Міністерство освіти і науки України
Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"
Фізико-технічний інститут

ЗВОРОТНА РОЗРОБКА ТА АНАЛІЗ ШКІДЛИВОГО ЗАБЕЗПЕЧЕННЯ

Лабораторна робота №1
Аналіз програмного коду мов високого рівня
Варіант 18

Виконала:
студентка 3 курсу
гр. ФБ-92
Шатковська Діана

Перевірив:
Якобчук Д.І.

Київ - 2021

Аналіз програмного коду мов високого рівня

Мета роботи:

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

Хід роботи

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/);

Для зміни архітектури для компіляції програми на Windows було використано скрипт-файл Visual Studio `vcvarsall.bat` із передачею аргументу, що визначає відповідну архітектуру(`amd64`,`x86_arm`, `x86_arm64`). Було виконано такі команди у cmd(із відповідними аргументами):

```
C:\Program Files (x86)\Microsoft Visual
Studio\2019\Community\VC\Auxiliary\Build>vcvarsall.bat amd64
**********************************************************************
** Visual Studio 2019 Developer Command Prompt v16.9.3
** Copyright (c) 2021 Microsoft Corporation
**********************************************************************
[vcvarsall.bat] Environment initialized for: 'x64'

D:\Study\5sem\RE\lab1>cl /FAcsu hanoi.c
Microsoft (R) C/C++ Optimizing Compiler Version 19.28.29913 for x64
Copyright (C) Microsoft Corporation.  All rights reserved.

hanoi.c
Microsoft (R) Incremental Linker Version 14.28.29913.0
Copyright (C) Microsoft Corporation.  All rights reserved.

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

На виході було отримано файли розширення .cod, перейменовані для зручності як `hanoi(arm).cod`, `hanoi(x64).cod`, `hanoi(arm64).cod`, у яких міститься машинний код скомпільованої програми у відповідних архітектурах(файли додаються).

У Linux необхідно було спочатку завантажити необхідне ПЗ. Результати компіляції були переправлені у файли .lst із машинним кодом.

Розглянемо результати та особливо звернемо увагу на опис функції hanoi().

Windows:

- x64

Тут ми бачимо декілька стовпців з даними: перший(наприклад: `00005`) - адреса інструкції; другий(`44 88 4c 24 20`) - бінарний код, який виконує процесор(для зручності читання записаний в хексі), з нього можна зрозуміти розмір команди(кожна пара - байт пам'яті, тобто дана команда має розмір в 5 байтів); третій стовпчик(`mov BYTE PTR [rsp+32], r9b`) - відповідна команда на асемблері та її параметри.

Архітектура x64 оперує такими регістрами:

8-byte register	Bytes 0-3	Bytes 0-1	Byte 0
%rax	%eax	%ax	%al
%rcx	%ecx	%cx	%cl
%rdx	%edx	%dx	%dl
%rbx	%ebx	%bx	%bl
%rsi	%esi	%si	%sil
%rdi	%edi	%di	%dil
%rsp	%esp	%sp	%spl
%rbp	%ebp	%bp	%bpl
%r8	%r8d	%r8w	%r8b
%r9	%r9d	%r9w	%r9b
%r10	%r10d	%r10w	%r10b
%r11	%r11d	%r11w	%r11b
%r12	%r12d	%r12w	%r12b
%r13	%r13d	%r13w	%r13b
%r14	%r14d	%r14w	%r14b
%r15	%r15d	%r15w	%r15b

За домовленістю, rax використовується для збереження виводу функції, якщо цей вивід існує та лише якщо він менше 64 бітів в розмірі(більші типи повертаються за допомогою стеку).

Регістри rbx, rbp, r12-r15 можуть використовуватись серед викликів функцій.

rsp використовується як вказівник стеку, що вказує на верхній елемент стеку.

Також rdi, rsi, rdx, rcx, r8, r9 використовуються, щоб передати перші шість параметрів до викликаної функції(всі інші параметри передаються за допомогою стеку).

На відміну від x86, у x64 ebp не використовується(або використовується в особливих випадках) для збереження значення основи поточного стек-фрейму, так як у x64 почали використовувати так звану динамічну алокацію стеку.

Також використовуються такі типи даних:

- "byte" - one-byte integer (suffix b),
- "word" - two-byte integer (suffix w),
- "doubleword" - four-byte integer (suffix l)
- "quadword" - eight-byte value (suffix q).

Та основні команди: mov - занесення значення у регістр, add/sub - додавання/віднімання значень із записом в останній, cmp - порівняння значень із модифікацією відповідного флагу, jne(та інші види умовних переходів) - відповідно до вмісту відповідного флагу виконує\не виконує перехід на вказану адресу(команду, або лейбл), movzx/movsx - переносить значення із розширенням розрядності/з урахуванням знаку, lea - записує адресу другого параметру в перший, тощо.

https://cs.brown.edu/courses/cs033/docs/guides/x64_cheatsheet.pdf

```
…
; Function compile flags: /Odtp
; File D:\Study\5sem\RE\lab1\hanoi.c
_TEXT SEGMENT
n$ = 48
from$ = 56
to$ = 64
aux$ = 72
hanoi PROC

; 4    : {

$LN5:
  00000     44 88 4c 24 20      mov   BYTE PTR [rsp+32], r9b
  00005     44 88 44 24 18      mov   BYTE PTR [rsp+24], r8b
  0000a     88 54 24 10  mov   BYTE PTR [rsp+16], dl
  0000e     89 4c 24 08  mov   DWORD PTR [rsp+8], ecx
  00012     48 83 ec 28  sub   rsp, 40                ; 00000028H

; 5    :     if (n == 1)

  00016     83 7c 24 30 01      cmp   DWORD PTR n$[rsp], 1
  0001b     75 1d         jne   SHORT $LN2@hanoi

; 6    :         printf("Move disk 1 from %c to %c\n", from, to);

  0001d     0f be 44 24 40      movsx eax, BYTE PTR to$[rsp]
  00022     0f be 4c 24 38      movsx ecx, BYTE PTR from$[rsp]
  00027     44 8b c0      mov   r8d, eax
  0002a     8b d1         mov   edx, ecx
  0002c     48 8d 0d 00 00
      00 00         lea   rcx, OFFSET FLAT:$SG8992
  00033     e8 00 00 00 00      call  printf
  00038     eb 5c         jmp   SHORT $LN3@hanoi
$LN2@hanoi:

; 7    :     else
; 8    :     {
; 9    :         hanoi(n - 1, from, aux, to);

  0003a     8b 44 24 30  mov   eax, DWORD PTR n$[rsp]
  0003e     ff c8         dec   eax
  00040     44 0f b6 4c 24
      40            movzx r9d, BYTE PTR to$[rsp]
  00046     44 0f b6 44 24
      48            movzx r8d, BYTE PTR aux$[rsp]
  0004c     0f b6 54 24 38      movzx edx, BYTE PTR from$[rsp]
  00051     8b c8         mov   ecx, eax
  00053     e8 00 00 00 00      call  hanoi

; 10   :         printf("Move disk %d from %c to %c\n", n, from, to);

  00058     0f be 44 24 40      movsx eax, BYTE PTR to$[rsp]
  0005d     0f be 4c 24 38      movsx ecx, BYTE PTR from$[rsp]
  00062     44 8b c8      mov   r9d, eax
  00065     44 8b c1      mov   r8d, ecx
  00068     8b 54 24 30  mov   edx, DWORD PTR n$[rsp]
  0006c     48 8d 0d 00 00
      00 00         lea   rcx, OFFSET FLAT:$SG8993
  00073     e8 00 00 00 00      call  printf
```

```
; 11   :         hanoi(n - 1, aux, to, from);

  00078     8b 44 24 30  mov   eax, DWORD PTR n$[rsp]
  0007c     ff c8        dec   eax
  0007e     44 0f b6 4c 24
      38           movzx r9d, BYTE PTR from$[rsp]
  00084     44 0f b6 44 24
      40           movzx r8d, BYTE PTR to$[rsp]
  0008a     0f b6 54 24 48     movzx edx, BYTE PTR aux$[rsp]
  0008f     8b c8        mov   ecx, eax
  00091     e8 00 00 00 00     call  hanoi
$LN3@hanoi:

; 12   :     }
; 13   : }

  00096     48 83 c4 28  add   rsp, 40                ; 00000028H
  0009a     c3           ret   0
hanoi ENDP
...
```

- ARM

На відміну від попередньої архітектури ARM - 32-бітна архітектура.
Серед відмінностей: тут використовуються 32-бітні регістри (ebp, esp тощо).

```
...
; Function compile flags: /Odtp
; File D:\Study\5sem\RE\lab1\hanoi.c
_TEXT SEGMENT
_n$ = 8                                   ; size = 4
_from$ = 12                               ; size = 1
_to$ = 16                                 ; size = 1
_aux$ = 20                                ; size = 1
_hanoi      PROC

; 4    : {

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

; 5    :     if (n == 1)

  00003     83 7d 08 01  cmp   DWORD PTR _n$[ebp], 1
  00007     75 19        jne   SHORT $LN2@hanoi

; 6    :         printf("Move disk 1 from %c to %c\n", from, to);

  00009     0f be 45 10  movsx eax, BYTE PTR _to$[ebp]
  0000d     50           push  eax
  0000e     0f be 4d 0c  movsx ecx, BYTE PTR _from$[ebp]
  00012     51           push  ecx
  00013     68 00 00 00 00     push  OFFSET $SG9258
  00018     e8 00 00 00 00     call  _printf
  0001d     83 c4 0c     add   esp, 12                ; 0000000cH
```

```
  00020     eb 57         jmp   SHORT $LN1@hanoi
$LN2@hanoi:

; 7    :     else
; 8    :     {
; 9    :         hanoi(n - 1, from, aux, to);

  00022     0f b6 55 10   movzx edx, BYTE PTR _to$[ebp]
  00026     52            push  edx
  00027     0f b6 45 14   movzx eax, BYTE PTR _aux$[ebp]
  0002b     50            push  eax
  0002c     0f b6 4d 0c   movzx ecx, BYTE PTR _from$[ebp]
  00030     51            push  ecx
  00031     8b 55 08      mov   edx, DWORD PTR _n$[ebp]
  00034     83 ea 01      sub   edx, 1
  00037     52            push  edx
  00038     e8 00 00 00 00      call  _hanoi
  0003d     83 c4 10      add   esp, 16                 ; 00000010H

; 10   :         printf("Move disk %d from %c to %c\n", n, from, to);

  00040     0f be 45 10   movsx eax, BYTE PTR _to$[ebp]
  00044     50            push  eax
  00045     0f be 4d 0c   movsx ecx, BYTE PTR _from$[ebp]
  00049     51            push  ecx
  0004a     8b 55 08      mov   edx, DWORD PTR _n$[ebp]
  0004d     52            push  edx
  0004e     68 00 00 00 00      push  OFFSET $SG9259
  00053     e8 00 00 00 00      call  _printf
  00058     83 c4 10      add   esp, 16                 ; 00000010H

; 11   :         hanoi(n - 1, aux, to, from);

  0005b     0f b6 45 0c   movzx eax, BYTE PTR _from$[ebp]
  0005f     50            push  eax
  00060     0f b6 4d 10   movzx ecx, BYTE PTR _to$[ebp]
  00064     51            push  ecx
  00065     0f b6 55 14   movzx edx, BYTE PTR _aux$[ebp]
  00069     52            push  edx
  0006a     8b 45 08      mov   eax, DWORD PTR _n$[ebp]
  0006d     83 e8 01      sub   eax, 1
  00070     50            push  eax
  00071     e8 00 00 00 00      call  _hanoi
  00076     83 c4 10      add   esp, 16                 ; 00000010H
$LN1@hanoi:

; 12   :     }
; 13   : }

  00079     5d            pop   ebp
  0007a     c3            ret   0
_hanoi ENDP
```

...

- ARM64

```
...
; Function compile flags: /Odtp
; File D:\Study\5sem\RE\lab1\hanoi.c
_TEXT SEGMENT
_n$ = 8                                         ; size = 4
_from$ = 12                                     ; size = 1
_to$ = 16                                       ; size = 1
_aux$ = 20                                      ; size = 1
_hanoi  PROC

; 4    : {

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

; 5    :     if (n == 1)

  00003     83 7d 08 01  cmp   DWORD PTR _n$[ebp], 1
  00007     75 19        jne   SHORT $LN2@hanoi

; 6    :         printf("Move disk 1 from %c to %c\n", from, to);

  00009     0f be 45 10  movsx eax, BYTE PTR _to$[ebp]
  0000d     50           push  eax
  0000e     0f be 4d 0c  movsx ecx, BYTE PTR _from$[ebp]
  00012     51           push  ecx
  00013     68 00 00 00 00     push  OFFSET $SG9258
  00018     e8 00 00 00 00     call  _printf
  0001d     83 c4 0c     add   esp, 12                ; 0000000cH
  00020     eb 57        jmp   SHORT $LN1@hanoi
$LN2@hanoi:

; 7    :     else
; 8    :     {
; 9    :         hanoi(n - 1, from, aux, to);

  00022     0f b6 55 10  movzx edx, BYTE PTR _to$[ebp]
  00026     52           push  edx
  00027     0f b6 45 14  movzx eax, BYTE PTR _aux$[ebp]
  0002b     50           push  eax
  0002c     0f b6 4d 0c  movzx ecx, BYTE PTR _from$[ebp]
  00030     51           push  ecx
  00031     8b 55 08     mov   edx, DWORD PTR _n$[ebp]
  00034     83 ea 01     sub   edx, 1
  00037     52           push  edx
  00038     e8 00 00 00 00     call  _hanoi
  0003d     83 c4 10     add   esp, 16                ; 00000010H
```

```
; 10   :         printf("Move disk %d from %c to %c\n", n, from, to);

  00040     0f be 45 10  movsx eax, BYTE PTR _to$[ebp]
  00044     50           push  eax
  00045     0f be 4d 0c  movsx ecx, BYTE PTR _from$[ebp]
  00049     51           push  ecx
  0004a     8b 55 08     mov   edx, DWORD PTR _n$[ebp]
  0004d     52           push  edx
  0004e     68 00 00 00 00     push  OFFSET $SG9259
  00053     e8 00 00 00 00     call  _printf
  00058     83 c4 10     add   esp, 16                ; 00000010H

; 11   :         hanoi(n - 1, aux, to, from);

  0005b     0f b6 45 0c  movzx eax, BYTE PTR _from$[ebp]
  0005f     50           push  eax
  00060     0f b6 4d 10  movzx ecx, BYTE PTR _to$[ebp]
  00064     51           push  ecx
  00065     0f b6 55 14  movzx edx, BYTE PTR _aux$[ebp]
  00069     52           push  edx
  0006a     8b 45 08     mov   eax, DWORD PTR _n$[ebp]
  0006d     83 e8 01     sub   eax, 1
  00070     50           push  eax
  00071     e8 00 00 00 00     call  _hanoi
  00076     83 c4 10     add   esp, 16                ; 00000010H
$LN1@hanoi:

; 12   :     }
; 13   : }

  00079     5d           pop   ebp
  0007a     c3           ret   0
_hanoiENDP
_TEXT ENDS

...
```

Linux:

```
diasha@diasha-lubuntu:~$ sudo apt install gcc gcc-i686-linux-gnu
gcc-arm-linux-gnueabi gcc-aarch64-linux-gnu binutils
binutils-arm-linux-gnueabi binutils-aarch64-linux-gnu
```

- amd64

```
diasha@diasha-lubuntu:~/RE_lab1$ gcc -Wa,-adhln -g hanoi.c > hanoi.amd64.lst

diasha@diasha-lubuntu:~/RE_lab1$ cat hanoi.amd64.lst
   1                            .file   "hanoi.c"
   2                            .text
   3                   .Ltext0:
```

