

Федеральное государственное бюджетное образовательное учреждение высшего образования  
«Сибирский государственный университет телекоммуникаций и информатики»  
(СибГУТИ)

09.03.01 "Информатика и вычислительная техника"  
профиль "Электронно-вычислительные машины,  
комплексы, системы и сети"

## **ЛАБОРАТОРНАЯ РАБОТА №2**

**по дисциплине «Архитектура вычислительных систем»**

**Тема: «Оценка производительности процессора»**

Выполнил:

студент гр.ИВ-021

\_\_\_\_\_/Савин А.Б. /

«17» сентября 2022г.

Принял:

Ассистент каф. ВС

\_\_\_\_\_/Романюта А.А./

Оценка \_\_\_\_\_

Новосибирск, 2022

## Содержание

1. Постановка задачи.....	3
2. Программная реализация.....	4
3. Результат работы.....	5
4. Приложение. Листинг.....	6

## Постановка задачи

Реализовать программу для оценки производительности процессора (benchmark).

1. Написать программу(ы) (benchmark) на языке C/C++/C# для оценки производительности процессора. В качестве набора типовых задач использовать либо минимум 3 функции выполняющих математические вычисления, либо одну функцию по работе с матрицами и векторами данных с несколькими типами данных. Можно использовать готовые функции из математической библиотеки (math.h) [3], библиотеки BLAS [4] (англ. Basic Linear Algebra Subprograms — базовые подпрограммы линейной алгебры) и/или библиотеки LAPACK [5] (Linear Algebra PACKage). Обеспечить возможность в качестве аргумента при вызове программы указать общее число испытаний для каждой типовой задачи (минимум 10). Входные данные для типовой задачи сгенерировать случайным образом.

2. С помощью системного таймера (библиотека time.h, функции clock() или gettimeofday()) или с помощью процессорного регистра счетчика TSC реализовать оценку в секундах среднего времени испытания каждой типовой задачи. Оценить точность и погрешность (абсолютную и относительную) измерения времени (рассчитать дисперсию и среднеквадратическое отклонение).

3. Результаты испытаний в самой программе (или с помощью скрипта) сохранить в файл в формате CSV со следующей структурой: [PModel;Task;OpType;Opt;InsCount;Timer;Time;LNum;AvTime;AbsErr;RelErr;TaskPerf], где

PModel – Processor Model, модель процессора, на котором проводятся испытания;

Task – название выбранной типовой задачи (например, sin, log, saxpy, dgemv, sgemm и др.);

OpType – Operand Type, тип операндов используемых при вычислениях типовой задачи;

Opt – Optimisations, используемы ключи оптимизации (None, O1, O2 и др.);

InsCount – Instruction Count, оценка числа инструкций при выполнении типовой задачи;

Timer – название функции обращения к таймеру (для измерения времени);

Time – время выполнения отдельного испытания;

LNum – Launch Numer, номер испытания типовой задачи.

AvTime – Average Time, среднее время выполнения типовой задачи из всех испытаний[секунды];

AbsError – Absolute Error, абсолютная погрешность измерения времени в секундах;

RelError – Relative Error, относительная погрешность измерения времени в %;

TaskPerf – Task Performance, производительность (быстродействие) процессора при выполнении типовой задачи.

## Программная реализация

В качестве языка программирования был использован C. В роли типовой задачи была выбрана перемножение матрицы на вектор в трех версиях для Integer, Float, Double.

