МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Вычислительная техника»

Дисциплина «Высокопроизводительные вычисления»

**Лабораторная работа №1.**

**Исследование способов измерения затрат времени исполнения кода**

Выполнил:

студент группы ИВТАПбд-31

Кондратьев П.С.

Проверил:

Негода В. Н.

Ульяновск, 2019

**Варианты заданий для my\_func**

|  |  |
| --- | --- |
| № | Функция |
| 11 | *ch x* |

**Цель работы:**

Изучение способов измерения затрат времени исполнения кода. Приобретение умений и навыков повышения точности измерения в различных условиях.

**Режимы конфигурации Debug и Release**

**Конфигурация сборки** (*build configuration*) – это набор настроек проекта, которые определяют, как ваша IDE будет строить ваш проект. Конфигурация сборки включает в себя: имя исполняемого файла, в какой директории исполняемый файл будет выводиться, в каких директорях IDE будет искать код и заголовочные файлы, сохранять или исключать информацию об отладке и насколько компилятор должен оптимизировать вашу программу.

**IDE** имеет две конфигурации сборки: Release (релиз) и Debug (отладка).

**Конфигурация Debug** предназначена для отладки вашей программы. Эта конфигурация отключает все настройки по оптимизации, включает информацию об отладке, что делает ваши программы больше и медленнее, но гораздо проще для проведения отладки. Debug обычно используется в качестве основной конфигурации по умолчанию.

**Конфигурация Release** используется для сборки программы для её дальнейшего выпуска. Таким образом версия программы будет более оптимизирована по размеру и производительности, и не будет содержать дополнительную информацию об отладке.

**Порядок работы:**

**1. Анализ базовых средств измерения затрат времени**

Анализ должен охватывать следующие средства:

* Функция сlock();
* Использование команды RDTSC;
* Использование таймера HPET (High Precision Event Timer).

Функция clock() описывается в time.h и возвращает число тиков от момента загрузки программы. Тик обычно равен 1 миллисекунде, но для возможности в последующем работать с другой длительностью такта в time.h фиксируется константа CLOCKS\_PER\_SEC (время изерения было взять в нс).

Фрагмент кода класса Clock():

class Clock : public Clocks { // часы на основе функции clock()

public:

Clock() { freq = CLOCKS\_PER\_SEC; name = "clock"; }

void measure(int sz) {

\_\_int64 t = clock(); // стартовая засечка

//my\_func(sz);

for (int i = 0; i < sz; i++)

dummy();

t = clock() - t; // замер продолжительности

time = 1000000000 \* t / freq;

}

};

Результаты применения этих средств для измерения затрат времени исполнения my\_func на компьютере 1 для класса Clock():

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | Режим Debug | Режим Release |
| Время (нс) Частота (гц) | Время (нс) Частота (гц) |
| 200000000 | 49961000000 1000  49387000000 1000  46873000000 1000  47918000000 1000 | 7818000000 1000  7552000000 1000  7257000000 1000  7389000000 1000 |
| 20000000 | 4780000000 1000  5080000000 1000  4848000000 1000  4923000000 1000 | 686000000 1000  719000000 1000  697000000 1000  706000000 1000 |
| 2000000 | 490000000 1000  526000000 1000  482000000 1000  468000000 1000 | 78000000 1000  82000000 1000  86000000 1000  71000000 1000 |
| 200000 | 46000000 1000  46000000 1000  46000000 1000  48000000 1000 | 6000000 1000  7000000 1000  7000000 1000  8000000 1000 |
| 20000 | 6000000 1000  5000000 1000  5000000 1000  5000000 1000 | 1000000 1000  1000000 1000  0 1000  1000000 1000 |
| 2000 | 0 1000  0 1000  1000000 1000  0 1000 | 1000000 1000  0 1000  0 1000  0 1000 |
| 200 | 0 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 20 | 0 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 2 | 0 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 1 | 0 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |

Результаты применения этих средств для измерения затрат времени исполнения my\_func на компьютере 2 для класса Clock():

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | Режим Debug | Режим Release |
| Время (нс) Частота (гц) | Время (нс) Частота (гц) |
| 200000000 | 61803000000 1000  64544000000 1000  59753000000 1000  65298000000 1000 | 4366000000 1000  3304000000 1000  3425000000 1000  3073000000 1000 |
| 20000000 | 6253000000 1000  6078000000 1000  5970000000 1000  5945000000 1000 | 302000000 1000  302000000 1000  302000000 1000  309000000 1000 |
| 2000000 | 591000000 1000  587000000 1000  602000000 1000  599000000 1000 | 30000000 1000  29000000 1000  29000000 1000  32000000 1000 |
| 200000 | 58000000 1000  59000000 1000  60000000 1000  58000000 1000 | 3000000 1000  5000000 1000  5000000 1000  3000000 1000 |
| 20000 | 5000000 1000  6000000 1000  6000000 1000  6000000 1000 | 0 1000  1000000 1000  1000000 1000  0 1000 |
| 2000 | 1000000 1000  0 1000  1000000 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 200 | 0 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 20 | 0 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 2 | 0 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 1 | 0 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |

Результаты применения этих средств для измерения затрат времени исполнения dummy на компьютере 1 для класса Clock():

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | Режим Debug | Режим Release |
| Время (нс) Частота (гц) | Время (нс) Частота (гц) |
| 200000000 | 4423000000 1000  4342000000 1000  5022000000 1000  4781000000 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 20000000 | 449000000 1000  429000000 1000  427000000 1000  418000000 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 2000000 | 42000000 1000  42000000 1000  42000000 1000  43000000 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 200000 | 5000000 1000  5000000 1000  5000000 1000  5000000 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 20000 | 1000000 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 2000 | 1000000 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 200 | 0 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 20 | 0 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 2 | 0 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 1 | 0 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |

Результаты применения этих средств для измерения затрат времени исполнения dummy на компьютере 2 для класса Clock():

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | Режим Debug | Режим Release |
| Время (нс) Частота (гц) | Время (нс) Частота (гц) |
| 200000000 | 5066000000 1000  3674000000 1000  3579000000 1000  3577000000 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 20000000 | 351000000 1000  353000000 1000  357000000 1000  359000000 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 2000000 | 36000000 1000  35000000 1000  36000000 1000  35000000 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 200000 | 3000000 1000  4000000 1000  3000000 1000  4000000 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 20000 | 1000000 1000  1000000 1000  1000000 1000  1000000 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 2000 | 0 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 200 | 0 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 20 | 0 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 2 | 0 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |
| 1 | 0 1000  0 1000  0 1000  0 1000 | 0 1000  0 1000  0 1000  0 1000 |

Счетчик тактов процессора TSC (Time Stamp Counter) встроен в ядро CPU и обычно прирастает на 1 с каждым тактом задающего генератора (это тот, что определяет гигагерцы процессора). В С/С++ он используется через функцию \_\_int64 \_\_ rdtsc(), описанную в windows.h.

Фрагмент кода класса TSC():

class TSC : public Clocks { // часы на основе Time Stamp Counter

public:

TSC() { name = "TSC"; }

void measure(int sz) {

freq = hz\_cpu();

\_\_int64 t = \_\_rdtsc(); // стартовая засечка

my\_func(sz);

/\*for (int i = 0; i < sz; i++)

dummy();\*/

t = \_\_rdtsc() - t; // замер продолжительности

time = 1000000000 \* t / freq;

}

\_\_int64 hz\_cpu() { // замер частоты

clock\_t t\_clock;

\_\_int64 t\_tsc;

t\_clock = clock() + CLOCKS\_PER\_SEC;

t\_tsc = \_\_rdtsc(); // взять TSC

while (clock() < t\_clock); // отсчет одной секунды

return (\_\_rdtsc() - t\_tsc); // частота в герцах

}

};

Результаты применения этих средств для измерения затрат времени исполнения my\_func на компьютере 1 для класса TSC():

