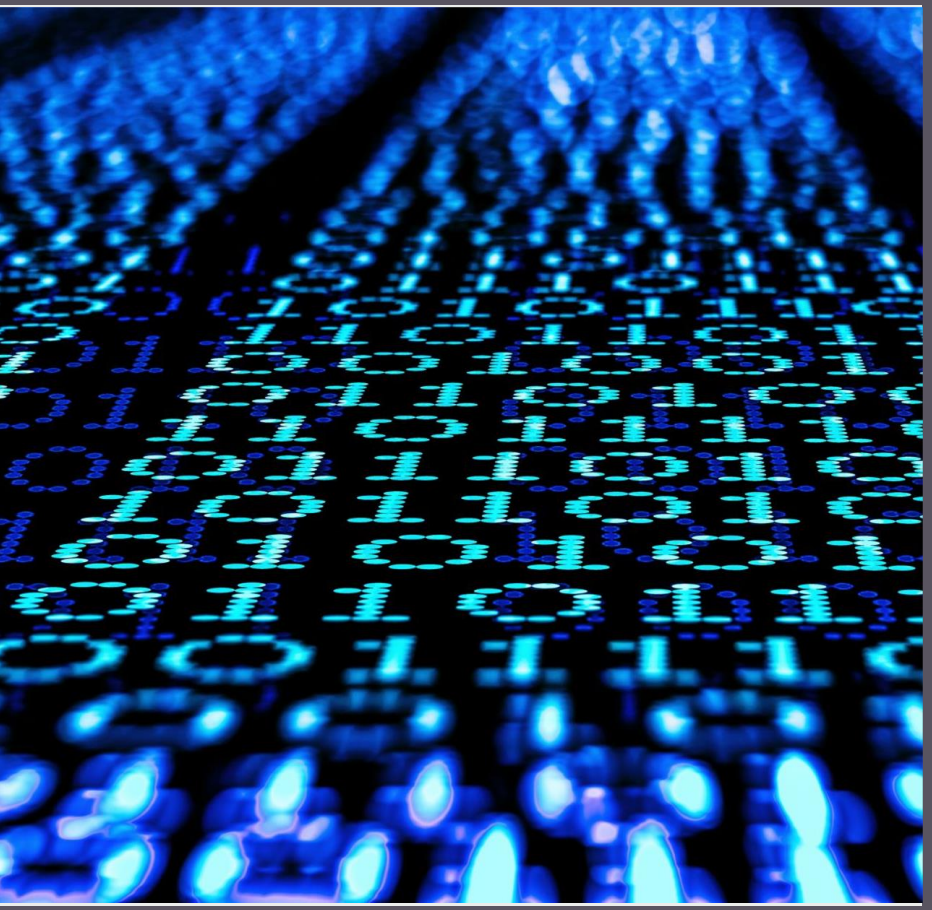

SC-369
CODE PROFILING AND OPTIMIZATION

Topic: Compiler Flags

Day 1

Instructors

Subhrajit Das

Pratyush Choudhary

GCC COMPILER OPTIMIZATION FLAGS

Diving deep into the abstractions of using GCC flags!

https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

Demo-1

```
int sample_function(int p)
{
   int n = 0;
   for (int i = 0; i < p; i++)
   {
      n = n + 1;
   }
   return n;
}
```

Compiler Flags

- **Focusing on GCC only**
- **Without any optimization option:**
- Compiler Goal:
- *Reduce Cost of Compilation*
- *To make debugging produce the expected results*

https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

Compiler Flags

- **Without any optimization option:**
- Compiler Goal:
- *Reduce Cost of Compilation*
- *To make debugging produce the expected results*
- **With Turning on Optimization Compilation Flags:**
- Compiler attempts to improve
- *Performance*
- *And/or Code Size*
- At the expense of compilation time and the ability to debug the program

https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

Types of Flags in GCC

- Performance flags
- Compiled code size flags
- Debugger flags

Usage: **gcc <file-name>.c <…> <Optimization Flag>**

Example:

gcc hello.c –O3

https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

So, what kind of Optimizations Compiler can do?

- Constant Propagation
- Loop unrolling
- Dead code elimination
- Loop invariants
- And many more!