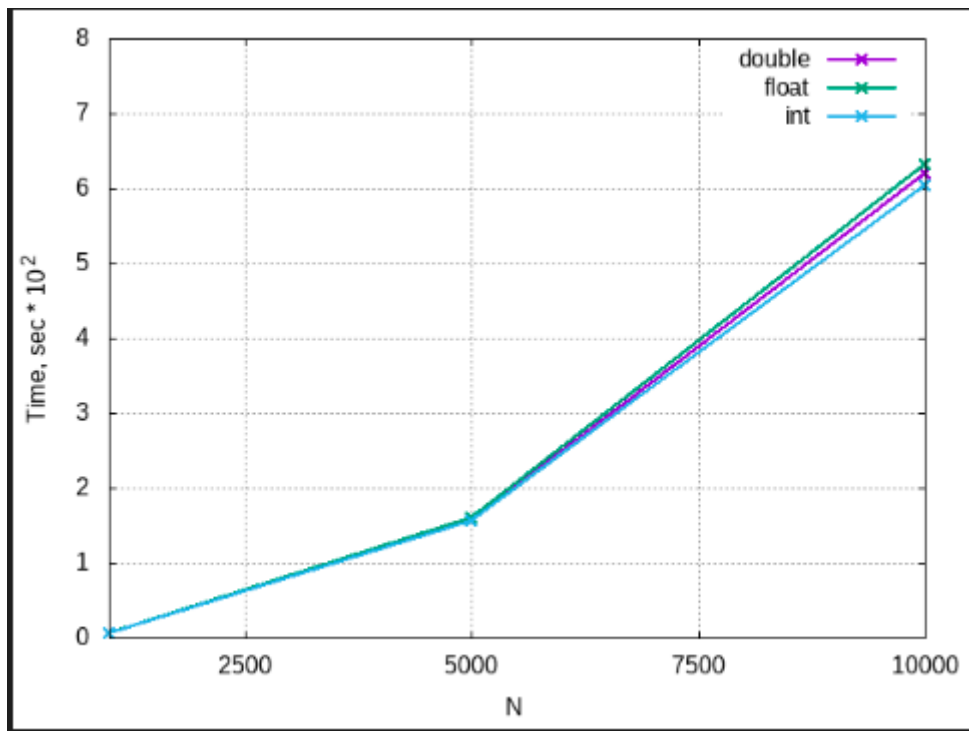
Для получения сведений о процессоре был использован файл /proc/cpuinfo. Затем данные были сохранены в csv файл. Для каждой версии количество проходов равняется 10 ( в сумме 30 ).

# Результат работы

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	double	O	16000000	gettimeofday	0.114110	1.0.114110	0.000000	0.000000 %		140215590.081318	
2	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	double	O	16000000	gettimeofday	0.232497	2.0.116249	0.002139	1.839691 %		137636057.114496	
3	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	double	O	16000000	gettimeofday	0.345050	3.0.115017	0.003371	2.930464 %		139110234.963672	
4	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	double	O	16000000	gettimeofday	0.487389	4.0.121847	0.020492	16.817545 %		131311926.381112	
5	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	double	O	16000000	gettimeofday	0.597819	5.0.119564	0.022775	19.048550 %		133819748.069242	
6	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	double	O	16000000	gettimeofday	0.709674	6.0.118279	0.024060	20.341707 %		135273356.180514	
7	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	double	O	16000000	gettimeofday	0.812019	7.0.116003	0.026336	22.703139 %		137927788.470149	
8	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	double	O	16000000	gettimeofday	0.920943	8.0.115118	0.027221	23.646295 %		138987968.825036	
9	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	double	O	16000000	gettimeofday	1.028634	9.0.114293	0.028046	24.539061 %		139991506.588513	
10	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	double	O	16000000	gettimeofday	1.135006	10.0.113501	0.028838	25.408145 %		140968423.882905	
11	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	float	O	16000000	gettimeofday	0.105264	1.0.105264	0.000000	0.000000 %		151998858.460416	
12	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	float	O	16000000	gettimeofday	0.219182	2.0.109591	0.004327	3.948370 %		145997380.661274	
13	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	float	O	16000000	gettimeofday	0.325047	3.0.108349	0.005569	5.139928 %		147670944.866527	
14	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	float	O	16000000	gettimeofday	0.429042	4.0.107261	0.006658	6.206888 %		149169510.089562	
15	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	float	O	16000000	gettimeofday	0.530024	5.0.106005	0.007913	7.464997 %		150936546.650101	
16	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	float	O	16000000	gettimeofday	0.630558	6.0.105093	0.008825	8.397385 %		152246102.470099	
17	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	float	O	16000000	gettimeofday	0.736887	7.0.105270	0.008648	8.215573 %		151990743.928291	
18	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	float	O	16000000	gettimeofday	0.854304	8.0.106788	0.010629	9.953447 %		149829555.669482	
19	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	float	O	16000000	gettimeofday	0.965934	9.0.107326	0.010091	9.402281 %		149078502.579712	
20	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	float	O	16000000	gettimeofday	1.077515	10.0.107751	0.009666	8.970278 %		148489827.798890	
21	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	integer	O	16000000	gettimeofday	0.113349	1.0.113349	0.000000	0.000000 %		1411157006.436414	
22	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	integer	O	16000000	gettimeofday	0.238246	2.0.119123	0.005774	4.847109 %		134314972.039757	
23	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	integer	O	16000000	gettimeofday	0.349562	3.0.116521	0.008376	7.188735 %		137314724.291674	
24	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	integer	O	16000000	gettimeofday	0.457725	4.0.114431	0.010466	9.145933 %		139822003.638851	
25	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	integer	O	16000000	gettimeofday	0.579054	5.0.115611	0.009286	8.032351 %		138395443.576144	
26	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	integer	O	16000000	gettimeofday	0.682716	6.0.113786	0.011111	9.764880 %		140614909.567500	
27	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	integer	O	16000000	gettimeofday	0.789642	7.0.112806	0.012091	10.718458 %		141836495.624845	
28	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	integer	O	16000000	gettimeofday	0.903476	8.0.112935	0.011962	10.592368 %		141674966.980740	
29	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz	multiplication of a matrix by a vector	integer	O	16000000	gettimeofday	1.012847	9.0.112539	0.012358	10.981477 %		142173437.628731	

