ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ”

Факультет безопасности информационных технологий

Дисциплина:

“Информатика”

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

Основные принципы функционирования вычислительных устройств.

Выполнил:

Студент Чу Ван Доан

Группы N3147

Проверил:

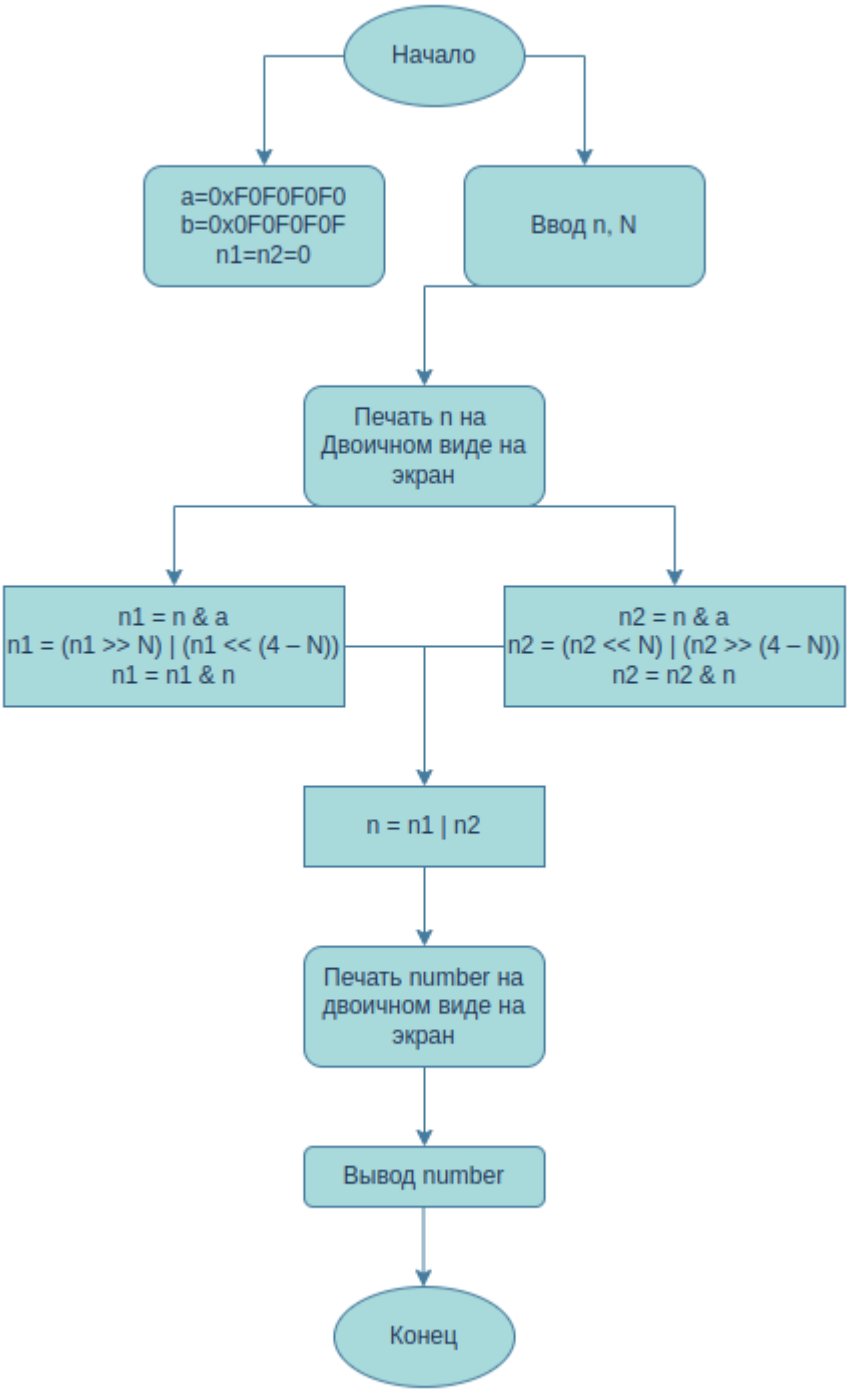
Горлина А.В.

Санкт-Петербург

2022г

Вариант 9: Назовем сверткой байта порядка N операцию циклического сдвига старшей тетрады на N битов вправо, а младшей тетрады на N битов влево. Выполнить свертку всех байтов на случайное число из диапазона 0..3.

I. Блок-схему алгоритма преобразования:



II.Текст программы с комментариями:

1. На языке С

```
#include <stdio.h>
void printBinary(unsigned int dec);
int main()
{
   unsigned int number, m1 = 0xF0F0F0F0, m2 = 0x0F0F0F0F, n1, n2;
   int N;
   printf("Enter your number: ");
   scanf("%x", &number);
   // Вводить число с клавиаптуры
   printf("Enter N: ");
   scanf("%d", &N);
   // Вводить значение N с клавиатуры
   printBinary(number);
   n1 = number & m1;
   n1 = (n1 >> N) | (n1 << (4 - N));
   // циклический сдвиг старшей тетрады на N битов впрова
   n1 = n1 & m1;
   n2 = number & m2;
   n2 = (n2 << N) | (n2 >> (4 - N));
   // циклический сдвиг старшей тетрады на N битов влева
   n2 = n2 & m2;
   number = n1 | n2;
   printBinary(number);
   return 0;
}
// выводить число на двоичном виде.
void printBinary(unsigned int dec)
{
   int k = 0, m;
   int mas[32];
   for (int i = 0; i < 32; i++)
       mas[i] = 0;
   while (dec > 0)
   {
       mas[k] = dec % 2;
       dec = dec / 2;
       k += 1;
   }
   for (m = 0; m < 32; m++)
   {
       printf("%d", mas[31 - m]);
       if (m % 4 == 3)
           printf(" ");
   }
   printf("\n");
}
```

