

Лабораторна робота №1

ФІ-91 Житкевич Іван

Стан реєстрів на кожному пункті

1. Массив елементів та дії віднімання та ділення на них

The screenshot shows the Immunity Debugger interface with two main panes. The left pane displays the assembly code for a C program, and the right pane shows the state of various registers at specific program points. A vertical blue line on the left marks the current assembly code being analyzed.

Assembly View:

```
[X] Disassembly [Cache] Off
255: int main (int argc, char **argv, char **envp);
    ; DATA XREF from entry0 @ 0x559f88928050
    ; var_int64_t var_30h @ rbp-0x38
    ; var_int64_t var_34h @ rbp-0x34
    ; var_int64_t var_30h @ rbp-0x30
    ; var_int64_t var_2ch @ rbp-0x2c
    ; var_int64_t var_28h @ rbp-0x28
    ; var_int64_t var_24h @ rbp-0x24
    ; var_int64_t var_20h @ rbp-0x20
    ; var_int64_t var_1ch @ rbp-0x1c
    ; var_int64_t var_10h @ rbp-0x10
    ; var_int64_t var_14h @ rbp-0x14
    ; var_int64_t var_10h @ rbp-0x10
    ; var_int64_t var_8h @ rbp-0x8
0x559f88928150 b 55 push rbp
0x559f88928151 489e5 mov rbp, rsp
0x559f88928154 483ec4b sub rsp, 0x40
0x559f88928158 64488b42528. mov rax, qword fs:[0x28]
0x559f88928161 4894f58 mov qword [var_8h], rax
0x559f88928165 31c0 xor eax, eax
;-- rip:
0x559f88928167 c745d0050000. mov dword [var_30h], 5
0x559f88928168 c745d4830000. mov dword [var_2ch], 3
0x559f88928175 c745d8880000. mov dword [var_28h], 8
0x559f88928176 c745dc010000. mov dword [var_24h], 1
0x559f88928183 8b55d8 mov edx, dword [var_30h]
0x559f88928186 8b45d8 mov eax, dword [var_28h]
0x559f88928189 29c2 sub edx, eax
0x559f8892818b 8955c8 mov dword [var_38h], edx
0x559f8892818d 8b45d4 mov eax, dword [var_2ch]
0x559f88928191 8b45dc mov ecx, dword [var_24h]
0x559f88928194 99 cdq
0x559f88928195 f7f9 idiv ecx
0x559f88928197 8945cc mov dword [var_34h], eax
0x559f88928198 8b45d4 mov eax, dword [var_2ch]
0x559f8892819d 83f82a cmp eax, 0x2a ; 42
0x559f889281a0 7597 jne 0x559f889281a9
0x559f889281a2 c745d4180000. mov dword [var_2ch], 0x18 ; 24
0x559f889281a9 8b45d0 mov eax, dword [var_30h]
0x559f889281ac 83f805 cmp eax, 5 ; 5
0x559f889281af 743a je 0x559f889281eb
0x559f889281b1 83f805 cmp eax, 5 ; 5
0x559f889281b4 74fa je 0x559f88928208
0x559f889281b6 83f804 cmp eax, 4 ; 4
0x559f889281b9 7427 je 0x559f889281e2
```

Registers View:

```
[X] Locals (Locals)
var var_8h = 0x7ffff3802da68 = (qword)0x65bda1a9f443500
var var_30h = 0x7ffff3802da40 = (qword)0x0000000000000000
var var_2ch = 0x7ffff3802da44 = (qword)0x0000000000000000
var var_28h = 0x7ffff3802da48 = (qword)0x0000000000000000
var var_24h = 0x7ffff3802da4c = (qword)0x0000000000000000
var var_38h = 0x7ffff3802da58 = (qword)0x0000000000000000
var var_20h = 0x7ffff3802da50 = (qword)0x0000000000000000
var var_1ch = 0x7ffff3802da54 = (qword)0x595a6b6000000000
var var_18h = 0x7ffff3802da58 = (qword)0x000007f3f595a6b0
var var_14h = 0x7ffff3802da5c = (qword)0x0000000000000000
var var_10h = 0x7ffff3802da60 = (qword)0x0000000000000000

[Tab [1]] [0x559f88928150]
[X] Registers (Registers)
rax = 0x00000000
rbx = 0x7ffff3802db88
rcx = 0x559f8892ad8
rdx = 0x7ffff3802db98
r8 = 0x00000000
r9 = 0x7f3f5958e890
r10 = 0x7ffff3802d7a0
r11 = 0x00000204
r12 = 0x00000000
r13 = 0x7ffff3802db98
r14 = 0x559f8892ad8
r15 = 0x7f3f595bd000
rsi = 0x7ffff3802db88
rdi = 0x00000001
rsp = 0x7ffff3802da30
rbp = 0x7ffff3802da70
rip = 0x559f88928167
rflags = 0x00000246
orax = 0xffffffffffffffffffff
```

За цей пункт відповідає код на С:

```
int arr[] = { 5, 3, 8, 1 };
int res1 = arr[0] - arr[2];
int res2 = arr[1] / arr[3];
```

