**ФЕДЕРАЛЬНО ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«**Национальный исследовательский университет ИТМО**»**

Факультет безопасности информационных технологий



**ЛАБОРАТОРНАЯ РАБОТА №2**

по дисциплине «Информатика»



Студент:

Чан Ван Хоанг

Группы: N3149



Преподаватель

Грозов В.А.

Санкт Петербург

2020

**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:**

m1= 0xF0F0F0F0

m2 = 0x0F0F0F0F

n1=n2=0

Ввод number, N

Печать number на двоичном виде на экран

n1 = number & m1

n1 = (n1 >> N) | (n1 << (4-N));

n1 = n1 & m1;

n2 = number & m2

n2 = (n2 << N) | (n2 >> (4-N));

n2 = n2 & m2;

number = n1 | n2

Печать number на двоичном виде на экран

Вывод number

**1. На языке C**

#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 клавиаnтуры***

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], 0x8ab6cd4e ;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 *;(n1 >> N) | (n1 << (4-N)).*

or eax, esi

**;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 ;вывод на экран

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 С с добавленными комментариями или пояснениями.**

Disassembly of section .init:

0000000000001000 <\_init>:

1000: f3 0f 1e fa endbr64

1004: 48 83 ec 08 sub rsp,0x8

1008: 48 8b 05 d9 2f 00 00 mov rax,QWORD PTR [rip+0x2fd9] # 3fe8 <\_\_gmon\_start\_\_>

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 rsp,0x8

101a: c3 ret

Disassembly of section .plt:

0000000000001020 <.plt>:

1020: ff 35 82 2f 00 00 push QWORD PTR [rip+0x2f82] # 3fa8 <\_GLOBAL\_OFFSET\_TABLE\_+0x8>

1026: f2 ff 25 83 2f 00 00 bnd jmp QWORD PTR [rip+0x2f83] # 3fb0 <\_GLOBAL\_OFFSET\_TABLE\_+0x10>

102d: 0f 1f 00 nop DWORD PTR [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 <.plt>

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 <.plt>

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 <.plt>

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 <.plt>

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 QWORD PTR [rip+0x2f7d] # 3ff8 <\_\_cxa\_finalize@GLIBC\_2.2.5>

107b: 0f 1f 44 00 00 nop DWORD PTR [rax+rax\*1+0x0]

Disassembly of section .plt.sec:

0000000000001080 <putchar@plt>:

1080: f3 0f 1e fa endbr64

1084: f2 ff 25 2d 2f 00 00 bnd jmp QWORD PTR [rip+0x2f2d] # 3fb8 <putchar@GLIBC\_2.2.5>

108b: 0f 1f 44 00 00 nop DWORD PTR [rax+rax\*1+0x0]

0000000000001090 <\_\_stack\_chk\_fail@plt>:

1090: f3 0f 1e fa endbr64

1094: f2 ff 25 25 2f 00 00 bnd jmp QWORD PTR [rip+0x2f25] # 3fc0 <\_\_stack\_chk\_fail@GLIBC\_2.4>

109b: 0f 1f 44 00 00 nop DWORD PTR [rax+rax\*1+0x0]

00000000000010a0 <printf@plt>:

10a0: f3 0f 1e fa endbr64

10a4: f2 ff 25 1d 2f 00 00 bnd jmp QWORD PTR [rip+0x2f1d] # 3fc8 <printf@GLIBC\_2.2.5>

10ab: 0f 1f 44 00 00 nop DWORD PTR [rax+rax\*1+0x0]

00000000000010b0 <\_\_isoc99\_scanf@plt>:

10b0: f3 0f 1e fa endbr64

10b4: f2 ff 25 15 2f 00 00 bnd jmp QWORD PTR [rip+0x2f15] # 3fd0 <\_\_isoc99\_scanf@GLIBC\_2.7>

10bb: 0f 1f 44 00 00 nop DWORD PTR [rax+rax\*1+0x0]

Disassembly of section .text:

00000000000010c0 <\_start>:

10c0: f3 0f 1e fa endbr64

10c4: 31 ed xor ebp,ebp

10c6: 49 89 d1 mov r9,rdx

10c9: 5e pop rsi

10ca: 48 89 e2 mov rdx,rsp

10cd: 48 83 e4 f0 and rsp,0xfffffffffffffff0

10d1: 50 push rax

10d2: 54 push rsp

10d3: 4c 8d 05 86 03 00 00 lea r8,[rip+0x386] # 1460 <\_\_libc\_csu\_fini>

10da: 48 8d 0d 0f 03 00 00 lea rcx,[rip+0x30f] # 13f0 <\_\_libc\_csu\_init>

10e1: 48 8d 3d c1 00 00 00 lea rdi,[rip+0xc1] # 11a9 <main>

10e8: ff 15 f2 2e 00 00 call QWORD PTR [rip+0x2ef2] # 3fe0 <\_\_libc\_start\_main@GLIBC\_2.2.5>

10ee: f4 hlt

10ef: 90 nop

00000000000010f0 <deregister\_tm\_clones>:

10f0: 48 8d 3d 19 2f 00 00 lea rdi,[rip+0x2f19] # 4010 <\_\_TMC\_END\_\_>

10f7: 48 8d 05 12 2f 00 00 lea rax,[rip+0x2f12] # 4010 <\_\_TMC\_END\_\_>

10fe: 48 39 f8 cmp rax,rdi

1101: 74 15 je 1118 <deregister\_tm\_clones+0x28>

1103: 48 8b 05 ce 2e 00 00 mov rax,QWORD PTR [rip+0x2ece] # 3fd8 <\_ITM\_deregisterTMCloneTable>

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 nop DWORD PTR [rax+0x0]

1118: c3 ret

1119: 0f 1f 80 00 00 00 00 nop DWORD PTR [rax+0x0]

0000000000001120 <register\_tm\_clones>:

1120: 48 8d 3d e9 2e 00 00 lea rdi,[rip+0x2ee9] # 4010 <\_\_TMC\_END\_\_>

1127: 48 8d 35 e2 2e 00 00 lea rsi,[rip+0x2ee2] # 4010 <\_\_TMC\_END\_\_>

112e: 48 29 fe sub rsi,rdi

1131: 48 89 f0 mov rax,rsi

1134: 48 c1 ee 3f shr rsi,0x3f

1138: 48 c1 f8 03 sar rax,0x3

113c: 48 01 c6 add rsi,rax

113f: 48 d1 fe sar rsi,1

1142: 74 14 je 1158 <register\_tm\_clones+0x38>

1144: 48 8b 05 a5 2e 00 00 mov rax,QWORD PTR [rip+0x2ea5] # 3ff0 <\_ITM\_registerTMCloneTable>

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 nop WORD PTR [rax+rax\*1+0x0]

1158: c3 ret

1159: 0f 1f 80 00 00 00 00 nop DWORD PTR [rax+0x0]

0000000000001160 <\_\_do\_global\_dtors\_aux>:

1160: f3 0f 1e fa endbr64

1164: 80 3d a5 2e 00 00 00 cmp BYTE PTR [rip+0x2ea5],0x0 # 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 cmp QWORD PTR [rip+0x2e82],0x0 # 3ff8 <\_\_cxa\_finalize@GLIBC\_2.2.5>

1175: 00

1176: 48 89 e5 mov rbp,rsp

1179: 74 0c je 1187 <\_\_do\_global\_dtors\_aux+0x27>

117b: 48 8b 3d 86 2e 00 00 mov rdi,QWORD PTR [rip+0x2e86] # 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 mov BYTE PTR [rip+0x2e7d],0x1 # 4010 <\_\_TMC\_END\_\_>

1193: 5d pop rbp

1194: c3 ret

1195: 0f 1f 00 nop DWORD PTR [rax]

1198: c3 ret

1199: 0f 1f 80 00 00 00 00 nop DWORD PTR [rax+0x0]

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 rbp,rsp

11b1: 48 83 ec 20 sub rsp,0x20

11b5: 64 48 8b 04 25 28 00 mov rax,QWORD PTR fs:0x28

11bc: 00 00

11be: 48 89 45 f8 mov QWORD PTR [rbp-0x8],rax

11c2: 31 c0 xor eax,eax

11c4: c7 45 e8 f0 f0 f0 f0 mov DWORD PTR [rbp-0x18],0xf0f0f0f0

11cb: c7 45 ec 0f 0f 0f 0f mov DWORD PTR [rbp-0x14],0xf0f0f0f

11d2: 48 8d 3d 2b 0e 00 00 lea rdi,[rip+0xe2b] # 2004 <\_IO\_stdin\_used+0x4>

11d9: b8 00 00 00 00 mov eax,0x0

11de: e8 bd fe ff ff call 10a0 <printf@plt>

11e3: 48 8d 45 e0 lea rax,[rbp-0x20]