2. На ассемблере:

```
section .data
       n db " "
section .text
       global _start
_start:
       push rbp
       mov rbp, rsp
       sub rsp, 0x20
       mov dword [rbp - 0x4], 627262 ;вводить значение числа.
       mov rax, [rbp - 0x4]
       call _printBinary       ;выводит число в двоичном виде на
экран.
       mov dword [rbp - 0x8], 0xf0f0f0f0
       mov dword [rbp - 0x12], 0x0f0f0f0f
       mov dword [rbp - 0x16], 0x2     ;вводить значение N.

       ;n1 = number & m1
       mov eax, dword [rbp - 0x4]
       and eax, dword [rbp - 0x8]      ;побитовое «И».
       mov dword [rbp - 0x20], eax

       ;n2 = number & m2
       mov eax, dword [rbp - 0x4]
       and eax, dword [rbp - 0x12]     ;побитовое «И» .
       mov dword [rbp - 0x24], eax

       ; циклический сдвиг старшей тетрады на N битов впрова
       mov eax, dword [rbp - 0x16]     ;n1 >> N.
       mov edx, dword [rbp - 0x20]
       mov esi, edx
       mov ecx, eax
       shr esi, cl     ;простой побитовый сдвиг вправо,количество
.
                       ;сдвига впрова хранил в регистре eax.

       mov eax, 0x4    ;n1 << (4-N).
       sub eax, dword [rbp - 0x16]
       mov edx, dword [rbp - 0x20]
       mov ecx, eax
       shl edx, cl     ;простой побитовый сдвиг влево,количество .
                       ;сдвига впрова хранил в регистре eax..
       mov eax, edx
       or eax, esi     ;(n1 >> N) | (n1 << (4-N)).

       ;n1 = n1 & m1
       mov dword [rbp - 0x20], eax
       mov eax, dword [rbp - 0x8]
```

```
        and dword [rbp - 0x20], eax

        ;циклический сдвиг старшей тетрады на N битов влева
        mov eax, dword [rbp - 0x16]     ;n2 << N.
        mov edx, dword [rbp - 0x24]
        mov esi, edx
        mov ecx, eax
        shl esi, cl

        mov eax, 0x4    ;n2 >> (4-N).
        sub eax, dword [rbp - 0x16]
        mov edx, dword [rbp - 0x24]
        mov ecx, eax
        shr edx, cl

        mov eax, edx    ;(n2 << N) | (n1 >> (4-N)).
        or eax, esi

        ;n2 = n2 & m2
        mov dword [rbp - 0x24], eax
        mov eax, dword [rbp - 0x12]
        and dword [rbp - 0x24], eax     ;побитовое «И».
        mov eax,dword [rbp - 0x18]

        ;number = n1 | n2
        mov eax, dword [rbp - 0x20]
        or eax, dword [rbp - 0x24]      ;побитовое «ИЛИ».
        mov dword [rbp - 0x4], eax

        mov rax, [rbp - 0x4]
        call _printBinary       ;выводить новое значение числа в
двоичном
                                ;виде на экран.
        mov rax, 60             ;Завершить программу.
        xor rdi, rdi
        syscall
_printBinary:
        mov rbx,0x0000000080000000
        mov rcx,32

for:
        mov rdx,rax
        and rdx,rbx
        cmp rdx,0       ;сравнить значение в rdx с 0.
        je zero         ;переход если значение в rdx равно 0.
        jne one         ;переход если значение в rdx не равно 0.

zero:   mov rdx,0
```

```
        add rdx,48      ;перевод символов в цифры.
        jmp endif       ; безусловный переход.

one:     mov rdx,1
        add rdx,48      ;перевод символов в цифры.

endif:
        push rax        ;Занесение значение rax в стек
        mov rax,1
        mov rdi,1
        mov [n],dl
        mov rsi,n
        mov rdx,1
        push rcx        ;Занесение значение rcx в стек

        syscall
        pop rcx         ;Извлечение значение rcx из стека
        pop rax         ;Извлечение значение rax из стека
        shr rbx,1       ;простой побитовый сдвиг вправо 1 раз
        loop for

        mov rax,1
        mov rdi,1
        mov rdx,10
        mov [n],rdx
        mov rsi,n
        mov rdx,1
        syscall         ;вывод на экран
     ret
```