Йому відповідає така діляка дизассемблізованого коду

| | | |
|----------------|---------------|--------------------------|
| 0x5622b6d2d167 | c745d0050000. | mov dword [var_30h], 5 |
| 0x5622b6d2d16e | c745d4030000. | mov dword [var_2ch], 3 |
| 0x5622b6d2d175 | c745d8080000. | mov dword [var_28h], 8 |
| 0x5622b6d2d17c | c745dc010000. | mov dword [var_24h], 1 |
| 0x5622b6d2d183 | 8b55d0 | mov edx, dword [var_30h] |
| 0x5622b6d2d186 | 8b45d8 | mov eax, dword [var_28h] |
| 0x5622b6d2d189 | 29c2 | sub edx, eax |
| 0x5622b6d2d18b | 8955c8 | mov dword [var_38h], edx |
| 0x5622b6d2d18e | 8b45d4 | mov eax, dword [var_2ch] |
| 0x5622b6d2d191 | 8b4ddc | mov ecx, dword [var_24h] |
| 0x5622b6d2d194 | 99 | cdq |
| 0x5622b6d2d195 | f7f9 | idiv ecx |
| 0x5622b6d2d197 | 8945cc | mov dword [var_34h], eax |

Як елементи масиву дебагер показує локальні змінні var_30h, var_2ch, var_28h, var_24h, що відповідають елементам 0, 1, 2 та 3 масиву.

Також ми бачимо:

- запис (адреси 67, 6e, 75, 7c) захардкожених даних у елементи масиву
- процес віднімання та запису результату виконання опкоду sub з регістру edx у область змінної var_38h по адресам 83, 86, 89, 8b. Для віднімання використовуються регістри edx (data register) та eax (primary accumulator).
- за ділення відповідають команди за адресами 8e, 91, 94, 95. idiv записує результат у регістри eax та edx

Значення у регістрах: - на момент запису нульового та другого елементів у edx та eax перед відніманням та після операції віднімання результат у регистрі rdx

```
[X] Registers (Registers)
rax = 0x00000008
rbx = 0x7fff3802db88
rcx = 0x559f8892add8
rdx = 0x00000005
r8 = 0x00000000
r9 = 0x7f3f5958e890
r10 = 0x7fff3802d7a0
r11 = 0x00000206
r12 = 0x00000000
r13 = 0x7fff3802db98
r14 = 0x559f8892add8
r15 = 0x7f3f595bd000
rsi = 0x7fff3802db88
rdi = 0x00000001
rsp = 0x7fff3802da30
rbp = 0x7fff3802da70
rip = 0x559f88928189
rflags = 0x00000246
orax = 0xffffffffffffffff
```

```
[X] Registers (Registers)
rax = 0x00000008
rbx = 0x7fff3802db88
rcx = 0x559f8892add8
rdx = 0xfffffffdf
r8 = 0x00000000
r9 = 0x7f3f5958e890
r10 = 0x7fff3802d7a0
r11 = 0x00000206
r12 = 0x00000000
r13 = 0x7fff3802db98
r14 = 0x559f8892add8
r15 = 0x7f3f595bd000
rsi = 0x7fff3802db88
rdi = 0x00000001
rsp = 0x7fff3802da30
rbp = 0x7fff3802da70
rip = 0x559f8892818b
rflags = 0x00000293
orax = 0xffffffffffffffff
```

- запис даних у регістри eax та ecx для ділення, після ділення

```
[X] Registers (Registers)
rax = 0x00000003
rbx = 0x7fff3802db88
rcx = 0x00000001
rdx = 0xfffffffffd
```

```
[X] Registers (Registers)
rax = 0x00000003
rbx = 0x7fff3802db88
rcx = 0x00000001
rdx = 0x00000000
```

2. Блок if

```
if (arr[1] == 42) {
    arr[1] = 24;
}
```

Йому відповідає

| | | |
|-----------------------------|----------------------|---------------------------------------|
| 0x5622b6d2d19a | 8b45d4 | mov eax, dword [var_2ch] |
| 0x5622b6d2d19d | 83f82a | cmp eax, 0x2a ; 42 |
| └< 0x5622b6d2d1a0 | 7507 | jne 0x5622b6d2d1a9 |
| 0x5622b6d2d1a2 | c745d4180000. | mov dword [var_2ch], 0x18 ; 24 |

Де записується значення першого елементу масиву у регістр eax та далі порівнюємо це значення із значенням 0x2a (42). Якщо ж два значення не рівні, то виконуємо перехід на

адресу 0x5622b6d2d1a9, інакше записуємо у перший елемент масиву 0x18 (24).