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | Режим Debug | Режим Release |
| Время (нс) Частота (гц) | Время (нс) Частота (гц) |
| 200000000 | 2888876871 2710192710  -1146402240 2711365760  2564370338 2709745192  -1368986936 2711778304 | 641599060 2709917754  284622732 2710032105  -28998186 2710305925  -46879352 2710454338 |
| 20000000 | -2188440671 2711494076  -2060089349 2710442430  -2237484733 2711914928  -2181927564 2710183753 | 678506251 2711370086  670473128 2711488011  676352034 2710709141  672961476 2709506573 |
| 2000000 | 454616064 2711700802  471372592 2711849030  640434784 2711830545  643338577 2709535783 | 66978781 2709843007  66526592 2710030674  66820266 2710057679  67820911 2710255352 |
| 200000 | 65694958 2712507850  52575530 2709356090  51728077 2711119341  55225291 2710867248 | 6640453 2709605419  7158683 2710353577  7153281 2710350852  7578078 2711439421 |
| 20000 | 13944490 2711858344  6734466 2709815953  4396498 2710903603  4453273 2709687347 | 775560 2710583404  655990 2710619182  656679 2710012098  657798 2710696487 |
| 2000 | 461925 2711205082  460177 2709853387  559427 2710558978  462674 2711225585 | 61448 2709682075  61350 2711129466  61363 2711017812  62207 2711283195 |
| 200 | 17143 2711661244  16793 2709930858  16781 2711874301  16734 2711177486 | 5653 2709848220  5511 2711292902  5584 2709418900  5657 2710006461 |
| 20 | 3237 2710010447  4344 2711495235  3660 2709630288  3896 2709776306 | 1181 2711980215  1357 2710009969  1215 2709481431  1349 2710122887 |
| 2 | 2137 2711580479  1558 2712918028  1642 2718711407  2307 2710114359 | 658 2710331977  1130 2710467466  1133 2710719149  1155 2710787471 |
| 1 | 1456 2710818038  2055 2711120309  1179 2711645489  1525 2711424420 | 8465 2711729331  1480 2710922688  1180 2711203530  1440 2709943815 |

В таблице наблюдалось отрицательное время в нс, это показывает что разрешающая способность TSC() превышена, произошло переполнение TSC().

Результаты применения этих средств для измерения затрат времени исполнения my\_func на компьютере 2 для класса TSC():

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | Режим Debug | Режим Release |
| Время (нс) Частота (гц) | Время (нс) Частота (гц) |
| 200000000 | 3014577969 2590213092  2142530738 2591551110  2397023471 2591426289  2394883466 2590379537 | -3169655021 2590030639  3111291650 2591504321  3137218908 2590236936  3115885542 2591878249 |
| 20000000 | -574317185 2591416108  -862630827 2591167856  -1162613884 2591085326  -1187994758 2590205545 | 309717474 2590749497  313268985 2591021406  311574657 2590849708  311524272 2589719086 |
| 2000000 | 586085450 2591436434  605914894 2590626726  589947276 2590500272  590553440 2591499456 | 29877342 2589551712  33542821 2591301271  35477718 2591955569  29384065 2591468815 |
| 200000 | 63829378 2591702957  65268048 2591029430  60154926 2590792433  62106983 2591051153 | 3067385 2589625233  2913972 2590363181  3056706 2591724044  2896530 2589804880 |
| 20000 | 6639630 2591219937  6025761 2590303083  5539695 2589465806  5518867 2591679687 | 276040 2591702023  285968 2590059189  276936 2591135541  276709 2590320138 |
| 2000 | 448956 2589833880  402123 2591233454  460511 2591464386  417870 2589987235 | 24915 2591821186  24950 2591508695  34020 2590193094  26520 2589832472 |
| 200 | 15344 2590998557  15071 2590397882  15336 2589706633  20313 2589781991 | 1971 2590804290  1853 2589827030  1985 2589511654  3154 2590880522 |
| 20 | 2536 2591879305  3535 2589483607  5247 2590567444  2867 2590381802 | 773 2591881237  879 2589842122  591 2589628121  557 2591679251 |
| 2 | 1315 2589841236  1368 2590398478  1408 2590758584  1543 2591579602 | 2027 2590234792  474 2590065015  450 2591082688  558 2589519988 |
| 1 | 1311 2590348347  1019 2589498248  916 2590280963  1012 2591963680 | 21081 2591330346  654 2590090477  613 2589803771  774 2591485180 |

В таблице наблюдалось отрицательное время в нс, это показывает что разрешающая способность TSC() превышена, произошло переполнение TSC().

Результаты применения этих средств для измерения затрат времени исполнения dummy на компьютере 1 для класса TSC():

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | Режим Debug | Режим Release |
| Время (нс) Частота (гц) | Время (нс) Частота (гц) |
| 200000000 | -2557473446 2710295343  -2622134880 2711682194  -2623735197 2709931202  -2615657235 2709624779 | 8 2711482690  8 2709953893  8 2711742714  8 2709742730 |
| 20000000 | 417198321 2711912843  420778558 2711696347  430824254 2710841756  452257134 2710355913 | 8 2710423127  7 2711444518  8 2710741039  8 2711455876 |
| 2000000 | 47896354 2711909380  44895830 2711506424  41967086 2711161127  41262656 2710449277 | 8 2711827988  8 2711185994  7 2709817468  7 2709500966 |
| 200000 | 4324324 2710767417  4046180 2710317815  4056269 2711380901  4085366 2711759439 | 7 2711362521  7 2709799511  7 2710363265  7 2710469313 |
| 20000 | 430179 2711658376  402446 2711475310  401478 2709624235  406110 2709638526 | 8 2710155115  7 2710112396  8 2710517971  8 2711008530 |
| 2000 | 40274 2710615900  40380 2709870722  40289 2711083560  40311 2710117618 | 7 2710023710  8 2710831190  7 2711601625  7 2710224022 |
| 200 | 4214 2711870044  4191 2711491694  4198 2709643983  4205 2711637666 | 8 2709743731  8 2711940853  8 2711387784  8 2709599463 |
| 20 | 599 2710223369  576 2711722340  610 2711532703  587 2710644789 | 8 2709341703  8 2710475386  8 2710251301  7 2710751442 |
| 2 | 2998 2711188072  201 2710496648  212 2711583692  285 2710273412 | 7 2711025493  8 2710608309  7 2710804412  7 2709601712 |
| 1 | 196 2709556382  228 2711983312  219 2710445876  223 2710863128 | 8 2709697369  8 2710947239  8 2710939555  7 2710915081 |

В таблице наблюдалось отрицательное время в нс, это показывает что разрешающая способность TSC() превышена, произошло переполнение TSC().

Результаты применения этих средств для измерения затрат времени исполнения dummy на компьютере 2 для класса TSC():

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | Режим Debug | Режим Release |
| Время (нс) Частота (гц) | Время (нс) Частота (гц) |
| 200000000 | -3323685451 2591717769  -3356046307 2589906067  -3345108547 2590772176  -3166905679 2591689247 | 9 2590079171  8 2590318048  7 2591576669  7 2591286048 |
| 20000000 | 376237046 2590401280  377074105 2590360455  377096808 2590680900  375112800 2591344463 | 8 2591650933  8 2590601435  8 2590628976  7 2590561201 |
| 2000000 | 37079692 2590490418  36616142 2591516146  37136103 2590419167  36874298 2591903931 | 6 2591755177  7 2589690091  8 2590287078  7 2591119374 |
| 200000 | 5013584 2591096062  3680592 2591062741  3698833 2591626445  3650564 2591587647 | 9 2590217282  8 2591732278  7 2591538315  8 2590943636 |
| 20000 | 386083 2590655843  362657 2589528402  349736 2589479133  349467 2591421512 | 9 2591364026  8 2590474775  8 2591105657  9 2591348948 |
| 2000 | 56299 2589675928  35132 2589441532  66577 2590135390  35160 2589667167 | 8 2590650535  8 2591353488  8 2590155321  9 2590700211 |
| 200 | 3682 2590262087  3699 2591392202  3797 2590243761  3683 2590583603 | 7 2591855793  8 2591284303  9 2589461677  8 2589758873 |
| 20 | 561 2591908714  528 2591886204  543 2590392855  531 2591216095 | 7 2591835509  8 2591684124  7 2589946031  8 2589614222 |
| 2 | 201 2590043539  217 2590579490  232 2591524008  235 2590003946 | 8 2590216985  8 2590237073  7 2591908090  8 2589415722 |
| 1 | 199 2591663875  17533 2590330609  205 2590745205  334 2591012037 | 8 2590957632  7 2589559828  7 2590304515  7 2591385666 |

В таблице наблюдалось отрицательное время в нс, это показывает что разрешающая способность TSC() превышена, произошло переполнение TSC().

