Homework 4:Compile Runtime Assignment

Justin Campbell
UT eID: jsc4348
SDS 335
Dr. Eijkhout

September 29th 2021

1 First Iteration of Compile Exercise: No Transformations

In this section, images of the source file, "rotate.c" without any transformations, and the compiler runtimes of the program as they appear in the terminal using compilers, "gcc - O0", "gcc -01", "gcc -02", and "gcc -03" will be provided. This serves as a baseline with which to compare the compiler runtimes for each successive code transformation.

Provided below is a screenshot of the program, "rotate.c" in the first iteration (with no transformations):

```c
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

void rotate(double *x,double *y,double alpha) {
  double x0 = *x, y0 = *y;
  *x = cos(alpha) * x0 - sin(alpha) * y0;
  *y = sin(alpha) * x0 + cos(alpha) * y0;
  return;
}

#include <sys/time.h>

double mysecond()
{
  struct timeval tp;
  //struct timezone tzp;
  int i;

  i = gettimeofday(&tp,NULL /* &tzp */);
  return ( (double) tp.tv_sec + (double) tp.tv_usec * 1.e-6 );
}

#define NREPS 10000000

int main() {

  double x=.5, y=.5, alpha=1.57;
  double start = mysecond();
  for (int i=0; i<NREPS; i++)
    rotate(&x,&y,alpha);
  double duration = mysecond()-start;
  printf("Done after %e\n",duration);

  return 0;
}
```

Figure 1: Screenshot of Program "rotate.c" for First Iteration

```
login1.frontera(873)$ gcc -O0 -o rotate_1 rotate_1.c -lm
login1.frontera(874)$ rotate_1
Done after 4.420528e-01
login1.frontera(875)$ gcc -O1 -o rotate_1 rotate_1.c -lm
login1.frontera(876)$ rotate_1
Done after 2.702091e-01
login1.frontera(877)$ gcc -O2 -o rotate_1 rotate_1.c -lm
login1.frontera(878)$ rotate_1
Done after 2.735569e-01
login1.frontera(879)$ gcc -O3 -o rotate_1 rotate_1.c -lm
login1.frontera(880)$ rotate_1
Done after 9.536743e-07
login1.frontera(881)$
```

Figure 2: Screenshot of Compiler Runtimes for First Iteration

Observations:

1. The screenshot presented above provides the compilation commands and associated runtimes for the program, "rotate.c" in decreasing order of runtime (increasing effiency) under the different "gcc" compilers.
2. We can see that the "gcc-03" compiler improves the performance by a magnitude of 10^6 relative to the "gcc-00" compiler, namely, an improvement from $4.42 * 10^{-1}$ to $9.54 * 10^{-7}$

2 Second Iteration of Compile Exercise: Transformations Applied to Trigonometric Functions

In this section, images of the source file, "rotate.c" with its first set of transformations, and the compiler runtimes of the program as they appear in the terminal using compilers, "gcc - O0", "gcc -01", "gcc -02", and "gcc -03" will be provided. In particular, variables "*cos_alpha*" and "*sin_alpha*" were defined in the "*rotate*()" function definition as a substitute for explicit function calls for "sin" and "cos" in the assignment for "*x" and "*y" as was provided in the original iteration of the source file. This is determined to be good programming practice, and will generally improve the performance of code. The results from this first set of transformations will be used to compare the runtime efficiencies of the first and second iterations of the program.

