Brian Barbu (brb9da) In-Lab 9

Optimized Code:

To analyze differences between optimized code made in the compiler using the -O2 versus regular code, I created a simple program and examined how their assembly files differed. Surprisingly, my first attempt at writing a C++ program did not result in a simpler assembly file when compiled with the -O2 flag. I attempted this using clang++ -mllvm --x86-asm-syntax=intel -m64 -S -O2 test1.cpp, but did not get any good results. I rewrote my program as a C program and was able to see definitive results. I compiled the code with the optimization flag using gcc -S -O2 test1.c.

The C Program:

```
#include <stdio.h>
int main(){
    int num;|
    printf( "Enter a number: ");
    scanf("%d", &num);
    int result = num;
    while(num>0){
        result = result * num;
        num--;
    }
    if( result % 2 == 0){
        result = result * 2;
    }
    else{
        result = result * 3;
    }
    return result;
}
```

There were 72 lines of assembly code in the non-optimized version of my program. However, the optimized version produced a much simpler 56 line file of assembly code.

Non-optimized Assembly:

```
.file
                   "test1.c"
          .text
          .section
                              .rodata
.LC0:
          .string "Enter a number: "
 LC1:
 Files
          .string "%d"
          .text
          .globl main
          .type
                    main, @function
main:
.LFB0:
                                              .L2:
          .cfi_startproc
                                                                  -16(%rbp), %eax
                                                        movl
          pushq %rbp
.cfi_def_cfa_offset 16
                                                        testl
                                                                  %eax, %eax
          .cri_der_cra_orrset 16
.cfi_offset 6, -16
movq %rsp, %rbp
.cfi_def_cfa_register 6
subq $16, %rsp
movq %fs:40, %rax
movq %rax, -8(%rbp)
xorl %eax, %eax
                                                        jg
                                                                  .L3
                                                                  -12(%rbp), %eax
                                                        movl
                                                        andl
                                                                  $1, %eax
                                                        testl
                                                                  %eax, %eax
                                                        jne
                                                                  .L4
                                                        sall
                                                                  -12(%rbp)
                                                        jmp
                                                                  .L5
                     .LCO(%rip), %rdi
          leaq
          movl
                    $0, %eax
                                                        movl
                                                                  -12(%rbp), %edx
          call
                    printf@PLT
                                                        movl
                                                                  %edx, %eax
          leaq
                     -16(%rbp), %rax
                                                        addl
                                                                  %eax, %eax
                    %rax, %rsi
.LC1(%rip), %rdi
          movq
                                                        addl
                                                                  %edx, %eax
          leaq
          movl
                    $0, %eax
                                                        movl
                                                                  %eax, -12(%rbp)
                    __isoc99_scanf@PLT
-16(%rbp), %eax
%eax, -12(%rbp)
          call
          movl
                                                        movl
                                                                  -12(%rbp), %eax
          movl
                                                        movq
                                                                  -8(%rbp), %rcx
          jmp
                    .L2
                                                        хога
                                                                  %fs:40, %rcx
.L3:
                    -16(%rbp), %eax
-12(%rbp), %edx
                                                        je
                                                                  .L7
          movl
                                                                  __stack_chk_fail@PLT
                                                        call
          movl
                    %edx, %eax
%eax, -12(%rbp)
-16(%rbp), %eax
                                              .L7:
          imull
          movl
                                                        leave
          movl
                                                        .cfi_def_cfa 7, 8
                    $1, %eax
%eax, -16(%rbp)
          subl
                                                        ret
          movl
                                                        .cfi_endproc
.L2:
                                              .LFE0:
          movl
                     -16(%rbp), %eax
                                                        .size
                                                                  main, .-main
          testl
                    %eax, %eax
                                                        .ident "GCC: (Ubuntu 7.3.0-27ubuntu1~18.04) 7.3.0"
          jg
          movl
                     -12(%rbp), %eax
                                                        .section
                                                                            .note.GNU-stack,"",@progbits
```

Optimized Assembly:

```
.file "test1.c"
       .section
                       .rodata.str1.1,"aMS",@progbits,1
.LCO:
       .string "Enter a number: "
.LC1:
        .strina "%d"
                       .text.startup, "ax", @progbits
        .p2align 4,,15
        .globl main
        .type
              main. @function
main:
.LFB23:
        .cfi_startproc
       subq
        subq $24, %rsp
.cfi_def_cfa_offset 32
        leaq
                .LCO(%rip), %rsi
               $1, %edi
%fs:40, %rax
       movl
       DVOM
               %rax, 8(%rsp)
               %eax, %eax
        xorl
                 _printf_chk@PLT
       call
               4(%rsp), %rsi
        leaq
       leaq
                .LC1(%rip), %rdi
       xorl
               %eax. %eax
                                                           .L2:
                 isoc99_scanf@PLT
       call
                                                                   leal
                                                                            (%rdx,%rdx), %ecx
               4(%rsp), %edx
       movl
                                                                   leal
                                                                            (%rdx,%rdx,2), %eax
       testl %edx, %edx
                                                                   andl
                                                                            $1, %edx
                                                                   cmove
                                                                            %ecx, %eax
       movl
               %edx, %eax
                                                                   movq
                                                                            8(%rsp), %rcx
       .p2align 4,,10
                                                                   хогq
       .p2align 3
                                                                            %fs:40. %rcx
.L3:
                                                                   jne
                                                                            .L10
       imull %eax, %edx
                                                                   addq
                                                                            $24, %rsp
               $1, %eax
.L3
       subl
                                                                   .cfi remember state
                                                                   .cfi_def_cfa_offset 8
.L2:
                                                                   ret
       leal
               (%rdx,%rdx), %ecx
                                                          .L10:
               (%rdx,%rdx,2), %eax
       leal
                                                                   .cfi restore state
       andl
               S1. %edx
                                                                            __stack_chk_fail@PLT
                                                                   call
               %ecx, %eax
       cmove
                                                                   .cfi_endproc
       pvom
               8(%rsp), %rcx
                                                          .LFE23:
       хога
               %fs:40, %rcx
                .L10
                                                                   .size main, .-main
       ine
                                                                   .ident "GCC: (Ubuntu 7.3.0-27ubuntu1~18.04) 7.3.0"
               $24, %rsp
        addq
                                                                                   .note.GNU-stack,"",@progbits
        .cfi_remember_state
                                                                   .section
```

When the -O2 flag is used, the compiler will try to minimize the instructions needed to run out the output. This means that instructions that do not directly influence the return value will often be omitted in the -O2 optimized version. One example of this is with the use of scanf. Scanf will have a return value, but it will be unused in creating the output and is thus omitted in the optimized assembly version. Another thing I noticed is that there were less loops used in the optimized version in comparison with the regular. Extraneous loops can be very detrimental for runtime as many iterations of useless instructions may be executed and slowing down the program. The optimized versions only executes the critical tasks. Another way -O2 is optimizing this program is by saving significant data early. Instructions are repositioned to make important data available in memory early and removes the need for certain mov load instructions that are called late in the original assembly and would slow runtime. Another example of how -O2 optimizes the code is by ignoring the base pointer. Typical assembly would push the pointer and allocate room on the stack, but this is significant memory usage and is not important to the simpler tasks the program is intended to complete. -O2 optimization then chooses to ignore managing the base pointer. Optimization in compilers using the -O2 flag is greatly effective in eliminating unnecessary instructions and generating the correct output in the shortest runtime.