Анализ повторяемости в результатах измерения показывает, что за исключением первого измерения за исключением первого счетчика clock(), все имеют небольшие отклонения друг от друга.

Высокоточный таймер HPET (High Precision Timer ) обычно имеет разрешение в 100 наносекунд. В С/С++ он используется через следующие функции, описанные в windows.h:

QueryPerformanceCounter(&tick) // прочитать номер тика таймера

QueryPerformanceFrequency(&freq) // получить частоту таймера

Фрагмент кода класса QPC():

class QPC : public Clocks { // часы на основе QPC

public:

QPC() { QueryPerformanceFrequency((LARGE\_INTEGER \*)&freq); name = "QPC"; }

void measure(int sz) {

LONGLONG t\_start, t\_end;

QueryPerformanceCounter((LARGE\_INTEGER \*)&t\_start); // стартовая засечка

//my\_func(sz);

for (int i = 0; i < sz; i++)

dummy();

QueryPerformanceCounter((LARGE\_INTEGER \*)&t\_end);

time = 1000000000\* (t\_end - t\_start) / freq;

}

};

Результаты применения этих средств для измерения затрат времени исполнения my\_func на компьютере 1 для класса QPC():

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | Режим Debug | Режим Release |
| Время (нс) Частота (гц) | Время (нс) Частота (гц) |
| 200000000 | 47162467329 2648434  46598158383 2648434  45761827555 2648434  45616419740 2648434 | 4630630402 2648434  3036317310 2648434  3009517699 2648434  2998951455 2648434 |
| 20000000 | 4526144506 2648434  4537585984 2648434  4536601251 2648434  4530551639 2648434 | 306110705 2648434  307739214 2648434  306849632 2648434  329864742 2648434 |
| 2000000 | 453755691 2648434  467596322 2648434  475015801 2648434  451231935 2648434 | 29031873 2648434  28838928 2648434  32420668 2648434  30440252 2648434 |
| 200000 | 46809548 2648434  45190478 2648434  44133627 2648434  43901414 2648434 | 3300440 2648434  3213219 2648434  2785797 2648434  3032735 2648434 |
| 20000 | 4482648 2648434  4507946 2648434  4445645 2648434  5076584 2648434 | 309616 2648434  313392 2648434  312259 2648434  310749 2648434 |
| 2000 | 482549 2648434  588649 2648434  462537 2648434  470466 2648434 | 30961 2648434  36247 2648434  33227 2648434  39268 2648434 |
| 200 | 17368 2648434  17368 2648434  32849 2648434  15480 2648434 | 2643 2648434  1887 2648434  2643 2648434  1887 2648434 |
| 20 | 3020 2648434  2643 2648434  2265 2648434  3020 2648434 | 755 2648434  755 2648434  755 2648434  755 2648434 |
| 2 | 1132 2648434  1132 2648434  1132 2648434  755 2648434 | 755 2648434  377 2648434  377 2648434  377 2648434 |
| 1 | 755 2648434  1132 2648434  755 2648434  2265 2648434 | 6418 2648434  377 2648434  755 2648434  755 2648434 |

Результаты применения этих средств для измерения затрат времени исполнения my\_func на компьютере 2 для класса QPC():

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | Режим Debug | Режим Release |
| Время (нс) Частота (гц) | Время (нс) Частота (гц) |
| 200000000 | 61033355878 2531248  59662813758 2531248  60237391199 2531248  59795686159 2531248 | 4405074097 2531248  3199765491 2531248  3199377145 2531248  3207420213 2531248 |
| 20000000 | 8966485701 2531248  8448432551 2531248  8359412037 2531248  8626052050 2531248 | 315474817 2531248  316586916 2531248  314549581 2531248  324194231 2531248 |
| 2000000 | 857111590 2531248  860615791 2531248  887408108 2531248  896231819 2531248 | 31343827 2531248  30965357 2531248  30737407 2531248  38622055 2531248 |
| 200000 | 87567871 2531248  92267924 2531248  87786736 2531248  87919476 2531248 | 3228051 2531248  3165632 2531248  2962175 2531248  3021039 2531248 |
| 20000 | 9445143 2531248  8737982 2531248  8781834 2531248  8924846 2531248 | 438913 2531248  339753 2531248  338568 2531248  365432 2531248 |
| 2000 | 791704 2531248  726124 2531248  624593 2531248  753778 2531248 | 26864 2531248  27259 2531248  57283 2531248  27654 2531248 |
| 200 | 19358 2531248  19753 2531248  18962 2531248  18962 2531248 | 3160 2531248  1975 2531248  2765 2531248  2370 2531248 |
| 20 | 4740 2531248  2765 2531248  3160 2531248  3950 2531248 | 790 2531248  790 2531248  395 2531248  790 2531248 |
| 2 | 2370 2531248  1185 2531248  1580 2531248  1580 2531248 | 395 2531248  1975 2531248  395 2531248  790 2531248 |
| 1 | 790 2531248  790 2531248  790 2531248  790 2531248 | 10271 2531248  1185 2531248  395 2531248  790 2531248 |

Результаты применения этих средств для измерения затрат времени исполнения dummy на компьютере 1 для класса QPC():

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | Режим Debug | Режим Release |
| Время (нс) Частота (гц) | Время (нс) Частота (гц) |
| 200000000 | 5344703700 2648434  4713224871 2648434  4282420479 2648434  4230703880 2648434 | 0 2648434  377 2648434  0 2648434  0 2648434 |
| 20000000 | 418873568 2648434  425761412 2648434  419189226 2648434  425219960 2648434 | 0 2648434  0 2648434  0 2648434  0 2648434 |
| 2000000 | 42786416 2648434  42536079 2648434  42785661 2648434  41247771 2648434 | 0 2648434  0 2648434  0 2648434  0 2648434 |
| 200000 | 4120925 2648434  4343698 2648434  5138508 2648434  5793989 2648434 | 0 2648434  0 2648434  0 2648434  0 2648434 |
| 20000 | 430443 2648434  417227 2648434  423268 2648434  417227 2648434 | 0 2648434  377 2648434  377 2648434  0 2648434 |
| 2000 | 43799 2648434  49463 2648434  44554 2648434  74383 2648434 | 377 2648434  0 2648434  0 2648434  0 2648434 |
| 200 | 4530 2648434  4153 2648434  4153 2648434  4153 2648434 | 0 2648434  0 2648434  0 2648434  377 2648434 |
| 20 | 377 2648434  377 2648434  377 2648434  755 2648434 | 0 2648434  0 2648434  0 2648434  377 2648434 |
| 2 | 377 2648434  0 2648434  0 2648434  0 2648434 | 0 2648434  377 2648434  0 2648434  377 2648434 |
| 1 | 0 2648434  0 2648434  377 2648434  0 2648434 | 0 2648434  0 2648434  0 2648434  755 2648434 |

Результаты применения этих средств для измерения затрат времени исполнения dummy на компьютере 2 для класса QPC():

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | Режим Debug | Режим Release |
| Время (нс) Частота (гц) | Время (нс) Частота (гц) |
| 200000000 | 4078947222 2531248  3838500218 2531248  3793475392 2531248  3762682281 2531248 | 0 2531248  0 2531248  0 2531248  0 2531248 |
| 20000000 | 373008492 2531248  376855211 2531248  391702037 2531248  375185284 2531248 | 0 2531248  0 2531248  0 2531248  0 2531248 |
| 2000000 | 37689313 2531248  36860868 2531248  37075782 2531248  37830745 2531248 | 0 2531248  395 2531248  0 2531248  0 2531248 |
| 200000 | 4611164 2531248  3810373 2531248  3953780 2531248  3736891 2531248 | 0 2531248  0 2531248  0 2531248  0 2531248 |
| 20000 | 391901 2531248  368197 2531248  378864 2531248  382815 2531248 | 0 2531248  0 2531248  0 2531248  0 2531248 |
| 2000 | 35950 2531248  51753 2531248  53728 2531248  36345 2531248 | 0 2531248  0 2531248  0 2531248  0 2531248 |
| 200 | 3555 2531248  3950 2531248  3555 2531248  3555 2531248 | 0 2531248  0 2531248  0 2531248  0 2531248 |
| 20 | 395 2531248  395 2531248  395 2531248  395 2531248 | 0 2531248  0 2531248  0 2531248  0 2531248 |
| 2 | 0 2531248  395 2531248  395 2531248  0 2531248 | 0 2531248  0 2531248  0 2531248  0 2531248 |
| 1 | 0 2531248  395 2531248  395 2531248  0 2531248 | 0 2531248  0 2531248  0 2531248  0 2531248 |

**Выводы:**

1. В результате измерений для различных таймеров получены значения одного порядка.
2. Результаты применения этих средств для измерения затрат времени исполнения my\_func и пустой функции dummy показали, что время исполнения my\_func больше в 11 раз, чем время исполнения пустой функции dummy. Это объясняется тем, что при выполнении my\_func рассчитывается значение математического выражения.
3. В конфигурации Release время исполнения my\_func меньше в 6 раз, чем в конфигурации Debug. В конфигурации Release время исполнения пустой функции dummy равно 0. Это объясняется тем, что в конфигурации Release удаляется отладочная информация из исполняемого файла. Это приводит к уменьшению размера исполняемого файла (обычно в несколько раз). Исключаются дополнительные проверки, например, инициализированы переменные или нет. Поэтому в конфигурации Release программа может работать значительно быстрее.