Loop Unrolling

- **Original code**

```
for (i = 1000; i>0; i=i-1)
    x[i] = x[i] + s;
```

- **Unrolled four times**

```
for (i = 1000; i>0; i=i-4) {
    x[i] = x[i] + s;
    x[i-1] = x[i-1] + s;
    x[i-2] = x[i-2] + s;
    x[i-3] = x[i-3] + s;
}
```

https://image3.slideserve.com/5635126/loop-unrolling-l.jpg

Dead Code Elimination

```c
int calculate_square(int x) {
    int result = x * x;

    // This code is never executed and will be eliminated by the compiler
    if (0) {
        result = 0;  // Dead code
    }

    return result;
}
```

https://image3.slideserve.com/5635126/loop-unrolling-l.jpg

```c
int multiply_array(int arr[], int n, int multiplier) {
    int product_sum = 0;

    for (int i = 0; i < n; i++) {
        // Loop-invariant expression
        int scaled_value = multiplier * 10;  // Doesn't depend on 'i' and can be moved out
        product_sum += arr[i] * scaled_value;
    }

    return product_sum;
}
```

LOOP INVARIANT

https://image3.slideserve.com/5635126/loop-unrolling-l.jpg

Compiler Flags

-O0 or not set

- Disables all the optimization
- This is the default
- Goal:
- *Reduce the cost of compilation*
- *To make debugging produce the expected results*

https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

Compiler Flags (Performance Flags)

-O or -O1

- Goal: to reduce code size and execution time, excluding any optimizations that take a great deal of compilation time
- Recommended for large machine-generated code as a sensible balance between time taken to compile and memory use
- Turns on these optimization flags: 01 Flags

Demo-1

```
int sample_function(int p)
{
   int n = 0;
   for (int i = 0; i < p; i++)
   {
      n = n + 1;
   }
   return n;
}
```

Basically, incrementing n by p
From 0 to p;
Simple can return p itself!!

Demo-1

```
int sample_function(int p)
{
    int n = 0;
    for (int i = 0; i < p; i++)
    {
        n = n + 1;
    }
    return n;
}
```

Compiler Flags (Performance Flags)

-O2

- Optimize even more than O1.
- Performs nearly all supported optimizations that do not involve a space-speed tradeoff
- Compared to –O1, this option increases both compilation time and the performance of the generated code.
- -O2 turns on all optimization flags specified by -O1. It also turns on the following optimization flags: O2 Flags

https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

O2 DOES MORE OPTIMIZATION THAN O1
SO, WILL IT RESULT IN FASTER COMPUTATION TIME?

Demo-2

```
int sum_of_two_arrays(int *array1, int *array2, int size)
{
   int total_sum = 0;
   for (int i = 0; i < size; i++)
   {
      int add = array1[i] + array2[i];
      total_sum += add;
   }
   return total_sum;
}
```

Compiler Flags (Performance Flags)

-O3

- Optimize yet more
- -O3 turns on all optimizations specified by -O2
- And also turns on the following optimization flags:O3 Flags

Compiler Flags (Performance Flags)

-Ofast

- Disregard strict standards compliance.
- -Ofast enables all -O3 optimizations
- Also enables optimizations that are not valid for all standard-compliant programs
- Turns on: Ofast flags

Demo-3

```
void sum_of_two_arrays(int *array1, int
*array2, int *array3, int size)
{
   int total_sum = 0;
   for (int i = 0; i < size; i++)
   {
      array3[size - i - 1] += array1[i] +
array2[i];
   }
}
```

https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

Compiler Flags (Size Flags)

-Os

- Optimize for size.
- -Os enables all -O2 optimizations except those that often increase code size: Os Exclusions

Compiler Flags (Size Flags)

-Oz

- Optimize aggressively for size rather than speed.
- Oz behaves similarly to -Os including enabling most -O2 optimizations
- This may increase the number of instructions executed if those instructions require fewer bytes to encode.

https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

Compiler Flags (Debugger Flag)

-Og

- Optimize debugging experience.
- -Og should be the optimization level of choice for the standard edit-compile-debug cycle, offering a reasonable level of optimization while maintaining fast compilation and a good debugging experience
- A better choice than -O0 for producing debuggable code because some compiler passes that collect debug information are disabled at -O0.
- -Og enables all -O1 optimization flags except for those that may interfere with debugging: -Og Exclusions

https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

Compiler Flags (Multiple Flags)

If you use multiple -O options, with or without level numbers, the last such option is the one that is effective.

https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

THANKS!

That’s it for now!

https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html