| | | | |
|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| [X] Registers (Registers) | [X] Registers (Registers) | [X] Registers (Registers) | [X] Registers (Registers) |
| rax = 0x00000003 | rax = 0x00000003 | rax = 0x00000003 | rax = 0x00000003 |
| rbx = 0x7ffffea9c9988 | rbx = 0x7ffffea9c9988 | rbx = 0x7ffffea9c9988 | rbx = 0x7fffcfeb70e98 |
| rcx = 0x00000001 | rcx = 0x00000001 | rcx = 0x00000001 | rcx = 0x00000001 |
| rdx = 0x00000000 | rdx = 0x00000000 | rdx = 0x00000000 | rdx = 0x00000000 |
| r8 = 0x00000000 | r8 = 0x00000000 | r8 = 0x00000000 | r8 = 0x00000000 |
| r9 = 0x7fc8393b9890 | r9 = 0x7fc8393b9890 | r9 = 0x7fc8393b9890 | r9 = 0x7f9ca9334890 |
| r10 = 0x7ffffea9c95a0 | r10 = 0x7ffffea9c95a0 | r10 = 0x7ffffea9c95a0 | r10 = 0x7fffcfeb70ab0 |
| r11 = 0x00000206 | r11 = 0x00000206 | r11 = 0x00000206 | r11 = 0x00000202 |
| r12 = 0x00000000 | r12 = 0x00000000 | r12 = 0x00000000 | r12 = 0x00000000 |
| r13 = 0x7ffffea9c9988 | r13 = 0x7ffffea9c9988 | r13 = 0x7ffffea9c9988 | r13 = 0x7fffcfeb70ea8 |
| r14 = 0x5622b6d2fdd8 | r14 = 0x5622b6d2fdd8 | r14 = 0x5622b6d2fdd8 | r14 = 0x560fb9686dd8 |
| r15 = 0x7fc8393e8000 | r15 = 0x7fc8393e8000 | r15 = 0x7fc8393e8000 | r15 = 0x7f9ca9363000 |
| rsi = 0x7ffffea9c9988 | rsi = 0x7ffffea9c9988 | rsi = 0x7ffffea9c9988 | rsi = 0x7fffcfeb70e98 |
| rdi = 0x00000001 | rdi = 0x00000001 | rdi = 0x00000001 | rdi = 0x00000001 |
| rsp = 0x7ffffea9c9830 | rsp = 0x7ffffea9c9830 | rsp = 0x7ffffea9c9830 | rsp = 0x7fffcfeb70d40 |
| rbp = 0x7ffffea9c9870 | rbp = 0x7ffffea9c9870 | rbp = 0x7ffffea9c9870 | rbp = 0x7fffcfeb70d80 |
| rip = 0x5622b6d2d19a | rip = 0x5622b6d2d19d | rip = 0x5622b6d2d1a0 | rip = 0x560fb96841a9 |
| rflags = 0x00000293 | rflags = 0x00000293 | rflags = 0x00000293 | rflags = 0x00000293 |
| orax = 0xffffffffffffffffffff | orax = 0xffffffffffffffffffff | orax = 0xffffffffffffffffffff | orax = 0xffffffffffffffffffff |

3. switch

```
switch (arr[0]) {
    case 1:
        arr[0]++;
        break;
    case 2:
        arr[1]--;
        break;
    case 4:
        arr[3]--;
    case 5:
        arr[2]++;
        break;
}
```

Значення у реєстрах:

| | | | |
|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| [X] Registers (Registers) | [X] Registers (Registers) | [X] Registers (Registers) | [X] Registers (Registers) |
| rax = 0x00000003 | rax = 0x00000005 | rax = 0x00000005 | rax = 0x00000008 |
| rbx = 0x7fff95c83178 | rbx = 0x7fff95c83178 | rbx = 0x7fff95c83178 | rbx = 0x7ff95c83178 |
| rcx = 0x00000001 | rcx = 0x00000001 | rcx = 0x00000001 | rcx = 0x00000001 |
| rdx = 0x00000000 | rdx = 0x00000000 | rdx = 0x00000000 | rdx = 0x00000000 |
| r8 = 0x00000000 | r8 = 0x00000000 | r8 = 0x00000000 | r8 = 0x00000000 |
| r9 = 0x7fe66530b890 | r9 = 0x7fe66530b890 | r9 = 0x7fe66530b890 | r9 = 0x7fe66530b890 |
| r10 = 0x7fff95c82d90 | r10 = 0x7fff95c82d90 | r10 = 0x7fff95c82d90 | r10 = 0x7fff95c82d90 |
| r11 = 0x00000202 | r11 = 0x00000202 | r11 = 0x00000202 | r11 = 0x00000202 |
| r12 = 0x00000000 | r12 = 0x00000000 | r12 = 0x00000000 | r12 = 0x00000000 |
| r13 = 0x7fff95c83188 | r13 = 0x7fff95c83188 | r13 = 0x7fff95c83188 | r13 = 0x7fff95c83188 |
| r14 = 0x556cccd28ed08 | r14 = 0x556cccd28ed08 | r14 = 0x556cccd28ed08 | r14 = 0x556cccd28ed08 |
| r15 = 0x7fe66533a000 | r15 = 0x7fe66533a000 | r15 = 0x7fe66533a000 | r15 = 0x7fe66533a000 |
| rsi = 0x7fff95c83178 | rsi = 0x7fff95c83178 | rsi = 0x7fff95c83178 | rsi = 0x7fff95c83178 |
| rdi = 0x00000001 | rdi = 0x00000001 | rdi = 0x00000001 | rdi = 0x00000001 |
| rsp = 0x7fff95c83020 | rsp = 0x7fff95c83020 | rsp = 0x7fff95c83020 | rsp = 0x7fff95c83020 |
| rbp = 0x7fff95c83060 | rbp = 0x7fff95c83060 | rbp = 0x7fff95c83060 | rbp = 0x7fff95c83060 |
| rip = 0x556cccd28c1b9 | rip = 0x556cccd28c1bc | rip = 0x556cccd28c1bf | rip = 0x556cccd28c1fe |
| rflags = 0x00000293 | rflags = 0x00000246 | rflags = 0x00000246 | rflags = 0x00000206 |
| orax = 0xffffffffffffffffffff | orax = 0xffffffffffffffffffff | orax = 0xffffffffffffffffffff | orax = 0xffffffffffffffffffff |