**2. Исследование разрешающей способности и точности измерений при использовании функции clock**

Разрешающая способность оценивается на основе измерения времени исполнения функции dummy в цикле с увеличением числа повторов, пока время не станет больше 0.

Фрагмент кода:

void Clocks::measSeries(int scount, int arcount) {

//cout << endl << name << ": " << arcount << " элементов" << endl

// << "Время (мc) \tЧастота (гц)\n";

for (int i = 0; i < scount; i++) {

measure(arcount);

cout << endl << name << ": " << arcount << " элементов" << endl

<< "Время (мc) \tЧастота (гц)\n";

cout.width(10);

cout.setf(ios::right);

cout << time << "\t" << freq;

}

}

class Clock : public Clocks { // часы на основе функции clock()

public:

Clock() { freq = CLOCKS\_PER\_SEC; name = "clock"; }

void measure(int sz) {

\_\_int64 t = clock(); // стартовая засечка

//my\_func(sz);

for (int i = 0; i < sz; i++)

dummy();

t = clock() - t; // замер продолжительности

time = 1000 \* t / freq;

}

};

int main()

{

Clock cl;

QPC qpc;

TSC tsc;

setlocale(LC\_CTYPE, "rus");

\_\_int64 sum = 0;

int sz = 1000;

do {

cl.measSeries(1, sz);

//qpc.measSeries(4, sz);

//tsc.measSeries(4, sz);

sz += 1000;

} while (cl.time < 3);

sum += sz;

system("pause");

return 0;

}

Результаты исполнения функции dummy:

|  |  |
| --- | --- |
| Количество элементов | Время (мc) |
| 1000 | 0 |
| 2000 | 0 |
| 3000 | 0 |
| 4000 | 1 |
| 5000 | 0 |
| 6000 | 0 |
| 7000 | 0 |
| 8000 | 0 |
| 9000 | 1 |
| 10000 | 0 |
| 11000 | 0 |
| 12000 | 0 |
| 13000 | 0 |
| 14000 | 0 |
| 15000 | 1 |
| 16000 | 0 |
| 17000 | 0 |
| 18000 | 0 |
| 19000 | 1 |
| 20000 | 0 |
| 21000 | 0 |
| 22000 | 1 |
| 23000 | 0 |
| 24000 | 0 |
| 25000 | 0 |
| 26000 | 0 |
| 27000 | 1 |
| 28000 | 0 |
| 29000 | 0 |
| 30000 | 1 |
| 31000 | 1 |
| 32000 | 1 |
| 33000 | 1 |
| 34000 | 0 |
| 35000 | 1 |
| 36000 | 0 |
| 37000 | 1 |
| 38000 | 1 |
| 39000 | 1 |
| 40000 | 1 |
| 41000 | 1 |
| 42000 | 1 |
| 43000 | 1 |
| 44000 | 1 |
| 45000 | 1 |
| 46000 | 2 |
| 47000 | 2 |
| 48000 | 2 |
| 49000 | 2 |
| 50000 | 1 |
| 51000 | 2 |
| 52000 | 2 |
| 53000 | 2 |
| 54000 | 1 |
| 55000 | 2 |
| 56000 | 1 |
| 57000 | 2 |
| 58000 | 3 |

**Среднее значение разрешающей способности: 1 мс**

Таблицы с результатами оценки линейности:

Точность измеряется на основе my\_func с линейно увеличивающимся объемом вычислительной работы.

1 запуск

Фрагмент кода:

void Clocks::measSeries(int scount, int arcount) {

//cout << endl << name << ": " << arcount << " элементов" << endl

// << "Время (м) \tЧастота (гц)\n";

for (int i = 0; i < scount; i++) {

measure(arcount);

if ((time >= 20)&&(time <= 2020)) {

cout.width(15);

cout.setf(ios::right);

cout << arcount << "\t\t\t" << time<<endl;

}

}

}

int main()

{

Clock cl;

QPC qpc;

TSC tsc;

setlocale(LC\_CTYPE, "rus");

cout << endl << "Количество элементов " << "Время (мc) \n";

int sz = 100000;

while (cl.time <= 2020) {

cl.measSeries(1, sz);

//qpc.measSeries(4, sz);

//tsc.measSeries(4, sz);

sz += 1000000;

}

system("pause");

return 0;

}

Результаты исполнения функции my\_func:

|  |  |
| --- | --- |
| Количество элементов | Время (мc) |
| 1000000 | 25 |
| 3000000 | 90 |
| 5000000 | 137 |
| 7000000 | 164 |
| 9000000 | 240 |
| 11000000 | 286 |
| 13000000 | 297 |
| 15000000 | 316 |
| 17000000 | 363 |
| 19000000 | 399 |
| 21000000 | 441 |
| 23000000 | 493 |
| 25000000 | 526 |
| 27000000 | 569 |
| 29000000 | 616 |
| 31000000 | 646 |
| 33000000 | 702 |
| 35000000 | 739 |
| 37000000 | 784 |
| 39000000 | 817 |
| 41000000 | 867 |
| 43000000 | 910 |
| 45000000 | 950 |
| 47000000 | 986 |
| 49000000 | 1049 |
| 51000000 | 1091 |
| 53000000 | 1128 |
| 55000000 | 1282 |
| 57000000 | 1222 |
| 59000000 | 1345 |
| 61000000 | 1340 |
| 63000000 | 1395 |
| 65000000 | 1605 |
| 67000000 | 1566 |
| 69000000 | 1485 |
| 71000000 | 1525 |
| 73000000 | 1664 |
| 75000000 | 1720 |
| 77000000 | 1791 |
| 79000000 | 1805 |
| 81000000 | 1962 |
| 83000000 | 1877 |
| 85000000 | 1835 |
| 87000000 | 1937 |
| 89000000 | 1930 |
| 91000000 | 1995 |
| 93000000 | 1993 |

Графики зависимости затрат времени T от количества элементов count

1 запуск

2 запуск (анализ повторяемости)

Построенные графики подтверждают линейную зависимость затрат времени T от количества элементов count. Нарушения линейности обнаруживаются при многократном запуске теста, что объясняется тем, что нужных данных в кэше не окажется и процессор вынужден будет простаивать, ожидая их из оперативной памяти, а алгоритмы выборки данных в кэш процессора как раз и настроены на чтение последовательных значений. Когда необходимых данных в кэше процессора не оказывалось и возникает вынужденная задержка в его работе из-за ожидания необходимых значений из оперативной памяти.