11e7: 48 89 c6 mov rsi,rax

11ea: 48 8d 3d 27 0e 00 00 lea rdi,[rip+0xe27] # 2018 <\_IO\_stdin\_used+0x18>

11f1: b8 00 00 00 00 mov eax,0x0

11f6: e8 b5 fe ff ff call 10b0 <\_\_isoc99\_scanf@plt>

11fb: 48 8d 3d 19 0e 00 00 lea rdi,[rip+0xe19] # 201b <\_IO\_stdin\_used+0x1b>

1202: b8 00 00 00 00 mov eax,0x0

1207: e8 94 fe ff ff call 10a0 <printf@plt>

120c: 48 8d 45 e4 lea rax,[rbp-0x1c]

1210: 48 89 c6 mov rsi,rax

1213: 48 8d 3d 0b 0e 00 00 lea rdi,[rip+0xe0b] # 2025 <\_IO\_stdin\_used+0x25>

121a: b8 00 00 00 00 mov eax,0x0

121f: e8 8c fe ff ff call 10b0 <\_\_isoc99\_scanf@plt>

1224: 8b 45 e0 mov eax,DWORD PTR [rbp-0x20]

1227: 89 c7 mov edi,eax

1229: e8 98 00 00 00 call 12c6 <printBinary>

122e: 8b 45 e0 mov eax,DWORD PTR [rbp-0x20]

1231: 23 45 e8 and eax,DWORD PTR [rbp-0x18]

1234: 89 45 f0 mov DWORD PTR [rbp-0x10],eax

1237: 8b 45 e4 mov eax,DWORD PTR [rbp-0x1c]

123a: 8b 55 f0 mov edx,DWORD PTR [rbp-0x10]

123d: 89 d6 mov esi,edx

123f: 89 c1 mov ecx,eax

1241: d3 ee shr esi,cl

1243: 8b 45 e4 mov eax,DWORD PTR [rbp-0x1c]

1246: ba 04 00 00 00 mov edx,0x4

124b: 29 c2 sub edx,eax

124d: 89 d0 mov eax,edx

124f: 8b 55 f0 mov edx,DWORD PTR [rbp-0x10]

1252: 89 c1 mov ecx,eax

1254: d3 e2 shl edx,cl

1256: 89 d0 mov eax,edx

1258: 09 f0 or eax,esi

125a: 89 45 f0 mov DWORD PTR [rbp-0x10],eax

125d: 8b 45 e8 mov eax,DWORD PTR [rbp-0x18]

1260: 21 45 f0 and DWORD PTR [rbp-0x10],eax

1263: 8b 45 e0 mov eax,DWORD PTR [rbp-0x20]

1266: 23 45 ec and eax,DWORD PTR [rbp-0x14]

1269: 89 45 f4 mov DWORD PTR [rbp-0xc],eax

126c: 8b 45 e4 mov eax,DWORD PTR [rbp-0x1c]

126f: 8b 55 f4 mov edx,DWORD PTR [rbp-0xc]

1272: 89 d6 mov esi,edx

1274: 89 c1 mov ecx,eax

1276: d3 e6 shl esi,cl

1278: 8b 45 e4 mov eax,DWORD PTR [rbp-0x1c]

127b: ba 04 00 00 00 mov edx,0x4

1280: 29 c2 sub edx,eax

1282: 89 d0 mov eax,edx

1284: 8b 55 f4 mov edx,DWORD PTR [rbp-0xc]

1287: 89 c1 mov ecx,eax

1289: d3 ea shr edx,cl

128b: 89 d0 mov eax,edx

128d: 09 f0 or eax,esi

128f: 89 45 f4 mov DWORD PTR [rbp-0xc],eax

1292: 8b 45 ec mov eax,DWORD PTR [rbp-0x14]

1295: 21 45 f4 and DWORD PTR [rbp-0xc],eax

1298: 8b 45 f0 mov eax,DWORD PTR [rbp-0x10]