```
   4                            .section        .rodata
   5                    .LC0:
   6 0000 4D6F7665              .string "Move disk 1 from %c to %c\n"
   6      20646973
   6      6B203120
   6      66726F6D
   6      20256320
   7                    .LC1:
   8 001b 4D6F7665              .string "Move disk %d from %c to %c\n"
   8      20646973
   8      6B202564
   8      2066726F
   8      6D202563
   9                            .text
  10                            .globl  hanoi
  12                    hanoi:
  13                    .LFB0:
  14                            .file 1 "hanoi.c"
   1:hanoi.c       **** #include <stdio.h>
   2:hanoi.c       ****
   3:hanoi.c       **** void hanoi(int n, char from, char to, char aux)
   4:hanoi.c       **** {
  15                            .loc 1 4 1
  16                            .cfi_startproc
  17 0000 F30F1EFA              endbr64
  18 0004 55                    pushq   %rbp
  19                            .cfi_def_cfa_offset 16
  20                            .cfi_offset 6, -16
  21 0005 4889E5                movq    %rsp, %rbp
  22                            .cfi_def_cfa_register 6
  23 0008 4883EC10              subq    $16, %rsp
  24 000c 897DFC                movl    %edi, -4(%rbp)
  25 000f 89C8                  movl    %ecx, %eax
  26 0011 89F1                  movl    %esi, %ecx
  27 0013 884DF8                movb    %cl, -8(%rbp)
  28 0016 8855F4                movb    %dl, -12(%rbp)
  29 0019 8845F0                movb    %al, -16(%rbp)
   5:hanoi.c       ****     if (n == 1)
  30                            .loc 1 5 8
  31 001c 837DFC01              cmpl    $1, -4(%rbp)
  32 0020 751D                  jne     .L2
   6:hanoi.c       ****         printf("Move disk 1 from %c to %c\n", from,
to);
  33                            .loc 1 6 9
  34 0022 0FBE55F4              movsbl  -12(%rbp), %edx
  35 0026 0FBE45F8              movsbl  -8(%rbp), %eax
  36 002a 89C6                  movl    %eax, %esi
  37 002c 488D3D00              leaq    .LC0(%rip), %rdi
  37      000000
  38 0033 B8000000              movl    $0, %eax
  38      00
  39 0038 E8000000              call    printf@PLT
  39      00
```

```
   7:hanoi.c       ****     else
   8:hanoi.c       ****     {
   9:hanoi.c       ****         hanoi(n - 1, from, aux, to);
  10:hanoi.c       ****         printf("Move disk %d from %c to %c\n", n,
from, to);
  11:hanoi.c       ****         hanoi(n - 1, aux, to, from);
  12:hanoi.c       ****     }
  13:hanoi.c       **** }
  40                            .loc 1 13 1
  41 003d EB50                  jmp     .L4
  42                    .L2:
   9:hanoi.c       ****         printf("Move disk %d from %c to %c\n", n,
from, to);
  43                            .loc 1 9 9
  44 003f 0FBE4DF4              movsbl  -12(%rbp), %ecx
  45 0043 0FBE55F0              movsbl  -16(%rbp), %edx
  46 0047 0FBE45F8              movsbl  -8(%rbp), %eax
  47 004b 8B75FC                movl    -4(%rbp), %esi
  48 004e 8D7EFF                leal    -1(%rsi), %edi
  49 0051 89C6                  movl    %eax, %esi
  50 0053 E8000000              call    hanoi
  50      00
  10:hanoi.c       ****         hanoi(n - 1, aux, to, from);
  51                            .loc 1 10 9
  52 0058 0FBE4DF4              movsbl  -12(%rbp), %ecx
  53 005c 0FBE55F8              movsbl  -8(%rbp), %edx
  54 0060 8B45FC                movl    -4(%rbp), %eax
  55 0063 89C6                  movl    %eax, %esi
  56 0065 488D3D00              leaq    .LC1(%rip), %rdi
  56      000000
  57 006c B8000000              movl    $0, %eax
  57      00
  58 0071 E8000000              call    printf@PLT
  58      00
  11:hanoi.c       ****     }
  59                            .loc 1 11 9
  60 0076 0FBE4DF8              movsbl  -8(%rbp), %ecx
  61 007a 0FBE55F4              movsbl  -12(%rbp), %edx
  62 007e 0FBE45F0              movsbl  -16(%rbp), %eax
  63 0082 8B75FC                movl    -4(%rbp), %esi
  64 0085 8D7EFF                leal    -1(%rsi), %edi
  65 0088 89C6                  movl    %eax, %esi
  66 008a E8000000              call    hanoi
  66      00
  67                    .L4:
  68                            .loc 1 13 1
  69 008f 90                    nop
  70 0090 C9                    leave
  71                            .cfi_def_cfa 7, 8
  72 0091 C3                    ret
  73                            .cfi_endproc
  74                    .LFE0:
  76                            .globl  main
```

```
 78                     main:
 79                     .LFB1:
 14:hanoi.c       ****
 15:hanoi.c       **** int main()
 16:hanoi.c       **** {
 80                             .loc 1 16 1
 81                             .cfi_startproc
 82 0092 F30F1EFA               endbr64
 83 0096 55                     pushq   %rbp
 84                             .cfi_def_cfa_offset 16
 85                             .cfi_offset 6, -16
 86 0097 4889E5                 movq    %rsp, %rbp
 87                             .cfi_def_cfa_register 6
 17:hanoi.c       ****     hanoi(3, 'A', 'C', 'B');
 88                             .loc 1 17 5
 89 009a B9420000               movl    $66, %ecx
 89      00
 90 009f BA430000               movl    $67, %edx
 90      00
 91 00a4 BE410000               movl    $65, %esi
 91      00
 92 00a9 BF030000               movl    $3, %edi
 92      00
 93 00ae E8000000               call    hanoi
 93      00
 94 00b3 B8000000               movl    $0, %eax
 94      00
 18:hanoi.c       **** }
 95                             .loc 1 18 1
 96 00b8 5D                     popq    %rbp
 97                             .cfi_def_cfa 7, 8
 98 00b9 C3                     ret
 99                             .cfi_endproc
100                     .LFE1:
102                     .Letext0:
383                             .section        .note.gnu.property,"a"
384                             .align 8
385 0000 04000000               .long    1f - 0f
386 0004 10000000               .long    4f - 1f
387 0008 05000000               .long    5
388                     0:
389 000c 474E5500               .string  "GNU"
390                     1:
391                             .align 8
392 0010 020000C0               .long    0xc0000002
393 0014 04000000               .long    3f - 2f
394                     2:
395 0018 03000000               .long    0x3
396                     3:
397 001c 00000000               .align 8
398                     4:
```

- arm

Використовуються такі регістри:

- 13 general-purpose registers R0-R12.
- One Stack Pointer (SP).
- One Link Register (LR).
- One Program Counter (PC).
- One Application Program Status Register (APSR)

http://www.cburch.com/cs/230/arm-ref.pdf

```
diasha@diasha-lubuntu:~/RE_lab1$ arm-linux-gnueabi-gcc -Wa,-adhln -g hanoi.c
> hanoi.arm.lst

diasha@diasha-lubuntu:~/RE_lab1$ cat hanoi.arm.lst
   1                            .arch armv5t
   2                            .eabi_attribute 20, 1
   3                            .eabi_attribute 21, 1
   4                            .eabi_attribute 23, 3
   5                            .eabi_attribute 24, 1
   6                            .eabi_attribute 25, 1
   7                            .eabi_attribute 26, 2
   8                            .eabi_attribute 30, 6
   9                            .eabi_attribute 34, 0
  10                            .eabi_attribute 18, 4
  11                            .file   "hanoi.c"
  12                            .text
  13                    .Ltext0:
  14                            .cfi_sections   .debug_frame
  15                            .section        .rodata
  16                            .align  2
  17                    .LC0:
  18 0000 4D6F7665              .ascii  "Move disk 1 from %c to %c\012\000"
  18      20646973
  18      6B203120
  18      66726F6D
  18      20256320
  19 001b 00                    .align  2
  20                    .LC1:
  21 001c 4D6F7665              .ascii  "Move disk %d from %c to %c\012\000"
  21      20646973
  21      6B202564
  21      2066726F
  21      6D202563
  22                            .text
  23                            .align  2
  24                            .global hanoi
  25                            .syntax unified
  26                            .arm
  27                            .fpu softvfp
  29                    hanoi:
  30                    .LFB0:
  31                            .file 1 "hanoi.c"
```

```
   1:hanoi.c       **** #include <stdio.h>
   2:hanoi.c       ****
   3:hanoi.c       **** void hanoi(int n, char from, char to, char aux)
   4:hanoi.c       **** {
  32                            .loc 1 4 1
  33                            .cfi_startproc
  34                            @ args = 0, pretend = 0, frame = 8
  35                            @ frame_needed = 1, uses_anonymous_args = 0
  36 0000 00482DE9              push    {fp, lr}
  37                            .cfi_def_cfa_offset 8
  38                            .cfi_offset 11, -8
  39                            .cfi_offset 14, -4
  40 0004 04B08DE2              add     fp, sp, #4
  41                            .cfi_def_cfa 11, 4
  42 0008 08D04DE2              sub     sp, sp, #8
  43 000c 08000BE5              str     r0, [fp, #-8]
  44 0010 0100A0E1              mov     r0, r1
  45 0014 0210A0E1              mov     r1, r2
  46 0018 0320A0E1              mov     r2, r3
  47 001c 0030A0E1              mov     r3, r0
  48 0020 09304BE5              strb    r3, [fp, #-9]
  49 0024 0130A0E1              mov     r3, r1
  50 0028 0A304BE5              strb    r3, [fp, #-10]
  51 002c 0230A0E1              mov     r3, r2
  52 0030 0B304BE5              strb    r3, [fp, #-11]
   5:hanoi.c       ****     if (n == 1)
  53                            .loc 1 5 8
  54 0034 08301BE5              ldr     r3, [fp, #-8]
  55 0038 010053E3              cmp     r3, #1
  56 003c 0500001A              bne     .L2
   6:hanoi.c       ****         printf("Move disk 1 from %c to %c\n", from,
to);
  57                            .loc 1 6 9
  58 0040 09305BE5              ldrb    r3, [fp, #-9]   @ zero_extendqisi2
  59 0044 0A205BE5              ldrb    r2, [fp, #-10]  @ zero_extendqisi2
  60 0048 0310A0E1              mov     r1, r3
  61 004c 54009FE5              ldr     r0, .L5
  62 0050 FEFFFFEB              bl      printf
   7:hanoi.c       ****     else
   8:hanoi.c       ****     {
   9:hanoi.c       ****         hanoi(n - 1, from, aux, to);
  10:hanoi.c       ****         printf("Move disk %d from %c to %c\n", n,
from, to);
  11:hanoi.c       ****         hanoi(n - 1, aux, to, from);
  12:hanoi.c       ****     }
  13:hanoi.c       **** }
  63                            .loc 1 13 1
  64 0054 100000EA              b       .L4
  65                        .L2:
   9:hanoi.c       ****         printf("Move disk %d from %c to %c\n", n,
from, to);
  66                            .loc 1 9 9
  67 0058 08301BE5              ldr     r3, [fp, #-8]
```

```
 68 005c 010043E2              sub     r0, r3, #1
 69 0060 0A305BE5              ldrb    r3, [fp, #-10]  @ zero_extendqisi2
 70 0064 0B205BE5              ldrb    r2, [fp, #-11]  @ zero_extendqisi2
 71 0068 09105BE5              ldrb    r1, [fp, #-9]   @ zero_extendqisi2
 72 006c FEFFFFEB              bl      hanoi
 10:hanoi.c       ****         hanoi(n - 1, aux, to, from);
 73                            .loc 1 10 9
 74 0070 09205BE5              ldrb    r2, [fp, #-9]   @ zero_extendqisi2
 75 0074 0A305BE5              ldrb    r3, [fp, #-10]  @ zero_extendqisi2
 76 0078 08101BE5              ldr     r1, [fp, #-8]
 77 007c 28009FE5              ldr     r0, .L5+4
 78 0080 FEFFFFEB              bl      printf
 11:hanoi.c       ****     }
 79                            .loc 1 11 9
 80 0084 08301BE5              ldr     r3, [fp, #-8]
 81 0088 010043E2              sub     r0, r3, #1
 82 008c 09305BE5              ldrb    r3, [fp, #-9]   @ zero_extendqisi2
 83 0090 0A205BE5              ldrb    r2, [fp, #-10]  @ zero_extendqisi2
 84 0094 0B105BE5              ldrb    r1, [fp, #-11]  @ zero_extendqisi2
 85 0098 FEFFFFEB              bl      hanoi
 86                        .L4:
 87                            .loc 1 13 1
 88 009c 0000A0E1              nop
 89 00a0 04D04BE2              sub     sp, fp, #4
 90                            .cfi_def_cfa 13, 8
 91                            @ sp needed
 92 00a4 0088BDE8              pop     {fp, pc}
 93                        .L6:
 94                            .align  2
 95                        .L5:
 96 00a8 00000000              .word   .LC0
 97 00ac 1C000000              .word   .LC1
 98                            .cfi_endproc
 99                        .LFE0:
101                            .align  2
102                            .global main
103                            .syntax unified
104                            .arm
105                            .fpu softvfp
107                         main:
108                         .LFB1:
 14:hanoi.c       ****
 15:hanoi.c       **** int main()
 16:hanoi.c       **** {
109                            .loc 1 16 1
110                            .cfi_startproc
111                            @ args = 0, pretend = 0, frame = 0
112                            @ frame_needed = 1, uses_anonymous_args = 0
113 00b0 00482DE9              push    {fp, lr}
114                            .cfi_def_cfa_offset 8
115                            .cfi_offset 11, -8
116                            .cfi_offset 14, -4
117 00b4 04B08DE2              add     fp, sp, #4
```

```
 118                            .cfi_def_cfa 11, 4
  17:hanoi.c       ****     hanoi(3, 'A', 'C', 'B');
 119                            .loc 1 17 5
 120 00b8 4230A0E3              mov     r3, #66
 121 00bc 4320A0E3              mov     r2, #67
 122 00c0 4110A0E3              mov     r1, #65
 123 00c4 0300A0E3              mov     r0, #3
 124 00c8 FEFFFFEB              bl      hanoi
 125 00cc 0030A0E3              mov     r3, #0
  18:hanoi.c       **** }
 126                            .loc 1 18 1
 127 00d0 0300A0E1              mov     r0, r3
 128 00d4 0088BDE8              pop     {fp, pc}
 129                            .cfi_endproc
 130                    .LFE1:
 132                    .Letext0:
```

- arm64

Є 32 основних регістри в arm64, x0–x30 (64-bit). x31 - спеціальний регістр - вказівник на верхній елемент стеку.

- x0–x7: аргументи функцій, інші дані (x0 також вихідне значення функції)
- x8–x18: інші дані (x8 номер системного виклику, x16–x18 деколи зарезервовані)
- x19–x28: регістри “callee-saved” (зберігаються в стек під час виклику функцій, відновлюються зі стеку перед виходом з функції)
- x29: вказівник фрейму
- x30: link register (save to stack for non-leaf functions(та що в собі не викликає інших функцій))
- sp: stack pointer

https://cit.dixie.edu/cs/2810/arm64-assembly.html

Основні функції: stp - зберігає пару регістрів вираховуючи адресу в пам’яті та розмір, mov - записує значення в регістр, str - зберігає значення у вказану адресу, strb - зберігає байти регістру у вказану адресу, ldr - завантажує значення у регістр, cmp - порівнює параметри і записує результат у флаг, bne(і подібні умовні переходи) - перехід якщо не рівне(переглядає відповідний флаг).

