Name – Anish Jangir
Id – 2020ucp1011

Usage of GodBolt

Introduction:

Godbolt is an online compiler and code analysis tool that allows developers to generate and compare the assembly code generated by different compilers and optimization levels. It provides an easy-to-use interface that makes it simple to see the effects of compiler optimization settings on generated code. In this report, we will explore the features and benefits of Godbolt for generating assembly language, analyzing code, and optimizing performance.

Generating Assembly Language:

One of the primary features of Godbolt is the ability to generate assembly code from various programming languages, including C++, Rust, and Go. Users can input their code into the editor and select the target compiler, optimization level, and CPU architecture to generate the assembly code. The generated code can be easily copied and pasted into other tools for further analysis, such as performance profiling or debugging.

Optimization Level Features:

Godbolt provides various optimization level features to optimize the generated assembly code. These optimization levels range from -O0 to -O3, with each level providing a different level of optimization. At -O0, the compiler generates code without any optimization, while at -O3, the compiler generates highly optimized code. Users can easily switch between optimization levels and compare the generated assembly code to understand the impact of optimization on performance.

Other Useful Features for Analysis of Code:

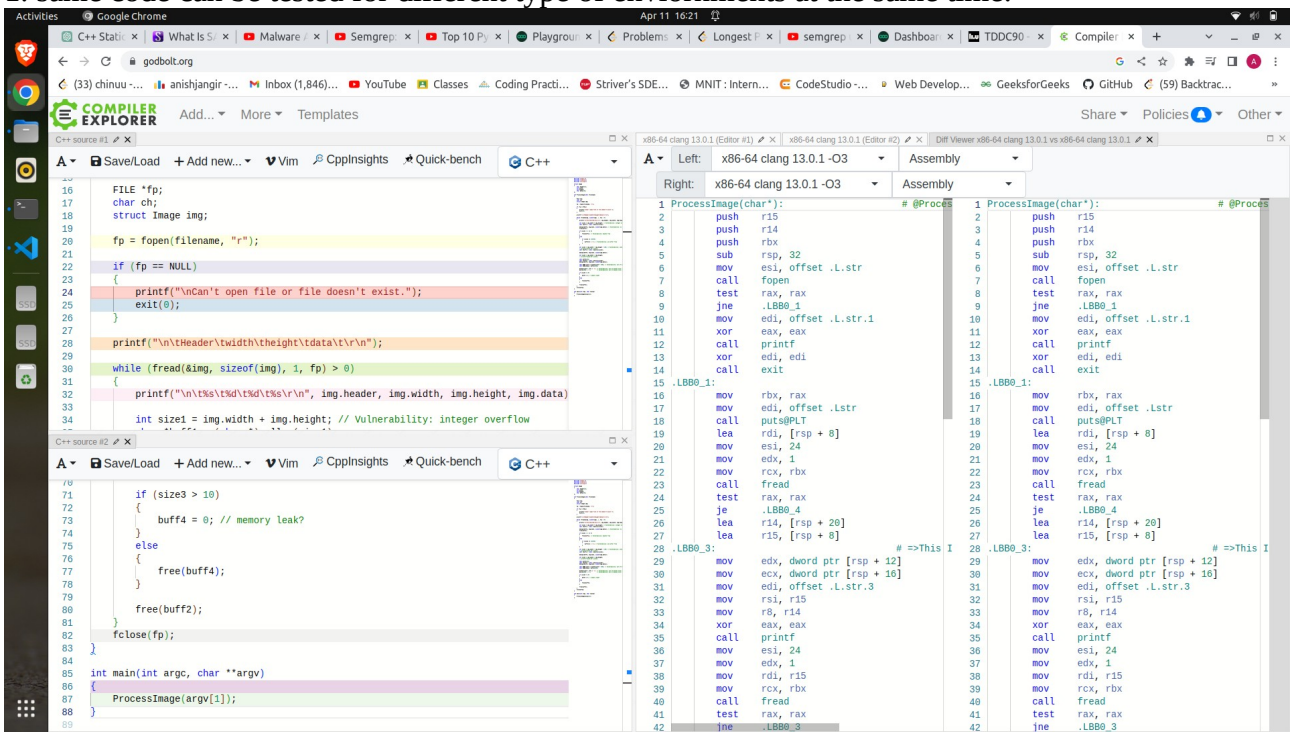
Godbolt provides many other useful features for code analysis, such as the ability to view the generated assembly code side by side with the original code. This makes it easy to understand how changes in the source code impact the generated assembly code. Additionally, Godbolt provides the ability to view the generated assembly code in different formats, including Intel and AT&T syntax, which can be useful for analyzing the generated code.