Результаты анализа повторяемости в 1000 измерениях

Фрагмент кода:

void Clocks::measSeries(int scount, int arcount) {

//cout << endl << name << ": " << arcount << " элементов" << endl

// << "Время (м) \tЧастота (гц)\n";

for (int i = 0; i < scount; i++) {

measure(arcount);

cout.width(15);

cout.setf(ios::right);

cout << arcount << "\t\t" << time<<endl;

}

}

int main()

{

Clock cl;

QPC qpc;

TSC tsc;

setlocale(LC\_CTYPE, "rus");

\_\_int64 sum = 0;

cout << endl << "Номер " << "Количество элементов " << "Время (мc) \n";

srand(time(NULL));

for (int i = 0; i < 1000; i++)

{

cout << i + 1;

cl.measSeries(1, 2000000 \* (rand() % 10 + 1));

//qpc.measSeries(4, sz);

//tsc.measSeries(4, sz);

}

cout << endl;

system("pause");

return 0;

}

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | count | mtime(мc) | ctime(мc) | i | xi | (xi - xср)^2 | Параметры линейного приближения по методу наименьших квадратов | |
| 1 | 16000000 | 4288 | 3739,3 | 778 | 548,7 | 252194,970 | k | b |
| 2 | 16000000 | 3742 | 3739,3 | 778 | 2,7 | 1919,301 | 2,34E-04 | - 6,1789 |
| 3 | 14000000 | 3371 | 3271,1 | 667 | 99,9 | 2848,527 | count min | count max |
| 4 | 12000000 | 2804 | 2802,9 | 556 | 1,1 | 2065,428 | 2000000 | 20000000 |
| 5 | 14000000 | 3554 | 3271,1 | 667 | 282,9 | 55871,532 | (countmax - countmin) /1000 |  |
| 6 | 4000000 | 1132 | 930,2 | 111 | 201,8 | 24111,478 | 18000 |  |
| 7 | 12000000 | 3348 | 2802,9 | 556 | 545,1 | 248555,114 | xср |  |
| 8 | 18000000 | 6000 | 4207,5 | 889 | 1792,5 | 3048546,557 | 46,537362 |  |
| 9 | 12000000 | 2928 | 2802,9 | 556 | 125,1 | 6170,577 | sum (xi - xср)^2 |  |
| 10 | 2000000 | 617 | 462,0 | 0 | 155,0 | 11763,601 | 19093395 |  |
| 11 | 16000000 | 5778 | 3739,3 | 778 | 2038,7 | 3968821,688 | S |  |
| 12 | 2000000 | 525 | 462,0 | 0 | 63,0 | 270,936 | 138,25 |  |
| 13 | 14000000 | 3771 | 3271,1 | 667 | 499,9 | 205545,805 | xср-S | xср+S |
| 14 | 18000000 | 5965 | 4207,5 | 889 | 1757,5 | 2927550,944 | -91,7 | 184,79 |
| 15 | 12000000 | 4229 | 2802,9 | 556 | 1426,1 | 1903166,534 |  |  |
| 16 | 20000000 | 6142 | 4675,6 | 1000 | 1466,4 | 2015909,640 |  |  |
| 17 | 20000000 | 5367 | 4675,6 | 1000 | 691,4 | 415802,282 |  |  |
| 18 | 12000000 | 3812 | 2802,9 | 556 | 1009,1 | 926508,316 |  |  |
| 19 | 4000000 | 1305 | 930,2 | 111 | 374,8 | 107766,913 |  |  |
| 20 | 12000000 | 3338 | 2802,9 | 556 | 535,1 | 238684,053 |  |  |
| … | … | … | … | … | … | … |  |  |
| 985 | 12000000 | 2787 | 2802,9 | 556 | 15,9 | 938,059 |  |  |
| 986 | 6000000 | 1374 | 1398,4 | 222 | 24,4 | 491,598 |  |  |
| 987 | 18000000 | 4356 | 4207,5 | 889 | 148,5 | 10405,785 |  |  |
| 988 | 10000000 | 2305 | 2334,7 | 444 | 29,7 | 282,548 |  |  |
| 989 | 18000000 | 4171 | 4207,5 | 889 | 36,5 | 101,676 |  |  |
| 990 | 6000000 | 1379 | 1398,4 | 222 | 19,4 | 738,318 |  |  |
| 991 | 16000000 | 3714 | 3739,3 | 778 | 25,3 | 452,196 |  |  |
| 992 | 16000000 | 3711 | 3739,3 | 778 | 28,3 | 333,606 |  |  |
| 993 | 8000000 | 1841 | 1866,5 | 333 | 25,5 | 440,605 |  |  |
| 994 | 18000000 | 4172 | 4207,5 | 889 | 35,5 | 122,843 |  |  |
| 995 | 10000000 | 2329 | 2334,7 | 444 | 5,7 | 1665,388 |  |  |
| 996 | 12000000 | 2787 | 2802,9 | 556 | 15,9 | 938,059 |  |  |
| 997 | 10000000 | 2313 | 2334,7 | 444 | 21,7 | 615,495 |  |  |
| 998 | 2000000 | 466 | 462,0 | 0 | 4,0 | 1809,640 |  |  |
| 999 | 18000000 | 4162 | 4207,5 | 889 | 45,5 | 1,174 |  |  |

**Выводы:** Среднее значение отклонения расчетной оценки времени ctime(count) от измеренной mtime(count) составило 46,537362 мс. Оценка среднеквадратического отклонения составила 138,25 мс., что говорит о том, что значения отклонения расчетной оценки времени ctime(count) от измеренной mtime(count) в среднем находятся в диапазоне от -91,7 мс. до 184,79мс.

**3. Исследование разрешающей способности и точности измерений при использовании команды RDTSC.**

Промежуточные и конечные результаты оценки разрешающей способности:

Разрешающая способность оценивается на основе измерения времени исполнения функции dummy в цикле с увеличением числа повторов, пока время не станет больше 0.

Фрагмент кода:

void Clocks::measSeries(int scount, int arcount) {

//cout << endl << name << ": " << arcount << " элементов" << endl

// << "Время (м) \tЧастота (гц)\n";

for (int i = 0; i < scount; i++) {

measure(arcount);

cout << endl << name << ": " << arcount << " элементов" << endl

<< "Время (нc) \tЧастота (гц)\n";

cout.width(10);

cout.setf(ios::right);

cout << time << "\t" << freq << endl;

}

}

class TSC : public Clocks { // часы на основе Time Stamp Counter

public:

TSC() { name = "TSC"; }

void measure(int sz) {

freq = hz\_cpu();

\_\_int64 t = \_\_rdtsc(); // стартовая засечка

//my\_func(sz);

for (int i = 0; i < sz; i++)

dummy();

t = \_\_rdtsc() - t; // замер продолжительности

time = 1000000000 \* t / freq;

}

\_\_int64 hz\_cpu() { // замер частоты

clock\_t t\_clock;

\_\_int64 t\_tsc;

t\_clock = clock() + CLOCKS\_PER\_SEC;

t\_tsc = \_\_rdtsc(); // взять TSC

while (clock() < t\_clock); // отсчет одной секунды

return (\_\_rdtsc() - t\_tsc); // частота в герцах

}

};

int main()

{

Clock cl;

QPC qpc;

TSC tsc;

setlocale(LC\_CTYPE, "rus");

\_\_int64 sum = 0;

int sz = 10;

for (int i = 0; i < 10; i++)

{

do {

//cl.measSeries(1, sz);

//qpc.measSeries(4, sz);

tsc.measSeries(1, sz);

sz += 10;

} while (tsc.time < 3);

sum += sz;

}

system("pause");

return 0;

}

|  |  |
| --- | --- |
| Количество элементов | Время (нc) |
| 10 | 386 |
| 20 | 711 |
| 30 | 845 |
| 40 | 1022 |
| 50 | 1204 |
| 60 | 1756 |
| 70 | 1802 |
| 80 | 1821 |
| 90 | 2014 |
| 100 | 2250 |

**Среднее значение разрешающей способности: 1 нс**

Таблицы с результатами оценки линейности:

Точность измеряется на основе my\_func с линейно увеличивающимся объемом вычислительной работы.