```
diasha@diasha-lubuntu:~/RE_lab1$ aarch64-linux-gnu-gcc -Wa,-adhln -g hanoi.c
> hanoi.arch64.lst
diasha@diasha-lubuntu:~/RE_lab1$ cat hanoi.arch64.lst
   1                            .arch armv8-a
   2                            .file   "hanoi.c"
   3                            .text
   4                    .Ltext0:
   5                            .section        .rodata
   6                            .align  3
   7                    .LC0:
   8 0000 4D6F7665              .string "Move disk 1 from %c to %c\n"
   8      20646973
   8      6B203120
```

```
   8      66726F6D
   8      20256320
   9 001b 00000000              .align  3
   9      00
  10                    .LC1:
  11 0020 4D6F7665              .string "Move disk %d from %c to %c\n"
  11      20646973
  11      6B202564
  11      2066726F
  11      6D202563
  12                            .text
  13                            .align  2
  14                            .global hanoi
  16                    hanoi:
  17                    .LFB0:
  18                            .file 1 "hanoi.c"
   1:hanoi.c       **** #include <stdio.h>
   2:hanoi.c       ****
   3:hanoi.c       **** void hanoi(int n, char from, char to, char aux)
   4:hanoi.c       **** {
  19                            .loc 1 4 1
  20                            .cfi_startproc
  21 0000 FD7BBEA9              stp     x29, x30, [sp, -32]!
  22                            .cfi_def_cfa_offset 32
  23                            .cfi_offset 29, -32
  24                            .cfi_offset 30, -24
  25 0004 FD030091              mov     x29, sp
  26 0008 E01F00B9              str     w0, [sp, 28]
  27 000c E16F0039              strb    w1, [sp, 27]
  28 0010 E26B0039              strb    w2, [sp, 26]
  29 0014 E3670039              strb    w3, [sp, 25]
   5:hanoi.c       ****     if (n == 1)
  30                            .loc 1 5 8
  31 0018 E01F40B9              ldr     w0, [sp, 28]
  32 001c 1F040071              cmp     w0, 1
  33 0020 21010054              bne     .L2
   6:hanoi.c       ****         printf("Move disk 1 from %c to %c\n", from,
to);
  34                            .loc 1 6 9
  35 0024 E06F4039              ldrb    w0, [sp, 27]
  36 0028 E16B4039              ldrb    w1, [sp, 26]
  37 002c E203012A              mov     w2, w1
  38 0030 E103002A              mov     w1, w0
  39 0034 00000090              adrp    x0, .LC0
  40 0038 00000091              add     x0, x0, :lo12:.LC0
  41 003c 00000094              bl      printf
   7:hanoi.c       ****     else
   8:hanoi.c       ****     {
   9:hanoi.c       ****         hanoi(n - 1, from, aux, to);
  10:hanoi.c       ****         printf("Move disk %d from %c to %c\n", n,
from, to);
  11:hanoi.c       ****         hanoi(n - 1, aux, to, from);
  12:hanoi.c       ****     }
```

```
  13:hanoi.c       **** }
  42                            .loc 1 13 1
  43 0040 15000014              b       .L4
  44                        .L2:
   9:hanoi.c       ****         printf("Move disk %d from %c to %c\n", n,
from, to);
  45                            .loc 1 9 9
  46 0044 E01F40B9              ldr     w0, [sp, 28]
  47 0048 00040051              sub     w0, w0, #1
  48 004c E36B4039              ldrb    w3, [sp, 26]
  49 0050 E2674039              ldrb    w2, [sp, 25]
  50 0054 E16F4039              ldrb    w1, [sp, 27]
  51 0058 00000094              bl      hanoi
  10:hanoi.c       ****         hanoi(n - 1, aux, to, from);
  52                            .loc 1 10 9
  53 005c E06F4039              ldrb    w0, [sp, 27]
  54 0060 E16B4039              ldrb    w1, [sp, 26]
  55 0064 E303012A              mov     w3, w1
  56 0068 E203002A              mov     w2, w0
  57 006c E11F40B9              ldr     w1, [sp, 28]
  58 0070 00000090              adrp    x0, .LC1
  59 0074 00000091              add     x0, x0, :lo12:.LC1
  60 0078 00000094              bl      printf
  11:hanoi.c       ****     }
  61                            .loc 1 11 9
  62 007c E01F40B9              ldr     w0, [sp, 28]
  63 0080 00040051              sub     w0, w0, #1
  64 0084 E36F4039              ldrb    w3, [sp, 27]
  65 0088 E26B4039              ldrb    w2, [sp, 26]
  66 008c E1674039              ldrb    w1, [sp, 25]
  67 0090 00000094              bl      hanoi
  68                        .L4:
  69                            .loc 1 13 1
  70 0094 1F2003D5              nop
  71 0098 FD7BC2A8              ldp     x29, x30, [sp], 32
  72                            .cfi_restore 30
  73                            .cfi_restore 29
  74                            .cfi_def_cfa_offset 0
  75 009c C0035FD6              ret
  76                            .cfi_endproc
  77                        .LFE0:
  79                            .align  2
  80                            .global main
  82                        main:
  83                        .LFB1:
  14:hanoi.c       ****
  15:hanoi.c       **** int main()
  16:hanoi.c       **** {
  84                            .loc 1 16 1
  85                            .cfi_startproc
  86 00a0 FD7BBFA9              stp     x29, x30, [sp, -16]!
  87                            .cfi_def_cfa_offset 16
  88                            .cfi_offset 29, -16
```

```
  89                            .cfi_offset 30, -8
  90 00a4 FD030091              mov     x29, sp
  17:hanoi.c       ****     hanoi(3, 'A', 'C', 'B');
  91                            .loc 1 17 5
  92 00a8 43088052              mov     w3, 66
  93 00ac 62088052              mov     w2, 67
  94 00b0 21088052              mov     w1, 65
  95 00b4 60008052              mov     w0, 3
  96 00b8 00000094              bl      hanoi
  97 00bc 00008052              mov     w0, 0
  18:hanoi.c       **** }
  98                            .loc 1 18 1
  99 00c0 FD7BC1A8              ldp     x29, x30, [sp], 16
 100                            .cfi_restore 30
 101                            .cfi_restore 29
 102                            .cfi_def_cfa_offset 0
 103 00c4 C0035FD6              ret
 104                            .cfi_endproc
 105                    .LFE1:
 107                    .Letext0:
```

- LLVM clang
-

```
diasha@diasha-lubuntu:~/RE_lab1$ sudo bash -c "$(wget -O -
https://apt.llvm.org/llvm.sh)"
diasha@diasha-lubuntu:~/RE_lab1$ sudo apt install llvm
diasha@diasha-lubuntu:~/RE_lab1$ sudo apt install clang

diasha@diasha-lubuntu:~/RE_lab1$ clang -target amd64-pc-linux-gnu hanoi.c -o
hanoi-clang
diasha@diasha-lubuntu:~/RE_lab1$ llvm-objdump -d hanoi-clang >
hanoi-clang.cod
diasha@diasha-lubuntu:~/RE_lab1$ cat hanoi-clang.cod

hanoi-clang:    file format elf64-x86-64


Disassembly of section .init:

0000000000401000 <_init>:
  401000: f3 0f 1e fa                   endbr64
  401004: 48 83 ec 08                   subq    $8, %rsp
  401008: 48 8b 05 e9 2f 00 00          movq    12265(%rip), %rax  # 403ff8
<printf@@GLIBC_2.2.5+0x403ff8>
  40100f: 48 85 c0                      testq   %rax, %rax
  401012: 74 02                         je      0x401016 <_init+0x16>
  401014: ff d0                         callq   *%rax
  401016: 48 83 c4 08                   addq    $8, %rsp
  40101a: c3                            retq

Disassembly of section .plt:

0000000000401020 <.plt>:
```

```
  401020: ff 35 e2 2f 00 00             pushq   12258(%rip)  # 404008
<_GLOBAL_OFFSET_TABLE_+0x8>
  401026: ff 25 e4 2f 00 00             jmpq    *12260(%rip)  # 404010
<_GLOBAL_OFFSET_TABLE_+0x10>
  40102c: 0f 1f 40 00                   nopl    (%rax)

0000000000401030 <printf@plt>:
  401030: ff 25 e2 2f 00 00             jmpq    *12258(%rip)  # 404018
<_GLOBAL_OFFSET_TABLE_+0x18>
  401036: 68 00 00 00 00                pushq   $0
  40103b: e9 e0 ff ff ff                jmp     0x401020 <.plt>

Disassembly of section .text:

0000000000401040 <_start>:
  401040: f3 0f 1e fa                   endbr64
  401044: 31 ed                         xorl    %ebp, %ebp
  401046: 49 89 d1                      movq    %rdx, %r9
  401049: 5e                            popq    %rsi
  40104a: 48 89 e2                      movq    %rsp, %rdx
  40104d: 48 83 e4 f0                   andq    $-16, %rsp
  401051: 50                            pushq   %rax
  401052: 54                            pushq   %rsp
  401053: 49 c7 c0 70 12 40 00          movq    $4199024, %r8
  40105a: 48 c7 c1 00 12 40 00          movq    $4198912, %rcx
  401061: 48 c7 c7 d0 11 40 00          movq    $4198864, %rdi
  401068: ff 15 82 2f 00 00             callq   *12162(%rip)  # 403ff0
<printf@@GLIBC_2.2.5+0x403ff0>
  40106e: f4                            hlt
  40106f: 90                            nop

0000000000401070 <_dl_relocate_static_pie>:
  401070: f3 0f 1e fa                   endbr64
  401074: c3                            retq
  401075: 66 2e 0f 1f 84 00 00 00 00 00 nopw    %cs:(%rax,%rax)
  40107f: 90                            nop

0000000000401080 <deregister_tm_clones>:
  401080: b8 30 40 40 00                movl    $4210736, %eax
  401085: 48 3d 30 40 40 00             cmpq    $4210736, %rax
  40108b: 74 13                         je      0x4010a0
<deregister_tm_clones+0x20>
  40108d: b8 00 00 00 00                movl    $0, %eax
  401092: 48 85 c0                      testq   %rax, %rax
  401095: 74 09                         je      0x4010a0
<deregister_tm_clones+0x20>
  401097: bf 30 40 40 00                movl    $4210736, %edi
  40109c: ff e0                         jmpq    *%rax
  40109e: 66 90                         nop
  4010a0: c3                            retq
  4010a1: 66 66 2e 0f 1f 84 00 00 00 00 00      nopw    %cs:(%rax,%rax)
  4010ac: 0f 1f 40 00                   nopl    (%rax)
```

```
00000000004010b0 <register_tm_clones>:
  4010b0: be 30 40 40 00                movl    $4210736, %esi
  4010b5: 48 81 ee 30 40 40 00          subq    $4210736, %rsi
  4010bc: 48 89 f0                      movq    %rsi, %rax
  4010bf: 48 c1 ee 3f                   shrq    $63, %rsi
  4010c3: 48 c1 f8 03                   sarq    $3, %rax
  4010c7: 48 01 c6                      addq    %rax, %rsi
  4010ca: 48 d1 fe                      sarq    %rsi
  4010cd: 74 11                         je      0x4010e0
<register_tm_clones+0x30>
  4010cf: b8 00 00 00 00                movl    $0, %eax
  4010d4: 48 85 c0                      testq   %rax, %rax
  4010d7: 74 07                         je      0x4010e0
<register_tm_clones+0x30>
  4010d9: bf 30 40 40 00                movl    $4210736, %edi
  4010de: ff e0                         jmpq    *%rax
  4010e0: c3                            retq
  4010e1: 66 66 2e 0f 1f 84 00 00 00 00 00      nopw    %cs:(%rax,%rax)
  4010ec: 0f 1f 40 00                   nopl    (%rax)

00000000004010f0 <__do_global_dtors_aux>:
  4010f0: f3 0f 1e fa                   endbr64
  4010f4: 80 3d 35 2f 00 00 00          cmpb    $0, 12085(%rip)  # 404030
<completed.0>
  4010fb: 75 13                         jne     0x401110
<__do_global_dtors_aux+0x20>
  4010fd: 55                            pushq   %rbp
  4010fe: 48 89 e5                      movq    %rsp, %rbp
  401101: e8 7a ff ff ff                callq   0x401080
<deregister_tm_clones>
  401106: c6 05 23 2f 00 00 01          movb    $1, 12067(%rip)  # 404030
<completed.0>
  40110d: 5d                            popq    %rbp
  40110e: c3                            retq
  40110f: 90                            nop
  401110: c3                            retq
  401111: 66 66 2e 0f 1f 84 00 00 00 00 00      nopw    %cs:(%rax,%rax)
  40111c: 0f 1f 40 00                   nopl    (%rax)

0000000000401120 <frame_dummy>:
  401120: f3 0f 1e fa                   endbr64
  401124: eb 8a                         jmp     0x4010b0 <register_tm_clones>
  401126: 66 2e 0f 1f 84 00 00 00 00 00 nopw    %cs:(%rax,%rax)

0000000000401130 <hanoi>:
  401130: 55                            pushq   %rbp
  401131: 48 89 e5                      movq    %rsp, %rbp
  401134: 48 83 ec 10                   subq    $16, %rsp
  401138: 89 7d fc                      movl    %edi, -4(%rbp)
  40113b: 40 88 75 fb                   movb    %sil, -5(%rbp)
  40113f: 88 55 fa                      movb    %dl, -6(%rbp)
  401142: 88 4d f9                      movb    %cl, -7(%rbp)
  401145: 83 7d fc 01                   cmpl    $1, -4(%rbp)
```

```
  401149: 0f 85 1e 00 00 00            jne     0x40116d <hanoi+0x3d>
  40114f: 0f be 75 fb                  movsbl  -5(%rbp), %esi
  401153: 0f be 55 fa                  movsbl  -6(%rbp), %edx
  401157: 48 bf 04 20 40 00 00 00 00 00 movabsq $4202500, %rdi
  401161: b0 00                        movb    $0, %al
  401163: e8 c8 fe ff ff               callq   0x401030 <printf@plt>
  401168: e9 5d 00 00 00               jmp     0x4011ca <hanoi+0x9a>
  40116d: 8b 45 fc                     movl    -4(%rbp), %eax
  401170: 83 e8 01                     subl    $1, %eax
  401173: 8a 4d fb                     movb    -5(%rbp), %cl
  401176: 8a 55 f9                     movb    -7(%rbp), %dl
  401179: 89 c7                        movl    %eax, %edi
  40117b: 0f be f1                     movsbl  %cl, %esi
  40117e: 0f be d2                     movsbl  %dl, %edx
  401181: 0f be 4d fa                  movsbl  -6(%rbp), %ecx
  401185: e8 a6 ff ff ff               callq   0x401130 <hanoi>
  40118a: 8b 75 fc                     movl    -4(%rbp), %esi
  40118d: 0f be 55 fb                  movsbl  -5(%rbp), %edx
  401191: 0f be 4d fa                  movsbl  -6(%rbp), %ecx
  401195: 48 bf 1f 20 40 00 00 00 00 00 movabsq $4202527, %rdi
  40119f: b0 00                        movb    $0, %al
  4011a1: e8 8a fe ff ff               callq   0x401030 <printf@plt>
  4011a6: 8b 4d fc                     movl    -4(%rbp), %ecx
  4011a9: 83 e9 01                     subl    $1, %ecx
  4011ac: 44 8a 45 f9                  movb    -7(%rbp), %r8b
  4011b0: 44 8a 4d fa                  movb    -6(%rbp), %r9b
  4011b4: 89 cf                        movl    %ecx, %edi
  4011b6: 41 0f be f0                  movsbl  %r8b, %esi
  4011ba: 41 0f be d1                  movsbl  %r9b, %edx
  4011be: 0f be 4d fb                  movsbl  -5(%rbp), %ecx
  4011c2: 89 45 f4                     movl    %eax, -12(%rbp)
  4011c5: e8 66 ff ff ff               callq   0x401130 <hanoi>
  4011ca: 48 83 c4 10                  addq    $16, %rsp
  4011ce: 5d                           popq    %rbp
  4011cf: c3                           retq

00000000004011d0 <main>:
  4011d0: 55                           pushq   %rbp
  4011d1: 48 89 e5                     movq    %rsp, %rbp
  4011d4: bf 03 00 00 00               movl    $3, %edi
  4011d9: be 41 00 00 00               movl    $65, %esi
  4011de: ba 43 00 00 00               movl    $67, %edx
  4011e3: b9 42 00 00 00               movl    $66, %ecx
  4011e8: e8 43 ff ff ff               callq   0x401130 <hanoi>
  4011ed: 31 c0                        xorl    %eax, %eax
  4011ef: 5d                           popq    %rbp
  4011f0: c3                           retq
  4011f1: 66 2e 0f 1f 84 00 00 00 00 00 nopw    %cs:(%rax,%rax)
  4011fb: 0f 1f 44 00 00               nopl    (%rax,%rax)

0000000000401200 <__libc_csu_init>:
  401200: f3 0f 1e fa                  endbr64
  401204: 41 57                        pushq   %r15
```