Та дизассембліваний код

| | | |
|------------------|--------|--------------------------|
| ↳ 0x556ccd28c1b9 | 8b45d0 | mov eax, dword [var_30h] |
| 0x556ccd28c1bc | 83f805 | cmp eax, 5 ; 5 |
| < 0x556ccd28c1bf | 743a | je 0x556ccd28c1fb |
| 0x556ccd28c1c1 | 83f805 | cmp eax, 5 ; 5 |
| < 0x556ccd28c1c4 | 7f3f | jg 0x556ccd28c205 |
| 0x556ccd28c1c6 | 83f804 | cmp eax, 4 ; 4 |
| < 0x556ccd28c1c9 | 7427 | je 0x556ccd28c1f2 |
| 0x556ccd28c1cb | 83f804 | cmp eax, 4 ; 4 |
| < 0x556ccd28c1ce | 7f35 | jg 0x556ccd28c205 |
| 0x556ccd28c1d0 | 83f801 | cmp eax, 1 ; rcx |
| < 0x556ccd28c1d3 | 7407 | je 0x556ccd28c1dc |
| 0x556ccd28c1d5 | 83f802 | cmp eax, 2 ; 2 |
| < 0x556ccd28c1d8 | 740d | je 0x556ccd28c1e7 |
| 0x556ccd28c1da | eb29 | jmp 0x556ccd28c205 |
| ↳ 0x556ccd28c1dc | 8b45d0 | mov eax, dword [var_30h] |
| 0x556ccd28c1df | 83c001 | add eax, 1 |
| 0x556ccd28c1e2 | 8945d0 | mov dword [var_30h], eax |
| < 0x556ccd28c1e5 | eb1e | jmp 0x556ccd28c205 |
| ↳ 0x556ccd28c1e7 | 8b45d4 | mov eax, dword [var_2ch] |
| 0x556ccd28c1ea | 83e801 | sub eax, 1 |
| 0x556ccd28c1ed | 8945d4 | mov dword [var_2ch], eax |
| < 0x556ccd28c1f0 | eb13 | jmp 0x556ccd28c205 |
| ↳ 0x556ccd28c1f2 | 8b45dc | mov eax, dword [var_24h] |
| 0x556ccd28c1f5 | 83e801 | sub eax, 1 |
| 0x556ccd28c1f8 | 8945dc | mov dword [var_24h], eax |
| < 0x556ccd28c1fb | 8b45d8 | mov eax, dword [var_28h] |
| 0x556ccd28c1fe | 83c001 | add eax, 1 |
| 0x556ccd28c201 | 8945d8 | mov dword [var_28h], eax |
| 0x556ccd28c204 | 90 | nop |

4. Global variable

Вивід значення змінної (виклик printf)

```
; CODE XREFS from main @ 0x556ccd28c1da, 0x556ccd28c1e5, 0x556ccd28c1f8
0x556ccd28c205    8b05152e0000  mov eax, dword [obj.global_variable] ; [0x556ccd28f020:4]=0x6ed6
0x556ccd28c20b    89c6          mov esi, eax
0x556ccd28c20d    488d05f00d00. lea rax, [0x556ccd28d004] ; "%d\n"
0x556ccd28c214    4889c7          mov rdi, rax
0x556ccd28c217    b800000000  mov eax, 0
0x556ccd28c21c    e81ffeffff  call sym.imp.printf ;[1] ; int printf(const char *format)
```

Значення глобальної змінної знаходитьться за адресою 0x556ccd28f020

| [X] Hexdump (Hexdump) | [Cache] Off |
|---|---------------|
| - offset - 0 1 2 3 4 5 6 7 8 9 A B C | 0123456789ABC |
| 0x556cccd28f020 d66e 0000 0000 0000 0000 00 .n..... ; obj.global_variable ; | |
| 0x556cccd28f02d 0000 0000 0000 0000 0000 00 | |
| 0x556cccd28f030 0000 0000 0000 0000 0000 00 | |

5. Цикл while

```
// 5. while cycle
while (arr[0] > 0) {
    arr[0]--;
}
```

| | |
|--|--------------------------|
| └< 0x55d89e9b3221 b eb09 | jmp 0x55d89e9b322c |
| └→ 0x55d89e9b3223 8b45d0 | mov eax, dword [var_30h] |
| 0x55d89e9b3226 83e801 | sub eax, 1 |
| 0x55d89e9b3229 8945d0 | mov dword [var_30h], eax |
| ; CODE XREF from main @ 0x55d89e9b3221 | |
| └→ 0x55d89e9b322c 8b45d0 | mov eax, dword [var_30h] |
| 0x55d89e9b322f 85c0 | test eax, eax |
| └< 0x55d89e9b3231 7ff0 | je 0x55d89e9b3223 |