1 запуск

Фрагмент кода:

void Clocks::measSeries(int scount, int arcount) {

//cout << endl << name << ": " << arcount << " элементов" << endl

// << "Время (м) \tЧастота (гц)\n";

for (int i = 0; i < scount; i++) {

measure(arcount);

if ((time >= 20000000) && (time <= 2020000000)) {

cout.width(15);

cout.setf(ios::right);

cout << arcount << "\t\t\t" << time << endl;

}

}

}

class TSC : public Clocks { // часы на основе Time Stamp Counter

public:

TSC() { name = "TSC"; }

void measure(int sz) {

freq = hz\_cpu();

\_\_int64 t = \_\_rdtsc(); // стартовая засечка

my\_func(sz);

//for (int i = 0; i < sz; i++)

//dummy();

t = \_\_rdtsc() - t; // замер продолжительности

time = 1000000000 \* t / freq;

}

\_\_int64 hz\_cpu() { // замер частоты

clock\_t t\_clock;

\_\_int64 t\_tsc;

t\_clock = clock() + CLOCKS\_PER\_SEC;

t\_tsc = \_\_rdtsc(); // взять TSC

while (clock() < t\_clock); // отсчет одной секунды

return (\_\_rdtsc() - t\_tsc); // частота в герцах

}

};

int main()

{

Clock cl;

QPC qpc;

TSC tsc;

setlocale(LC\_CTYPE, "rus");

cout << endl << "Количество элементов " << "Время (нc) \n";

int sz = 1000000;

while (tsc.time <= 2020000000) {

//cl.measSeries(1, sz);

//qpc.measSeries(4, sz);

tsc.measSeries(1, sz);

sz += 1000000;

}

system("pause");

return 0;

}

Результаты исполнения функции my\_func:

|  |  |
| --- | --- |
| Количество элементов | Время (нc) |
| 1000000 | 39492800 |
| 2000000 | 57934445 |
| 3000000 | 91716614 |
| 4000000 | 111287950 |
| 5000000 | 145211081 |
| 6000000 | 171423481 |
| 7000000 | 198604556 |
| 8000000 | 228574574 |
| 9000000 | 253833859 |
| 10000000 | 298306829 |
| 11000000 | 306808516 |
| 12000000 | 339685192 |
| 13000000 | 375384566 |
| 14000000 | 421365576 |
| 15000000 | 481435239 |
| 16000000 | 445739412 |
| 17000000 | 478976572 |
| 18000000 | 508339713 |
| 19000000 | 536482502 |
| 20000000 | 642477924 |
| 21000000 | 660885323 |
| 1000000 | 39492800 |
| 2000000 | 57934445 |
| 3000000 | 91716614 |
| 4000000 | 111287950 |
| 5000000 | 145211081 |
| 6000000 | 171423481 |
| 7000000 | 198604556 |
| 8000000 | 228574574 |
| 9000000 | 253833859 |
| 10000000 | 298306829 |
| 11000000 | 306808516 |
| 12000000 | 339685192 |
| 13000000 | 375384566 |
| 14000000 | 421365576 |
| 15000000 | 481435239 |
| 16000000 | 445739412 |
| 17000000 | 478976572 |
| 18000000 | 508339713 |
| 19000000 | 536482502 |
| 20000000 | 642477924 |
| 21000000 | 660885323 |
| … | … |
| 59000000 | 1640416570 |
| 60000000 | 1829373947 |
| 61000000 | 1773067300 |
| 62000000 | 1716511900 |
| 63000000 | 1759418601 |
| 64000000 | 1785960515 |
| 65000000 | 1817904652 |
| 66000000 | 1827515269 |
| 67000000 | 1860562441 |
| 68000000 | 1894277546 |
| 69000000 | 1918626926 |
| 70000000 | 1943920467 |
| 71000000 | 1974714727 |
| 59000000 | 1640416570 |
| 60000000 | 1829373947 |
| 61000000 | 1773067300 |
| 62000000 | 1716511900 |
| 63000000 | 1759418601 |

Графики зависимости затрат времени T от количества элементов count

1 запуск

2 запуск (анализ повторяемости)

Построенные графики подтверждают линейную зависимость затрат времени T от количества элементов count. Нарушения линейности обнаруживаются при многократном запуске теста, что объясняется тем, что нужных данных в кэше не окажется и процессор вынужден будет простаивать, ожидая их из оперативной памяти, а алгоритмы выборки данных в кэш процессора как раз и настроены на чтение последовательных значений. Когда необходимых данных в кэше процессора не оказывалось и возникает вынужденная задержка в его работе из-за ожидания необходимых значений из оперативной памяти.

Результаты анализа повторяемости в 1000 измерениях

Фрагмент кода:

int main()

{

Clock cl;

QPC qpc;

TSC tsc;

setlocale(LC\_CTYPE, "rus");

\_\_int64 sum = 0;

cout << endl << "Номер " << "Количество элементов " << "Время (нc) \n";

srand(time(NULL));

for (int i = 0; i < 1000; i++)

{

cout << i + 1;

tsc.measSeries(1, 6000000 \* (rand() % 10 + 1));

}

cout << endl;

system("pause");

return 0;

}

void Clocks::measSeries(int scount, int arcount) {

//cout << endl << name << ": " << arcount << " элементов" << endl

// << "Время (м) \tЧастота (гц)\n";

for (int i = 0; i < scount; i++) {

measure(arcount);

cout.width(15);

cout.setf(ios::right);

cout << arcount << "\t\t" << time << endl;

}

}

class TSC : public Clocks { // часы на основе Time Stamp Counter

public:

TSC() { name = "TSC"; }

void measure(int sz) {

freq = hz\_cpu();

\_\_int64 t = \_\_rdtsc(); // стартовая засечка

my\_func(sz);

//for (int i = 0; i < sz; i++)

//dummy();

t = \_\_rdtsc() - t; // замер продолжительности

time = 1000000000 \* t / freq;

}

\_\_int64 hz\_cpu() { // замер частоты

clock\_t t\_clock;

\_\_int64 t\_tsc;

t\_clock = clock() + CLOCKS\_PER\_SEC;

t\_tsc = \_\_rdtsc(); // взять TSC

while (clock() < t\_clock); // отсчет одной секунды

return (\_\_rdtsc() - t\_tsc); // частота в герцах

}

};

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| count | mtime(мc) | ctime(мc) | i | xi | (xi - xср)^2 | Параметры линейного приближения по методу наименьших квадратов | |
| 54000000 | 1759,356274 | 1584,7 | 889 | 174,7 | 12971,739 | k | b |
| 6000000 | 167,64769 | 175,6 | 0 | 7,9 | 2796,246 | 2,94E-05 | - 0,5746 |
| 24000000 | 670,69788 | 704,0 | 333 | 33,3 | 757,115 | count min | count max |
| 48000000 | 1346,333254 | 1408,5 | 778 | 62,2 | 1,963 | 6000000 | 60000000 |
| 24000000 | 676,551972 | 704,0 | 333 | 27,4 | 1113,545 | (countmax - countmin) /1000 |  |
| 6000000 | 170,493803 | 175,6 | 0 | 5,1 | 3105,349 | 54000 |  |
| 48000000 | 1731,204672 | 1408,5 | 778 | 322,7 | 68581,103 | xср |  |
| 30000000 | 856,146938 | 880,1 | 444 | 24,0 | 1356,212 | 60,795288 |  |
| 42000000 | 1417,777818 | 1232,4 | 667 | 185,4 | 15522,939 | sum (xi - xср)^2 |  |
| 48000000 | 1388,635995 | 1408,5 | 778 | 19,9 | 1672,961 | 9079333 |  |
| 24000000 | 677,46181 | 704,0 | 333 | 26,5 | 1175,095 | S |  |
| 24000000 | 676,607085 | 704,0 | 333 | 27,4 | 1117,226 | 95,33 |  |
| 42000000 | 1242,216778 | 1232,4 | 667 | 9,8 | 2597,936 | xср-S | xср+S |
| 54000000 | 1618,064533 | 1584,7 | 889 | 33,4 | 750,661 | -34,5 | 156,13 |
| 48000000 | 1405,504006 | 1408,5 | 778 | 3,0 | 3337,356 |  |  |
| 24000000 | 669,796277 | 704,0 | 333 | 34,2 | 708,312 |  |  |
| 60000000 | 1898,72276 | 1760,8 | 1000 | 137,9 | 5947,807 |  |  |
| 18000000 | 552,701176 | 527,8 | 222 | 24,9 | 1291,220 |  |  |
| 30000000 | 842,380848 | 880,1 | 444 | 37,7 | 531,796 |  |  |
| … | … | … | … | … | … |  |  |
| 54000000 | 1533,504648 | 1584,7 | 889 | 51,2 | 92,785 |  |  |
| 54000000 | 1529,872804 | 1584,7 | 889 | 54,8 | 36,008 |  |  |
| 24000000 | 685,827105 | 704,0 | 333 | 18,2 | 1818,592 |  |  |
| 12000000 | 337,755512 | 351,7 | 111 | 13,9 | 2194,863 |  |  |
| 30000000 | 869,314005 | 880,1 | 444 | 10,8 | 2499,385 |  |  |
| 24000000 | 684,524432 | 704,0 | 333 | 19,5 | 1709,184 |  |  |
| 48000000 | 1423,450457 | 1408,5 | 778 | 14,9 | 2104,449 |  |  |
| 6000000 | 174,561452 | 175,6 | 0 | 1,0 | 3575,240 |  |  |
| 30000000 | 852,478157 | 880,1 | 444 | 27,6 | 1099,453 |  |  |
| 6000000 | 169,655066 | 175,6 | 0 | 5,9 | 3012,574 |  |  |
| 48000000 | 1361,155295 | 1408,5 | 778 | 47,4 | 180,127 |  |  |
| 12000000 | 335,329247 | 351,7 | 111 | 16,4 | 1973,412 |  |  |
| 30000000 | 873,770734 | 880,1 | 444 | 6,3 | 2964,866 |  |  |
| 6000000 | 166,532366 | 175,6 | 0 | 9,0 | 2679,535 |  |  |
| 48000000 | 1359,168725 | 1408,5 | 778 | 49,4 | 130,749 |  |  |

**Выводы:** Для более точного вычисления реальные значения времени, которые были взяты переведены из нс в мс. Среднее значение отклонения расчетной оценки времени ctime(count) от измеренной mtime(count) составило 60,795288 мс. Оценка среднеквадратического отклонения составила 95,33 мс., что говорит о том, что значения отклонения расчетной оценки времени ctime(count) от измеренной mtime(count) в среднем находятся в диапазоне от -34,5 мс. до 156,13 мс.

