Новосибирский Государственный Университет

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Курс “ЭВМ и переферийные устройства”

Лабораторная работа №2

«ИЗУЧЕНИЕ ОПТИМИЗИРУЕЩЕГО КОМПИЛЯТОРА»

Выполнил:
Пятаев Егор, гр. 15206
Преподаватель:
Городничев Максим Александрович

Новосибирск 2016

Цели работы

1. Изучение основных функций оптимизирующего компилятора, и некоторых примеров оптимизирующих преобразований и уровней оптимизации.
2. Получение базовых навыков работы с компилятором GCC.
3. Исследование влияния оптимизационных настроек компилятора GCC на время исполнения программы.

Вариант задания

Алгоритм вычисления функции e^x с помощью разложения в ряд Маклорена по первым N членам этого ряда.

Таблица зависимости времени выполнения программы
с различными уровнями оптимизации

-O*	Время, сек	N
0	0.0002814040	50000
	0.0043513880	1000000
	0.2047690470	50000000
	4.0528589620	1000000000
	40.8347387140	10000000000
1	0.0002834700	50000
	0.0042388900	1000000
	0.2034364010	50000000
	4.0632819700	1000000000
	40.7227969360	10000000000
2	0.0002832220	50000
	0.0042903630	1000000
	0.2039566750	50000000
	4.0650748900	1000000000
	40.7270321770	10000000000
3		50000
		1000000
		50000000
		1000000000
		10000000000

Таблица зависимости времени выполнения программы
с различными уровнями оптимизации

-O*	Время, сек	N
s	0.0002819850	50000
	0.0042807640	1000000
	0.2021854730	50000000
	4.0306666810	1000000000
	40.4808239020	10000000000
fast	0.0002827250	50000
	0.0042777450	1000000
	0.2034549610	50000000
	4.0687560770	1000000000
	40.7685852050	10000000000
g	0.0002813930	50000
	0.0042691690	1000000
	0.2028085350	50000000
	4.0374056030	1000000000
	40.5295140140	10000000000

Листинг реализованной программы

header.h:

```
#ifndef H_1
#define H_1
#define BAD_ARGS 1
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

double calc_ex(double, double);

#endif
```

calc_ex.c:

```
#include "header.h"
double calc_ex(double n, double x) {

  double ex = 1;
  double i;
  double j = 1;

  /*Calculate ex*/
  if (x != 0) {
    for (i = 1; i < n; i++){
      j*=(x/i);
      ex+=j;
    }
  } else return ex;

  return ex;
}
```

main.c:

```
#include "header.h"

int main(int argc, char *argv[]) {
  double ex;
  double n;
  double x;
  struct timespec start, end;

if (argc == 3 ) {

  n = atof(argv[1]);
  x = atof(argv[2]);

  if(n <= 0){
    printf("Enter N > 0");
    return BAD_ARGS;
  }

  clock_gettime(CLOCK_MONOTONIC_RAW, &start);

  ex = calc_ex(n, x);

  clock_gettime(CLOCK_MONOTONIC_RAW, &end);

  printf("Time taken: %.10lf sec.\n",end.tv_sec-start.tv_sec+
0.000000001*(end.tv_nsec-start.tv_nsec));

  printf("e^x = %.10f\n", ex);

} else printf("Bad arguments");

  return 0;

}
```

Команда компиляции: gcc [-O*] main.c calc.c -o calc [-lrt]

Команда запуска: ./calc [значение N] [значение X]

Выводы

Для выполнения замеров времени при разных уровнях оптимизации компилятора gcc была использована программа с алгоритмом вычисления функции e^x с помощью разложения в ряд Маклорена по первым N членам этого ряда.

Из приведенной таблицы зависимости времени работы программы от уровня компиляции видно, что для данной программы изменение времени работы с разных случаях незначительно.