Рисунок 1 — Пример работы программы (сверху)(запись в файл)

Рисунок 2 — График (снизу)



## Приложение. Листинг.

```
1  #include <math.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <sys/time.h>
6  #include <time.h>
7
8  #define INT_MAX 2147483647
9
10 void matrix_vector_product_double(double* a, double* b, double* c, int m, int n)
11 {
12     for (int i = 0; i < m; i++) {
13         {
14             c[i] = 0.0;
15         }
16         for (int j = 0; j < n; j++) {
17             c[i] += a[i * n + j] * b[j];
18         }
19     }
20 }
21
22 void matrix_vector_product_float(float* a, float* b, float* c, int m, int n)
23 {
24     for (int i = 0; i < m; i++) {
25         {
26             c[i] = 0.0;
27         }
28         for (int j = 0; j < n; j++) {
29             c[i] += a[i * n + j] * b[j];
30         }
31     }
32 }
33
34 void matrix_vector_product_int(
35     int* a, int* b, int* c, int m, int n) // m + (m*(n-1))
36 {
37     for (int i = 0; i < m; i++) {
38         {
39             c[i] = 0.0;
40         }
41         for (int j = 0; j < n; j++) {
42             c[i] += a[i * n + j] * b[j];
43         }
44     }
45 }
46
47 double wtime() // saod
48 {
49     struct timeval t;
50     gettimeofday(&t, NULL);
51     return (double)t.tv_sec + (double)t.tv_usec * 1E-6;
52 }
53
54 double getrand(int min, int max) // [min;max)
55 {
56     return (double)rand() / (RAND_MAX + 1.0) * (max - min) + min;
57 }
58
59 void init_matrix_d(double* a, double* b, int m, int n, int min, int max)
```

```

60 {
61     for (int i = 0; i < m; i++) {
62         for (int j = 0; j < n; j++) {
63             a[i * n + j] = getrand(min, max);
64         }
65     }
66     for (int j = 0; j < n; j++) {
67         b[j] = getrand(min, max);
68     }
69 }
70
71 void init_matrix_f(float* a, float* b, int m, int n, int min, int max)
72 {
73     for (int i = 0; i < m; i++) {
74         for (int j = 0; j < n; j++) {
75             a[i * n + j] = getrand(min, max);
76         }
77     }
78     for (int j = 0; j < n; j++) {
79         b[j] = getrand(min, max);
80     }
81 }
82
83 void init_matrix_i(int* a, int* b, int m, int n, int min, int max)
84 {
85     for (int i = 0; i < m; i++) {
86         for (int j = 0; j < n; j++) {
87             a[i * n + j] = getrand(min, max);
88         }
89     }
90     for (int j = 0; j < n; j++) {
91         b[j] = getrand(min, max);
92     }
93 }
94
95 char* readln(FILE* stream)
96 {
97     static char* str = NULL;
98     static size_t i = 0;
99     int ch = fgetc(stream);
100
101     if ((ch == '\n') || (ch == EOF)) {
102         str = malloc(i + 1);
103         str[i] = 0;
104     } else {
105         i++;
106         readln(stream);
107         str[--i] = ch;
108     }
109     return str;
110 }
111
112 void output_result(
113     FILE* out,
114     char* optype,
115     double i_time,
116     int test_cnt,
117     int m,
118     int n,
119     double max_time,
120     double min_time)
121 {
122     int insCount = m + (m * (n - 1));

