Лабораторная работа №1

Студента группы ПА-19-2

Ильяшенко Егора

**Постановка задачи:** На мові С/C++ написати програму яка буде перемножувати квадратну матрицю розміром NxN на вектор розмірність N. Величину N вибрати таким чином, щоб максимально використати всю можливу оперативну пам’ять. Обчислити затрати часу на дану дію. Перемножити також вектор на матрицю. Порівняти затрачений час в обох випадках. Зробити висновки. Запропонувати зміну алгоритму таким чином, щоб затрачений час був приблизно однаковий.

Описание:

Программа выполняет 3 перемножения.

1. Матрица на вектор
2. Вектор на матрицу
3. Вектор на матрицу оптимизированным методом

Оптимизированный метод заключается в последовательном доступе к элементам в памяти.

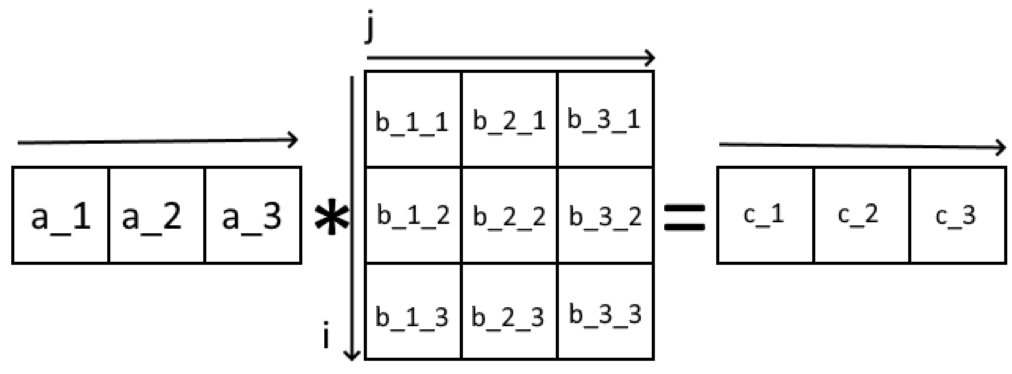
Поскольку после конца первой строки начинаются элементы следующей. Такой доступ осуществляется значительно быстрее.

Вектор на матрицу

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

for (int j = 0; j < g; j++)

c[i] += b[j] \* a[j][i];



Вектор на матрицу(Оптимизированное)

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

for (int j = 0; j < g; j++)

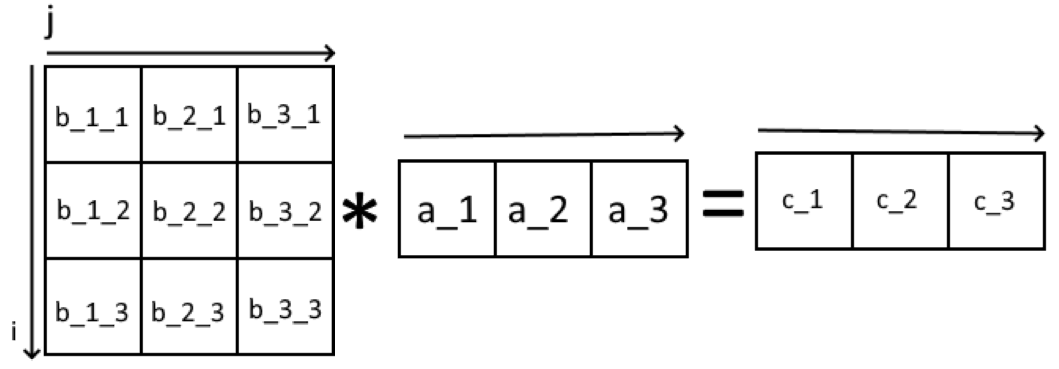
c[j] += b[i] \* a[i][j];

Матрица на вектор

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

for (int j = 0; j < g; j++)

c[i] += a[i][j] \* b[j];



M - Матрица

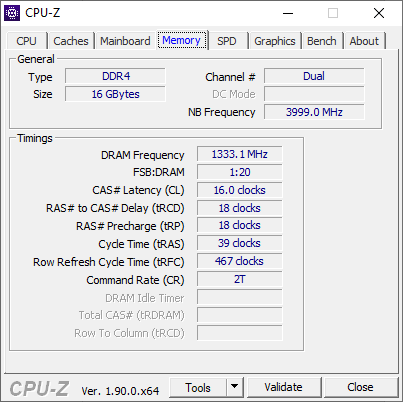
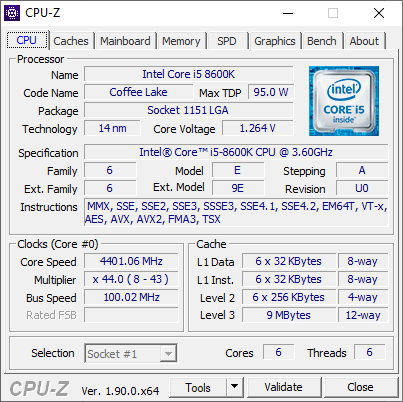
V – Вектор

