# C Program: Leap Year

The purpose is this program is to determine if a given year is a leap by using mathematical logic

## Original code from:

https://www.sanfoundry.com/c-program-check-year-leap/

```
1. /*
 2. * C program to find whether a given year is leap year or not
3.
4. void main()
 5. {
 6.
        int year;
 7.
8.
       printf("Enter a year \n");
 9.
       scanf("%d", &year);
       if ((year % 400) == 0)
10.
           printf("%d is a leap year \n", year);
11.
       else if ((year % 100) == 0)
12.
13.
           printf("%d is a not leap year \n", year);
       else if ((year % 4) == 0)
15.
           printf("%d is a leap year \n", year);
16.
       else
          printf("%d is not a leap year \n", year);
17.
18. }
```

The code from sanfoundry.com would not compile right so I changed some operators/operands in the code and modified the if-else statement to produce a cleaner and easier to digest code, user preference of course.

Moreover, I added a greeting (Welcome!) with instructions in the beginning of the code to greet and instruct user. Further I added some line advances (\n)

# Code outputs: ASM generation compiler returned: 0 Execution build compiler returned: 0 Program returned: 0 Welcome! To find out if a given year is a leap year, please enter a year below. Enter year: 0 is a Leap Year

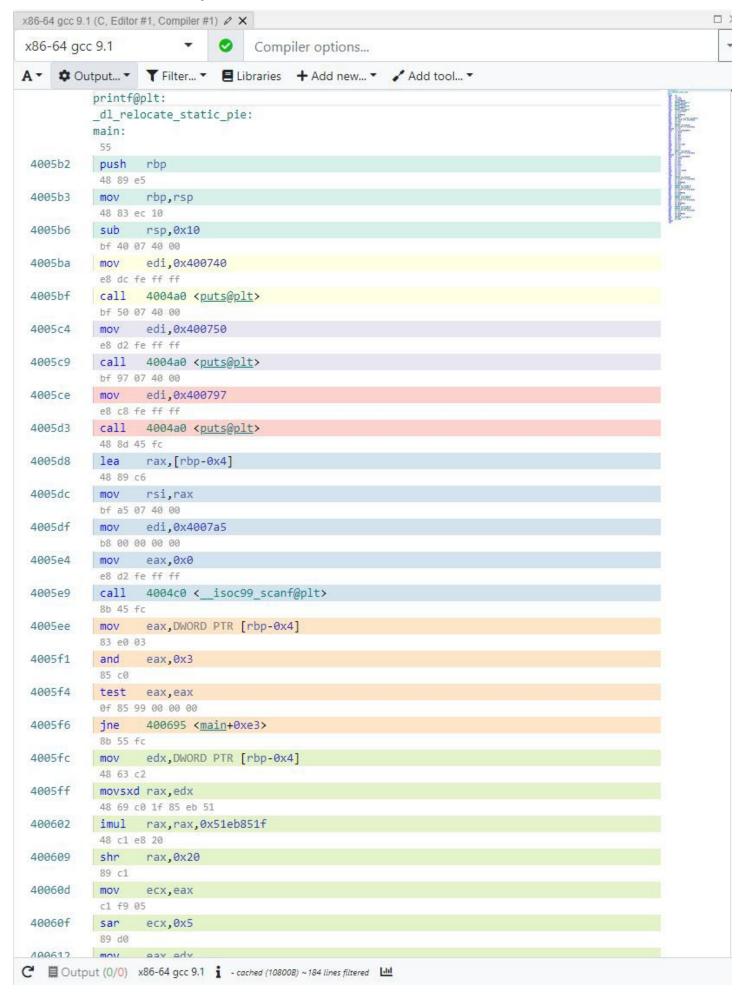
### Modified code used for this exercise:

```
#include <stdio.h>
1
2
3
     int main()
4 V
     {
5
         int y;
6
7
         printf("Welcome!\n");
         printf("To find out if a given year is a leap year, please enter a year below.\n");
8
9
         printf("\nEnter year: \n");
10
         scanf("%d",&y);
11
12
         if(y \% 4 == 0)
13
14 V
             if( y % 100 == 0)
15
16
                  if( y % 400 == 0)
17
                      printf("\n%d is a Leap Year", y);
18
19 V
                  else
                      printf("\n%d is not a Leap Year", y);
20
21
              }
22
             else
                 printf("%d is a Leap Year", y );
23
24
         }
25 V
         else
26
             printf("%d is not a Leap Year", y);
27
28
         return 0;
29
```