III. Дизассемблерный листинг существенных частей программы на C с добавленными комментариями или пояснениями.

```
Disassembly of section .init:

0000000000001000 <_init>:
    1000:  f3 0f 1e fa           endbr64
    1004:  48 83 ec 08           sub    $0x8,%rsp
    1008:  48 8b 05 d9 2f 00 00  mov    0x2fd9(%rip),%rax        #
3fe8 <__gmon_start__@Base>
    100f:  48 85 c0              test   %rax,%rax
    1012:  74 02                 je     1016 <_init+0x16>
    1014:  ff d0                 call   *%rax
    1016:  48 83 c4 08           add    $0x8,%rsp
    101a:  c3                    ret

Disassembly of section .plt:
```

```
0000000000001020 <.plt>:
    1020:  ff 35 82 2f 00 00     push   0x2f82(%rip)        # 3fa8
<_GLOBAL_OFFSET_TABLE_+0x8>
    1026:  f2 ff 25 83 2f 00 00  bnd jmp *0x2f83(%rip)        #
3fb0 <_GLOBAL_OFFSET_TABLE_+0x10>
    102d:  0f 1f 00              nopl   (%rax)
    1030:  f3 0f 1e fa           endbr64
    1034:  68 00 00 00 00        push   $0x0
    1039:  f2 e9 e1 ff ff ff     bnd jmp 1020 <_init+0x20>
    103f:  90                    nop
    1040:  f3 0f 1e fa           endbr64
    1044:  68 01 00 00 00        push   $0x1
    1049:  f2 e9 d1 ff ff ff     bnd jmp 1020 <_init+0x20>
    104f:  90                    nop
    1050:  f3 0f 1e fa           endbr64
    1054:  68 02 00 00 00        push   $0x2
    1059:  f2 e9 c1 ff ff ff     bnd jmp 1020 <_init+0x20>
    105f:  90                    nop
    1060:  f3 0f 1e fa           endbr64
    1064:  68 03 00 00 00        push   $0x3
    1069:  f2 e9 b1 ff ff ff     bnd jmp 1020 <_init+0x20>
    106f:  90                    nop

Disassembly of section .plt.got:

0000000000001070 <__cxa_finalize@plt>:
    1070:  f3 0f 1e fa           endbr64
    1074:  f2 ff 25 7d 2f 00 00  bnd jmp *0x2f7d(%rip)        #
3ff8 <__cxa_finalize@GLIBC_2.2.5>
    107b:  0f 1f 44 00 00        nopl   0x0(%rax,%rax,1)

Disassembly of section .plt.sec:

0000000000001080 <putchar@plt>:
    1080:  f3 0f 1e fa           endbr64
    1084:  f2 ff 25 2d 2f 00 00  bnd jmp *0x2f2d(%rip)        #
3fb8 <putchar@GLIBC_2.2.5>
    108b:  0f 1f 44 00 00        nopl   0x0(%rax,%rax,1)

0000000000001090 <__stack_chk_fail@plt>:
    1090:  f3 0f 1e fa           endbr64
    1094:  f2 ff 25 25 2f 00 00  bnd jmp *0x2f25(%rip)        #
3fc0 <__stack_chk_fail@GLIBC_2.4>
    109b:  0f 1f 44 00 00        nopl   0x0(%rax,%rax,1)

00000000000010a0 <printf@plt>:
    10a0:  f3 0f 1e fa           endbr64
```

```
    10a4:  f2 ff 25 1d 2f 00 00 bnd jmp *0x2f1d(%rip)        #
3fc8 <printf@GLIBC_2.2.5>
    10ab:  0f 1f 44 00 00       nopl   0x0(%rax,%rax,1)

00000000000010b0 <__isoc99_scanf@plt>:
    10b0:  f3 0f 1e fa          endbr64
    10b4:  f2 ff 25 15 2f 00 00 bnd jmp *0x2f15(%rip)        #
3fd0 <__isoc99_scanf@GLIBC_2.7>
    10bb:  0f 1f 44 00 00       nopl   0x0(%rax,%rax,1)

Disassembly of section .text:

00000000000010c0 <_start>:
    10c0:  f3 0f 1e fa          endbr64
    10c4:  31 ed                xor    %ebp,%ebp
    10c6:  49 89 d1             mov    %rdx,%r9
    10c9:  5e                   pop    %rsi
    10ca:  48 89 e2             mov    %rsp,%rdx
    10cd:  48 83 e4 f0          and    $0xfffffffffffffff0,%rsp
    10d1:  50                   push   %rax
    10d2:  54                   push   %rsp
    10d3:  45 31 c0             xor    %r8d,%r8d
    10d6:  31 c9                xor    %ecx,%ecx
    10d8:  48 8d 3d ca 00 00 00 lea    0xca(%rip),%rdi        #
11a9 <main>
    10df:  ff 15 f3 2e 00 00    call   *0x2ef3(%rip)        # 3fd8
<__libc_start_main@GLIBC_2.34>
    10e5:  f4                   hlt
    10e6:  66 2e 0f 1f 84 00 00 cs nopw 0x0(%rax,%rax,1)
    10ed:  00 00 00

00000000000010f0 <deregister_tm_clones>:
    10f0:  48 8d 3d 19 2f 00 00 lea    0x2f19(%rip),%rdi        #
4010 <__TMC_END__>
    10f7:  48 8d 05 12 2f 00 00 lea    0x2f12(%rip),%rax        #
4010 <__TMC_END__>
    10fe:  48 39 f8             cmp    %rdi,%rax
    1101:  74 15                je     1118
<deregister_tm_clones+0x28>
    1103:  48 8b 05 d6 2e 00 00 mov    0x2ed6(%rip),%rax        #
3fe0 <_ITM_deregisterTMCloneTable@Base>
    110a:  48 85 c0             test   %rax,%rax
    110d:  74 09                je     1118
<deregister_tm_clones+0x28>
    110f:  ff e0                jmp    *%rax
    1111:  0f 1f 80 00 00 00 00 nopl   0x0(%rax)
    1118:  c3                   ret
    1119:  0f 1f 80 00 00 00 00 nopl   0x0(%rax)
```

```