**4. Исследование разрешающей способности и точности измерений при использовании таймера HPET.**

Промежуточные и конечные результаты оценки разрешающей способности:

Разрешающая способность оценивается на основе измерения времени исполнения функции dummy в цикле с увеличением числа повторов, пока время не станет больше 0.

Фрагмент кода:

int main()

{

Clock cl;

QPC qpc;

TSC tsc;

setlocale(LC\_CTYPE, "rus");

\_\_int64 sum = 0;

for (int i = 0; i < 1; i++)

{

int sz = 10;

do {

//cl.measSeries(1, sz);

qpc.measSeries(1, sz);

//tsc.measSeries(1, sz);

sz += 10;

} while (qpc.time < 3);

sum += sz;

}

system("pause");

return 0;

}void Clocks::measSeries(int scount, int arcount) {

//cout << endl << name << ": " << arcount << " элементов" << endl

// << "Время (м) \tЧастота (гц)\n";

for (int i = 0; i < scount; i++) {

measure(arcount);

cout << endl << name << ": " << arcount << " элементов" << endl

<< "Время (мкc) \tЧастота (гц)\n";

cout.width(10);

cout.setf(ios::right);

cout << time << "\t" << freq << endl;

}

}

class QPC : public Clocks { // часы на основе QPC

public:

QPC() { QueryPerformanceFrequency((LARGE\_INTEGER \*)&freq); name = "QPC"; }

void measure(int sz) {

LONGLONG t\_start, t\_end;

QueryPerformanceCounter((LARGE\_INTEGER \*)&t\_start); // количеств тактов в секунду в начеле

//my\_func(sz);

for (int i = 0; i < sz; i++)

dummy();

QueryPerformanceCounter((LARGE\_INTEGER \*)&t\_end);// количеств тактов в секунду в конце

time = 1000000 \* (t\_end - t\_start) / freq;

}

};

|  |  |
| --- | --- |
| Количество элементов | Время (мкc) |
| 10 | 0 |
| 20 | 0 |
| 30 | 0 |
| 40 | 1 |
| 50 | 1 |
| 60 | 1 |
| 70 | 1 |
| 80 | 1 |
| 90 | 2 |
| 100 | 1 |
| 110 | 2 |
| 120 | 2 |
| 130 | 3 |

**Среднее значение разрешающей способности: 1 мкс**

Таблицы с результатами оценки линейности:

Точность измеряется на основе my\_func с линейно увеличивающимся объемом вычислительной работы.

1 запуск

Фрагмент кода:

void Clocks::measSeries(int scount, int arcount) {

//cout << endl << name << ": " << arcount << " элементов" << endl

// << "Время (м) \tЧастота (гц)\n";

for (int i = 0; i < scount; i++) {

measure(arcount);

if ((time >= 500) && (time <= 2500)) {

cout.width(15);

cout.setf(ios::right);

cout << arcount << "\t\t\t" << time << endl;

}

}

}

class QPC : public Clocks { // часы на основе QPC

public:

QPC() { QueryPerformanceFrequency((LARGE\_INTEGER \*)&freq); name = "QPC"; }

void measure(int sz) {

LONGLONG t\_start, t\_end;

QueryPerformanceCounter((LARGE\_INTEGER \*)&t\_start); // количеств тактов в секунду в начеле

my\_func(sz);

/\*for (int i = 0; i < sz; i++)

dummy();\*/

QueryPerformanceCounter((LARGE\_INTEGER \*)&t\_end);// количеств тактов в секунду в конце

time = 1000000 \* (t\_end - t\_start) / freq;

}

};

int main()

{

Clock cl;

QPC qpc;

TSC tsc;

setlocale(LC\_CTYPE, "rus");

cout << endl << "Количество элементов " << "Время (мкc) \n";

int sz = 500;

while (qpc.time <= 2500) {

//cl.measSeries(1, sz);

qpc.measSeries(1, sz);

//tsc.measSeries(1, sz);

sz += 1000;

}

system("pause");

return 0;

}

Результаты исполнения функции my\_func:

|  |  |
| --- | --- |
| Количество элементов | Время (мкc) |
| 3000 | 878 |
| 3100 | 967 |
| 3200 | 986 |
| 3300 | 809 |
| 3400 | 837 |
| 3500 | 1098 |
| 3600 | 1052 |
| 3700 | 911 |
| 3800 | 1256 |
| 3900 | 1236 |
| 4000 | 1283 |
| 4100 | 1291 |
| 4200 | 1652 |
| 4300 | 1536 |
| 4400 | 1187 |
| 4500 | 1350 |
| 4600 | 1622 |
| 4700 | 1482 |
| 4800 | 1478 |
| 4900 | 1505 |
| 5000 | 1131 |
| 5100 | 1155 |
| 5200 | 1387 |
| 5300 | 1765 |
| 5400 | 1639 |
| 5500 | 1450 |
| 5600 | 1928 |
| 5700 | 1684 |
| 5800 | 1568 |
| 5900 | 1850 |
| 6000 | 2133 |
| 6100 | 1497 |
| 6200 | 1562 |
| 6300 | 1877 |

Графики зависимости затрат времени T от количества элементов count

1 запуск

2 запуск (анализ повторяемости)

Построенные графики подтверждают линейную зависимость затрат времени T от количества элементов count. Нарушения линейности обнаруживаются при многократном запуске теста, что объясняется тем, что нужных данных в кэше не окажется и процессор вынужден будет простаивать, ожидая их из оперативной памяти, а алгоритмы выборки данных в кэш процессора как раз и настроены на чтение последовательных значений. Когда необходимых данных в кэше процессора не оказывалось и возникает вынужденная задержка в его работе из-за ожидания необходимых значений из оперативной памяти.

Результаты анализа повторяемости в 1000 измерениях

Фрагмент кода:

int main()

{

Clock cl;

QPC qpc;

TSC tsc;

setlocale(LC\_CTYPE, "rus");

\_\_int64 sum = 0;

cout << endl << "Номер " << "Количество элементов " << "Время (мкc) \n";

srand(time(NULL));

for (k; k < 1000; k++)

{

cl.measSeries(1, 600000 \* (rand() % 10 + 1));

//qpc.measSeries(4, sz);

//tsc.measSeries(4, sz);

}

cout << endl;

system("pause");

return 0;

}

void Clocks::measSeries(int scount, int arcount) {

//cout << endl << name << ": " << arcount << " элементов" << endl

// << "Время (м) \tЧастота (гц)\n";

for (int i = 0; i < scount; i++) {

measure(arcount);

cout << k + 1;

cout.width(15);

cout.setf(ios::right);

cout << arcount << "\t\t" << time << endl;

}

}

class QPC : public Clocks { // часы на основе QPC

public:

QPC() { QueryPerformanceFrequency((LARGE\_INTEGER \*)&freq); name = "QPC"; }

void measure(int sz) {

LONGLONG t\_start, t\_end;

QueryPerformanceCounter((LARGE\_INTEGER \*)&t\_start); // количеств тактов в секунду в начеле

my\_func(sz);

/\*for (int i = 0; i < sz; i++)

dummy();\*/

QueryPerformanceCounter((LARGE\_INTEGER \*)&t\_end);// количеств тактов в секунду в конце

time = 1000000 \* (t\_end - t\_start) / freq;