```
  401206: 4c 8d 3d 03 2c 00 00          leaq    11267(%rip), %r15  # 403e10
<__init_array_start>
  40120d: 41 56                         pushq   %r14
  40120f: 49 89 d6                      movq    %rdx, %r14
  401212: 41 55                         pushq   %r13
  401214: 49 89 f5                      movq    %rsi, %r13
  401217: 41 54                         pushq   %r12
  401219: 41 89 fc                      movl    %edi, %r12d
  40121c: 55                            pushq   %rbp
  40121d: 48 8d 2d f4 2b 00 00          leaq    11252(%rip), %rbp  # 403e18
<__do_global_dtors_aux_fini_array_entry>
  401224: 53                            pushq   %rbx
  401225: 4c 29 fd                      subq    %r15, %rbp
  401228: 48 83 ec 08                   subq    $8, %rsp
  40122c: e8 cf fd ff ff                callq   0x401000 <_init>
  401231: 48 c1 fd 03                   sarq    $3, %rbp
  401235: 74 1f                         je      0x401256
<__libc_csu_init+0x56>
  401237: 31 db                         xorl    %ebx, %ebx
  401239: 0f 1f 80 00 00 00 00          nopl    (%rax)
  401240: 4c 89 f2                      movq    %r14, %rdx
  401243: 4c 89 ee                      movq    %r13, %rsi
  401246: 44 89 e7                      movl    %r12d, %edi
  401249: 41 ff 14 df                   callq   *(%r15,%rbx,8)
  40124d: 48 83 c3 01                   addq    $1, %rbx
  401251: 48 39 dd                      cmpq    %rbx, %rbp
  401254: 75 ea                         jne     0x401240
<__libc_csu_init+0x40>
  401256: 48 83 c4 08                   addq    $8, %rsp
  40125a: 5b                            popq    %rbx
  40125b: 5d                            popq    %rbp
  40125c: 41 5c                         popq    %r12
  40125e: 41 5d                         popq    %r13
  401260: 41 5e                         popq    %r14
  401262: 41 5f                         popq    %r15
  401264: c3                            retq
  401265: 66 66 2e 0f 1f 84 00 00 00 00 00      nopw    %cs:(%rax,%rax)

0000000000401270 <__libc_csu_fini>:
  401270: f3 0f 1e fa                   endbr64
  401274: c3                            retq

Disassembly of section .fini:

0000000000401278 <_fini>:
  401278: f3 0f 1e fa                   endbr64
  40127c: 48 83 ec 08                   subq    $8, %rsp
  401280: 48 83 c4 08                   addq    $8, %rsp
  401284: c3                            retq
```

Завдання 2

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

– Комбінаторні алгоритми, будь-який на Ваш вибір з вказаного класу:

19. сортування – злиттям;

Програма merge sort на мові c:

```
#include <stdio.h>
#include <stdlib.h>

void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;

    int L[n1], R[n2];

    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}
```

```
void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}

void printArray(int A[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", A[i]);
    printf("\n");
}

int main()
{
    int arr[] = { 12, 11, 13, 5, 6, 7 };
    int arr_size = sizeof(arr) / sizeof(arr[0]);

    printf("Given array is \n");
    printArray(arr, arr_size);

    mergeSort(arr, 0, arr_size - 1);

    printf("\nSorted array is \n");
    printArray(arr, arr_size);
    return 0;
}
```

Виконаємо компіляцію для потрібних платформ:

- i686

```
diasha@diasha-lubuntu:~/RE_lab1$ i686-linux-gnu-gcc -Wa,-adhln -g
merge_sort.c > merge_sort.i686.lst
```

Звернемо увагу на функцію merge() у цій архітектурі (повний код цього та інших результатів додається).

```
…
   4:merge_sort.c  **** void merge(int arr[], int l, int m, int r)
   5:merge_sort.c  **** {
   9                            .loc 1 5 1
  10                            .cfi_startproc
  11 0000 F30F1EFB              endbr32
  12 0004 55                    pushl   %ebp
  13                            .cfi_def_cfa_offset 8
  14                            .cfi_offset 5, -8
```

```
15 0005 89E5                movl    %esp, %ebp
16                          .cfi_def_cfa_register 5
17 0007 53                  pushl   %ebx
18 0008 83EC44              subl    $68, %esp
19                          .cfi_offset 3, -12
20 000b E8FCFFFF            call    __x86.get_pc_thunk.ax
20      FF
21 0010 05010000            addl    $_GLOBAL_OFFSET_TABLE_, %eax
21      00
22 0015 8B4508              movl    8(%ebp), %eax
23 0018 8945C4              movl    %eax, -60(%ebp)
24                          .loc 1 5 1
25 001b 65A11400            movl    %gs:20, %eax
25      0000
26 0021 8945F4              movl    %eax, -12(%ebp)
27 0024 31C0                xorl    %eax, %eax
28 0026 89E0                movl    %esp, %eax
29 0028 89C3                movl    %eax, %ebx
 6:merge_sort.c  ****     int i, j, k;
 7:merge_sort.c  ****     int n1 = m - l + 1;
30                          .loc 1 7 16
31 002a 8B4510              movl    16(%ebp), %eax
32 002d 2B450C              subl    12(%ebp), %eax
33                          .loc 1 7 9
34 0030 83C001              addl    $1, %eax
35 0033 8945DC              movl    %eax, -36(%ebp)
 8:merge_sort.c  ****     int n2 = r - m;
36                          .loc 1 8 9
37 0036 8B4514              movl    20(%ebp), %eax
38 0039 2B4510              subl    16(%ebp), %eax
39 003c 8945E0              movl    %eax, -32(%ebp)
 9:merge_sort.c  ****
10:merge_sort.c  ****     int L[n1], R[n2];
40                          .loc 1 10 5
41 003f 8B45DC              movl    -36(%ebp), %eax
42                          .loc 1 10 9
43 0042 8D50FF              leal    -1(%eax), %edx
44 0045 8955E4              movl    %edx, -28(%ebp)
45 0048 8D148500            leal    0(,%eax,4), %edx
45      000000
46 004f B8100000            movl    $16, %eax
46      00
47 0054 83E801              subl    $1, %eax
48 0057 01D0                addl    %edx, %eax
49 0059 B9100000            movl    $16, %ecx
49      00
50 005e BA000000            movl    $0, %edx
50      00
51 0063 F7F1                divl    %ecx
52 0065 6BC010              imull   $16, %eax, %eax
53 0068 89C1                movl    %eax, %ecx
54 006a 81E100F0            andl    $-4096, %ecx
54      FFFF
```

```
55 0070 89E2                movl    %esp, %edx
56 0072 29CA                subl    %ecx, %edx
57                  .L2:
58 0074 39D4                cmpl    %edx, %esp
59 0076 7410                je      .L3
60 0078 81EC0010            subl    $4096, %esp
60      0000
61 007e 838C24FC            orl     $0, 4092(%esp)
61      0F000000
62 0086 EBEC                jmp     .L2
63                  .L3:
64 0088 89C2                movl    %eax, %edx
65 008a 81E2FF0F            andl    $4095, %edx
65      0000
66 0090 29D4                subl    %edx, %esp
67 0092 89C2                movl    %eax, %edx
68 0094 81E2FF0F            andl    $4095, %edx
68      0000
69 009a 85D2                testl   %edx, %edx
70 009c 740D                je      .L4
71 009e 25FF0F00            andl    $4095, %eax
71      00
72 00a3 83E804              subl    $4, %eax
73 00a6 01E0                addl    %esp, %eax
74 00a8 830800              orl     $0, (%eax)
75                  .L4:
76 00ab 89E0                movl    %esp, %eax
77 00ad 83C003              addl    $3, %eax
78 00b0 C1E802              shrl    $2, %eax
79 00b3 C1E002              sall    $2, %eax
80 00b6 8945E8              movl    %eax, -24(%ebp)
81                          .loc 1 10 5
82 00b9 8B45E0              movl    -32(%ebp), %eax
83                          .loc 1 10 16
84 00bc 8D50FF              leal    -1(%eax), %edx
85 00bf 8955EC              movl    %edx, -20(%ebp)
86 00c2 8D148500            leal    0(,%eax,4), %edx
86      000000
87 00c9 B8100000            movl    $16, %eax
87      00
88 00ce 83E801              subl    $1, %eax
89 00d1 01D0                addl    %edx, %eax
90 00d3 B9100000            movl    $16, %ecx
90      00
91 00d8 BA000000            movl    $0, %edx
91      00
92 00dd F7F1                divl    %ecx
93 00df 6BC010              imull   $16, %eax, %eax
94 00e2 89C1                movl    %eax, %ecx
95 00e4 81E100F0            andl    $-4096, %ecx
95      FFFF
96 00ea 89E2                movl    %esp, %edx
97 00ec 29CA                subl    %ecx, %edx
```

```
 98                     .L5:
 99 00ee 39D4                    cmpl    %edx, %esp
100 00f0 7410                    je      .L6
101 00f2 81EC0010                subl    $4096, %esp
101      0000
102 00f8 838C24FC                orl     $0, 4092(%esp)
102      0F000000
103 0100 EBEC                    jmp     .L5
104                     .L6:
105 0102 89C2                    movl    %eax, %edx
106 0104 81E2FF0F                andl    $4095, %edx
106      0000
107 010a 29D4                    subl    %edx, %esp
108 010c 89C2                    movl    %eax, %edx
109 010e 81E2FF0F                andl    $4095, %edx
109      0000
110 0114 85D2                    testl   %edx, %edx
111 0116 740D                    je      .L7
112 0118 25FF0F00                andl    $4095, %eax
112      00
113 011d 83E804                  subl    $4, %eax
114 0120 01E0                    addl    %esp, %eax
115 0122 830800                  orl     $0, (%eax)
116                     .L7:
117 0125 89E0                    movl    %esp, %eax
118 0127 83C003                  addl    $3, %eax
119 012a C1E802                  shrl    $2, %eax
120 012d C1E002                  sall    $2, %eax
121 0130 8945F0                  movl    %eax, -16(%ebp)
 11:merge_sort.c  ****
 12:merge_sort.c  ****     for (i = 0; i < n1; i++)
122                              .loc 1 12 12
123 0133 C745D800                movl    $0, -40(%ebp)
123      000000
124                              .loc 1 12 5
125 013a EB23                    jmp     .L8
126                     .L9:
 13:merge_sort.c  ****         L[i] = arr[l + i];
127                              .loc 1 13 22 discriminator 3
128 013c 8B550C                  movl    12(%ebp), %edx
129 013f 8B45D8                  movl    -40(%ebp), %eax
130 0142 01D0                    addl    %edx, %eax
131                              .loc 1 13 19 discriminator 3
132 0144 8D148500                leal    0(,%eax,4), %edx
132      000000
133 014b 8B45C4                  movl    -60(%ebp), %eax
134 014e 01D0                    addl    %edx, %eax
135 0150 8B08                    movl    (%eax), %ecx
136                              .loc 1 13 14 discriminator 3
137 0152 8B45E8                  movl    -24(%ebp), %eax
138 0155 8B55D8                  movl    -40(%ebp), %edx
139 0158 890C90                  movl    %ecx, (%eax,%edx,4)
 12:merge_sort.c  ****         L[i] = arr[l + i];
```

```
140                             .loc 1 12 26 discriminator 3
141 015b 8345D801               addl    $1, -40(%ebp)
142                     .L8:
 12:merge_sort.c  ****          L[i] = arr[l + i];
143                             .loc 1 12 5 discriminator 1
144 015f 8B45D8                 movl    -40(%ebp), %eax
145 0162 3B45DC                 cmpl    -36(%ebp), %eax
146 0165 7CD5                   jl      .L9
 14:merge_sort.c  ****      for (j = 0; j < n2; j++)
147                             .loc 1 14 12
148 0167 C745D400               movl    $0, -44(%ebp)
148      000000
149                             .loc 1 14 5
150 016e EB26                   jmp     .L10
151                     .L11:
 15:merge_sort.c  ****          R[j] = arr[m + 1 + j];
152                             .loc 1 15 22 discriminator 3
153 0170 8B4510                 movl    16(%ebp), %eax
154 0173 8D5001                 leal    1(%eax), %edx
155                             .loc 1 15 26 discriminator 3
156 0176 8B45D4                 movl    -44(%ebp), %eax
157 0179 01D0                   addl    %edx, %eax
158                             .loc 1 15 19 discriminator 3
159 017b 8D148500               leal    0(,%eax,4), %edx
159      000000
160 0182 8B45C4                 movl    -60(%ebp), %eax
161 0185 01D0                   addl    %edx, %eax
162 0187 8B08                   movl    (%eax), %ecx
163                             .loc 1 15 14 discriminator 3
164 0189 8B45F0                 movl    -16(%ebp), %eax
165 018c 8B55D4                 movl    -44(%ebp), %edx
166 018f 890C90                 movl    %ecx, (%eax,%edx,4)
 14:merge_sort.c  ****      for (j = 0; j < n2; j++)
167                             .loc 1 14 26 discriminator 3
168 0192 8345D401               addl    $1, -44(%ebp)
169                     .L10:
 14:merge_sort.c  ****      for (j = 0; j < n2; j++)
170                             .loc 1 14 5 discriminator 1
171 0196 8B45D4                 movl    -44(%ebp), %eax
172 0199 3B45E0                 cmpl    -32(%ebp), %eax
173 019c 7CD2                   jl      .L11
 16:merge_sort.c  ****
 17:merge_sort.c  ****      i = 0;
174                             .loc 1 17 7
175 019e C745D800               movl    $0, -40(%ebp)
175      000000
 18:merge_sort.c  ****      j = 0;
176                             .loc 1 18 7
177 01a5 C745D400               movl    $0, -44(%ebp)
177      000000
 19:merge_sort.c  ****      k = l;
178                             .loc 1 19 7
179 01ac 8B450C                 movl    12(%ebp), %eax
```

```
180 01af 8945D0               movl    %eax, -48(%ebp)
 20:merge_sort.c  ****     while (i < n1 && j < n2) {
181                           .loc 1 20 11
182 01b2 EB5A                 jmp     .L12
183                   .L16:
 21:merge_sort.c  ****         if (L[i] <= R[j]) {
184                           .loc 1 21 14
185 01b4 8B45E8               movl    -24(%ebp), %eax
186 01b7 8B55D8               movl    -40(%ebp), %edx
187 01ba 8B0C90               movl    (%eax,%edx,4), %ecx
188                           .loc 1 21 22
189 01bd 8B45F0               movl    -16(%ebp), %eax
190 01c0 8B55D4               movl    -44(%ebp), %edx
191 01c3 8B0490               movl    (%eax,%edx,4), %eax
192                           .loc 1 21 12
193 01c6 39C1                 cmpl    %eax, %ecx
194 01c8 7F21                 jg      .L13
 22:merge_sort.c  ****             arr[k] = L[i];
195                           .loc 1 22 16
196 01ca 8B45D0               movl    -48(%ebp), %eax
197 01cd 8D148500             leal    0(,%eax,4), %edx
197      000000
198 01d4 8B45C4               movl    -60(%ebp), %eax
199 01d7 8D0C02               leal    (%edx,%eax), %ecx
200                           .loc 1 22 23
201 01da 8B45E8               movl    -24(%ebp), %eax
202 01dd 8B55D8               movl    -40(%ebp), %edx
203 01e0 8B0490               movl    (%eax,%edx,4), %eax
204                           .loc 1 22 20
205 01e3 8901                 movl    %eax, (%ecx)
 23:merge_sort.c  ****             i++;
206                           .loc 1 23 14
207 01e5 8345D801             addl    $1, -40(%ebp)
208 01e9 EB1F                 jmp     .L14
209                   .L13:
 24:merge_sort.c  ****         }
 25:merge_sort.c  ****         else {
 26:merge_sort.c  ****             arr[k] = R[j];
210                           .loc 1 26 16
211 01eb 8B45D0               movl    -48(%ebp), %eax
212 01ee 8D148500             leal    0(,%eax,4), %edx
212      000000
213 01f5 8B45C4               movl    -60(%ebp), %eax
214 01f8 8D0C02               leal    (%edx,%eax), %ecx
215                           .loc 1 26 23
216 01fb 8B45F0               movl    -16(%ebp), %eax
217 01fe 8B55D4               movl    -44(%ebp), %edx
218 0201 8B0490               movl    (%eax,%edx,4), %eax
219                           .loc 1 26 20
220 0204 8901                 movl    %eax, (%ecx)
 27:merge_sort.c  ****             j++;
221                           .loc 1 27 14
222 0206 8345D401             addl    $1, -44(%ebp)
```