0000000000001120 <register_tm_clones>:
    1120:  48 8d 3d e9 2e 00 00  lea    0x2ee9(%rip),%rdi        #
4010 <__TMC_END__>
    1127:  48 8d 35 e2 2e 00 00  lea    0x2ee2(%rip),%rsi        #
4010 <__TMC_END__>
    112e:  48 29 fe              sub    %rdi,%rsi
    1131:  48 89 f0              mov    %rsi,%rax
    1134:  48 c1 ee 3f           shr    $0x3f,%rsi
    1138:  48 c1 f8 03           sar    $0x3,%rax
    113c:  48 01 c6              add    %rax,%rsi
    113f:  48 d1 fe              sar    %rsi
    1142:  74 14                 je     1158
<register_tm_clones+0x38>
    1144:  48 8b 05 a5 2e 00 00  mov    0x2ea5(%rip),%rax        #
3ff0 <_ITM_registerTMCloneTable@Base>
    114b:  48 85 c0              test   %rax,%rax
    114e:  74 08                 je     1158
<register_tm_clones+0x38>
    1150:  ff e0                 jmp    *%rax
    1152:  66 0f 1f 44 00 00     nopw   0x0(%rax,%rax,1)
    1158:  c3                    ret
    1159:  0f 1f 80 00 00 00 00  nopl   0x0(%rax)

0000000000001160 <__do_global_dtors_aux>:
    1160:  f3 0f 1e fa           endbr64
    1164:  80 3d a5 2e 00 00 00  cmpb   $0x0,0x2ea5(%rip)        #
4010 <__TMC_END__>
    116b:  75 2b                 jne    1198
<__do_global_dtors_aux+0x38>
    116d:  55                    push   %rbp
    116e:  48 83 3d 82 2e 00 00  cmpq   $0x0,0x2e82(%rip)        #
3ff8 <__cxa_finalize@GLIBC_2.2.5>
    1175:  00
    1176:  48 89 e5              mov    %rsp,%rbp
    1179:  74 0c                 je     1187
<__do_global_dtors_aux+0x27>
    117b:  48 8b 3d 86 2e 00 00  mov    0x2e86(%rip),%rdi        #
4008 <__dso_handle>
    1182:  e8 e9 fe ff ff        call   1070 <__cxa_finalize@plt>
    1187:  e8 64 ff ff ff        call   10f0 <deregister_tm_clones>
    118c:  c6 05 7d 2e 00 00 01  movb   $0x1,0x2e7d(%rip)        #
4010 <__TMC_END__>
    1193:  5d                    pop    %rbp
    1194:  c3                    ret
    1195:  0f 1f 00              nopl   (%rax)
    1198:  c3                    ret
    1199:  0f 1f 80 00 00 00 00  nopl   0x0(%rax)
```

```
00000000000011a0 <frame_dummy>:
    11a0:  f3 0f 1e fa           endbr64
    11a4:  e9 77 ff ff ff        jmp    1120 <register_tm_clones>

00000000000011a9 <main>:
    11a9:  f3 0f 1e fa           endbr64
    11ad:  55                    push   %rbp
    11ae:  48 89 e5              mov    %rsp,%rbp
    11b1:  48 83 ec 20           sub    $0x20,%rsp
    11b5:  64 48 8b 04 25 28 00  mov    %fs:0x28,%rax
    11bc:  00 00
    11be:  48 89 45 f8           mov    %rax,-0x8(%rbp)
    11c2:  31 c0                 xor    %eax,%eax
    11c4:  c7 45 e8 f0 f0 f0 f0  movl   $0xf0f0f0f0,-0x18(%rbp)
    11cb:  c7 45 ec 0f 0f 0f 0f  movl   $0xf0f0f0f,-0x14(%rbp)
    11d2:  48 8d 05 2b 0e 00 00  lea    0xe2b(%rip),%rax        #
2004 <_IO_stdin_used+0x4>
    11d9:  48 89 c7              mov    %rax,%rdi
    11dc:  b8 00 00 00 00        mov    $0x0,%eax
    11e1:  e8 ba fe ff ff        call   10a0 <printf@plt>
    11e6:  48 8d 45 e0           lea    -0x20(%rbp),%rax
    11ea:  48 89 c6              mov    %rax,%rsi
    11ed:  48 8d 05 24 0e 00 00  lea    0xe24(%rip),%rax        #
2018 <_IO_stdin_used+0x18>
    11f4:  48 89 c7              mov    %rax,%rdi
    11f7:  b8 00 00 00 00        mov    $0x0,%eax
    11fc:  e8 af fe ff ff        call   10b0 <__isoc99_scanf@plt>
    1201:  48 8d 05 13 0e 00 00  lea    0xe13(%rip),%rax        #
201b <_IO_stdin_used+0x1b>
    1208:  48 89 c7              mov    %rax,%rdi
    120b:  b8 00 00 00 00        mov    $0x0,%eax
    1210:  e8 8b fe ff ff        call   10a0 <printf@plt>
    1215:  48 8d 45 e4           lea    -0x1c(%rbp),%rax
    1219:  48 89 c6              mov    %rax,%rsi
    121c:  48 8d 05 02 0e 00 00  lea    0xe02(%rip),%rax        #
2025 <_IO_stdin_used+0x25>
    1223:  48 89 c7              mov    %rax,%rdi
    1226:  b8 00 00 00 00        mov    $0x0,%eax
    122b:  e8 80 fe ff ff        call   10b0 <__isoc99_scanf@plt>
    1230:  8b 45 e0              mov    -0x20(%rbp),%eax
    1233:  89 c7                 mov    %eax,%edi
    1235:  e8 94 00 00 00        call   12ce <printBinary>
    123a:  8b 45 e0              mov    -0x20(%rbp),%eax
    123d:  23 45 e8              and    -0x18(%rbp),%eax
    1240:  89 45 f0              mov    %eax,-0x10(%rbp)
    1243:  8b 45 e4              mov    -0x1c(%rbp),%eax
    1246:  8b 55 f0              mov    -0x10(%rbp),%edx
```

```
1249:  89 d6                mov    %edx,%esi
