gdc function declaration:

2 ورودی int و خروجی هایی با حالت های مختلف که هریک باعث ایجاد label های مختلف میشوند

```
gcd(int, int):
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     DWORD PTR [rbp-4], edi
        mov     DWORD PTR [rbp-8], esi
        cmp     DWORD PTR [rbp-4], 0
        je      .L2
        cmp     DWORD PTR [rbp-8], 0
        jne     .L3
```

label 2:

در این بخش یک کپی انجام شده و به label2 میرود

```
.L2:
        mov     eax, 0
        jmp     .L4
```

label 3:

در این بخش یک کپی انجام شده و سپس 2 متغیر را مقایسه میکند و اختلاف label5 را محاسبه کرده و یک کپی انجام میدهد و به label4 میپرد

```
.L3:
        mov     eax, DWORD PTR [rbp-4]
        cmp     eax, DWORD PTR [rbp-8]
        jne     .L5
        mov     eax, DWORD PTR [rbp-4]
        jmp     .L4
```

label 5:

در این بخش یک کپی انجام شده و سپس 2 متغیر را مقایسه میکند و اختلاف label6 را محاسبه کرده و چند کپی انجام میدهد و به دوباره خود تابع را فراخوانی میکند که سبب پرش میشود

```
.L5:
        mov     eax, DWORD PTR [rbp-4]
        cmp     eax, DWORD PTR [rbp-8]
        jle     .L6
        mov     eax, DWORD PTR [rbp-4]
        sub     eax, DWORD PTR [rbp-8]
        mov     edx, DWORD PTR [rbp-8]
        mov     esi, edx
        mov     edi, eax
        call    gcd(int, int)
        jmp     .L4
```

label 6:

در این بخش چند کپی انجام شده و دوباره خود تابع را فراخوانی میکند که سبب پرش میشود

```
.L6:
        mov     eax, DWORD PTR [rbp-8]
        sub     eax, DWORD PTR [rbp-4]
        mov     edx, eax
        mov     eax, DWORD PTR [rbp-4]
        mov     esi, edx
        mov     edi, eax
        call    gcd(int, int)
        nop
```

label 4:

در این بخش استک ساخته شده را خالی کرده و ان را بر میگرداند

```
.L4:
        leave
        ret
```

در این بخش مقادیر ثابت رشته ها مشخص میشوند

در

```
.LC0:
        .string "GCD of "
```

```
.LC1:
        .string " and "
```

```
.LC2:
        .string " is "
```

Main function:

در این قسمت تابع **main** شروع به کار کرده و به ترتیب متغیر ها را تعریف کرده و از **gcd** استفاده میکند و در نهایت یک رشته را به همراه متغیر های درون ان چاپ میکند
بقیه دستورات مربوط به استفاده از استک و مدیریت مقادیر موجود در توابع ایجاد شده توسط **main** است و انجام دستورات برای چک کردن شرط های موجود در کد

```
main:
        push    rbp
        mov     rbp, rsp
        push    rbx
        sub     rsp, 24
        mov     DWORD PTR [rbp-20], 105
        mov     DWORD PTR [rbp-24], 30
        mov     esi, OFFSET FLAT:.LC0
        mov     edi, OFFSET FLAT:_ZSt4cout
        call    std::basic_ostream<char, std::char_traits<char> >&
std::operator<< <std::char_traits<char> >(std::basic_ostream<char,
std::char_traits<char> >&, char const*)
        mov     rdx, rax
        mov     eax, DWORD PTR [rbp-20]
        mov     esi, eax
        mov     rdi, rdx
        call    std::basic_ostream<char, std::char_traits<char>
>::operator<<(int)
        mov     esi, OFFSET FLAT:.LC1
```

```
        mov     rdi, rax
        call    std::basic_ostream<char, std::char_traits<char> >&
std::operator<< <std::char_traits<char> >(std::basic_ostream<char,
std::char_traits<char> >&, char const*)
        mov     rdx, rax
        mov     eax, DWORD PTR [rbp-24]
        mov     esi, eax
        mov     rdi, rdx
        call    std::basic_ostream<char, std::char_traits<char>
>::operator<<(int)
        mov     esi, OFFSET FLAT:.LC2
        mov     rdi, rax
        call    std::basic_ostream<char, std::char_traits<char> >&
std::operator<< <std::char_traits<char> >(std::basic_ostream<char,
std::char_traits<char> >&, char const*)
        mov     rbx, rax
        mov     edx, DWORD PTR [rbp-24]
        mov     eax, DWORD PTR [rbp-20]
        mov     esi, edx
        mov     edi, eax
        call    gcd(int, int)
        mov     esi, eax
        mov     rdi, rbx
        call    std::basic_ostream<char, std::char_traits<char>
>::operator<<(int)
        mov     eax, 0
        mov     rbx, QWORD PTR [rbp-8]
        leave
        ret
__static_initialization_and_destruction_0(int, int):
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     DWORD PTR [rbp-4], edi
        mov     DWORD PTR [rbp-8], esi
        cmp     DWORD PTR [rbp-4], 1
        jne     .L11
        cmp     DWORD PTR [rbp-8], 65535
        jne     .L11
        mov     edi, OFFSET FLAT:_ZStL8__ioinit
```

```
        call    std::ios_base::Init::Init() [complete object constructor]
        mov     edx, OFFSET FLAT:__dso_handle
        mov     esi, OFFSET FLAT:_ZStL8__ioinit
        mov     edi, OFFSET FLAT:_ZNSt8ios_base4InitD1Ev
        call    __cxa_atexit
```

label 11:

در این بخش دستور nop کاری انجام نمیدهد و دستور leave استک را خالی میکند

```
.L11:
        nop
        leave
        ret
```

در این بخش مقادیر تابع gcd تخصیص داده میشوند و هر بار که تغییر میکنند مقادیر مورد نیاز را مشخص میکنند تا تابع درست کار کند

```
_GLOBAL__sub_I_gcd(int, int):
        push    rbp
        mov     rbp, rsp
        mov     esi, 65535
        mov     edi, 1
        call    __static_initialization_and_destruction_0(int, int)
        pop     rbp
        ret
```