129b: 0b 45 f4 or eax,DWORD PTR [rbp-0xc]

129e: 89 45 e0 mov DWORD PTR [rbp-0x20],eax

12a1: 8b 45 e0 mov eax,DWORD PTR [rbp-0x20]

12a4: 89 c7 mov edi,eax

12a6: e8 1b 00 00 00 call 12c6 <printBinary>

12ab: b8 00 00 00 00 mov eax,0x0

12b0: 48 8b 7d f8 mov rdi,QWORD PTR [rbp-0x8]

12b4: 64 48 33 3c 25 28 00 xor rdi,QWORD PTR fs:0x28

12bb: 00 00

12bd: 74 05 je 12c4 <main+0x11b>

12bf: e8 cc fd ff ff call 1090 <\_\_stack\_chk\_fail@plt>

12c4: c9 leave

12c5: c3 ret

00000000000012c6 <printBinary>:

12c6: f3 0f 1e fa endbr64

12ca: 55 push rbp

12cb: 48 89 e5 mov rbp,rsp

12ce: 48 81 ec b0 00 00 00 sub rsp,0xb0

12d5: 89 bd 5c ff ff ff mov DWORD PTR [rbp-0xa4],edi

12db: 64 48 8b 04 25 28 00 mov rax,QWORD PTR fs:0x28

12e2: 00 00

12e4: 48 89 45 f8 mov QWORD PTR [rbp-0x8],rax

12e8: 31 c0 xor eax,eax

12ea: c7 85 64 ff ff ff 00 mov DWORD PTR [rbp-0x9c],0x0

12f1: 00 00 00

12f4: c7 85 6c ff ff ff 00 mov DWORD PTR [rbp-0x94],0x0

12fb: 00 00 00

12fe: eb 1a jmp 131a <printBinary+0x54>

1300: 8b 85 6c ff ff ff mov eax,DWORD PTR [rbp-0x94]

1306: 48 98 cdqe

1308: c7 84 85 70 ff ff ff mov DWORD PTR [rbp+rax\*4-0x90],0x0

130f: 00 00 00 00

1313: 83 85 6c ff ff ff 01 add DWORD PTR [rbp-0x94],0x1

131a: 83 bd 6c ff ff ff 1f cmp DWORD PTR [rbp-0x94],0x1f

1321: 7e dd jle 1300 <printBinary+0x3a>

1323: eb 2f jmp 1354 <printBinary+0x8e>

1325: 8b 85 5c ff ff ff mov eax,DWORD PTR [rbp-0xa4]

132b: 83 e0 01 and eax,0x1

132e: 89 c2 mov edx,eax

1330: 8b 85 64 ff ff ff mov eax,DWORD PTR [rbp-0x9c]

1336: 48 98 cdqe

1338: 89 94 85 70 ff ff ff mov DWORD PTR [rbp+rax\*4-0x90],edx

133f: 8b 85 5c ff ff ff mov eax,DWORD PTR [rbp-0xa4]

1345: d1 e8 shr eax,1

1347: 89 85 5c ff ff ff mov DWORD PTR [rbp-0xa4],eax

134d: 83 85 64 ff ff ff 01 add DWORD PTR [rbp-0x9c],0x1

1354: 83 bd 5c ff ff ff 00 cmp DWORD PTR [rbp-0xa4],0x0

135b: 75 c8 jne 1325 <printBinary+0x5f>

135d: c7 85 68 ff ff ff 00 mov DWORD PTR [rbp-0x98],0x0

1364: 00 00 00

1367: eb 4e jmp 13b7 <printBinary+0xf1>

1369: b8 1f 00 00 00 mov eax,0x1f

136e: 2b 85 68 ff ff ff sub eax,DWORD PTR [rbp-0x98]

1374: 48 98 cdqe

1376: 8b 84 85 70 ff ff ff mov eax,DWORD PTR [rbp+rax\*4-0x90]

137d: 89 c6 mov esi,eax

137f: 48 8d 3d 9f 0c 00 00 lea rdi,[rip+0xc9f] # 2025 <\_IO\_stdin\_used+0x25>

1386: b8 00 00 00 00 mov eax,0x0

138b: e8 10 fd ff ff call 10a0 <printf@plt>

1390: 8b 85 68 ff ff ff mov eax,DWORD PTR [rbp-0x98]