Перевіряється умова циклу

| | | | |
|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| [X] Registers (Registers) | [X] Registers (Registers) | [X] Registers (Registers) | [X] Registers (Registers) |
| rax = 0x00000006 | rax = 0x00000006 | rax = 0x00000005 | rax = 0x00000005 |
| rbx = 0x7ffeb1dbed18 | rbx = 0x7ffeb1dbed18 | rbx = 0x7ffeb1dbed18 | rbx = 0x7ffeb1dbed18 |
| rcx = 0x00000000 | rcx = 0x00000000 | rcx = 0x00000000 | rcx = 0x00000000 |
| rdx = 0x00000000 | rdx = 0x00000000 | rdx = 0x00000000 | rdx = 0x00000000 |
| r8 = 0x00000400 | r8 = 0x00000400 | r8 = 0x00000400 | r8 = 0x00000400 |
| r9 = 0x00000064 | r9 = 0x00000064 | r9 = 0x00000064 | r9 = 0x00000064 |
| r10 = 0x7fb95a141c00 | r10 = 0x7fb95a141c00 | r10 = 0x7fb95a141c00 | r10 = 0x7fb95a141c00 |
| r11 = 0x00000202 | r11 = 0x00000202 | r11 = 0x00000202 | r11 = 0x00000202 |
| r12 = 0x00000000 | r12 = 0x00000000 | r12 = 0x00000000 | r12 = 0x00000000 |
| r13 = 0x7ffeb1dbed28 | r13 = 0x7ffeb1dbed28 | r13 = 0x7ffeb1dbed28 | r13 = 0x7ffeb1dbed28 |
| r14 = 0x55d89e9b5dd8 | r14 = 0x55d89e9b5dd8 | r14 = 0x55d89e9b5dd8 | r14 = 0x55d89e9b5dd8 |
| r15 = 0x7fb95a1b3000 | r15 = 0x7fb95a1b3000 | r15 = 0x7fb95a1b3000 | r15 = 0x7fb95a1b3000 |
| rsi = 0x55d89f5112a0 | rsi = 0x55d89f5112a0 | rsi = 0x55d89f5112a0 | rsi = 0x55d89f5112a0 |
| rdi = 0x7ffeb1dbe660 | rdi = 0x7ffeb1dbe660 | rdi = 0x7ffeb1dbe660 | rdi = 0x7ffeb1dbe660 |
| rsp = 0x7ffeb1dbebc0 | rsp = 0x7ffeb1dbebc0 | rsp = 0x7ffeb1dbebc0 | rsp = 0x7ffeb1dbebc0 |
| rbp = 0x7ffeb1dbec00 | rbp = 0x7ffeb1dbec00 | rbp = 0x7ffeb1dbec00 | rbp = 0x7ffeb1dbec00 |
| rip = 0x55d89e9b3221 | rip = 0x55d89e9b322c | rip = 0x55d89e9b322f | rip = 0x55d89e9b3231 |
| rflags = 0x00000206 | rflags = 0x00000206 | rflags = 0x00000206 | rflags = 0x00000206 |
| orax = 0xffffffffffffffffffff | orax = 0xffffffffffffffffffff | orax = 0xffffffffffffffffffff | orax = 0xffffffffffffffffffff |

Перейшли у тіло циклу

```
[X] Registers (Registers)
rax = 0x00000005
rbx = 0x7ffeb1dbed18
rcx = 0x00000000
rdx = 0x00000000
r8 = 0x00000400
r9 = 0x00000064
r10 = 0x7fb95a141c00
r11 = 0x00000202
r12 = 0x00000000
r13 = 0x7ffeb1dbed28
r14 = 0x55d89e9b5dd8
r15 = 0x7fb95a1b3000
rsi = 0x55d89f5112a0
rdi = 0x7ffeb1dbe660
rsp = 0x7ffeb1dbebc0
rbp = 0x7ffeb1dbec00
rip = 0x55d89e9b3223
rflags = 0x00000206
orax = 0xffffffffffffffffffff
```

```
[X] Registers (Registers)
rax = 0x00000005
rbx = 0x7ffeb1dbed18
rcx = 0x00000000
rdx = 0x00000000
r8 = 0x00000400
r9 = 0x00000064
r10 = 0x7fb95a141c00
r11 = 0x00000202
r12 = 0x00000000
r13 = 0x7ffeb1dbed28
r14 = 0x55d89e9b5dd8
r15 = 0x7fb95a1b3000
rsi = 0x55d89f5112a0
rdi = 0x7ffeb1dbe660
rsp = 0x7ffeb1dbebc0
rbp = 0x7ffeb1dbec00
rip = 0x55d89e9b3226
rflags = 0x00000206
orax = 0xffffffffffffffffffff
```

```
[X] Registers (Registers)
rax = 0x00000004
rbx = 0x7ffeb1dbed18
rcx = 0x00000000
rdx = 0x00000000
r8 = 0x00000400
r9 = 0x00000064
r10 = 0x7fb95a141c00
r11 = 0x00000202
r12 = 0x00000000
r13 = 0x7ffeb1dbed28
r14 = 0x55d89e9b5dd8
r15 = 0x7fb95a1b3000
rsi = 0x55d89f5112a0
rdi = 0x7ffeb1dbe660
rsp = 0x7ffeb1dbebc0
rbp = 0x7ffeb1dbec00
rip = 0x55d89e9b3229
rflags = 0x00000202
orax = 0xffffffffffffffffffff
```

Та знову перейшли на область перевірки умови

```
[X] Registers (Registers)
rax = 0x00000004
rbx = 0x7ffeb1dbed18
rcx = 0x00000000
rdx = 0x00000000
r8 = 0x00000400
r9 = 0x00000064
r10 = 0x7fb95a141c00
r11 = 0x00000202
r12 = 0x00000000
r13 = 0x7ffeb1dbed28
r14 = 0x55d89e9b5dd8
r15 = 0x7fb95a1b3000
rsi = 0x55d89f5112a0
rdi = 0x7ffeb1dbe660
rsp = 0x7ffeb1dbebc0
rbp = 0x7ffeb1dbec00
rip = 0x55d89e9b322c
rflags = 0x00000202
orax = 0xffffffffffffffffffff
```

6. Реалізована функція `is_even`, що перевіряє `int` число на подільність на 2.