}

};

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | count | mtime(мкc) | ctime(мкc) | i | xi | (xi - xср)^2 | Параметры линейного приближения по методу наименьших квадратов | |
| 1 | 4200000 | 1245 | 1173,3 | 667 | 71,7 | 5144,579 | k | b |
| 2 | 4200000 | 1019 | 1173,3 | 667 | -154,3 | 23800,553 | 0,000 | 3,8767 |
| 3 | 2400000 | 566 | 672,1 | 333 | -106,1 | 11258,038 | count min | count max |
| 4 | 1800000 | 425 | 505,0 | 222 | -80,0 | 6407,540 | 600000 | 6000000 |
| 5 | 600000 | 138 | 170,9 | 0 | -32,9 | 1084,617 | (countmax - countmin) /1000 |  |
| 6 | 4200000 | 971 | 1173,3 | 667 | -202,3 | 40914,883 | 5400 |  |
| 7 | 1800000 | 430 | 505,0 | 222 | -75,0 | 5632,069 | xср |  |
| 8 | 6000000 | 1405 | 1674,4 | 1000 | -269,4 | 72600,421 | 0,000000 |  |
| 9 | 3600000 | 841 | 1006,2 | 556 | -165,2 | 27296,817 | sum (xi - xср)^2 |  |
| 10 | 1800000 | 424 | 505,0 | 222 | -81,0 | 6568,634 | 31580801 |  |
| 11 | 1800000 | 419 | 505,0 | 222 | -86,0 | 7404,105 | S |  |
| 12 | 5400000 | 1261 | 1507,4 | 889 | -246,4 | 60706,978 | 177,80 |  |
| 13 | 2400000 | 562 | 672,1 | 333 | -110,1 | 12122,869 | xср-S | xср+S |
| 14 | 3000000 | 698 | 839,2 | 444 | -141,2 | 19926,341 | -177,8 | 177,80 |
| 15 | 1800000 | 416 | 505,0 | 222 | -89,0 | 7929,388 |  |  |
| 16 | 5400000 | 1264 | 1507,4 | 889 | -243,4 | 59237,651 |  |  |
| 17 | 6000000 | 1393 | 1674,4 | 1000 | -281,4 | 79211,093 |  |  |
| 18 | 1200000 | 285 | 338,0 | 111 | -53,0 | 2807,974 |  |  |
| 19 | 2400000 | 549 | 672,1 | 333 | -123,1 | 15154,570 |  |  |
| 20 | 2400000 | 579 | 672,1 | 333 | -93,1 | 8668,336 |  |  |
| … | … | … | … | … | … | … |  |  |
| 985 | 3000000 | 779 | 839,2 | 444 | -60,2 | 3619,309 |  |  |
| 986 | 3600000 | 903 | 1006,2 | 556 | -103,2 | 10653,849 |  |  |
| 987 | 3600000 | 900 | 1006,2 | 556 | -106,2 | 11282,154 |  |  |
| 988 | 2400000 | 814 | 672,1 | 333 | 141,9 | 20134,503 |  |  |
| 989 | 1800000 | 532 | 505,0 | 222 | 27,0 | 726,458 |  |  |
| 990 | 3600000 | 963 | 1006,2 | 556 | -43,2 | 1867,751 |  |  |
| 991 | 4800000 | 1311 | 1340,3 | 778 | -29,3 | 860,312 |  |  |
| 992 | 6000000 | 1498 | 1674,4 | 1000 | -176,4 | 31132,715 |  |  |
| 993 | 3600000 | 896 | 1006,2 | 556 | -110,2 | 12147,894 |  |  |
| 994 | 4200000 | 1054 | 1173,3 | 667 | -119,3 | 14226,353 |  |  |
| 995 | 1200000 | 311 | 338,0 | 111 | -27,0 | 728,477 |  |  |
| 996 | 1800000 | 489 | 505,0 | 222 | -16,0 | 257,510 |  |  |
| 997 | 6000000 | 1511 | 1674,4 | 1000 | -163,4 | 26714,155 |  |  |
| 998 | 2400000 | 583 | 672,1 | 333 | -89,1 | 7939,505 |  |  |
| 999 | 5400000 | 1379 | 1507,4 | 889 | -128,4 | 16483,443 |  |  |

**Выводы:** Среднее значение отклонения расчетной оценкой времени ctime(count) от измеренной mtime(count) составило 0 мкс. Оценка среднеквадратического отклонения составила 177,80 мкс., что говорит о том, что значения отклонения расчетной оценкой времени ctime(count) от измеренной mtime(count) в среднем находятся в диапазоне от -177,8 мкс. до -177,8 мкс.

Параметры машины, в среде которой проводились измерения, приведены в **Приложение 1**.

Базовые сведения о средствах измерения времени, приведены в **Приложение 2**.

**Приложение 1.**

Сводная таблица характеристик, тестируемых ЭВМ;

|  |  |  |
| --- | --- | --- |
| Наименование характеристик | Компьютер 1 | Компьютер 2 |
| Тип процессора | DualCore Intel Core i5-7200U | Intel Core i7 - 6500 U |
| Исходная частота | 2500 МГц | 2500 ГГц |
| Максимальная частота | 3100 MHz | 3700 MHz |
| Архитектура процессора | Kaby Lake-U | - |
| Cash-памяти | Кэш L1 — 2x32+2x32 Кб per core  Кэш L2 — 256x2 Кб per core (On-Die, ECC, Full-Speed)  Кэш L3 — 3 Мб (On-Die, ECC, Full-Speed) | Кэш L1 — 256x2 Кб per core (On-Die, ECC, Full-Speed)  - |
| Тип сокета | BGA1356 | - |
| Напряжение питания процессора | 0.6 V | - |
| Мощность процессора | 15 W | - |
| Количество ядер/потоков | 2/4 | 2/4 |
| Производитель | Asus | - |
| Материнская плата | Asus VivoBook X556UQK | - |
| Наличие и тип слотов PCI | PCI  PCI Express 2.0 | - |
| Тип видеокарты (встроенная, дополнительная) | Intel HD Graphics 620  NVIDIA GeForce 940MX | Intel HD Graphics 620  NVIDIA GeForce GTX 1050TI |
| Тип шины | Встроено  PCI-устройство 8086-5916 / 1043-1490 (Rev 02) | - |
| Объем памяти видеокарты | 1 Гб  2 Гб | 4Гб |
| Физическая память ЭВМ | HGST HTS721010A9E630 (1 ТБ, 7200 RPM, SATA-III) | 1 ТБ |
| Оперативная память (объем, тип и скорость) | 8Гб SO-DIMM DDR4 15-15-15-35 4-50-17-8 2T  DDR3 SO-DIMM DDR4  2x 1ГГц (2.13ГГц) | 8Гб SO-DIMM DDR4 |
| Драйвер видеокарты | 23.20.16.4905 | - |
| Интерфейсы ввода-вывода | USB 2.0, USB 3.0, USB type C, HDMI, microSD, Audio 3.5, LAN, VGA | USB 2.0, USB 3.0, USB type C, HDMI, microSD, Audio 3.5, LAN, VGA |
| ОС | Windows 10 | Windows 10 |

**Приложение 2. Базовые сведения о средствах измерения времени.**

**Функция** **clock()** описывается в time.h и возвращает число тиков от момента загрузки программы. Тик обычно равен 1 миллисекунде, но для возможности в последующем работать с другой длительностью такта в time.h фиксируется константа CLOCKS\_PER\_SEC.

**Высокоточный таймер HPET** (High Precision Timer ) обычно имеет разрешение в 100 наносекунд. В С/С++ он используется через следующие функции, описанные в windows.h:

QueryPerformanceCounter(&tick) // прочитать номер тика таймера

QueryPerformanceFrequency(&freq) // получить частоту таймера

Это единственный таймер, который активируется через Windows. Для его отключения используйтся команда: bcdedit /deletevalue useplatformclock

Если таймер отключен, то Windows использует таймеры, встроенные непосредственно в чипсет материнской платы.

**Счетчик тактов процессора TSC** (Time Stamp Counter) встроен в ядро CPU и обычно прирастает на 1 с каждым тактом задающего генератора (это тот, что определяет гигагерцы процессора). В С/С++ он используется через функцию \_\_int64 \_\_ rdtsc(), описанную в windows.h.