Provided below is a screenshot of the program, "rotate.c" in the second iteration (with its first set of transformations):

File Edit Options Buffers Tools C Help

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

void rotate(double *x,double *y,double alpha) {
  double x0 = *x, y0 = *y;
  double cos_alpha = cos(alpha), sin_alpha = sin(alpha);
  *x = cos_alpha * x0 - sin_alpha * y0;
  *y = sin_alpha * x0 + cos_alpha * y0;
  return;
}

#include <sys/time.h>

double mysecond()
{
  struct timeval tp;
  //struct timezone tzp;
  int i;

  i = gettimeofday(&tp,NULL /* &tzp */);
  return ( (double) tp.tv_sec + (double) tp.tv_usec * 1.e-6 );
}

#define NREPS 10000000

int main() {

  double x=.5, y=.5, alpha=1.57;
  double start = mysecond();
  for (int i=0; i<NREPS; i++)
    rotate(&x,&y,alpha);
  double duration = mysecond()-start;
  printf("Done after %e\n",duration);

  return 0;
}
```

Figure 3: Screenshot of Program "rotate.c" for Second Iteration

```
login1.frontera(881)$ gcc -O0 -o rotate_2 rotate_1.c -lm
login1.frontera(882)$ rotate_2
Done after 2.443218e-01
login1.frontera(883)$ gcc -O1 -o rotate_2 rotate_1.c -lm
login1.frontera(884)$ rotate_2
Done after 2.623241e-01
login1.frontera(885)$ gcc -O2 -o rotate_2 rotate_1.c -lm
login1.frontera(886)$ rotate_2
Done after 2.624061e-01
login1.frontera(887)$ gcc -O3 -o rotate_2 rotate_1.c -lm
login1.frontera(888)$ rotate_2
Done after 9.536743e-07
login1.frontera(889)$
```

Figure 4: Screenshot of Compiler Runtimes for Second Iteration

Observations:

1. The screenshot presented above provides the compilation commands and associated runtimes for the program, "rotate.c" in decreasing order of runtime (increasing effiency) after the aforementioned changes to the computations of "*x" and "*y" were applied.

2. We can see that the "gcc -03" compiler again improves the performance by a magnitude of 10^6 relative to the "gcc -00" compiler, namely, an improvement from $2.44 * 10^{-1}$ to $9.54 * 10^{-7}$.

3. Overall, the performance improvement for the runtime for each compiler is comparable to that of the runtime of the associated compiler for the first iteration of the program. Put differently, this transformation marginally improved the runtime for each compiler and the differences in runtime across the compilers has a similar pattern.

3 Third Iteration of Compile Exercise: Transformations Applied to "printf()" Function Call

In this section, images of the source file, "rotate.c" with its second set of transformations, and the compiler runtimes of the program as they appear in the terminal using compilers, "gcc - O0", "gcc -01", "gcc -02", and "gcc -03" will be provided. The second set of transformations involved removing the variable declaration "double duration = mysecond() - start", and replacing the argument "duration" with "mysecond() - start" in the "printf()" function call. The results from these transformations will be used to compare the runtime efficiencies of the second and third iterations of the program.

Provided below is a screenshot of the program, "rotate.c" in the third iteration (with its second set of transformations):

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

void rotate(double *x,double *y,double alpha) {
  double x0 = *x, y0 = *y;
  double cos_alpha = cos(alpha), sin_alpha = sin(alpha);
  *x = cos_alpha * x0 - sin_alpha * y0;
  *y = sin_alpha * x0 + cos_alpha * y0;
  return;
}

#include <sys/time.h>

double mysecond()
{
  struct timeval tp;
  //struct timezone tzp;
  int i;

  i = gettimeofday(&tp,NULL /* &tzp */);
  return ( (double) tp.tv_sec + (double) tp.tv_usec * 1.e-6 );
}

#define NREPS 10000000

int main() {

  double x=.5, y=.5, alpha=1.57;
  double start = mysecond();
  for (int i=0; i<NREPS; i++)
    rotate(&x,&y,alpha);
  printf("Done after %e\n",mysecond()-start);

  return 0;
}
```