124b:  89 c1                mov    %eax,%ecx
124d:  d3 ee                shr    %cl,%esi
124f:  8b 55 e4             mov    -0x1c(%rbp),%edx
1252:  b8 04 00 00 00       mov    $0x4,%eax
1257:  29 d0                sub    %edx,%eax
1259:  8b 55 f0             mov    -0x10(%rbp),%edx
125c:  89 c1                mov    %eax,%ecx
125e:  d3 e2                shl    %cl,%edx
1260:  89 d0                mov    %edx,%eax
1262:  09 f0                or     %esi,%eax
1264:  89 45 f0             mov    %eax,-0x10(%rbp)
1267:  8b 45 e8             mov    -0x18(%rbp),%eax
126a:  21 45 f0             and    %eax,-0x10(%rbp)
126d:  8b 45 e0             mov    -0x20(%rbp),%eax
1270:  23 45 ec             and    -0x14(%rbp),%eax
1273:  89 45 f4             mov    %eax,-0xc(%rbp)
1276:  8b 45 e4             mov    -0x1c(%rbp),%eax
1279:  8b 55 f4             mov    -0xc(%rbp),%edx
127c:  89 d6                mov    %edx,%esi
127e:  89 c1                mov    %eax,%ecx
1280:  d3 e6                shl    %cl,%esi
1282:  8b 55 e4             mov    -0x1c(%rbp),%edx
1285:  b8 04 00 00 00       mov    $0x4,%eax
128a:  29 d0                sub    %edx,%eax
128c:  8b 55 f4             mov    -0xc(%rbp),%edx
128f:  89 c1                mov    %eax,%ecx
1291:  d3 ea                shr    %cl,%edx
1293:  89 d0                mov    %edx,%eax
1295:  09 f0                or     %esi,%eax
1297:  89 45 f4             mov    %eax,-0xc(%rbp)
129a:  8b 45 ec             mov    -0x14(%rbp),%eax
129d:  21 45 f4             and    %eax,-0xc(%rbp)
12a0:  8b 45 f0             mov    -0x10(%rbp),%eax
12a3:  0b 45 f4             or     -0xc(%rbp),%eax
12a6:  89 45 e0             mov    %eax,-0x20(%rbp)
12a9:  8b 45 e0             mov    -0x20(%rbp),%eax
12ac:  89 c7                mov    %eax,%edi
12ae:  e8 1b 00 00 00       call   12ce <printBinary>
12b3:  b8 00 00 00 00       mov    $0x0,%eax
12b8:  48 8b 55 f8          mov    -0x8(%rbp),%rdx
12bc:  64 48 2b 14 25 28 00 sub    %fs:0x28,%rdx
12c3:  00 00
12c5:  74 05                je     12cc <main+0x123>
12c7:  e8 c4 fd ff ff       call   1090 <__stack_chk_fail@plt>
12cc:  c9                   leave
12cd:  c3                   ret
```

```
00000000000012ce <printBinary>:
    12ce:  f3 0f 1e fa           endbr64
    12d2:  55                    push   %rbp
    12d3:  48 89 e5              mov    %rsp,%rbp
    12d6:  48 81 ec b0 00 00 00  sub    $0xb0,%rsp
    12dd:  89 bd 5c ff ff ff     mov    %edi,-0xa4(%rbp)
    12e3:  64 48 8b 04 25 28 00  mov    %fs:0x28,%rax
    12ea:  00 00
    12ec:  48 89 45 f8           mov    %rax,-0x8(%rbp)
    12f0:  31 c0                 xor    %eax,%eax
    12f2:  c7 85 64 ff ff ff 00  movl   $0x0,-0x9c(%rbp)
    12f9:  00 00 00
    12fc:  c7 85 6c ff ff ff 00  movl   $0x0,-0x94(%rbp)
    1303:  00 00 00
    1306:  eb 1a                 jmp    1322 <printBinary+0x54>
    1308:  8b 85 6c ff ff ff     mov    -0x94(%rbp),%eax
    130e:  48 98                 cltq
    1310:  c7 84 85 70 ff ff ff  movl   $0x0,-0x90(%rbp,%rax,4)
    1317:  00 00 00 00
    131b:  83 85 6c ff ff ff 01  addl   $0x1,-0x94(%rbp)
    1322:  83 bd 6c ff ff ff 1f  cmpl   $0x1f,-0x94(%rbp)
    1329:  7e dd                 jle    1308 <printBinary+0x3a>
    132b:  eb 2f                 jmp    135c <printBinary+0x8e>
    132d:  8b 85 5c ff ff ff     mov    -0xa4(%rbp),%eax
    1333:  83 e0 01              and    $0x1,%eax
    1336:  89 c2                 mov    %eax,%edx
    1338:  8b 85 64 ff ff ff     mov    -0x9c(%rbp),%eax
    133e:  48 98                 cltq
    1340:  89 94 85 70 ff ff ff  mov    %edx,-0x90(%rbp,%rax,4)
    1347:  8b 85 5c ff ff ff     mov    -0xa4(%rbp),%eax
    134d:  d1 e8                 shr    %eax
    134f:  89 85 5c ff ff ff     mov    %eax,-0xa4(%rbp)
    1355:  83 85 64 ff ff ff 01  addl   $0x1,-0x9c(%rbp)
    135c:  83 bd 5c ff ff ff 00  cmpl   $0x0,-0xa4(%rbp)
    1363:  75 c8                 jne    132d <printBinary+0x5f>
    1365:  c7 85 68 ff ff ff 00  movl   $0x0,-0x98(%rbp)
    136c:  00 00 00
    136f:  eb 51                 jmp    13c2 <printBinary+0xf4>
    1371:  b8 1f 00 00 00        mov    $0x1f,%eax
    1376:  2b 85 68 ff ff ff     sub    -0x98(%rbp),%eax
    137c:  48 98                 cltq
    137e:  8b 84 85 70 ff ff ff  mov    -0x90(%rbp,%rax,4),%eax
    1385:  89 c6                 mov    %eax,%esi
    1387:  48 8d 05 97 0c 00 00  lea    0xc97(%rip),%rax        #