```
223                     .L14:
 28:merge_sort.c  ****          }
 29:merge_sort.c  ****          k++;
224                             .loc 1 29 10
225 020a 8345D001               addl    $1, -48(%ebp)
226                     .L12:
 20:merge_sort.c  ****          if (L[i] <= R[j]) {
227                             .loc 1 20 11
228 020e 8B45D8                movl    -40(%ebp), %eax
229 0211 3B45DC                cmpl    -36(%ebp), %eax
230 0214 7D2D                   jge     .L17
 20:merge_sort.c  ****          if (L[i] <= R[j]) {
231                             .loc 1 20 19 discriminator 1
232 0216 8B45D4                movl    -44(%ebp), %eax
233 0219 3B45E0                cmpl    -32(%ebp), %eax
234 021c 7C96                   jl      .L16
 30:merge_sort.c  ****      }
 31:merge_sort.c  ****
 32:merge_sort.c  ****      while (i < n1) {
235                             .loc 1 32 11
236 021e EB23                   jmp     .L17
237                     .L18:
 33:merge_sort.c  ****          arr[k] = L[i];
238                             .loc 1 33 12
239 0220 8B45D0                movl    -48(%ebp), %eax
240 0223 8D148500               leal    0(,%eax,4), %edx
240      000000
241 022a 8B45C4                movl    -60(%ebp), %eax
242 022d 8D0C02                leal    (%edx,%eax), %ecx
243                             .loc 1 33 19
244 0230 8B45E8                movl    -24(%ebp), %eax
245 0233 8B55D8                movl    -40(%ebp), %edx
246 0236 8B0490                movl    (%eax,%edx,4), %eax
247                             .loc 1 33 16
248 0239 8901                   movl    %eax, (%ecx)
 34:merge_sort.c  ****          i++;
249                             .loc 1 34 10
250 023b 8345D801               addl    $1, -40(%ebp)
 35:merge_sort.c  ****          k++;
251                             .loc 1 35 10
252 023f 8345D001               addl    $1, -48(%ebp)
253                     .L17:
 32:merge_sort.c  ****          arr[k] = L[i];
254                             .loc 1 32 11
255 0243 8B45D8                movl    -40(%ebp), %eax
256 0246 3B45DC                cmpl    -36(%ebp), %eax
257 0249 7CD5                   jl      .L18
 36:merge_sort.c  ****      }
 37:merge_sort.c  ****
 38:merge_sort.c  ****      while (j < n2) {
258                             .loc 1 38 11
259 024b EB23                   jmp     .L19
260                     .L20:
```

```
 39:merge_sort.c  ****         arr[k] = R[j];
261                            .loc 1 39 12
262 024d 8B45D0                movl    -48(%ebp), %eax
263 0250 8D148500              leal    0(,%eax,4), %edx
263      000000
264 0257 8B45C4                movl    -60(%ebp), %eax
265 025a 8D0C02                leal    (%edx,%eax), %ecx
266                            .loc 1 39 19
267 025d 8B45F0                movl    -16(%ebp), %eax
268 0260 8B55D4                movl    -44(%ebp), %edx
269 0263 8B0490                movl    (%eax,%edx,4), %eax
270                            .loc 1 39 16
271 0266 8901                  movl    %eax, (%ecx)
 40:merge_sort.c  ****         j++;
272                            .loc 1 40 10
273 0268 8345D401              addl    $1, -44(%ebp)
 41:merge_sort.c  ****         k++;
274                            .loc 1 41 10
275 026c 8345D001              addl    $1, -48(%ebp)
276                    .L19:
 38:merge_sort.c  ****         arr[k] = R[j];
277                            .loc 1 38 11
278 0270 8B45D4                movl    -44(%ebp), %eax
279 0273 3B45E0                cmpl    -32(%ebp), %eax
280 0276 7CD5                  jl      .L20
281 0278 89DC                  movl    %ebx, %esp
 42:merge_sort.c  ****     }
 43:merge_sort.c  **** }
282                            .loc 1 43 1
283 027a 90                    nop
284 027b 8B45F4                movl    -12(%ebp), %eax
285 027e 652B0514              subl    %gs:20, %eax
285      000000
286 0285 7405                  je      .L21
287 0287 E8FCFFFF              call    __stack_chk_fail_local
287      FF
288                    .L21:
289 028c 8B5DFC                movl    -4(%ebp), %ebx
290 028f C9                    leave
291                            .cfi_restore 5
292                            .cfi_restore 3
293                            .cfi_def_cfa 4, 4
294 0290 C3                    ret
295                            .cfi_endproc
296                    .LFE6:
298                            .globl  mergeSort
300                    mergeSort:
301                    .LFB7:
...
```

- amd64

```
diasha@diasha-lubuntu:~/RE_lab1$ gcc -Wa,-adhln -g merge_sort.c >
merge_sort.amd64.lst
```

- arm

```
diasha@diasha-lubuntu:~/RE_lab1$ arm-linux-gnueabi-gcc -Wa,-adhln -g
merge_sort.c > merge_sort.arm.lst
```

- aarch64

```
diasha@diasha-lubuntu:~/RE_lab1$ aarch64-linux-gnu-gcc -Wa,-adhln -g
merge_sort.c > merge_sort.aarch64.lst
```

– Криптографічні алгоритми, алгоритми кодування та контролю цілісності:

1. AES;

Реалізацію даного алгоритму було знайдено у [відкритому доступі](), звідки було взято потрібні файли(`aes.h, aes.c`) та трохи модифіковані для зручності використання.
Результати компіляції у різних архітектурах надані у відповідних файлах(`aes.i686.lst, aes.amd64.lst, aes.arm.lst, aes.aarch64.lst`)

```
diasha@diasha-lubuntu:~/RE_lab1$ i686-linux-gnu-gcc -Wa,-adhln -g aes.c >
aes.i686.lst
diasha@diasha-lubuntu:~/RE_lab1$ gcc -Wa,-adhln -g aes.c > aes.amd64.lst
diasha@diasha-lubuntu:~/RE_lab1$ arm-linux-gnueabi-gcc -Wa,-adhln -g aes.c >
aes.arm.lst
diasha@diasha-lubuntu:~/RE_lab1$ aarch64-linux-gnu-gcc -Wa,-adhln -g aes.c >
aes.aarch64.lst
```

Поглянемо на частину лістинга `aes.i686.lst` :

```
223:aes.c         **** void xor_buf(const BYTE in[], BYTE out[], size_t len)
 224:aes.c         **** {
 318                            .loc 1 224 1
 319                            .cfi_startproc
 320 0000 F30F1EFB              endbr32
 321 0004 55                    pushl   %ebp
 322                            .cfi_def_cfa_offset 8
 323                            .cfi_offset 5, -8
 324 0005 89E5                  movl    %esp, %ebp
 325                            .cfi_def_cfa_register 5
 326 0007 53                    pushl   %ebx
 327 0008 83EC10                subl    $16, %esp
 328                            .cfi_offset 3, -12
 329 000b E8FCFFFF              call    __x86.get_pc_thunk.ax
 329      FF
 330 0010 05010000              addl    $_GLOBAL_OFFSET_TABLE_, %eax
 330      00
 225:aes.c         ****         size_t idx;
 226:aes.c         ****
 227:aes.c         ****         for (idx = 0; idx < len; idx++)
 331                            .loc 1 227 11
```

```
 332 0015 C745F800              movl    $0, -8(%ebp)
 332      000000
 333                            .loc 1 227 2
 334 001c EB26                  jmp     .L2
 335                    .L3:
 228:aes.c         ****                 out[idx] ^= in[idx];
 336                            .loc 1 228 12 discriminator 3
 337 001e 8B550C                movl    12(%ebp), %edx
 338 0021 8B45F8                movl    -8(%ebp), %eax
 339 0024 01D0                  addl    %edx, %eax
 340 0026 0FB608                movzbl  (%eax), %ecx
 341                            .loc 1 228 17 discriminator 3
 342 0029 8B5508                movl    8(%ebp), %edx
 343 002c 8B45F8                movl    -8(%ebp), %eax
 344 002f 01D0                  addl    %edx, %eax
 345 0031 0FB610                movzbl  (%eax), %edx
 346                            .loc 1 228 12 discriminator 3
 347 0034 8B5D0C                movl    12(%ebp), %ebx
 348 0037 8B45F8                movl    -8(%ebp), %eax
 349 003a 01D8                  addl    %ebx, %eax
 350 003c 31CA                  xorl    %ecx, %edx
 351 003e 8810                  movb    %dl, (%eax)
 227:aes.c         ****                 out[idx] ^= in[idx];
 352                            .loc 1 227 30 discriminator 3
 353 0040 8345F801              addl    $1, -8(%ebp)
 354                    .L2:
 227:aes.c         ****                 out[idx] ^= in[idx];
 355                            .loc 1 227 2 discriminator 1
 356 0044 8B45F8                movl    -8(%ebp), %eax
 357 0047 3B4510                cmpl    16(%ebp), %eax
 358 004a 72D2                  jb      .L3
 229:aes.c         **** }
 359                            .loc 1 229 1
 360 004c 90                    nop
 361 004d 90                    nop
 362 004e 8B5DFC                movl    -4(%ebp), %ebx
 363 0051 C9                    leave
 364                            .cfi_restore 5
 365                            .cfi_restore 3
 366                            .cfi_def_cfa 4, 4
 367 0052 C3                    ret
 368                            .cfi_endproc
 369                    .LFE6:
 371                            .globl  aes_encrypt_cbc
 373                    aes_encrypt_cbc:
 374                    .LFB7:
```

Завдання 3

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

5(19 mod 7). musl

Функції:

- стандартного вводу-виводу printf, puts

printf:

```
000000000005eb30 <printf>:
   5eb30:    f3 0f 1e fa              endbr64
   5eb34:    48 81 ec d8 00 00 00     sub    $0xd8,%rsp
   5eb3b:    49 89 fa                 mov    %rdi,%r10
   5eb3e:    48 89 74 24 28           mov    %rsi,0x28(%rsp)
   5eb43:    48 89 54 24 30           mov    %rdx,0x30(%rsp)
   5eb48:    48 89 4c 24 38           mov    %rcx,0x38(%rsp)
   5eb4d:    4c 89 44 24 40           mov    %r8,0x40(%rsp)
   5eb52:    4c 89 4c 24 48           mov    %r9,0x48(%rsp)
   5eb57:    84 c0                    test   %al,%al
   5eb59:    74 37                    je     5eb92 <printf+0x62>
   5eb5b:    0f 29 44 24 50           movaps %xmm0,0x50(%rsp)
   5eb60:    0f 29 4c 24 60           movaps %xmm1,0x60(%rsp)
   5eb65:    0f 29 54 24 70           movaps %xmm2,0x70(%rsp)
   5eb6a:    0f 29 9c 24 80 00 00     movaps %xmm3,0x80(%rsp)
   5eb71:    00
   5eb72:    0f 29 a4 24 90 00 00     movaps %xmm4,0x90(%rsp)
   5eb79:    00
   5eb7a:    0f 29 ac 24 a0 00 00     movaps %xmm5,0xa0(%rsp)
   5eb81:    00
   5eb82:    0f 29 b4 24 b0 00 00     movaps %xmm6,0xb0(%rsp)
   5eb89:    00
   5eb8a:    0f 29 bc 24 c0 00 00     movaps %xmm7,0xc0(%rsp)
   5eb91:    00
   5eb92:    64 48 8b 04 25 28 00     mov    %fs:0x28,%rax
   5eb99:    00 00
   5eb9b:    48 89 44 24 18           mov    %rax,0x18(%rsp)
   5eba0:    31 c0                    xor    %eax,%eax
   5eba2:    48 8d 84 24 e0 00 00     lea    0xe0(%rsp),%rax
   5eba9:    00
   5ebaa:    48 89 e2                 mov    %rsp,%rdx
   5ebad:    4c 89 d6                 mov    %r10,%rsi
   5ebb0:    48 89 44 24 08           mov    %rax,0x8(%rsp)
   5ebb5:    48 8d 3d c4 16 05 00     lea    0x516c4(%rip),%rdi        # b0280
<stdout+0x4c8>
   5ebbc:    48 8d 44 24 20           lea    0x20(%rsp),%rax
   5ebc1:    c7 04 24 08 00 00 00     movl   $0x8,(%rsp)
   5ebc8:    c7 44 24 04 30 00 00     movl   $0x30,0x4(%rsp)
   5ebcf:    00
   5ebd0:    48 89 44 24 10           mov    %rax,0x10(%rsp)
   5ebd5:    e8 66 34 00 00           callq  62040 <vfprintf>
   5ebda:    48 8b 4c 24 18           mov    0x18(%rsp),%rcx
   5ebdf:    64 48 33 0c 25 28 00     xor    %fs:0x28,%rcx
   5ebe6:    00 00
   5ebe8:    75 08                    jne    5ebf2 <printf+0xc2>
   5ebea:    48 81 c4 d8 00 00 00     add    $0xd8,%rsp
   5ebf1:    c3                       retq
   5ebf2:    e8 99 01 fc ff           callq  1ed90 <__stack_chk_fail>
   5ebf7:    66 0f 1f 84 00 00 00     nopw   0x0(%rax,%rax,1)
   5ebfe:    00 00
   5ec00:    41 54                    push   %r12
   5ec02:    31 c0                    xor    %eax,%eax
   5ec04:    41 89 fc                 mov    %edi,%r12d
   5ec07:    ba ff ff ff 3f           mov    $0x3fffffff,%edx
   5ec0c:    55                       push   %rbp
   5ec0d:    48 8d ae 8c 00 00 00     lea    0x8c(%rsi),%rbp
   5ec14:    53                       push   %rbx
   5ec15:    48 89 f3                 mov    %rsi,%rbx
   5ec18:    f0 0f b1 96 8c 00 00     lock cmpxchg %edx,0x8c(%rsi)
   5ec1f:    00
```

```
   5ec20:    85 c0                   test   %eax,%eax
   5ec22:    75 6c                   jne    5ec90 <printf+0x160>
   5ec24:    45 0f b6 c4             movzbl %r12b,%r8d
   5ec28:    44 39 83 90 00 00 00    cmp    %r8d,0x90(%rbx)
   5ec2f:    74 4f                   je     5ec80 <printf+0x150>
   5ec31:    48 8b 43 28             mov    0x28(%rbx),%rax
   5ec35:    48 3b 43 20             cmp    0x20(%rbx),%rax
   5ec39:    74 45                   je     5ec80 <printf+0x150>
   5ec3b:    48 8d 50 01             lea    0x1(%rax),%rdx
   5ec3f:    48 89 53 28             mov    %rdx,0x28(%rbx)
   5ec43:    44 88 20                mov    %r12b,(%rax)
   5ec46:    31 c9                   xor    %ecx,%ecx
   5ec48:    87 8b 8c 00 00 00       xchg   %ecx,0x8c(%rbx)
   5ec4e:    81 e1 00 00 00 40       and    $0x40000000,%ecx
   5ec54:    74 1e                   je     5ec74 <printf+0x144>
   5ec56:    41 b9 ca 00 00 00       mov    $0xca,%r9d
   5ec5c:    ba 01 00 00 00          mov    $0x1,%edx
   5ec61:    be 81 00 00 00          mov    $0x81,%esi
   5ec66:    48 89 ef                mov    %rbp,%rdi
   5ec69:    4c 89 c8                mov    %r9,%rax
   5ec6c:    0f 05                   syscall
   5ec6e:    48 83 f8 da             cmp    $0xffffffffffffffda,%rax
   5ec72:    74 2c                   je     5eca0 <printf+0x170>
   5ec74:    5b                      pop    %rbx
   5ec75:    44 89 c0                mov    %r8d,%eax
   5ec78:    5d                      pop    %rbp
   5ec79:    41 5c                   pop    %r12
   5ec7b:    c3                      retq
   5ec7c:    0f 1f 40 00             nopl   0x0(%rax)
   5ec80:    44 89 c6                mov    %r8d,%esi
   5ec83:    48 89 df                mov    %rbx,%rdi
   5ec86:    e8 c5 be ff ff          callq  5ab50 <__overflow>
   5ec8b:    41 89 c0                mov    %eax,%r8d
   5ec8e:    eb b6                   jmp    5ec46 <printf+0x116>
   5ec90:    48 89 f7                mov    %rsi,%rdi
   5ec93:    e8 88 bd ff ff          callq  5aa20 <fdopen+0x370>
   5ec98:    eb 8a                   jmp    5ec24 <printf+0xf4>
   5ec9a:    66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
   5eca0:    4c 89 c8                mov    %r9,%rax
   5eca3:    48 89 d6                mov    %rdx,%rsi
   5eca6:    0f 05                   syscall
   5eca8:    5b                      pop    %rbx
   5eca9:    44 89 c0                mov    %r8d,%eax
   5ecac:    5d                      pop    %rbp
   5ecad:    41 5c                   pop    %r12
   5ecaf:    c3                      retq
```

puts:

```
000000000005eed0 <puts>:
   5eed0:    f3 0f 1e fa             endbr64
   5eed4:    8b 05 32 14 05 00       mov    0x51432(%rip),%eax        # b030c
<stdout+0x554>
   5eeda:    41 54                   push   %r12
   5eedc:    55                      push   %rbp
   5eedd:    48 89 fd                mov    %rdi,%rbp
   5eee0:    53                      push   %rbx
   5eee1:    85 c0                   test   %eax,%eax
   5eee3:    79 7b                   jns    5ef60 <puts+0x90>
   5eee5:    48 8d 35 94 13 05 00    lea    0x51394(%rip),%rsi        # b0280
<stdout+0x4c8>
   5eeec:    41 bc ff ff ff ff       mov    $0xffffffff,%r12d
   5eef2:    31 db                   xor    %ebx,%ebx
   5eef4:    e8 37 db ff ff          callq  5ca30 <fputs>
   5eef9:    85 c0                   test   %eax,%eax
   5eefb:    78 3a                   js     5ef37 <puts+0x67>
   5eefd:    83 3d 0c 14 05 00 0a    cmpl   $0xa,0x5140c(%rip)        # b0310
```

```
<stdout+0x558>
   5ef04:    74 3a                    je     5ef40 <puts+0x70>
   5ef06:    48 8b 05 9b 13 05 00     mov    0x5139b(%rip),%rax        # b02a8
<stdout+0x4f0>
   5ef0d:    48 3b 05 8c 13 05 00     cmp    0x5138c(%rip),%rax        # b02a0
<stdout+0x4e8>
   5ef14:    74 2a                    je     5ef40 <puts+0x70>
   5ef16:    48 8d 50 01              lea    0x1(%rax),%rdx
   5ef1a:    45 31 e4                 xor    %r12d,%r12d
   5ef1d:    48 89 15 84 13 05 00     mov    %rdx,0x51384(%rip)        # b02a8
<stdout+0x4f0>
   5ef24:    c6 00 0a                 movb   $0xa,(%rax)
   5ef27:    85 db                    test   %ebx,%ebx
   5ef29:    74 0c                    je     5ef37 <puts+0x67>
   5ef2b:    48 8d 3d 4e 13 05 00     lea    0x5134e(%rip),%rdi        # b0280
<stdout+0x4c8>
   5ef32:    e8 c9 bb ff ff           callq  5ab00 <fdopen+0x450>
   5ef37:    44 89 e0                 mov    %r12d,%eax
   5ef3a:    5b                       pop    %rbx
   5ef3b:    5d                       pop    %rbp
   5ef3c:    41 5c                    pop    %r12
   5ef3e:    c3                       retq
   5ef3f:    90                       nop
   5ef40:    be 0a 00 00 00           mov    $0xa,%esi
   5ef45:    48 8d 3d 34 13 05 00     lea    0x51334(%rip),%rdi        # b0280
<stdout+0x4c8>
   5ef4c:    e8 ff bb ff ff           callq  5ab50 <__overflow>
   5ef51:    c1 f8 1f                 sar    $0x1f,%eax
   5ef54:    41 89 c4                 mov    %eax,%r12d
   5ef57:    eb ce                    jmp    5ef27 <puts+0x57>
   5ef59:    0f 1f 80 00 00 00 00     nopl   0x0(%rax)
   5ef60:    48 8d 3d 19 13 05 00     lea    0x51319(%rip),%rdi        # b0280
<stdout+0x4c8>
   5ef67:    41 bc ff ff ff ff        mov    $0xffffffff,%r12d
   5ef6d:    e8 ae ba ff ff           callq  5aa20 <fdopen+0x370>
   5ef72:    48 8d 35 07 13 05 00     lea    0x51307(%rip),%rsi        # b0280
<stdout+0x4c8>
   5ef79:    48 89 ef                 mov    %rbp,%rdi
   5ef7c:    89 c3                    mov    %eax,%ebx
   5ef7e:    e8 ad da ff ff           callq  5ca30 <fputs>
   5ef83:    85 c0                    test   %eax,%eax
   5ef85:    78 a0                    js     5ef27 <puts+0x57>
   5ef87:    83 3d 82 13 05 00 0a     cmpl   $0xa,0x51382(%rip)        # b0310
<stdout+0x558>
   5ef8e:    0f 85 72 ff ff ff        jne    5ef06 <puts+0x36>
   5ef94:    eb aa                    jmp    5ef40 <puts+0x70>
   5ef96:    66 2e 0f 1f 84 00 00     nopw   %cs:0x0(%rax,%rax,1)
   5ef9d:    00 00 00
```

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

fopen:

```
000000000005c3f0 <fopen>:
   5c3f0:    f3 0f 1e fa              endbr64
   5c3f4:    41 55                    push   %r13
   5c3f6:    41 54                    push   %r12
   5c3f8:    55                       push   %rbp
   5c3f9:    48 89 f5                 mov    %rsi,%rbp
   5c3fc:    53                       push   %rbx
   5c3fd:    48 89 fb                 mov    %rdi,%rbx
   5c400:    48 8d 3d 31 1c 05 00     lea    0x51c31(%rip),%rdi        # ae038
<in6addr_loopback+0x4688>
   5c407:    48 83 ec 08              sub    $0x8,%rsp
   5c40b:    0f be 36                 movsbl (%rsi),%esi
   5c40e:    e8 6d b0 00 00           callq  67480 <strchr>
   5c413:    48 85 c0                 test   %rax,%rax
```

```
5c416:    0f 84 94 00 00 00         je     5c4b0 <fopen+0xc0>
5c41c:    48 89 ef                  mov    %rbp,%rdi
5c41f:    41 bd 02 00 00 00         mov    $0x2,%r13d
5c425:    e8 76 e4 ff ff            callq  5a8a0 <fdopen+0x1f0>
5c42a:    ba b6 01 00 00            mov    $0x1b6,%edx
5c42f:    48 89 df                  mov    %rbx,%rdi
5c432:    48 63 f0                  movslq %eax,%rsi
5c435:    4c 89 e8                  mov    %r13,%rax
5c438:    49 89 f4                  mov    %rsi,%r12
5c43b:    0f 05                     syscall
5c43d:    48 89 c7                  mov    %rax,%rdi
5c440:    e8 db 55 fc ff            callq  21a20 <feupdateenv+0x1e70>
5c445:    48 89 c3                  mov    %rax,%rbx
5c448:    85 c0                     test   %eax,%eax
5c44a:    0f 88 80 00 00 00         js     5c4d0 <fopen+0xe0>
5c450:    41 81 e4 00 00 08 00      and    $0x80000,%r12d
5c457:    75 27                     jne    5c480 <fopen+0x90>
5c459:    48 89 ee                  mov    %rbp,%rsi
5c45c:    89 df                     mov    %ebx,%edi
5c45e:    e8 4d e2 ff ff            callq  5a6b0 <fdopen>
5c463:    49 89 c4                  mov    %rax,%r12
5c466:    48 85 c0                  test   %rax,%rax
5c469:    74 2d                     je     5c498 <fopen+0xa8>
5c46b:    48 83 c4 08               add    $0x8,%rsp
5c46f:    4c 89 e0                  mov    %r12,%rax
5c472:    5b                        pop    %rbx
5c473:    5d                        pop    %rbp
5c474:    41 5c                     pop    %r12
5c476:    41 5d                     pop    %r13
5c478:    c3                        retq
5c479:    0f 1f 80 00 00 00 00      nopl   0x0(%rax)
5c480:    48 63 f8                  movslq %eax,%rdi
5c483:    ba 01 00 00 00            mov    $0x1,%edx
5c488:    b8 48 00 00 00            mov    $0x48,%eax
5c48d:    4c 89 ee                  mov    %r13,%rsi
5c490:    0f 05                     syscall
5c492:    eb c5                     jmp    5c459 <fopen+0x69>
5c494:    0f 1f 40 00               nopl   0x0(%rax)
5c498:    48 63 fb                  movslq %ebx,%rdi
5c49b:    b8 03 00 00 00            mov    $0x3,%eax
5c4a0:    0f 05                     syscall
5c4a2:    48 83 c4 08               add    $0x8,%rsp
5c4a6:    4c 89 e0                  mov    %r12,%rax
5c4a9:    5b                        pop    %rbx
5c4aa:    5d                        pop    %rbp
5c4ab:    41 5c                     pop    %r12
5c4ad:    41 5d                     pop    %r13
5c4af:    c3                        retq
5c4b0:    49 89 c4                  mov    %rax,%r12
5c4b3:    e8 18 2e fc ff            callq  1f2d0 <__errno_location>
5c4b8:    c7 00 16 00 00 00         movl   $0x16,(%rax)
5c4be:    48 83 c4 08               add    $0x8,%rsp
5c4c2:    4c 89 e0                  mov    %r12,%rax
5c4c5:    5b                        pop    %rbx
5c4c6:    5d                        pop    %rbp
5c4c7:    41 5c                     pop    %r12
5c4c9:    41 5d                     pop    %r13
5c4cb:    c3                        retq
5c4cc:    0f 1f 40 00               nopl   0x0(%rax)
5c4d0:    48 83 c4 08               add    $0x8,%rsp
5c4d4:    45 31 e4                  xor    %r12d,%r12d
5c4d7:    5b                        pop    %rbx
5c4d8:    4c 89 e0                  mov    %r12,%rax
5c4db:    5d                        pop    %rbp
5c4dc:    41 5c                     pop    %r12
5c4de:    41 5d                     pop    %r13
5c4e0:    c3                        retq
5c4e1:    66 2e 0f 1f 84 00 00      nopw   %cs:0x0(%rax,%rax,1)
```

```
5c4e8:    00 00 00
5c4eb:    0f 1f 44 00 00           nopl   0x0(%rax,%rax,1)
5c4f0:    f3 0f 1e fa              endbr64
5c4f4:    41 56                    push   %r14
5c4f6:    31 c0                    xor    %eax,%eax
5c4f8:    41 55                    push   %r13
5c4fa:    41 54                    push   %r12
5c4fc:    55                       push   %rbp
5c4fd:    48 89 d5                 mov    %rdx,%rbp
5c500:    53                       push   %rbx
5c501:    48 8b 57 60              mov    0x60(%rdi),%rdx
5c505:    49 89 ec                 mov    %rbp,%r12
5c508:    48 89 fb                 mov    %rdi,%rbx
5c50b:    4c 8b b7 98 00 00 00     mov    0x98(%rdi),%r14
5c512:    48 85 d2                 test   %rdx,%rdx
5c515:    0f 95 c0                 setne  %al
5c518:    49 29 c4                 sub    %rax,%r12
5c51b:    49 8b 46 08              mov    0x8(%r14),%rax
5c51f:    48 85 c0                 test   %rax,%rax
5c522:    0f 84 a0 00 00 00        je     5c5c8 <fopen+0x1d8>
5c528:    49 89 f5                 mov    %rsi,%r13
5c52b:    4d 85 e4                 test   %r12,%r12
5c52e:    74 17                    je     5c547 <fopen+0x157>
5c530:    49 8b 3e                 mov    (%r14),%rdi
5c533:    4c 89 e2                 mov    %r12,%rdx
5c536:    ff d0                    callq  *%rax
5c538:    48 85 c0                 test   %rax,%rax
5c53b:    7e 5b                    jle    5c598 <fopen+0x1a8>
5c53d:    48 8b 53 60              mov    0x60(%rbx),%rdx
5c541:    49 89 c4                 mov    %rax,%r12
5c544:    48 29 c5                 sub    %rax,%rbp
5c547:    48 83 fd 01              cmp    $0x1,%rbp
5c54b:    77 38                    ja     5c585 <fopen+0x195>
5c54d:    48 85 d2                 test   %rdx,%rdx
5c550:    74 33                    je     5c585 <fopen+0x195>
5c552:    48 8b 73 58              mov    0x58(%rbx),%rsi
5c556:    49 8b 3e                 mov    (%r14),%rdi
5c559:    48 89 73 08              mov    %rsi,0x8(%rbx)
5c55d:    41 ff 56 08              callq  *0x8(%r14)
5c561:    48 85 c0                 test   %rax,%rax
5c564:    7e 35                    jle    5c59b <fopen+0x1ab>
5c566:    48 8b 53 08              mov    0x8(%rbx),%rdx
5c56a:    48 01 d0                 add    %rdx,%rax
5c56d:    48 89 43 10              mov    %rax,0x10(%rbx)
5c571:    48 8d 42 01              lea    0x1(%rdx),%rax
5c575:    48 89 43 08              mov    %rax,0x8(%rbx)
5c579:    0f b6 02                 movzbl (%rdx),%eax
5c57c:    43 88 44 25 00           mov    %al,0x0(%r13,%r12,1)
5c581:    49 83 c4 01              add    $0x1,%r12
5c585:    5b                       pop    %rbx
5c586:    4c 89 e0                 mov    %r12,%rax
5c589:    5d                       pop    %rbp
5c58a:    41 5c                    pop    %r12
5c58c:    41 5d                    pop    %r13
5c58e:    41 5e                    pop    %r14
5c590:    c3                       retq
5c591:    0f 1f 80 00 00 00 00     nopl   0x0(%rax)
5c598:    45 31 e4                 xor    %r12d,%r12d
5c59b:    48 83 f8 01              cmp    $0x1,%rax
5c59f:    8b 13                    mov    (%rbx),%edx
5c5a1:    19 c0                    sbb    %eax,%eax
5c5a3:    83 e0 f0                 and    $0xfffffff0,%eax
5c5a6:    83 c0 20                 add    $0x20,%eax
5c5a9:    09 d0                    or     %edx,%eax
5c5ab:    89 03                    mov    %eax,(%rbx)
5c5ad:    48 8b 43 58              mov    0x58(%rbx),%rax
5c5b1:    48 89 43 10              mov    %rax,0x10(%rbx)
5c5b5:    48 89 43 08              mov    %rax,0x8(%rbx)
```

```
5c5b9:    5b                      pop    %rbx
5c5ba:    4c 89 e0                mov    %r12,%rax
5c5bd:    5d                      pop    %rbp
5c5be:    41 5c                   pop    %r12
5c5c0:    41 5d                   pop    %r13
5c5c2:    41 5e                   pop    %r14
5c5c4:    c3                      retq
5c5c5:    0f 1f 00                nopl   (%rax)
5c5c8:    8b 17                   mov    (%rdi),%edx
5c5ca:    45 31 e4                xor    %r12d,%r12d
5c5cd:    b8 20 00 00 00          mov    $0x20,%eax
5c5d2:    eb d5                   jmp    5c5a9 <fopen+0x1b9>
5c5d4:    66 2e 0f 1f 84 00 00    nopw   %cs:0x0(%rax,%rax,1)
5c5db:    00 00 00
5c5de:    66 90                   xchg   %ax,%ax
5c5e0:    f3 0f 1e fa             endbr64
5c5e4:    48 8b 97 98 00 00 00    mov    0x98(%rdi),%rdx
5c5eb:    48 8b 42 20             mov    0x20(%rdx),%rax
5c5ef:    48 85 c0                test   %rax,%rax
5c5f2:    74 0c                   je     5c600 <fopen+0x210>
5c5f4:    48 8b 3a                mov    (%rdx),%rdi
5c5f7:    ff e0                   jmpq   *%rax
5c5f9:    0f 1f 80 00 00 00 00    nopl   0x0(%rax)
5c600:    31 c0                   xor    %eax,%eax
5c602:    c3                      retq
5c603:    66 2e 0f 1f 84 00 00    nopw   %cs:0x0(%rax,%rax,1)
5c60a:    00 00 00
5c60d:    0f 1f 00                nopl   (%rax)
5c610:    f3 0f 1e fa             endbr64
5c614:    41 56                   push   %r14
5c616:    41 55                   push   %r13
5c618:    49 89 f5                mov    %rsi,%r13
5c61b:    41 54                   push   %r12
5c61d:    49 89 d4                mov    %rdx,%r12
5c620:    55                      push   %rbp
5c621:    53                      push   %rbx
5c622:    4c 8b b7 98 00 00 00    mov    0x98(%rdi),%r14
5c629:    48 8b 77 38             mov    0x38(%rdi),%rsi
5c62d:    48 8b 6f 28             mov    0x28(%rdi),%rbp
5c631:    49 8b 46 10             mov    0x10(%r14),%rax
5c635:    48 29 f5                sub    %rsi,%rbp
5c638:    48 85 c0                test   %rax,%rax
5c63b:    74 73                   je     5c6b0 <fopen+0x2c0>
5c63d:    48 89 fb                mov    %rdi,%rbx
5c640:    48 85 ed                test   %rbp,%rbp
5c643:    75 1b                   jne    5c660 <fopen+0x270>
5c645:    49 8b 3e                mov    (%r14),%rdi
5c648:    4c 89 e2                mov    %r12,%rdx
5c64b:    4c 89 ee                mov    %r13,%rsi
5c64e:    ff d0                   callq  *%rax
5c650:    48 85 c0                test   %rax,%rax
5c653:    78 35                   js     5c68a <fopen+0x29a>
5c655:    5b                      pop    %rbx
5c656:    5d                      pop    %rbp
5c657:    41 5c                   pop    %r12
5c659:    41 5d                   pop    %r13
5c65b:    41 5e                   pop    %r14
5c65d:    c3                      retq
5c65e:    66 90                   xchg   %ax,%ax
5c660:    48 89 77 28             mov    %rsi,0x28(%rdi)
5c664:    48 89 ea                mov    %rbp,%rdx
5c667:    e8 a4 ff ff ff          callq  5c610 <fopen+0x220>
5c66c:    49 89 c0                mov    %rax,%r8
5c66f:    31 c0                   xor    %eax,%eax
5c671:    49 39 e8                cmp    %rbp,%r8
5c674:    72 df                   jb     5c655 <fopen+0x265>
5c676:    49 8b 46 10             mov    0x10(%r14),%rax
5c67a:    49 8b 3e                mov    (%r14),%rdi
```