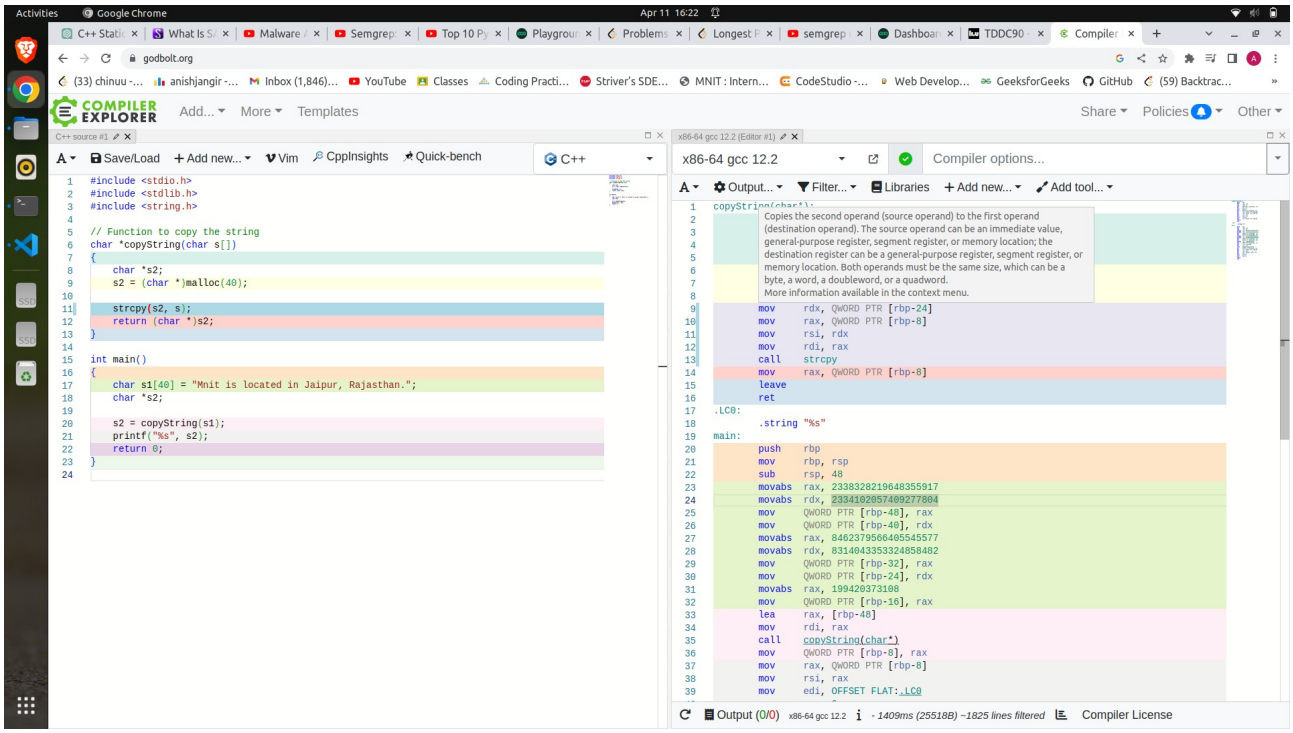
Conclusion:

Godbolt is a powerful online tool for generating assembly code, analyzing code, and optimizing performance. With its easy-to-use interface and powerful features, Godbolt is a valuable tool for developers who need to optimize their code for performance. Its ability to generate assembly code from various programming languages, provide different optimization levels, and other useful features makes it a must-have tool for any developer looking to improve the performance of their code.