2025 <_IO_stdin_used+0x25>
    138e:  48 89 c7              mov    %rax,%rdi
    1391:  b8 00 00 00 00        mov    $0x0,%eax
    1396:  e8 05 fd ff ff        call   10a0 <printf@plt>
```

```
    139b:  8b 85 68 ff ff ff    mov    -0x98(%rbp),%eax
    13a1:  99                   cltd
    13a2:  c1 ea 1e             shr    $0x1e,%edx
    13a5:  01 d0                add    %edx,%eax
    13a7:  83 e0 03             and    $0x3,%eax
    13aa:  29 d0                sub    %edx,%eax
    13ac:  83 f8 03             cmp    $0x3,%eax
    13af:  75 0a                jne    13bb <printBinary+0xed>
    13b1:  bf 20 00 00 00       mov    $0x20,%edi
    13b6:  e8 c5 fc ff ff       call   1080 <putchar@plt>
    13bb:  83 85 68 ff ff ff 01 addl   $0x1,-0x98(%rbp)
    13c2:  83 bd 68 ff ff ff 1f cmpl   $0x1f,-0x98(%rbp)
    13c9:  7e a6                jle    1371 <printBinary+0xa3>
    13cb:  bf 0a 00 00 00       mov    $0xa,%edi
    13d0:  e8 ab fc ff ff       call   1080 <putchar@plt>
    13d5:  90                   nop
    13d6:  48 8b 45 f8          mov    -0x8(%rbp),%rax
    13da:  64 48 2b 04 25 28 00 sub    %fs:0x28,%rax
    13e1:  00 00
    13e3:  74 05                je     13ea <printBinary+0x11c>
    13e5:  e8 a6 fc ff ff       call   1090 <__stack_chk_fail@plt>
    13ea:  c9                   leave
    13eb:  c3                   ret

Disassembly of section .fini:

00000000000013ec <_fini>:
    13ec:  f3 0f 1e fa          endbr64
    13f0:  48 83 ec 08          sub    $0x8,%rsp
    13f4:  48 83 c4 08          add    $0x8,%rsp
    13f8:  c3                   ret

Disassembly of section .rodata:

0000000000002000 <_IO_stdin_used>:
    2000:  01 00                add    %eax,(%rax)
    2002:  02 00                add    (%rax),%al
    2004:  45 6e                rex.RB outsb %ds:(%rsi),(%dx)
    2006:  74 65                je     206d
<__GNU_EH_FRAME_HDR+0x45>
    2008:  72 20                jb     202a
<__GNU_EH_FRAME_HDR+0x2>
    200a:  79 6f                jns    207b
<__GNU_EH_FRAME_HDR+0x53>
    200c:  75 72                jne    2080
<__GNU_EH_FRAME_HDR+0x58>
    200e:  20 6e 75             and    %ch,0x75(%rsi)
    2011:  6d                   insl   (%dx),%es:(%rdi)
```

```
    2012:  62 65                (bad)
    2014:  72 3a                jb     2050
<__GNU_EH_FRAME_HDR+0x28>
    2016:  20 00                and    %al,(%rax)
    2018:  25 78 00 45 6e       and    $0x6e450078,%eax
    201d:  74 65                je     2084
<__GNU_EH_FRAME_HDR+0x5c>
    201f:  72 20                jb     2041
<__GNU_EH_FRAME_HDR+0x19>
    2021:  4e 3a 20             rex.WRX cmp (%rax),%r12b
    2024:  00                   .byte 0x0
    2025:  25                   .byte 0x25
    2026:  64                   fs
      ...

Disassembly of section .eh_frame_hdr:

0000000000002028 <__GNU_EH_FRAME_HDR>:
    2028:  01 1b                add    %ebx,(%rbx)
    202a:  03 3b                add    (%rbx),%edi
    202c:  3c 00                cmp    $0x0,%al
    202e:  00 00                add    %al,(%rax)
    2030:  06                   (bad)
    2031:  00 00                add    %al,(%rax)
    2033:  00 f8                add    %bh,%al
    2035:  ef                   out    %eax,(%dx)
    2036:  ff                   (bad)
    2037:  ff 70 00             push   0x0(%rax)
    203a:  00 00                add    %al,(%rax)
    203c:  48                   rex.W
    203d:  f0 ff                lock (bad)
    203f:  ff 98 00 00 00 58    lcall  *0x58000000(%rax)
    2045:  f0 ff                lock (bad)
    2047:  ff b0 00 00 00 98    push   -0x68000000(%rax)
    204d:  f0 ff                lock (bad)
    204f:  ff 58 00             lcall  *0x0(%rax)
    2052:  00 00                add    %al,(%rax)
    2054:  81 f1 ff ff c8 00    xor    $0xc8ffff,%ecx
    205a:  00 00                add    %al,(%rax)
    205c:  a6                   cmpsb  %es:(%rdi),%ds:(%rsi)
    205d:  f2 ff                repnz (bad)
    205f:  ff                   (bad)
    2060:  e8                   .byte 0xe8
    2061:  00 00                add    %al,(%rax)
      ...

Disassembly of section .eh_frame:
```

```
0000000000002068 <__FRAME_END__-0xc8>:
    2068:  14 00                 adc    $0x0,%al
    206a:  00 00                 add    %al,(%rax)
    206c:  00 00                 add    %al,(%rax)
    206e:  00 00                 add    %al,(%rax)
    2070:  01 7a 52              add    %edi,0x52(%rdx)