# **Code Assembly Equivalent:**

```
x86-64 gcc 9.1 (C, Editor #1, Compiler #1) & X
                                                                                                               \square \times
                                     Compiler options...
x86-64 gcc 9.1
     Output... T Filter... E Libraries + Add new... Add tool...
  1
       .LC0:
               .string "Welcome!"
  2
                                                                                                         3
       .LC1:
  4
               .string "To find out if a given year is a leap year, please enter a year below."
  5
       .LC2:
                                                                                                         Action because III.
               .string "\nEnter year: "
  6
  7
       .LC3:
               .string "%d"
  8
       .LC4:
 9
 10
               .string "\n%d is a Leap Year"
       .LC5:
 11
 12
               .string "\n%d is not a Leap Year"
 13
       .LC6:
               .string "%d is a Leap Year"
 14
 15
       .LC7:
               .string "%d is not a Leap Year"
 16
 17
      main:
               push
18
                       rbp
19
               mov
                       rbp, rsp
                       rsp, 16
 20
               sub
 21
                       edi, OFFSET FLAT: .LCO
               mov
 22
               call
 23
                       edi, OFFSET FLAT: .LC1
               mov
 24
               call
                        puts
 25
               mov
                       edi, OFFSET FLAT: .LC2
 26
               call
                       puts
 27
               lea
                       rax, [rbp-4]
 28
               mov
                       rsi, rax
                       edi, OFFSET FLAT: LC3
 29
               mov
 30
               mov
                       eax, 0
                        isoc99_scanf
 31
               call
                       eax, DWORD PTR [rbp-4]
 32
               mov
 33
               and
                       eax, 3
 34
               test
                        eax, eax
 35
                        .L2
               jne
                        edx, DWORD PTR [rbp-4]
 36
               mov
 37
               movsx
                       rax, edx
                       rax, rax, 1374389535
 38
               imul
 39
               shr
                       rax, 32
 40
               mov
                       ecx, eax
                       ecx, 5
 41
               sar
42
               mov
                       eax, edx
 43
               sar
                       eax, 31
 44
                       ecx, eax
               sub
 45
               mov
                       eax, ecx
 46
               imul
                       eax, eax, 100
47
               sub
                       edx, eax
48
               mov
                       eax, edx
 49
                       eax, eax
               test
 50
                        .L3
               jne
 51
               mov
                       edx, DWORD PTR [rbp-4]
 52
               movsx rax, edx
 53
               imul nav nav 127/1280525
   Utput (0/0) x86-64 gcc 9.1 : - cached (12556B) ~790 lines filtered
```

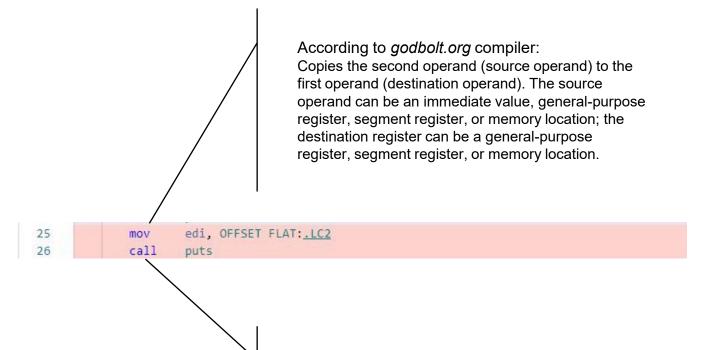
# **Code Hexadecimal Equivalent:**



# **Analyzing One Line of Code:**

```
printf("\nEnter year: \n");
```

This line of code <code>printf("\nEnter year: \n");</code> is executing two things. mov It's moving data back and from a hardware-supported stack in RAM into CPU registers. Additionally, call calls near procedures, causes the procedure named in the operand to be executed.



According to *godbolt.org* compiler:

Saves procedure linking information on the stack and branches to the called procedure specified using the target operand. The target operand specifies the address of the first instruction in the called procedure. The operand can be an immediate value, a general-purpose register, or a memory location.