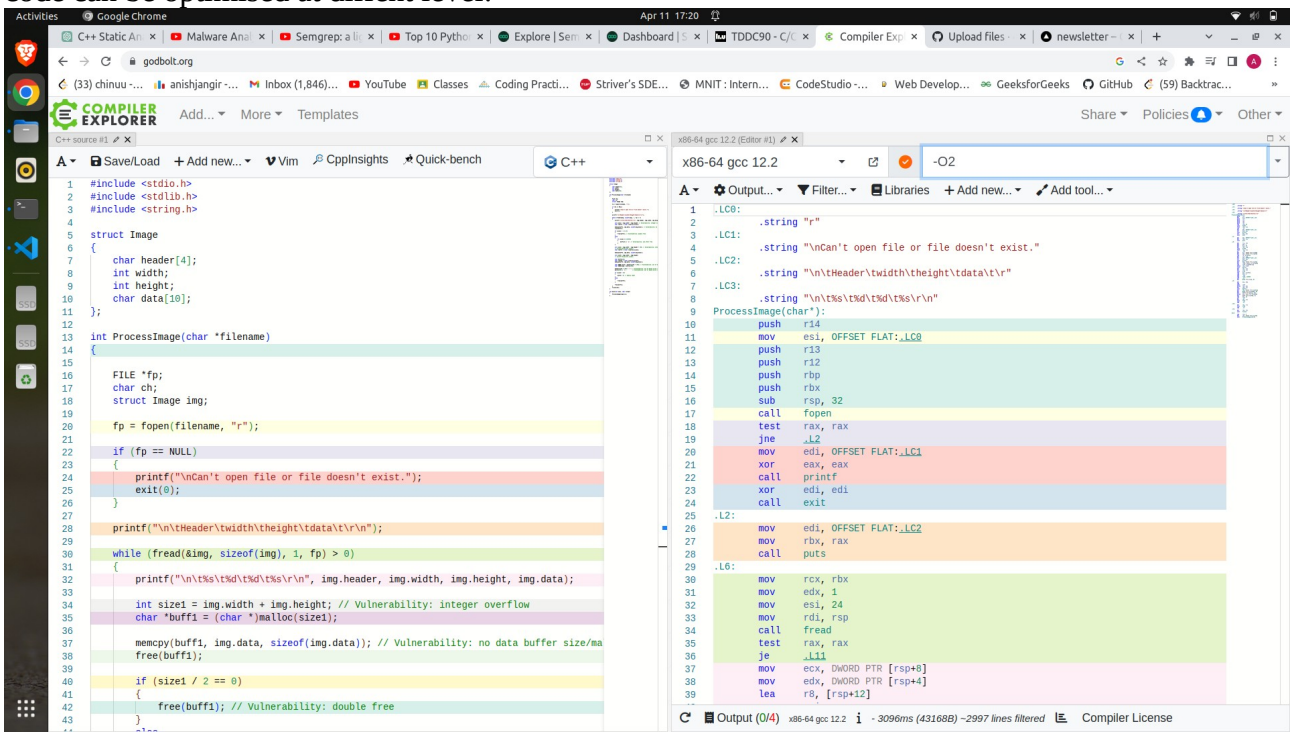
Below are some of the screenshots :

1. same code can be tested for different type of enviornments at the same time:

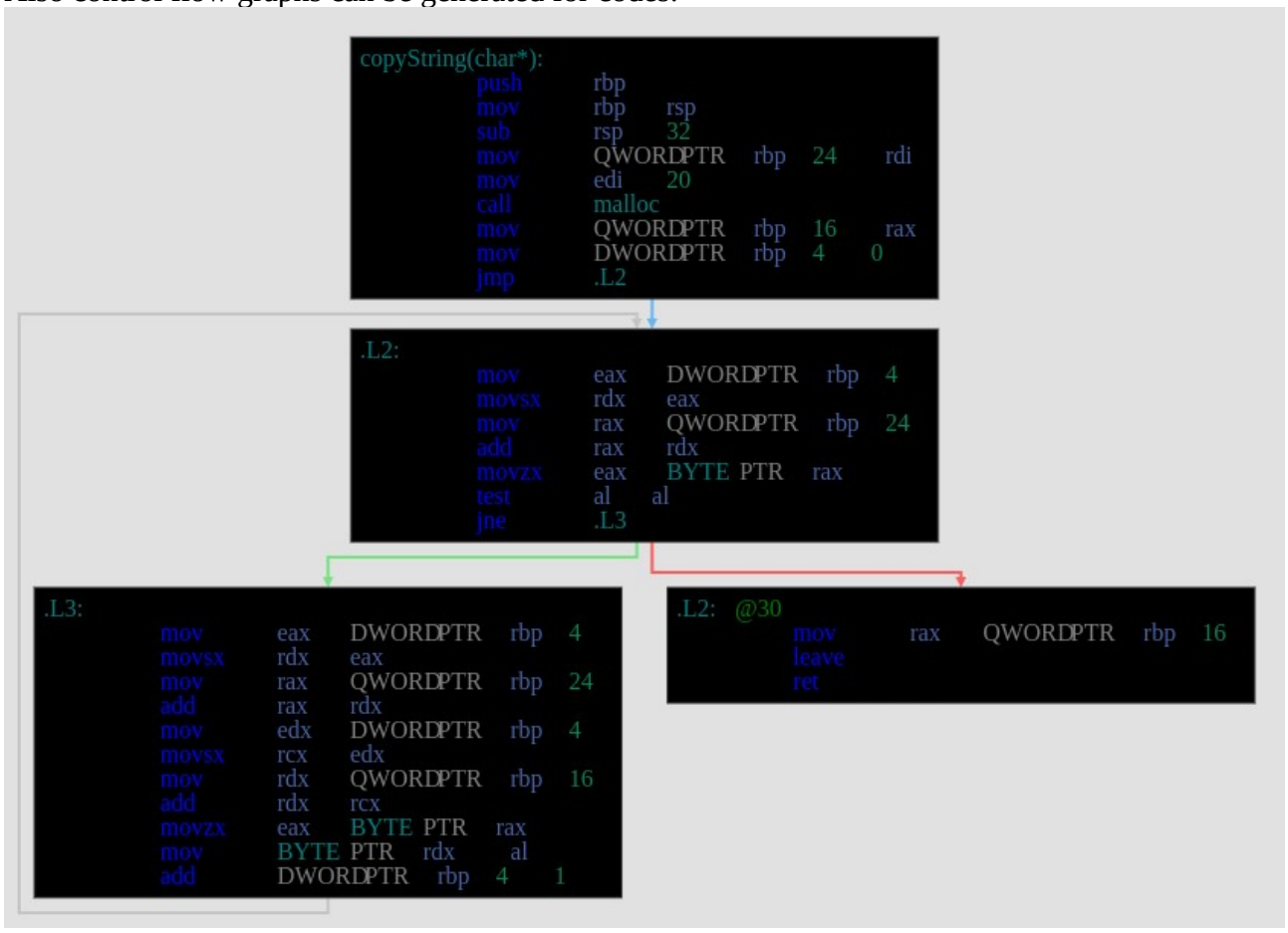


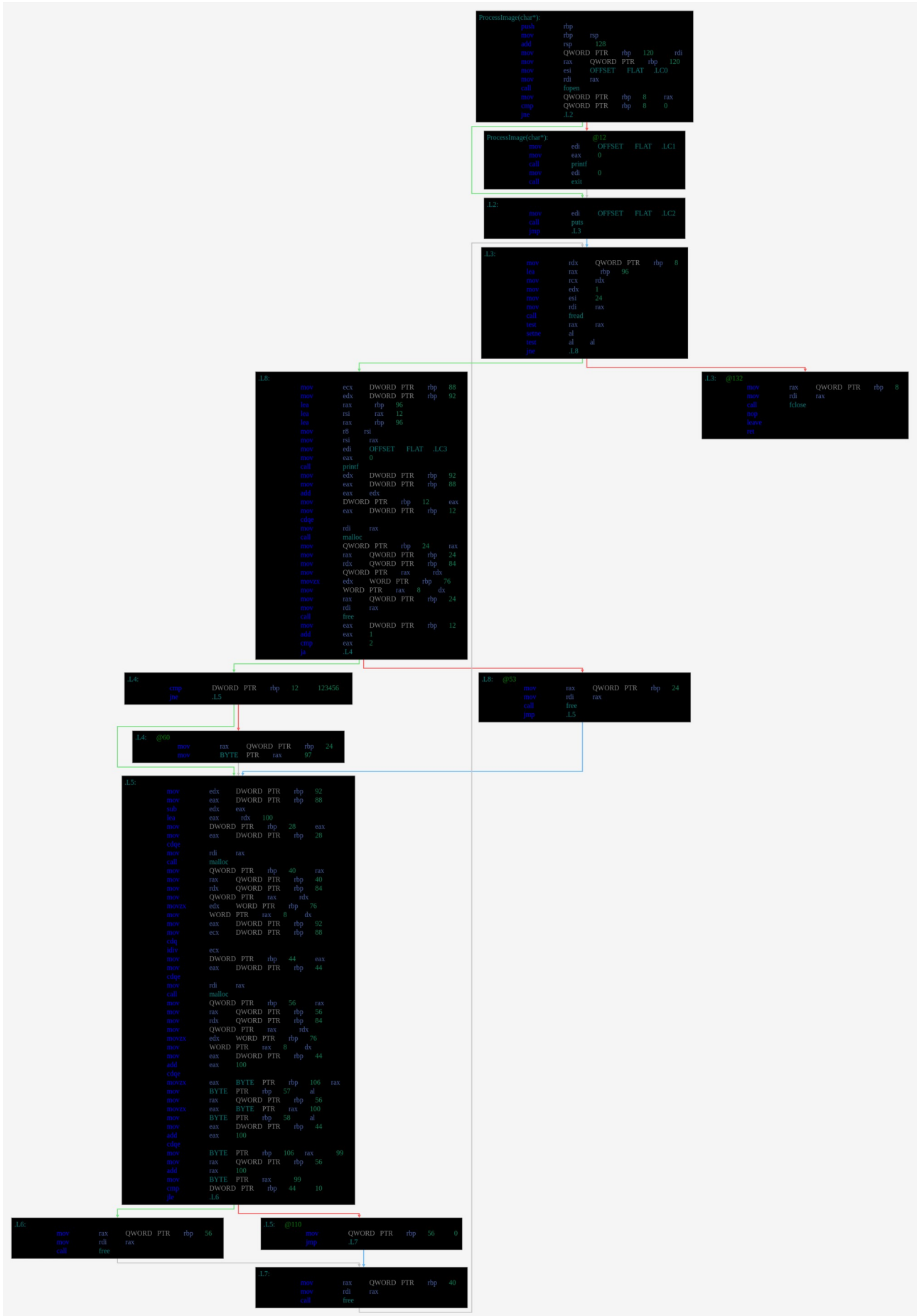


code can be optimised at diffent level:



Also control flow graphs can be generated for codes:




ProcessImage(char*):
push rbp
mov rbp rsp
add rsp -128
mov QWORD PTR rbp 120 rdi
mov rax QWORD PTR rbp 120
mov esi OFFSET FLAT .LC0
mov rdi rax
call fopen
mov QWORD PTR rbp 8 rax
cmp QWORD PTR rbp 8 0
jne .L2
ProcessImage(char*): @12
mov edi OFFSET FLAT .LC1
mov eax 0
call printf
mov edi 0
call exit
.L2:
mov edi OFFSET FLAT .LC2
call puts
jmp .L3
.L3:
mov rdx QWORD PTR rbp 8
lea rax rbp 96
mov rcx rdx
mov edx 1
mov esi 24
mov rdi rax
call fread
test rax rax
sete al
test al al
jne .L8
.L3: @132
mov rax QWORD PTR rbp 8
mov rdi rax
call fclose
nop
leave
ret
.L8:
mov ecx DWORD PTR rbp 88
mov edx DWORD PTR rbp 92
lea rax rbp 96
lea rsi rax 12
lea rax rbp 96
mov r8 rsi
mov rsi rax
mov edi OFFSET FLAT .LC3
mov eax 0
call printf
mov edx DWORD PTR rbp 92
mov eax DWORD PTR rbp 88
add eax edx
mov DWORD PTR rbp 12 eax
mov eax DWORD PTR rbp 12
cdqe
mov rdi rax
call malloc
mov QWORD PTR rbp 24 rax
mov rax QWORD PTR rbp 24
mov rdx QWORD PTR rbp 84
mov QWORD PTR rax rdx
movzx edx WORD PTR rbp 76
mov WORD PTR rax 8 dx
mov rax QWORD PTR rbp 24
mov rdi rax
call free
mov eax DWORD PTR rbp 12
add eax 1
cmp eax 2
jg .L4
.L4:
cmp DWORD PTR rbp 12 123456
jne .L5
.L8: @53
mov rax QWORD PTR rbp 24
mov rdi rax
call free
jmp .L5
.L4: @60
mov rax QWORD PTR rbp 24
mov BYTE PTR rax 97
.L5:
mov edx DWORD PTR rbp 92
mov eax DWORD PTR rbp 88
sub edx eax
lea eax rdx 100
mov DWORD PTR rbp 28 eax
mov eax DWORD PTR rbp 28
cdqe
mov rdi rax
call malloc
mov QWORD PTR rbp 40 rax
mov rax QWORD PTR rbp 40
mov rdx QWORD PTR rbp 84
mov QWORD PTR rax rdx
movzx edx WORD PTR rbp 76
mov WORD PTR rax 8 dx
mov eax DWORD PTR rbp 92
mov ecx DWORD PTR rbp 88
cdq
idiv ecx
mov DWORD PTR rbp 44 eax
mov eax DWORD PTR rbp 44
cdqe
mov rdi rax
call malloc
mov QWORD PTR rbp 56 rax
mov rax QWORD PTR rbp 56
mov rdx QWORD PTR rbp 84
mov QWORD PTR rax rdx
movzx edx WORD PTR rbp 76
mov WORD PTR rax 8 dx
mov eax DWORD PTR rbp 44
add eax 100
cdqe
movzx eax BYTE PTR rbp 106 rax
mov BYTE PTR rbp 57 al
mov rax QWORD PTR rbp 56
movzx eax BYTE PTR rax 100
mov BYTE PTR rbp 58 al
mov eax DWORD PTR rbp 44
add eax 100
cdqe
mov BYTE PTR rbp 106 rax 99
mov rax QWORD PTR rbp 56
add rax 100
mov BYTE PTR rax 99
cmp DWORD PTR rbp 44 10
jle .L6
.L6:
mov rax QWORD PTR rbp 56
mov rdi rax
call free
.L5: @110
mov QWORD PTR rbp 56 0
jmp .L7
.L7:
mov rax QWORD PTR rbp 40
mov rdi rax
call free