1396: 99 cdq

1397: c1 ea 1e shr edx,0x1e

139a: 01 d0 add eax,edx

139c: 83 e0 03 and eax,0x3

139f: 29 d0 sub eax,edx

13a1: 83 f8 03 cmp eax,0x3

13a4: 75 0a jne 13b0 <printBinary+0xea>

13a6: bf 20 00 00 00 mov edi,0x20

13ab: e8 d0 fc ff ff call 1080 <putchar@plt>

13b0: 83 85 68 ff ff ff 01 add DWORD PTR [rbp-0x98],0x1

13b7: 83 bd 68 ff ff ff 1f cmp DWORD PTR [rbp-0x98],0x1f

13be: 7e a9 jle 1369 <printBinary+0xa3>

13c0: bf 0a 00 00 00 mov edi,0xa

13c5: e8 b6 fc ff ff call 1080 <putchar@plt>

13ca: 90 nop

13cb: 48 8b 45 f8 mov rax,QWORD PTR [rbp-0x8]

13cf: 64 48 33 04 25 28 00 xor rax,QWORD PTR fs:0x28

13d6: 00 00

13d8: 74 05 je 13df <printBinary+0x119>

13da: e8 b1 fc ff ff call 1090 <\_\_stack\_chk\_fail@plt>

13df: c9 leave

13e0: c3 ret

13e1: 66 2e 0f 1f 84 00 00 nop WORD PTR cs:[rax+rax\*1+0x0]

13e8: 00 00 00

13eb: 0f 1f 44 00 00 nop DWORD PTR [rax+rax\*1+0x0]

00000000000013f0 <\_\_libc\_csu\_init>:

13f0: f3 0f 1e fa endbr64

13f4: 41 57 push r15

13f6: 4c 8d 3d a3 29 00 00 lea r15,[rip+0x29a3] # 3da0 <\_\_frame\_dummy\_init\_array\_entry>

13fd: 41 56 push r14

13ff: 49 89 d6 mov r14,rdx

1402: 41 55 push r13

1404: 49 89 f5 mov r13,rsi

1407: 41 54 push r12

1409: 41 89 fc mov r12d,edi

140c: 55 push rbp

140d: 48 8d 2d 94 29 00 00 lea rbp,[rip+0x2994] # 3da8 <\_\_do\_global\_dtors\_aux\_fini\_array\_entry>

1414: 53 push rbx

1415: 4c 29 fd sub rbp,r15

1418: 48 83 ec 08 sub rsp,0x8

141c: e8 df fb ff ff call 1000 <\_init>

1421: 48 c1 fd 03 sar rbp,0x3

1425: 74 1f je 1446 <\_\_libc\_csu\_init+0x56>

1427: 31 db xor ebx,ebx

1429: 0f 1f 80 00 00 00 00 nop DWORD PTR [rax+0x0]

1430: 4c 89 f2 mov rdx,r14

1433: 4c 89 ee mov rsi,r13

1436: 44 89 e7 mov edi,r12d

1439: 41 ff 14 df call QWORD PTR [r15+rbx\*8]

143d: 48 83 c3 01 add rbx,0x1

1441: 48 39 dd cmp rbp,rbx

1444: 75 ea jne 1430 <\_\_libc\_csu\_init+0x40>

1446: 48 83 c4 08 add rsp,0x8

144a: 5b pop rbx

144b: 5d pop rbp

144c: 41 5c pop r12

144e: 41 5d pop r13

1450: 41 5e pop r14

1452: 41 5f pop r15

1454: c3 ret

1455: 66 66 2e 0f 1f 84 00 data16 nop WORD PTR cs:[rax+rax\*1+0x0]

145c: 00 00 00 00

0000000000001460 <\_\_libc\_csu\_fini>:

1460: f3 0f 1e fa endbr64

1464: c3 ret

Disassembly of section .fini:

0000000000001468 <\_fini>:

1468: f3 0f 1e fa endbr64

146c: 48 83 ec 08 sub rsp,0x8

1470: 48 83 c4 08 add rsp,0x8

1474: c3 ret

- 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 С.**

**-** Программа на ассемблере короче чем, дизассемблированняа программы на С.

- 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.**







