-64
```

feof (https://github.com/wbx-github/uclibc-ng/blob/master/libc/stdio/feof.c):

```
anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib/uClibc-ng-1.0.54$ find . -name "feof.os"
./libc/stdio/feof.os
anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib/uClibc-ng-1.0.54$ objdump -d libc/stdio/feof.os > feof.os.x86_64.lst
```

```
 GNU nano 7.2                                                feof.os.x86_64.lst

libc/stdio/feof.os:     file format elf64-x86-64
```

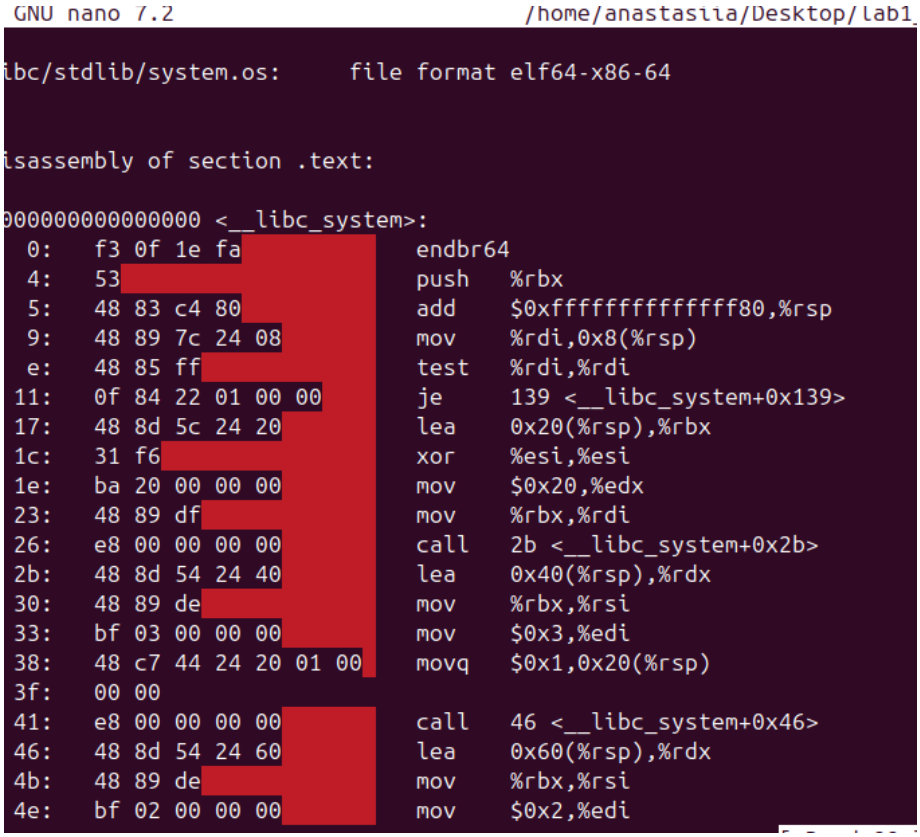
fclose (https://github.com/wbx-github/uclibc-ng/blob/master/libc/stdio/fclose.c):

```
anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib/uClibc-ng-1.0.54$ find . -name "fclose.os"
./libc/stdio/fclose.os
anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib/uClibc-ng-1.0.54$ objdump -d libc/stdio/fclose.os > fclose.os.x86_64.lst
```

```
libc/stdio/fclose.os:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <__GI_fclose>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   %rbp
   5:   53                      push   %rbx
   6:   48 89 fb                mov    %rdi,%rbx
   9:   51                      push   %rcx
   a:   48 8b 15 00 00 00 00    mov    0x0(%rip),%rdx        # 11 <__GI_fclose+0x11>
  11:   48 8b 02                mov    (%rdx),%rax
  14:   48 39 f8                cmp    %rdi,%rax
  17:   75 0c                   jne    25 <__GI_fclose+0x25>
  19:   48 8b 47 38             mov    0x38(%rdi),%rax
  1d:   48 89 02                mov    %rax,(%rdx)
  20:   eb 19                   jmp    3b <__GI_fclose+0x3b>
  22:   48 89 d0                mov    %rdx,%rax
  25:   48 85 c0                test   %rax,%rax
  28:   74 11                   je     3b <__GI_fclose+0x3b>
  2a:   48 8b 50 38             mov    0x38(%rax),%rdx
  2e:   48 39 da                cmp    %rbx,%rdx
  31:   75 ef                   jne    22 <__GI_fclose+0x22>
  33:   48 8b 53 38             mov    0x38(%rbx),%rdx
  37:   48 89 50 38             mov    %rdx,0x38(%rax)
```

system (https://github.com/wbx-github/uclibc-ng/blob/master/libc/stdlib/system.c):

```
anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib/uClibc-ng-1.0.54$ find . -name "system.os"
./libc/stdlib/system.os
anastasiia@anastasiia:~/Desktop/lab1_rev/uClibc-ng/lib/uClibc-ng-1.0.54$ objdump -d libc/stdlib/system.os  > system.x86_64.lst
```



Загалом бібліотека uClibc-ng це компактна реалізація стандартної бібліотеки C. На скріншотах видно що деякі функції не вийшло компіллювати, лише виводить заголовок. Загалом всі асемблерні коди вище складаються з:

- Секцій (.text, .data...)
- Заголовків
- Імена функцій/змінних (__GI_fopen, __libc_system, __GI_fclose...) — вказують на позиції в коді/даних.
- Інструкцій (mov, jmp, call, add, sub...)
- Виклики зовнішніх функцій

Висновок: один з висновків це встановлювати останню версію/сумісну з програмою. Бо довгий час думала чому не виходить зробити перше завдання, виявляється встановила компоненти 2017 року. Навчилась компілювати файли/функції з бібліотеки (amd) на ОС Windows/Linux, порозбиралась в асеблерному коді, базові нобори інструкцій зрозуміла. Описала різницю між компілюваним кодами на архітектурах AMD64, ARM, ARM64.