    2073:  00 01                 add    %al,(%rcx)
    2075:  78 10                 js     2087
<__GNU_EH_FRAME_HDR+0x5f>
    2077:  01 1b                 add    %ebx,(%rbx)
    2079:  0c 07                 or     $0x7,%al
    207b:  08 90 01 00 00 14     or     %dl,0x14000001(%rax)
    2081:  00 00                 add    %al,(%rax)
    2083:  00 1c 00              add    %bl,(%rax,%rax,1)
    2086:  00 00                 add    %al,(%rax)
    2088:  38 f0                 cmp    %dh,%al
    208a:  ff                    (bad)
    208b:  ff 26                 jmp    *(%rsi)
    208d:  00 00                 add    %al,(%rax)
    208f:  00 00                 add    %al,(%rax)
    2091:  44 07                 rex.R (bad)
    2093:  10 00                 adc    %al,(%rax)
    2095:  00 00                 add    %al,(%rax)
    2097:  00 24 00              add    %ah,(%rax,%rax,1)
    209a:  00 00                 add    %al,(%rax)
    209c:  34 00                 xor    $0x0,%al
    209e:  00 00                 add    %al,(%rax)
    20a0:  80 ef ff              sub    $0xff,%bh
    20a3:  ff 50 00              call   *0x0(%rax)
    20a6:  00 00                 add    %al,(%rax)
    20a8:  00 0e                 add    %cl,(%rsi)
    20aa:  10 46 0e              adc    %al,0xe(%rsi)
    20ad:  18 4a 0f              sbb    %cl,0xf(%rdx)
    20b0:  0b 77 08              or     0x8(%rdi),%esi
    20b3:  80 00 3f              addb   $0x3f,(%rax)
    20b6:  1a 3a                 sbb    (%rdx),%bh
    20b8:  2a 33                 sub    (%rbx),%dh
    20ba:  24 22                 and    $0x22,%al
    20bc:  00 00                 add    %al,(%rax)
    20be:  00 00                 add    %al,(%rax)
    20c0:  14 00                 adc    $0x0,%al
    20c2:  00 00                 add    %al,(%rax)
    20c4:  5c                    pop    %rsp
    20c5:  00 00                 add    %al,(%rax)
    20c7:  00 a8 ef ff ff 10     add    %ch,0x10ffffef(%rax)
      ...
    20d5:  00 00                 add    %al,(%rax)
    20d7:  00 14 00              add    %dl,(%rax,%rax,1)
```

```
    20da:  00 00                 add    %al,(%rax)
    20dc:  74 00                 je     20de
<__GNU_EH_FRAME_HDR+0xb6>
    20de:  00 00                 add    %al,(%rax)
    20e0:  a0 ef ff ff 40 00 00 movabs 0x40ffffef,%al
    20e7:  00 00
    20e9:  00 00                 add    %al,(%rax)
    20eb:  00 00                 add    %al,(%rax)
    20ed:  00 00                 add    %al,(%rax)
    20ef:  00 1c 00              add    %bl,(%rax,%rax,1)
    20f2:  00 00                 add    %al,(%rax)
    20f4:  8c 00                 mov    %es,(%rax)
    20f6:  00 00                 add    %al,(%rax)
    20f8:  b1 f0                 mov    $0xf0,%cl
    20fa:  ff                    (bad)
    20fb:  ff 25 01 00 00 00     jmp    *0x1(%rip)        # 2102
<__GNU_EH_FRAME_HDR+0xda>
    2101:  45 0e                 rex.RB (bad)
    2103:  10 86 02 43 0d 06     adc    %al,0x60d4302(%rsi)
    2109:  03 1c 01              add    (%rcx,%rax,1),%ebx
    210c:  0c 07                 or     $0x7,%al
    210e:  08 00                 or     %al,(%rax)
    2110:  1c 00                 sbb    $0x0,%al
    2112:  00 00                 add    %al,(%rax)
    2114:  ac                    lods   %ds:(%rsi),%al
    2115:  00 00                 add    %al,(%rax)
    2117:  00 b6 f1 ff ff 1e     add    %dh,0x1efffff1(%rsi)
    211d:  01 00                 add    %eax,(%rax)
    211f:  00 00                 add    %al,(%rax)
    2121:  45 0e                 rex.RB (bad)
    2123:  10 86 02 43 0d 06     adc    %al,0x60d4302(%rsi)
    2129:  03 15 01 0c 07 08     add    0x8070c01(%rip),%edx
# 8072d30 <_end+0x806ed18>
      ...

0000000000002130 <__FRAME_END__>:
    2130:  00 00                 add    %al,(%rax)
      ...

Disassembly of section .init_array:

0000000000003da0 <__frame_dummy_init_array_entry>:
    3da0:  a0                    .byte 0xa0
    3da1:  11 00                 adc    %eax,(%rax)
    3da3:  00 00                 add    %al,(%rax)
    3da5:  00 00                 add    %al,(%rax)
      ...
```

```
Disassembly of section .fini_array:

0000000000003da8 <__do_global_dtors_aux_fini_array_entry>:
    3da8:  60                    (bad)
    3da9:  11 00                 adc    %eax,(%rax)
    3dab:  00 00                 add    %al,(%rax)
    3dad:  00 00                 add    %al,(%rax)
      ...
```