```



```

123     double avg_time = i_time / test_cnt;
124     double performance = pow((1 / (insCount / avg_time)), -1);
125     double abs_error, rel_error, delta_max, delta_min;
126     delta_max = fabs(max_time - avg_time);
127     delta_min = fabs(min_time - avg_time);
128     if (delta_max >= delta_min) {
129         abs_error = delta_max;
130     } else {
131         abs_error = delta_min;
132     }
133     rel_error = (abs_error / avg_time) * 100;
134
135     char* str2 = (char*)malloc(256);
136     FILE* cpu_inf = fopen("/proc/cpuinfo", "r");
137     fgets(str2, 255, cpu_inf);
138
139     for (int i = 0; i < 4; i++) {
140         str2 = readln(cpu_inf);
141         if (i == 3) {
142             str2 = str2 + 13;
143         }
144     }
145
146     // fprintf (out, "PModel - %s\n", pipe_fp);
147     /* for(int i = 0; i < 5 ; i++){ */
148     fprintf(out, "%s;", str2);
149     fprintf(out, "multiplication of a matrix by a vector;");
150     fprintf(out, "%s;", optype);
151     fprintf(out, "O;");
152     fprintf(out, "%d;", insCount);
153     fprintf(out, "gettimeofday;");
154     fprintf(out, "%lf;", i_time);
155     fprintf(out, "%d;", test_cnt);
156     fprintf(out, "%lf;", avg_time);
157     fprintf(out, "%lf;", abs_error);
158     fprintf(out, "%lf %c;", rel_error, 37);
159     fprintf(out, "%lf;\n", performance);
160 }
161
162 int main()
163 {
164     srand(time(NULL));
165     FILE* out;
166     out = fopen("result.csv", "w");
167     int n, m, min, max, test_cnt;
168     min = 0;
169     max = 64;
170     n = m = 4000;
171     double *a, *b, *c, first_time, second_time, res_time, min_time, max_time,
172           cur_time;
173     a = (double*)malloc(sizeof(*a) * m * n);
174     b = (double*)malloc(sizeof(*b) * n);
175     c = (double*)malloc(sizeof(*c) * m);
176
177     printf("Enter number of tests\n");
178     scanf("%d", &test_cnt);
179     if (test_cnt < 10) {
180         return 0;
181     }
182
183     for (int i = 0; i < 3; i++) {
184         min_time = INT_MAX;
185         max_time = 0;

```

```

186     if (i == 0) {
187         res_time = 0;
188         for (int j = 1; j <= test_cnt; j++) {
189             init_matrix_d(a, b, m, n, min, max);
190             first_time = wtime();
191             matrix_vector_product_double(a, b, c, m, n);
192             second_time = wtime();
193             cur_time = second_time - first_time;
194             res_time += cur_time;
195             if (cur_time > max_time)
196                 max_time = cur_time;
197             if (cur_time < min_time)
198                 min_time = cur_time;
199             output_result(
200                 out, "double", res_time, j, m, n, max_time, min_time);
201         }
202     }
203     if (i == 1) {
204         res_time = 0;
205         for (int j = 1; j <= test_cnt; j++) {
206             init_matrix_f((float*)a, (float*)b, m, n, min, max);
207             first_time = wtime();
208             matrix_vector_product_float(
209                 (float*)a, (float*)b, (float*)c, m, n);
210             second_time = wtime();
211             cur_time = second_time - first_time;
212             res_time += cur_time;
213             if (cur_time > max_time)
214                 max_time = cur_time;
215             if (cur_time < min_time)
216                 min_time = cur_time;
217             output_result(
218                 out, "float", res_time, j, m, n, max_time, min_time);
219         }
220     }
221     if (i == 2) {
222         res_time = 0;
223         for (int j = 1; j <= test_cnt; j++) {
224             init_matrix_i((int*)a, (int*)b, m, n, min, max);
225             first_time = wtime();
226             matrix_vector_product_int((int*)a, (int*)b, (int*)c, m, n);
227             second_time = wtime();
228             cur_time = second_time - first_time;
229             res_time += cur_time;
230             if (cur_time > max_time)
231                 max_time = cur_time;
232             if (cur_time < min_time)
233                 min_time = cur_time;
234             output_result(
235                 out, "integer", res_time, j, m, n, max_time, min_time);
236         }
237     }
238
239     printf("%f\n", res_time);
240 }
241 free(a);
242 free(b);
243 free(c);
244 return 0;
245 }

```