Її виклик в `main`

| | | |
|-----------------------------|-------------------------|---------------------------------------|
| <code>0x55d89e9b3233</code> | <code>8b45dc</code> | <code>mov eax, dword [var_24h]</code> |
| <code>0x55d89e9b3236</code> | <code>89c7</code> | <code>mov edi, eax</code> |
| <code>0x55d89e9b3238</code> | <code>e80cffffff</code> | <code>call sym.is_even ;[2]</code> |

Значення регістрів при виклику

```
[X] Registers (Registers)
rax = 0x00000000
rbx = 0x7ffeb1dbed18
rcx = 0x00000000
rdx = 0x00000000
r8 = 0x00000400
r9 = 0x00000064
r10 = 0x7fb95a141c00
r11 = 0x00000202
r12 = 0x00000000
r13 = 0x7ffeb1dbed28
r14 = 0x55d89e9b5dd8
r15 = 0x7fb95a1b3000
rsi = 0x55d89f5112a0
rdi = 0x7ffeb1dbe660
rsp = 0x7ffeb1dbebc0
rbp = 0x7ffeb1dbecc0
rip = 0x55d89e9b3233
rflags = 0x00000246
orax = 0xffffffffffffffff
```

```
[X] Registers (Registers)
rax = 0x00000001
rbx = 0x7ffeb1dbed18
rcx = 0x00000000
rdx = 0x00000000
r8 = 0x00000400
r9 = 0x00000064
r10 = 0x7fb95a141c00
r11 = 0x00000202
r12 = 0x00000000
r13 = 0x7ffeb1dbed28
r14 = 0x55d89e9b5dd8
r15 = 0x7fb95a1b3000
rsi = 0x55d89f5112a0
rdi = 0x7ffeb1dbe660
rsp = 0x7ffeb1dbebc0
rbp = 0x7ffeb1dbecc0
rip = 0x55d89e9b3236
rflags = 0x00000246
orax = 0xffffffffffffffff
```

```
[X] Registers (Registers)
rax = 0x00000001
rbx = 0x7ffeb1dbed18
rcx = 0x00000000
rdx = 0x00000000
r8 = 0x00000400
r9 = 0x00000064
r10 = 0x7fb95a141c00
r11 = 0x00000202
r12 = 0x00000000
r13 = 0x7ffeb1dbed28
r14 = 0x55d89e9b5dd8
r15 = 0x7fb95a1b3000
rsi = 0x55d89f5112a0
rdi = 0x00000001
rsp = 0x7ffeb1dbebc0
rbp = 0x7ffeb1dbecc0
rip = 0x55d89e9b3238
rflags = 0x00000246
orax = 0xffffffffffffffff
```

Функція is_even

```
23: sym.is_even (int64_t arg1);
; var int64_t var_4h @ rbp-0x4
; arg int64_t arg1 @ rdi
0x55d89e9b3149      55          push rbp
0x55d89e9b314a      4889e5      mov rbp, rsp
0x55d89e9b314d      897dfc      mov dword [var_4h], edi    ; arg1
0x55d89e9b3150      8b45fc      mov eax, dword [var_4h]
0x55d89e9b3153      83e001      and eax, 1
0x55d89e9b3156      85c0        test eax, eax
0x55d89e9b3158      0f94c0      sete al
0x55d89e9b315b      0fb6c0      movzx eax, al
0x55d89e9b315e      5d          pop rbp
0x55d89e9b315f      c3          ret
```