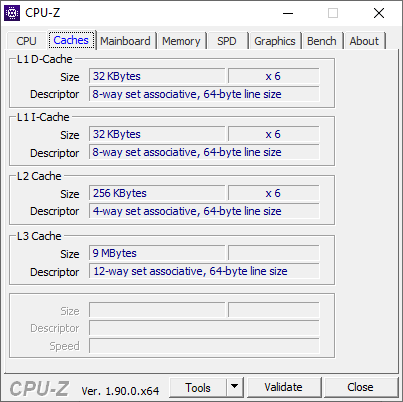
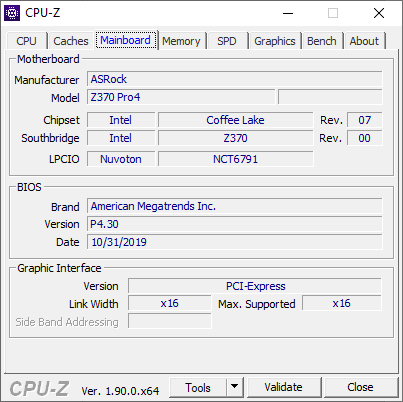
OPT – Оптимизация

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| INT | M\*V | V\*M | V\*M(OPT) | FLOAT | M\*V | V\*M | V\*M(OPT) | DOUBLE | M\*V | V\*M | V\*M(OPT) | M size |
|  | 0,003 | 0,003 | 0,002 |  | 0,003 | 0,004 | 0,003 |  | 0,002 | 0,004 | 0,003 | 1000 |
|  | 0,008 | 0,026 | 0,007 |  | 0,01 | 0,026 | 0,01 |  | 0,014 | 0,034 | 0,009 | 2000 |
|  | 0,018 | 0,067 | 0,018 |  | 0,022 | 0,067 | 0,02 |  | 0,024 | 0,085 | 0,021 | 3000 |
|  | 0,031 | 0,119 | 0,03 |  | 0,041 | 0,122 | 0,037 |  | 0,043 | 0,142 | 0,037 | 4000 |
|  | 0,05 | 0,211 | 0,048 |  | 0,063 | 0,189 | 0,057 |  | 0,065 | 0,221 | 0,057 | 5000 |
|  | 0,071 | 0,274 | 0,068 |  | 0,091 | 0,273 | 0,081 |  | 0,095 | 0,326 | 0,084 | 6000 |
|  | 0,097 | 0,376 | 0,094 |  | 0,125 | 0,384 | 0,112 |  | 0,124 | 0,441 | 0,113 | 7000 |
|  | 0,126 | 0,497 | 0,121 |  | 0,16 | 0,504 | 0,147 |  | 0,163 | 0,606 | 0,149 | 8000 |
|  | 0,159 | 0,626 | 0,154 |  | 0,208 | 0,683 | 0,182 |  | 0,21 | 0,803 | 0,185 | 9000 |
|  | 0,198 | 0,782 | 0,189 |  | 0,255 | 0,807 | 0,228 |  | 0,258 | 0,989 | 0,229 | 10000 |
|  | 0,238 | 0,958 | 0,229 |  | 0,306 | 0,954 | 0,271 |  | 0,312 | 1,19 | 0,283 | 11000 |
|  | 0,283 | 1,243 | 0,272 |  | 0,363 | 1,17 | 0,325 |  | 0,372 | 1,394 | 0,332 | 12000 |
|  | 0,333 | 1,511 | 0,327 |  | 0,423 | 1,43 | 0,382 |  | 0,435 | 1,646 | 0,387 | 13000 |
|  | 0,391 | 1,91 | 0,373 |  | 0,19 | 1,688 | 0,445 |  | 0,51 | 1,974 | 0,45 | 14000 |
|  | 0,453 | 2,074 | 0,432 |  | 0,583 | 2,041 | 0,509 |  | 0,583 | 2,454 | 0,524 | 15000 |
|  | 0,525 | 2,366 | 0,485 |  | 0,641 | 2,255 | 0,573 |  |  |  |  | 16000 |
|  | 0,577 | 3,09 | 0,567 |  | 0,727 | 2,655 | 0,65 |  |  |  |  | 17000 |
|  | 0,64 | 3,035 | 0,616 |  | 0,827 | 3,77 | 0,742 |  |  |  |  | 18000 |
|  | 0,722 | 3,519 | 0,692 |  | 0,914 | 3,627 | 0,811 |  |  |  |  | 19000 |
|  | 0,792 | 4,156 | 0,763 |  | 1,007 | 3,995 | 0,897 |  |  |  |  | 20000 |
|  | 0,895 | 4,974 | 0,847 |  | 1,107 | 4,58 | 1,011 |  |  |  |  | 21000 |
|  | 0,962 | 5,139 | 0,926 |  | 1,23 | 5,037 | 1,093 |  |  |  |  | 22000 |

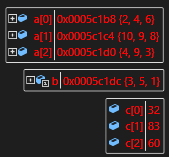
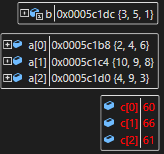
INT

FLOAT





Matrix\*Vector Vector\*Matrix

Вывод: Максимальное значение памяти, которое удалось выделить – 2gb RAM.

При этом максимальный размер матрицы и векторов типа int и float составил – [22000]

Максимальный размер матрицы и векторов типа double составил – [16000]

Время вычислений возрастает параболически.

Время доступа к элементам один за другим намного выше, чем произвольный доступ.