```
   5c67d:    4c 89 e2                mov    %r12,%rdx
   5c680:    4c 89 ee                mov    %r13,%rsi
   5c683:    ff d0                   callq  *%rax
   5c685:    48 85 c0                test   %rax,%rax
   5c688:    79 cb                   jns    5c655 <fopen+0x265>
   5c68a:    83 0b 20                orl    $0x20,(%rbx)
   5c68d:    31 c0                   xor    %eax,%eax
   5c68f:    48 c7 43 20 00 00 00    movq   $0x0,0x20(%rbx)
   5c696:    00
   5c697:    48 c7 43 38 00 00 00    movq   $0x0,0x38(%rbx)
   5c69e:    00
   5c69f:    48 c7 43 28 00 00 00    movq   $0x0,0x28(%rbx)
   5c6a6:    00
   5c6a7:    5b                      pop    %rbx
   5c6a8:    5d                      pop    %rbp
   5c6a9:    41 5c                   pop    %r12
   5c6ab:    41 5d                   pop    %r13
   5c6ad:    41 5e                   pop    %r14
   5c6af:    c3                      retq
   5c6b0:    5b                      pop    %rbx
   5c6b1:    48 89 d0                mov    %rdx,%rax
   5c6b4:    5d                      pop    %rbp
   5c6b5:    41 5c                   pop    %r12
   5c6b7:    41 5d                   pop    %r13
   5c6b9:    41 5e                   pop    %r14
   5c6bb:    c3                      retq
   5c6bc:    0f 1f 40 00             nopl   0x0(%rax)
   5c6c0:    f3 0f 1e fa             endbr64
   5c6c4:    48 83 ec 18             sub    $0x18,%rsp
   5c6c8:    48 8b 87 98 00 00 00    mov    0x98(%rdi),%rax
   5c6cf:    48 89 74 24 08          mov    %rsi,0x8(%rsp)
   5c6d4:    83 fa 02                cmp    $0x2,%edx
   5c6d7:    77 27                   ja     5c700 <fopen+0x310>
   5c6d9:    48 8b 48 18             mov    0x18(%rax),%rcx
   5c6dd:    48 85 c9                test   %rcx,%rcx
   5c6e0:    74 32                   je     5c714 <fopen+0x324>
   5c6e2:    48 8b 38                mov    (%rax),%rdi
   5c6e5:    48 8d 74 24 08          lea    0x8(%rsp),%rsi
   5c6ea:    ff d1                   callq  *%rcx
   5c6ec:    48 63 d0                movslq %eax,%rdx
   5c6ef:    85 c0                   test   %eax,%eax
   5c6f1:    48 89 d0                mov    %rdx,%rax
   5c6f4:    48 0f 49 44 24 08       cmovns 0x8(%rsp),%rax
   5c6fa:    48 83 c4 18             add    $0x18,%rsp
   5c6fe:    c3                      retq
   5c6ff:    90                      nop
   5c700:    e8 cb 2b fc ff          callq  1f2d0 <__errno_location>
   5c705:    c7 00 16 00 00 00       movl   $0x16,(%rax)
   5c70b:    48 c7 c0 ff ff ff ff    mov    $0xffffffffffffffff,%rax
   5c712:    eb e6                   jmp    5c6fa <fopen+0x30a>
   5c714:    e8 b7 2b fc ff          callq  1f2d0 <__errno_location>
   5c719:    c7 00 5f 00 00 00       movl   $0x5f,(%rax)
   5c71f:    48 c7 c0 ff ff ff ff    mov    $0xffffffffffffffff,%rax
   5c726:    eb d2                   jmp    5c6fa <fopen+0x30a>
   5c728:    0f 1f 84 00 00 00 00    nopl   0x0(%rax,%rax,1)
   5c72f:    00
```

fread:

```
000000000005cd60 <fread>:
   5cd60:    f3 0f 1e fa             endbr64
   5cd64:    41 57                   push   %r15
   5cd66:    b8 00 00 00 00          mov    $0x0,%eax
   5cd6b:    49 89 cf                mov    %rcx,%r15
   5cd6e:    41 56                   push   %r14
   5cd70:    49 89 f6                mov    %rsi,%r14
   5cd73:    41 55                   push   %r13
   5cd75:    49 89 d5                mov    %rdx,%r13
```

```
5cd78:    41 54                   push   %r12
5cd7a:    49 89 f4                mov    %rsi,%r12
5cd7d:    55                      push   %rbp
5cd7e:    4c 0f af e2             imul   %rdx,%r12
5cd82:    48 89 fd                mov    %rdi,%rbp
5cd85:    53                      push   %rbx
5cd86:    48 83 ec 18             sub    $0x18,%rsp
5cd8a:    48 85 f6                test   %rsi,%rsi
5cd8d:    4c 0f 44 e8             cmove  %rax,%r13
5cd91:    8b 81 8c 00 00 00       mov    0x8c(%rcx),%eax
5cd97:    c7 44 24 08 00 00 00    movl   $0x0,0x8(%rsp)
5cd9e:    00
5cd9f:    85 c0                   test   %eax,%eax
5cda1:    78 0c                   js     5cdaf <fread+0x4f>
5cda3:    48 89 cf                mov    %rcx,%rdi
5cda6:    e8 75 dc ff ff          callq  5aa20 <fdopen+0x370>
5cdab:    89 44 24 08             mov    %eax,0x8(%rsp)
5cdaf:    41 8b 97 88 00 00 00    mov    0x88(%r15),%edx
5cdb6:    49 8b 77 08             mov    0x8(%r15),%rsi
5cdba:    4c 89 e3                mov    %r12,%rbx
5cdbd:    8d 42 ff                lea    -0x1(%rdx),%eax
5cdc0:    09 d0                   or     %edx,%eax
5cdc2:    41 89 87 88 00 00 00    mov    %eax,0x88(%r15)
5cdc9:    49 8b 47 10             mov    0x10(%r15),%rax
5cdcd:    48 39 c6                cmp    %rax,%rsi
5cdd0:    74 28                   je     5cdfa <fread+0x9a>
5cdd2:    48 29 f0                sub    %rsi,%rax
5cdd5:    48 89 ef                mov    %rbp,%rdi
5cdd8:    4c 39 e0                cmp    %r12,%rax
5cddb:    49 0f 47 c4             cmova  %r12,%rax
5cddf:    48 89 c2                mov    %rax,%rdx
5cde2:    48 89 c3                mov    %rax,%rbx
5cde5:    e8 a6 bf 01 00          callq  78d90 <memcpy>
5cdea:    4c 89 e0                mov    %r12,%rax
5cded:    49 01 5f 08             add    %rbx,0x8(%r15)
5cdf1:    48 01 dd                add    %rbx,%rbp
5cdf4:    48 29 d8                sub    %rbx,%rax
5cdf7:    48 89 c3                mov    %rax,%rbx
5cdfa:    48 85 db                test   %rbx,%rbx
5cdfd:    75 23                   jne    5ce22 <fread+0xc2>
5cdff:    eb 5f                   jmp    5ce60 <fread+0x100>
5ce01:    0f 1f 80 00 00 00 00    nopl   0x0(%rax)
5ce08:    48 89 da                mov    %rbx,%rdx
5ce0b:    48 89 ee                mov    %rbp,%rsi
5ce0e:    4c 89 ff                mov    %r15,%rdi
5ce11:    41 ff 57 40             callq  *0x40(%r15)
5ce15:    48 85 c0                test   %rax,%rax
5ce18:    74 14                   je     5ce2e <fread+0xce>
5ce1a:    48 01 c5                add    %rax,%rbp
5ce1d:    48 29 c3                sub    %rax,%rbx
5ce20:    74 3e                   je     5ce60 <fread+0x100>
5ce22:    4c 89 ff                mov    %r15,%rdi
5ce25:    e8 16 e2 ff ff          callq  5b040 <__overflow+0x4f0>
5ce2a:    85 c0                   test   %eax,%eax
5ce2c:    74 da                   je     5ce08 <fread+0xa8>
5ce2e:    8b 4c 24 08             mov    0x8(%rsp),%ecx
5ce32:    85 c9                   test   %ecx,%ecx
5ce34:    75 1a                   jne    5ce50 <fread+0xf0>
5ce36:    4c 89 e0                mov    %r12,%rax
5ce39:    31 d2                   xor    %edx,%edx
5ce3b:    48 29 d8                sub    %rbx,%rax
5ce3e:    49 f7 f6                div    %r14
5ce41:    48 83 c4 18             add    $0x18,%rsp
5ce45:    5b                      pop    %rbx
5ce46:    5d                      pop    %rbp
5ce47:    41 5c                   pop    %r12
5ce49:    41 5d                   pop    %r13
5ce4b:    41 5e                   pop    %r14
```

```
   5ce4d:    41 5f                   pop    %r15
   5ce4f:    c3                      retq
   5ce50:    4c 89 ff                mov    %r15,%rdi
   5ce53:    e8 a8 dc ff ff          callq  5ab00 <fdopen+0x450>
   5ce58:    eb dc                   jmp    5ce36 <fread+0xd6>
   5ce5a:    66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
   5ce60:    8b 54 24 08             mov    0x8(%rsp),%edx
   5ce64:    4c 89 e8                mov    %r13,%rax
   5ce67:    85 d2                   test   %edx,%edx
   5ce69:    74 d6                   je     5ce41 <fread+0xe1>
   5ce6b:    4c 89 ff                mov    %r15,%rdi
   5ce6e:    4c 89 6c 24 08          mov    %r13,0x8(%rsp)
   5ce73:    e8 88 dc ff ff          callq  5ab00 <fdopen+0x450>
   5ce78:    48 8b 44 24 08          mov    0x8(%rsp),%rax
   5ce7d:    48 83 c4 18             add    $0x18,%rsp
   5ce81:    5b                      pop    %rbx
   5ce82:    5d                      pop    %rbp
   5ce83:    41 5c                   pop    %r12
   5ce85:    41 5d                   pop    %r13
   5ce87:    41 5e                   pop    %r14
   5ce89:    41 5f                   pop    %r15
   5ce8b:    c3                      retq
   5ce8c:    0f 1f 40 00             nopl   0x0(%rax)
```

fwrite:

```
000000000005d710 <fwrite>:
   5d710:    f3 0f 1e fa             endbr64
   5d714:    41 57                   push   %r15
   5d716:    b8 00 00 00 00          mov    $0x0,%eax
   5d71b:    41 56                   push   %r14
   5d71d:    49 89 f6                mov    %rsi,%r14
   5d720:    41 55                   push   %r13
   5d722:    4c 0f af f2             imul   %rdx,%r14
   5d726:    49 89 cd                mov    %rcx,%r13
   5d729:    41 54                   push   %r12
   5d72b:    49 89 fc                mov    %rdi,%r12
   5d72e:    55                      push   %rbp
   5d72f:    48 89 f5                mov    %rsi,%rbp
   5d732:    53                      push   %rbx
   5d733:    48 89 d3                mov    %rdx,%rbx
   5d736:    48 83 ec 08             sub    $0x8,%rsp
   5d73a:    48 85 f6                test   %rsi,%rsi
   5d73d:    48 0f 44 d8             cmove  %rax,%rbx
   5d741:    8b 81 8c 00 00 00       mov    0x8c(%rcx),%eax
   5d747:    85 c0                   test   %eax,%eax
   5d749:    79 35                   jns    5d780 <fwrite+0x70>
   5d74b:    48 89 ca                mov    %rcx,%rdx
   5d74e:    4c 89 f6                mov    %r14,%rsi
   5d751:    e8 ba fe ff ff          callq  5d610 <fwprintf+0xc0>
   5d756:    49 89 c4                mov    %rax,%r12
   5d759:    48 89 d8                mov    %rbx,%rax
   5d75c:    4d 39 e6                cmp    %r12,%r14
   5d75f:    74 08                   je     5d769 <fwrite+0x59>
   5d761:    4c 89 e0                mov    %r12,%rax
   5d764:    31 d2                   xor    %edx,%edx
   5d766:    48 f7 f5                div    %rbp
   5d769:    48 83 c4 08             add    $0x8,%rsp
   5d76d:    5b                      pop    %rbx
   5d76e:    5d                      pop    %rbp
   5d76f:    41 5c                   pop    %r12
   5d771:    41 5d                   pop    %r13
   5d773:    41 5e                   pop    %r14
   5d775:    41 5f                   pop    %r15
   5d777:    c3                      retq
   5d778:    0f 1f 84 00 00 00 00    nopl   0x0(%rax,%rax,1)
   5d77f:    00
   5d780:    48 89 cf                mov    %rcx,%rdi
```

```
   5d783:    e8 98 d2 ff ff           callq  5aa20 <fdopen+0x370>
   5d788:    4c 89 e7                 mov    %r12,%rdi
   5d78b:    4c 89 ea                 mov    %r13,%rdx
   5d78e:    4c 89 f6                 mov    %r14,%rsi
   5d791:    41 89 c7                 mov    %eax,%r15d
   5d794:    e8 77 fe ff ff           callq  5d610 <fwprintf+0xc0>
   5d799:    49 89 c4                 mov    %rax,%r12
   5d79c:    45 85 ff                 test   %r15d,%r15d
   5d79f:    74 b8                    je     5d759 <fwrite+0x49>
   5d7a1:    4c 89 ef                 mov    %r13,%rdi
   5d7a4:    e8 57 d3 ff ff           callq  5ab00 <fdopen+0x450>
   5d7a9:    eb ae                    jmp    5d759 <fwrite+0x49>
   5d7ab:    0f 1f 44 00 00           nopl   0x0(%rax,%rax,1)
```

feof:

```
000000000005b580 <feof>:
   5b580:    f3 0f 1e fa              endbr64
   5b584:    41 54                    push   %r12
   5b586:    55                       push   %rbp
   5b587:    48 89 fd                 mov    %rdi,%rbp
   5b58a:    48 83 ec 08              sub    $0x8,%rsp
   5b58e:    8b 87 8c 00 00 00        mov    0x8c(%rdi),%eax
   5b594:    85 c0                    test   %eax,%eax
   5b596:    79 18                    jns    5b5b0 <feof+0x30>
   5b598:    44 8b 27                 mov    (%rdi),%r12d
   5b59b:    41 c1 ec 04              shr    $0x4,%r12d
   5b59f:    41 83 e4 01              and    $0x1,%r12d
   5b5a3:    48 83 c4 08              add    $0x8,%rsp
   5b5a7:    44 89 e0                 mov    %r12d,%eax
   5b5aa:    5d                       pop    %rbp
   5b5ab:    41 5c                    pop    %r12
   5b5ad:    c3                       retq
   5b5ae:    66 90                    xchg   %ax,%ax
   5b5b0:    e8 6b f4 ff ff           callq  5aa20 <fdopen+0x370>
   5b5b5:    44 8b 65 00              mov    0x0(%rbp),%r12d
   5b5b9:    41 c1 ec 04              shr    $0x4,%r12d
   5b5bd:    41 83 e4 01              and    $0x1,%r12d
   5b5c1:    85 c0                    test   %eax,%eax
   5b5c3:    74 de                    je     5b5a3 <feof+0x23>
   5b5c5:    48 89 ef                 mov    %rbp,%rdi
   5b5c8:    e8 33 f5 ff ff           callq  5ab00 <fdopen+0x450>
   5b5cd:    48 83 c4 08              add    $0x8,%rsp
   5b5d1:    44 89 e0                 mov    %r12d,%eax
   5b5d4:    5d                       pop    %rbp
   5b5d5:    41 5c                    pop    %r12
   5b5d7:    c3                       retq
   5b5d8:    0f 1f 84 00 00 00 00     nopl   0x0(%rax,%rax,1)
   5b5df:    00
```

fclose:

```
000000000005b4a0 <fclose>:
   5b4a0:    f3 0f 1e fa              endbr64
   5b4a4:    41 55                    push   %r13
   5b4a6:    41 54                    push   %r12
   5b4a8:    55                       push   %rbp
   5b4a9:    8b 87 8c 00 00 00        mov    0x8c(%rdi),%eax
   5b4af:    48 89 fd                 mov    %rdi,%rbp
   5b4b2:    85 c0                    test   %eax,%eax
   5b4b4:    0f 89 7e 00 00 00        jns    5b538 <fclose+0x98>
   5b4ba:    e8 81 01 00 00           callq  5b640 <fflush>
   5b4bf:    48 89 ef                 mov    %rbp,%rdi
   5b4c2:    41 89 c4                 mov    %eax,%r12d
   5b4c5:    ff 55 18                 callq  *0x18(%rbp)
   5b4c8:    41 09 c4                 or     %eax,%r12d
   5b4cb:    f6 45 00 01              testb  $0x1,0x0(%rbp)
```

```
   5b4cf:    74 0f                   je     5b4e0 <fclose+0x40>
   5b4d1:    44 89 e0                mov    %r12d,%eax
   5b4d4:    5d                      pop    %rbp
   5b4d5:    41 5c                   pop    %r12
   5b4d7:    41 5d                   pop    %r13
   5b4d9:    c3                      retq
   5b4da:    66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
   5b4e0:    48 89 ef                mov    %rbp,%rdi
   5b4e3:    e8 08 1e 00 00          callq  5d2f0 <ftell+0x50>
   5b4e8:    e8 23 2b 00 00          callq  5e010 <getwchar+0x10>
   5b4ed:    48 8b 55 68             mov    0x68(%rbp),%rdx
   5b4f1:    48 85 d2                test   %rdx,%rdx
   5b4f4:    74 08                   je     5b4fe <fclose+0x5e>
   5b4f6:    48 8b 4d 70             mov    0x70(%rbp),%rcx
   5b4fa:    48 89 4a 70             mov    %rcx,0x70(%rdx)
   5b4fe:    48 8b 4d 70             mov    0x70(%rbp),%rcx
   5b502:    48 85 c9                test   %rcx,%rcx
   5b505:    74 04                   je     5b50b <fclose+0x6b>
   5b507:    48 89 51 68             mov    %rdx,0x68(%rcx)
   5b50b:    48 39 28                cmp    %rbp,(%rax)
   5b50e:    74 60                   je     5b570 <fclose+0xd0>
   5b510:    e8 1b 2b 00 00          callq  5e030 <getwchar+0x30>
   5b515:    48 8b bd a8 00 00 00    mov    0xa8(%rbp),%rdi
   5b51c:    e8 2f 9b fb ff          callq  15050 <free@plt>
   5b521:    48 89 ef                mov    %rbp,%rdi
   5b524:    e8 27 9b fb ff          callq  15050 <free@plt>
   5b529:    44 89 e0                mov    %r12d,%eax
   5b52c:    5d                      pop    %rbp
   5b52d:    41 5c                   pop    %r12
   5b52f:    41 5d                   pop    %r13
   5b531:    c3                      retq
   5b532:    66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
   5b538:    e8 e3 f4 ff ff          callq  5aa20 <fdopen+0x370>
   5b53d:    48 89 ef                mov    %rbp,%rdi
   5b540:    41 89 c5                mov    %eax,%r13d
   5b543:    e8 f8 00 00 00          callq  5b640 <fflush>
   5b548:    48 89 ef                mov    %rbp,%rdi
   5b54b:    41 89 c4                mov    %eax,%r12d
   5b54e:    ff 55 18                callq  *0x18(%rbp)
   5b551:    41 09 c4                or     %eax,%r12d
   5b554:    45 85 ed                test   %r13d,%r13d
   5b557:    0f 84 6e ff ff ff       je     5b4cb <fclose+0x2b>
   5b55d:    48 89 ef                mov    %rbp,%rdi
   5b560:    e8 9b f5 ff ff          callq  5ab00 <fdopen+0x450>
   5b565:    e9 61 ff ff ff          jmpq   5b4cb <fclose+0x2b>
   5b56a:    66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
   5b570:    48 89 08                mov    %rcx,(%rax)
   5b573:    eb 9b                   jmp    5b510 <fclose+0x70>
   5b575:    66 2e 0f 1f 84 00 00    nopw   %cs:0x0(%rax,%rax,1)
   5b57c:    00 00 00
   5b57f:    90                      nop
```

- виконання команд операційної системи system

```
000000000004ef80 <system>:
   4ef80:    f3 0f 1e fa             endbr64
   4ef84:    41 57                   push   %r15
   4ef86:    b9 12 00 00 00          mov    $0x12,%ecx
   4ef8b:    41 56                   push   %r14
   4ef8d:    41 55                   push   %r13
   4ef8f:    41 54                   push   %r12
   4ef91:    55                      push   %rbp
   4ef92:    53                      push   %rbx
   4ef93:    48 89 fb                mov    %rdi,%rbx
   4ef96:    48 81 ec 78 04 00 00    sub    $0x478,%rsp
   4ef9d:    64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
   4efa4:    00 00
```

```
   4efa6:    48 89 84 24 68 04 00       mov    %rax,0x468(%rsp)
   4efad:    00
   4efae:    31 c0                      xor    %eax,%eax
   4efb0:    48 8d ac 24 38 01 00       lea    0x138(%rsp),%rbp
   4efb7:    00
   4efb8:    48 c7 84 24 30 01 00       movq   $0x1,0x130(%rsp)
   4efbf:    00 01 00 00 00
   4efc4:    48 89 ef                   mov    %rbp,%rdi
   4efc7:    c7 44 24 0c ff ff ff       movl   $0xffffffff,0xc(%rsp)
   4efce:    ff
   4efcf:    f3 48 ab                   rep stos %rax,%es:(%rdi)
   4efd2:    e8 09 e5 01 00             callq  6d4e0 <pthread_testcancel>
   4efd7:    b8 01 00 00 00             mov    $0x1,%eax
   4efdc:    48 85 db                   test   %rbx,%rbx
   4efdf:    0f 84 80 01 00 00          je     4f165 <system+0x1e5>
   4efe5:    4c 8d a4 24 30 01 00       lea    0x130(%rsp),%r12
   4efec:    00
   4efed:    4c 8d b4 24 d0 01 00       lea    0x1d0(%rsp),%r14
   4eff4:    00
   4eff5:    bf 02 00 00 00             mov    $0x2,%edi
   4effa:    4c 89 e6                   mov    %r12,%rsi
   4effd:    4c 89 f2                   mov    %r14,%rdx
   4f000:    4c 8d ac 24 70 02 00       lea    0x270(%rsp),%r13
   4f007:    00
   4f008:    e8 d3 a2 00 00             callq  592e0 <sigaction>
   4f00d:    4c 89 ea                   mov    %r13,%rdx
   4f010:    4c 89 e6                   mov    %r12,%rsi
   4f013:    bf 03 00 00 00             mov    $0x3,%edi
   4f018:    e8 c3 a2 00 00             callq  592e0 <sigaction>
   4f01d:    4c 8d 64 24 30             lea    0x30(%rsp),%r12
   4f022:    be 11 00 00 00             mov    $0x11,%esi
   4f027:    48 89 ef                   mov    %rbp,%rdi
   4f02a:    e8 f1 a2 00 00             callq  59320 <sigaddset>
   4f02f:    31 ff                      xor    %edi,%edi
   4f031:    4c 89 e2                   mov    %r12,%rdx
   4f034:    48 89 ee                   mov    %rbp,%rsi
   4f037:    e8 34 a7 00 00             callq  59770 <sigprocmask>
   4f03c:    4c 8d bc 24 b0 00 00       lea    0xb0(%rsp),%r15
   4f043:    00
   4f044:    4c 89 ff                   mov    %r15,%rdi
   4f047:    e8 e4 a3 00 00             callq  59430 <sigemptyset>
   4f04c:    48 83 bc 24 d0 01 00       cmpq   $0x1,0x1d0(%rsp)
   4f053:    00 01
   4f055:    74 0d                      je     4f064 <system+0xe4>
   4f057:    be 02 00 00 00             mov    $0x2,%esi
   4f05c:    4c 89 ff                   mov    %r15,%rdi
   4f05f:    e8 bc a2 00 00             callq  59320 <sigaddset>
   4f064:    48 83 bc 24 70 02 00       cmpq   $0x1,0x270(%rsp)
   4f06b:    00 01
   4f06d:    74 0d                      je     4f07c <system+0xfc>
   4f06f:    be 03 00 00 00             mov    $0x3,%esi
   4f074:    4c 89 ff                   mov    %r15,%rdi
   4f077:    e8 a4 a2 00 00             callq  59320 <sigaddset>
   4f07c:    48 8d ac 24 10 03 00       lea    0x310(%rsp),%rbp
   4f083:    00
   4f084:    48 89 ef                   mov    %rbp,%rdi
   4f087:    e8 24 fd ff ff             callq  4edb0 <posix_spawnattr_init>
   4f08c:    4c 89 e6                   mov    %r12,%rsi
   4f08f:    48 89 ef                   mov    %rbp,%rdi
   4f092:    e8 e9 fd ff ff             callq  4ee80 <posix_spawnattr_setsigmask>
   4f097:    4c 89 fe                   mov    %r15,%rsi
   4f09a:    48 89 ef                   mov    %rbp,%rdi
   4f09d:    e8 8e fd ff ff             callq  4ee30
<posix_spawnattr_setsigdefault>
   4f0a2:    be 0c 00 00 00             mov    $0xc,%esi
   4f0a7:    48 89 ef                   mov    %rbp,%rdi
   4f0aa:    e8 51 fd ff ff             callq  4ee00 <posix_spawnattr_setflags>
   4f0af:    48 8d 7c 24 08             lea    0x8(%rsp),%rdi
```

```
   4f0b4:    31 d2                   xor    %edx,%edx
   4f0b6:    48 89 e9                mov    %rbp,%rcx
   4f0b9:    48 8d 05 6d ed 05 00    lea    0x5ed6d(%rip),%rax        # ade2d
<in6addr_loopback+0x447d>
   4f0c0:    4c 8d 44 24 10          lea    0x10(%rsp),%r8
   4f0c5:    48 89 5c 24 20          mov    %rbx,0x20(%rsp)
   4f0ca:    48 89 44 24 10          mov    %rax,0x10(%rsp)
   4f0cf:    48 8d 05 4f ed 05 00    lea    0x5ed4f(%rip),%rax        # ade25
<in6addr_loopback+0x4475>
   4f0d6:    48 8d 35 4b ed 05 00    lea    0x5ed4b(%rip),%rsi        # ade28
<in6addr_loopback+0x4478>
   4f0dd:    48 89 44 24 18          mov    %rax,0x18(%rsp)
   4f0e2:    48 8b 05 b7 0e 06 00    mov    0x60eb7(%rip),%rax        # affa0
<__environ-0x2f98>
   4f0e9:    48 c7 44 24 28 00 00    movq   $0x0,0x28(%rsp)
   4f0f0:    00 00
   4f0f2:    4c 8b 08                mov    (%rax),%r9
   4f0f5:    e8 46 f7 ff ff          callq  4e840 <posix_spawn>
   4f0fa:    48 89 ef                mov    %rbp,%rdi
   4f0fd:    89 c3                   mov    %eax,%ebx
   4f0ff:    e8 bc fb ff ff          callq  4ecc0 <posix_spawnattr_destroy>
   4f104:    85 db                   test   %ebx,%ebx
   4f106:    0f 85 84 00 00 00       jne    4f190 <system+0x210>
   4f10c:    48 8d 5c 24 0c          lea    0xc(%rsp),%rbx
   4f111:    eb 0f                   jmp    4f122 <system+0x1a2>
   4f113:    0f 1f 44 00 00          nopl   0x0(%rax,%rax,1)
   4f118:    e8 b3 01 fd ff          callq  1f2d0 <__errno_location>
   4f11d:    83 38 04                cmpl   $0x4,(%rax)
   4f120:    75 12                   jne    4f134 <system+0x1b4>
   4f122:    8b 7c 24 08             mov    0x8(%rsp),%edi
   4f126:    31 d2                   xor    %edx,%edx
   4f128:    48 89 de                mov    %rbx,%rsi
   4f12b:    e8 00 01 00 00          callq  4f230 <waitpid>
   4f130:    85 c0                   test   %eax,%eax
   4f132:    78 e4                   js     4f118 <system+0x198>
   4f134:    31 d2                   xor    %edx,%edx
   4f136:    4c 89 f6                mov    %r14,%rsi
   4f139:    bf 02 00 00 00          mov    $0x2,%edi
   4f13e:    e8 9d a1 00 00          callq  592e0 <sigaction>
   4f143:    31 d2                   xor    %edx,%edx
   4f145:    4c 89 ee                mov    %r13,%rsi
   4f148:    bf 03 00 00 00          mov    $0x3,%edi
   4f14d:    e8 8e a1 00 00          callq  592e0 <sigaction>
   4f152:    31 d2                   xor    %edx,%edx
   4f154:    4c 89 e6                mov    %r12,%rsi
   4f157:    bf 02 00 00 00          mov    $0x2,%edi
   4f15c:    e8 0f a6 00 00          callq  59770 <sigprocmask>
   4f161:    8b 44 24 0c             mov    0xc(%rsp),%eax
   4f165:    48 8b 8c 24 68 04 00    mov    0x468(%rsp),%rcx
   4f16c:    00
   4f16d:    64 48 33 0c 25 28 00    xor    %fs:0x28,%rcx
   4f174:    00 00
   4f176:    75 52                   jne    4f1ca <system+0x24a>
   4f178:    48 81 c4 78 04 00 00    add    $0x478,%rsp
   4f17f:    5b                      pop    %rbx
   4f180:    5d                      pop    %rbp
   4f181:    41 5c                   pop    %r12
   4f183:    41 5d                   pop    %r13
   4f185:    41 5e                   pop    %r14
   4f187:    41 5f                   pop    %r15
   4f189:    c3                      retq
   4f18a:    66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
   4f190:    31 d2                   xor    %edx,%edx
   4f192:    4c 89 f6                mov    %r14,%rsi
   4f195:    bf 02 00 00 00          mov    $0x2,%edi
   4f19a:    e8 41 a1 00 00          callq  592e0 <sigaction>
   4f19f:    31 d2                   xor    %edx,%edx
   4f1a1:    4c 89 ee                mov    %r13,%rsi
```

```
4f1a4:    bf 03 00 00 00           mov    $0x3,%edi
4f1a9:    e8 32 a1 00 00           callq  592e0 <sigaction>
4f1ae:    31 d2                    xor    %edx,%edx
4f1b0:    4c 89 e6                 mov    %r12,%rsi
4f1b3:    bf 02 00 00 00           mov    $0x2,%edi
4f1b8:    e8 b3 a5 00 00           callq  59770 <sigprocmask>
4f1bd:    e8 0e 01 fd ff           callq  1f2d0 <__errno_location>
4f1c2:    89 18                    mov    %ebx,(%rax)
4f1c4:    8b 44 24 0c              mov    0xc(%rsp),%eax
4f1c8:    eb 9b                    jmp    4f165 <system+0x1e5>
4f1ca:    e8 c1 fb fc ff           callq  1ed90 <__stack_chk_fail>
4f1cf:    90                       nop
```