Та значення регістрів при її виконанні

```
[X] Registers (Registers)
rax = 0x00000001
rbx = 0x7ffeb1dbed18
rcx = 0x00000000
rdx = 0x00000000
r8 = 0x00000400
r9 = 0x00000064
r10 = 0x7fb95a141c00
r11 = 0x00000202
r12 = 0x00000000
r13 = 0x7ffeb1dbed28
r14 = 0x55d89e9b5dd8
r15 = 0x7fb95a1b3000
rsi = 0x55d89f5112a0
rdi = 0x00000001
rsp = 0x7ffeb1dbebb8
rbp = 0x7ffeb1dbecc0
rip = 0x55d89e9b3149
rflags = 0x00000246
orax = 0xffffffffffffffff
```

```
[X] Registers (Registers)
rax = 0x00000001
rbx = 0x7ffeb1dbed18
rcx = 0x00000000
rdx = 0x00000000
r8 = 0x00000400
r9 = 0x00000064
r10 = 0x7fb95a141c00
r11 = 0x00000202
r12 = 0x00000000
r13 = 0x7ffeb1dbed28
r14 = 0x55d89e9b5dd8
r15 = 0x7fb95a1b3000
rsi = 0x55d89f5112a0
rdi = 0x00000001
rsp = 0x7ffeb1dbebb0
rbp = 0x7ffeb1dbebb0
rip = 0x55d89e9b314d
rflags = 0x00000246
orax = 0xffffffffffffffff
```

```
[X] Registers (Registers)
rax = 0x00000001
rbx = 0x7ffeb1dbed18
rcx = 0x00000000
rdx = 0x00000000
r8 = 0x00000400
r9 = 0x00000064
r10 = 0x7fb95a141c00
r11 = 0x00000202
r12 = 0x00000000
r13 = 0x7ffeb1dbed28
r14 = 0x55d89e9b5dd8
r15 = 0x7fb95a1b3000
rsi = 0x55d89f5112a0
rdi = 0x00000001
rsp = 0x7ffeb1dbebb0
rbp = 0x7ffeb1dbebb0
rip = 0x55d89e9b3150
rflags = 0x00000246
orax = 0xffffffffffffffff
```

```
[X] Registers (Registers)
rax = 0x00000001
rbx = 0x7ffeb1dbed18
rcx = 0x00000000
rdx = 0x00000000
r8 = 0x00000400
r9 = 0x00000064
r10 = 0x7fb95a141c00
r11 = 0x00000202
r12 = 0x00000000
r13 = 0x7ffeb1dbed28
r14 = 0x55d89e9b5dd8
r15 = 0x7fb95a1b3000
rsi = 0x55d89f5112a0
rdi = 0x00000001
rsp = 0x7ffeb1dbebb0
rbp = 0x7ffeb1dbebb0
rip = 0x55d89e9b3153
rflags = 0x00000246
orax = 0xffffffffffffffff
```

```
[X] Registers (Registers)
rax = 0x00000001
rbx = 0x7ffeb1dbed18
rcx = 0x00000000
rdx = 0x00000000
r8 = 0x00000400
r9 = 0x00000064
r10 = 0x7fb95a141c00
r11 = 0x00000202
r12 = 0x00000000
r13 = 0x7ffeb1dbed28
r14 = 0x55d89e9b5dd8
r15 = 0x7fb95a1b3000
rsi = 0x55d89f5112a0
rdi = 0x00000001
rsp = 0x7ffeb1dbebb0
rbp = 0x7ffeb1dbebb0
rip = 0x55d89e9b3153
rflags = 0x00000246
orax = 0xffffffffffffffff
```