Figure 5: Screenshot of Program "rotate.c" for Third Iteration

```
login1.frontera(889)$ gcc -O0 -o rotate_3 rotate_1.c -lm
login1.frontera(890)$ rotate_3
Done after 2.516639e-01
login1.frontera(891)$ gcc -O1 -o rotate_3 rotate_1.c -lm
login1.frontera(892)$ rotate_3
Done after 2.713890e-01
login1.frontera(893)$ gcc -O2 -o rotate_3 rotate_1.c -lm
login1.frontera(894)$ rotate_3
Done after 2.708981e-01
login1.frontera(895)$ gcc -O3 -o rotate_3 rotate_1.c -lm
login1.frontera(896)$ rotate_3
Done after 9.536743e-07
login1.frontera(897)$
```

Figure 6: Screenshot of Compiler Runtimes for Third Iteration

Observations:

1. The screenshot presented above provides the compilation commands and associated runtimes for the program, ”rotate.c” in decreasing order of runtime (increasing efficiency) after the aforementioned changes to the variable declarations were applied.
2. As shown in the screenshot above, the transformation marginally increases the runtime of the code.
3. It should be noted again that the performance improvement of the runtime for each of the compilers is roughly the same as that for the runtimes in the first and second iterations of the program.

4 Fourth Iteration of Compile Exercise: Implementation of Macro Define Directive Statements

In this section, images of the source file, "rotate.c" with its third set of transformations, and the compiler runtimes of the program as they appear in the terminal using compilers, "gcc - O0", "gcc -01", "gcc -02", and "gcc -03" will be provided. In this set of transformations, the variable declarations for "*cos_alpha*" and "*sin_alpha*" in the "rotate()" function were replaced with macro "define" directive statements. The results from these transformations will be used to compare the runtime efficiencies of the third and fourth iterations of the program.

Provided below is a screenshot of the program, "rotate.c" in the fourth iteration (with its third set of transformations):

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

void rotate(double *x,double *y,double alpha) {
  double x0 = *x, y0 = *y;
  #define cos_alpha cos(alpha)
  #define sin_alpha sin(alpha)
  *x = cos_alpha * x0 - sin_alpha * y0;
  *y = sin_alpha * x0 + cos_alpha * y0;
  return;
}

#include <sys/time.h>

double mysecond()
{
  struct timeval tp;
  //struct timezone tzp;
  int i;

  i = gettimeofday(&tp,NULL /* &tzp */);
  return ( (double) tp.tv_sec + (double) tp.tv_usec * 1.e-6 );
}

#define NREPS 10000000

int main() {

  double x=.5, y=.5, alpha=1.57;
  double start = mysecond();
  for (int i=0; i<NREPS; i++)
    rotate(&x,&y,alpha);
  printf("Done after %e\n",mysecond()-start);

  return 0;
}
```

Figure 7: Screenshot of Program "rotate.c" for Fourth Iteration

```
login1.frontera(898)$ gcc -O0 -o rotate_4 rotate_1.c -lm
login1.frontera(899)$ rotate_4
Done after 4.419219e-01
login1.frontera(900)$ gcc -O1 -o rotate_4 rotate_1.c -lm
login1.frontera(901)$ rotate_4
Done after 2.619650e-01
login1.frontera(902)$ gcc -O2 -o rotate_4 rotate_1.c -lm
login1.frontera(903)$ rotate_4
Done after 2.622778e-01
login1.frontera(904)$ gcc -O3 -o rotate_4 rotate_1.c -lm
login1.frontera(905)$ rotate_4
Done after 9.536743e-07
login1.frontera(906)$
```

Figure 8: Screenshot of Compiler Runtimes for Fourth Iteration

Observations:

1. The screenshot presented above provides the compilation commands and associated runtimes for the program, "rotate.c" in decreasing order of runtime (increasing effiency) after the aforementioned changes to the variable declarations of "*cos_alpha*" and "*sin_alpha*" were applied.

2. Similar to the previous transformations, this change marginally increases the runtime of the code when the first compiler is used, marginally decreases the runtime when the second and third compiler are used, and nearly replicates the runtime when the fourth compiler is used.

3. Again, it should be noted that the performance improvement of the runtime for each of the compilers is roughly the same as that for the runtime in the first and second iterations of the code.