- Шестнадцатеричные числа слева, начиная с 0x1000, являются адресами памяти.
- Второй столбец содержит инструкции машинного языка, которые процессор x64 считывает как двоичные значения. Наример 01001110110111, objdump будет отображать двоичный файл как шестнадцатеричный, чтобы сделать его более удобочитаемым форматом.
- Последний правый столбец содержит ассемблерную версию инструкций машинного языка.

IV. Краткий анализ по результатам сравнения программы на ассемблере и дизассемблированной программы на C.

- Программа на ассемблере короче чем, дизассемблированняа программы на C.
- Программа, написанная на языке ассемблера, может состоять из нескольких частей, называемых модулями. В каждом модуле могут быть определены один или несколько сегментов данных, стека и кода. Любая законченная программа на ассемблере должна включать один главный, или основной, модуль, с которого начинается ее выполнение.

V. Скриншоты прогонов программ на различных исходных данных.

```
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$ gcc code.c -o code
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$ ./code
Enter your number: 123abcdf
Enter N: 1
0001 0010 0011 1010 1011 1100 1101 1111
1000 0100 1001 0101 1101 1001 1110 1111
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$ nasm -f elf64 -o main.o main.asm
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$ ld main.o -o main
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$ ./main
00010010001110101011110011011111
10000100100101011101100111101111
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$
```

```
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$ gcc code.c -o code
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$ ./code
Enter your number: 12345678
Enter N: 2
0001 0010 0011 0100 0101 0110 0111 1000
0100 1000 1100 0001 0101 1001 1101 0010
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$ nasm -f elf64 -o main.o main.asm
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$ ld main.o -o main
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$ ./main
00010010001101000101011001111000
01001000110000010101100111010010
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$
```

```
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$ gcc code.c -o code
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$ ./code
Enter your number: abcdef
Enter N: 3
0000 0000 1010 1011 1100 1101 1110 1111
0000 0000 0101 1101 1001 1110 1101 1111
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$ nasm -f elf64 -o main.o main.asm
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$ ld main.o -o main
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$ ./main
00000000101010111001101111011111
00000000010111011001111011011111
chudoan@chudoan-Latitude-5510:~/Desktop/lab2$
```