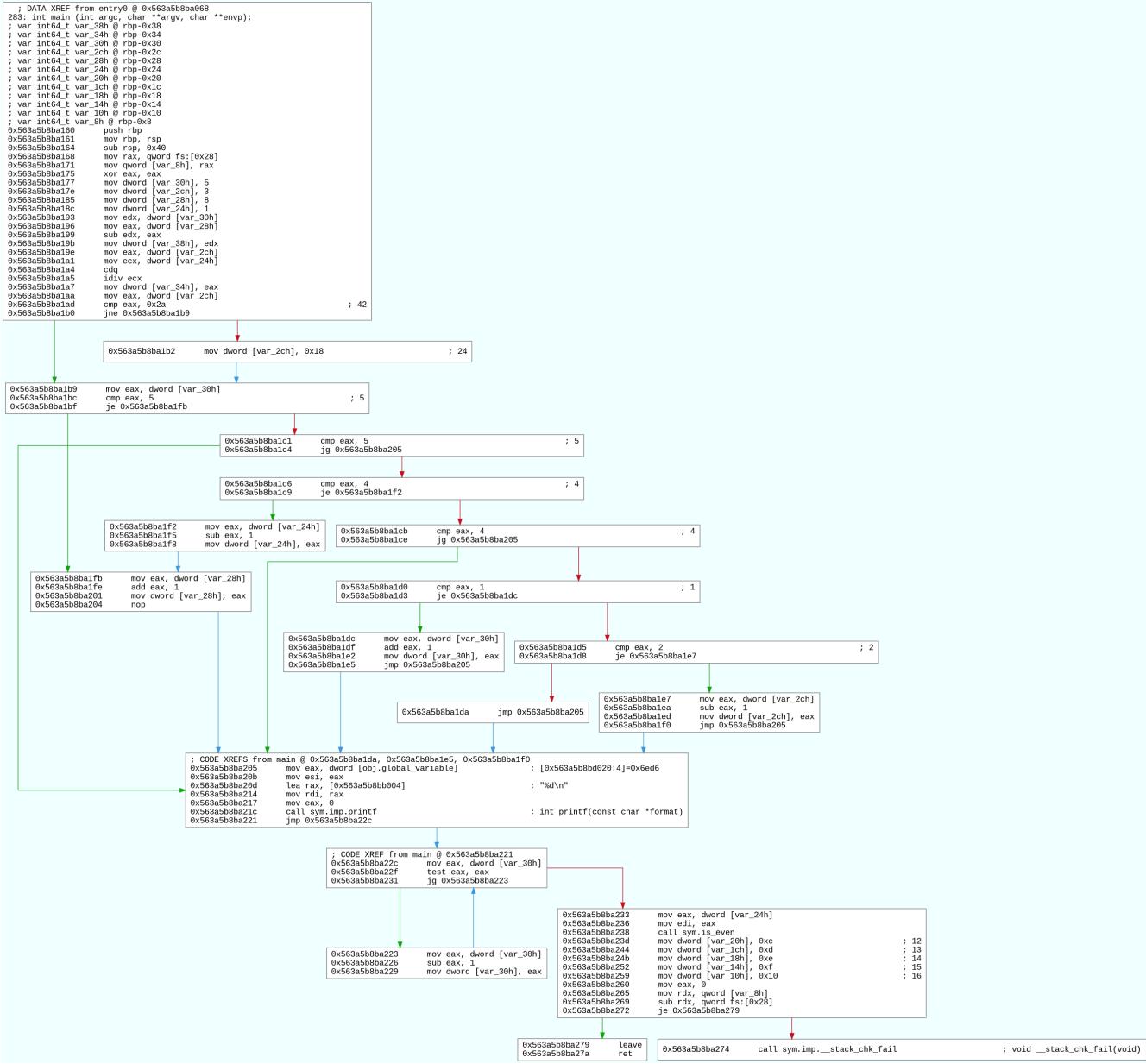
| | | | |
|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| [X] Registers (Registers) | [X] Registers (Registers) | [X] Registers (Registers) | [X] Registers (Registers) |
| rax = 0x00000001 | rax = 0x00000001 | rax = 0x00000000 | rax = 0x00000000 |
| rbx = 0x7ffeb1dbed18 | rbx = 0x7ffeb1dbed18 | rbx = 0x7ffeb1dbed18 | rbx = 0x7ffeb1dbed18 |
| rcx = 0x00000000 | rcx = 0x00000000 | rcx = 0x00000000 | rcx = 0x00000000 |
| rdx = 0x00000000 | rdx = 0x00000000 | rdx = 0x00000000 | rdx = 0x00000000 |
| r8 = 0x00000400 | r8 = 0x00000400 | r8 = 0x00000400 | r8 = 0x00000400 |
| r9 = 0x00000064 | r9 = 0x00000064 | r9 = 0x00000064 | r9 = 0x00000064 |
| r10 = 0x7fb95a141c00 | r10 = 0x7fb95a141c00 | r10 = 0x7fb95a141c00 | r10 = 0x7fb95a141c00 |
| r11 = 0x00000202 | r11 = 0x00000202 | r11 = 0x00000202 | r11 = 0x00000202 |
| r12 = 0x00000000 | r12 = 0x00000000 | r12 = 0x00000000 | r12 = 0x00000000 |
| r13 = 0x7ffeb1dbed28 | r13 = 0x7ffeb1dbed28 | r13 = 0x7ffeb1dbed28 | r13 = 0x7ffeb1dbed28 |
| r14 = 0x55d89e9b5dd8 | r14 = 0x55d89e9b5dd8 | r14 = 0x55d89e9b5dd8 | r14 = 0x55d89e9b5dd8 |
| r15 = 0x7fb95a1b3000 | r15 = 0x7fb95a1b3000 | r15 = 0x7fb95a1b3000 | r15 = 0x7fb95a1b3000 |
| rsi = 0x55d89f5112a0 | rsi = 0x55d89f5112a0 | rsi = 0x55d89f5112a0 | rsi = 0x55d89f5112a0 |
| rdi = 0x00000001 | rdi = 0x00000001 | rdi = 0x00000001 | rdi = 0x00000001 |
| rsp = 0x7ffeb1dbebb0 | rsp = 0x7ffeb1dbebb0 | rsp = 0x7ffeb1dbebb0 | rsp = 0x7ffeb1dbebb0 |
| rbp = 0x7ffeb1dbebb0 | rbp = 0x7ffeb1dbebb0 | rbp = 0x7ffeb1dbebb0 | rbp = 0x7ffeb1dbebb0 |
| rip = 0x55d89e9b3156 | rip = 0x55d89e9b3158 | rip = 0x55d89e9b315b | rip = 0x55d89e9b315e |
| rflags = 0x00000202 | rflags = 0x00000202 | rflags = 0x00000202 | rflags = 0x00000202 |
| orax = 0xffffffffffffffffffff | orax = 0xffffffffffffffffffff | orax = 0xffffffffffffffffffff | orax = 0xffffffffffffffffffff |

7. Список зі значеннями

```
// 7. array with specific elements
int arr2[] = { 12, 13, 14, 15, 16 };
```

| | | |
|----------------|------------------------|--------------------------------|
| 0x55d89e9b323d | c745e00c 0000 . | mov dword [var_20h], 0xc ; 12 |
| 0x55d89e9b3244 | c745e40d 0000 . | mov dword [var_1ch], 0xd ; 13 |
| 0x55d89e9b324b | c745e80e 0000 . | mov dword [var_18h], 0xe ; 14 |
| 0x55d89e9b3252 | c745ec0f 0000 . | mov dword [var_14h], 0xf ; 15 |
| 0x55d89e9b3259 | c745f010 0000 . | mov dword [var_10h], 0x10 ; 16 |

Дерево програми (функція main)



Лістинг програми на С

```
#include <stdio.h>

// 4. global variable
int global_variable = 28374;
```

```
// 6. own function
int is_even(int num) {
    return num % 2 == 0;
}
```

```

int main() {
    // 1. array elements and actions (division, subtraction)
    int arr[] = { 5, 3, 8, 1 };
    int res1 = arr[0] - arr[2]; int res2 = arr[1] / arr[3];

    // 2. if statement
    if (arr[1] == 42) {
        arr[1] = 24;
    }

    // 3. switch on 4 cases
    switch (arr[0]) {
        case 1:
            arr[0]++;
            break;
        case 2:
            arr[1]--;
            break;
        case 4:
            arr[3]--;
        case 5:
            arr[2]++;
            break;
    }

    // 4. print global variable
    printf("%d\n", global_variable);

    // 5. while cycle
    while (arr[0] > 0) {
        arr[0]--;
    }

    // 6. function call
    is_even(arr[3]);

    // 7. array with specific elements
    int arr2[] = { 12, 13, 14, 15, 16 };

    return 0;
}

```