Московский физико-технический институт (государственный университет)
Факультет радиотехники и вычислительной техники Кафедра информатики и вычислительной техники

Выпускная квалификационная работа магистра

Оптимизации алгоритмов умножения матрицы на скаляр, вектор и матрицу с сохранением точности для архитектуры Эльбрус-2С+

Выполнил: Мустафин Т.Р. 913 группа

Научный руководитель: к.т.н. Логинов В.Е.

Введение

Тестовая утилита Coremark состоит из алгоритмов:

- работа со списками, поиск данных через указатели;
- сложение, умножение матриц;
- работа конечного автомата, операции сравнения, перехода;
- вычисление циклического избыточного кода (cycling redundancy check).

Используются умножения матрицы (int_16s) на:

- скаляр (int_16s), результат – матрица (int_32s);
- вектор (int_16s), результат – вектор (int_32s);
- матрицу (int_16s), результат – матрица (int_32s).

Цель работы

Адаптация существующих архитектурно-оптимизированных функций высокопроизводительной библиотеки Elbrus Mathematical Library для использования в Coremark на архитектуре Эльбрус-2С+.

Структура широкой команды Эльбрус-2С+

- **Слоги арифметико-логических каналов широкой команды** **до 6**

0 I, M0	1 I, M1	2 I, S	3 I, M0	4 I, M1	5 I, S

I – integer, арифметические и логические операции над целыми числами
S – store, сохранение значений в память
M0, M1 – арифметические и логические операции над упакованными числами

- **Слоги каналов обращения к массивам** **до 6**

Кодируют до 4 операций пересылки элементов массивов из буфера предварительной подкачки массивов (APB) в регистровый файл.

- **Слоги управления CS** **до 2**

Кодируют операции передачи управления.

- **Слог коротких операций SS** **1**

Кодирует различные короткие фрагменты команды. В частности, содержит поля: bap/eap – начать/закончить предварительную подкачку массива, ctcond - условие операции передачи управления.

Умножение матрицы на скаляр

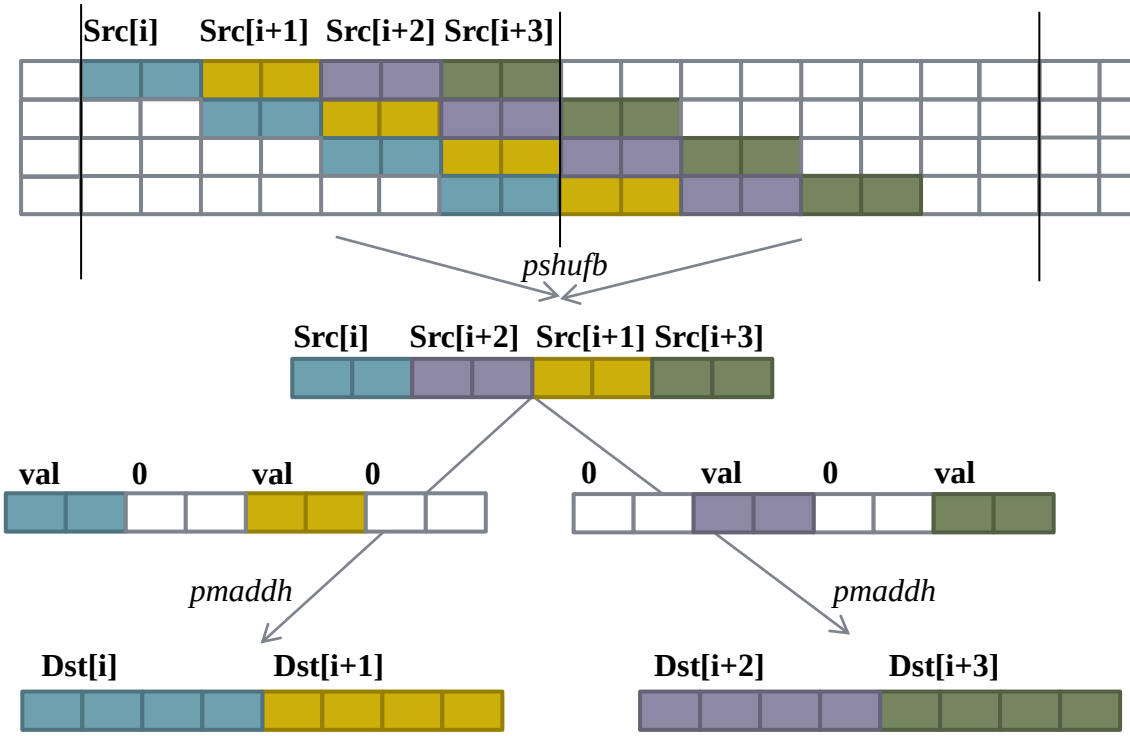
Умножение матрицы на скаляр с точки зрения функциональности можно свести к более общему умножению вектора на скаляр, если строки матрицы хранятся подряд, в ином случае к каждой строке матрицы можно применить умножение вектора на скаляр.

За основу для реализации взята функция умножения вектора (float_32) на число (float_32) с вектором (float_32) в качестве результата.
Особенности:
+ Вещественное умножение => нет операций сдвига;
+ После умножения 32 разряда => формат записи, выравнивание идентичны;
- До умножения 32 разряда => формат чтения, выравнивание различаются.

Задачи:
1. Подобрать состав упакованных операций (int_16s)*(int_16s) → (int_32s).
2. Решить проблему выравнивания при чтении исходного вектора из памяти.

Умножение матрицы на скаляр

Решения поставленных задач



Описание схемы:

- Pshufb решает проблему выравнивания и меняет местами средние элементы
- Две pmaddh умножают четыре элемента
- Две std сохраняют результаты умножения

- За один такт умножаются 4 элемента входного вектора.
  В случае невыровненных векторов скорость остается 4 элемента/такт.
- Загрузка элементов исходного вектора кодируется в каналах обращения к массивам.
  Условие завершения цикла кодируется в слоге коротких операций.
  Операция выхода из цикла кодируется в слоге управления.

Умножение матрицы на скаляр

Стандартный и оптимизированный циклы умножения вектора на скаляр

```
#pragma unroll(1)
for(i = 0; i < len; i++) {
        pDst[i] = (eml_32s)pSrc[i] * val;
    }
```

```
M_5240:loop_mode
ct %ctpr1 ? #NOT_LOOP_END
ipd 3
abn abnf=1, abnt=1
abp abpf=1, abpt=1
alc alcf=1, alct=1
muls,0,sm %b[13], %r5, %b[1]
addd,1,sm 0x4, %dr0, %dr0 ? %pcnt0
stw,2 %b[9], [ %dr2 + %dr0 ]
getfs,5,sm %b[10], %r1, %b[9]

movah,1 area = 0, ind = 0, am = 1, be = 0, %db[0]

rlp,cd00 %pcnt0, >alc1
```

С раскруткой на 7 стандартный цикл планируется в 4 такта, теоретическая скорость – 1,75 элемента/такт.

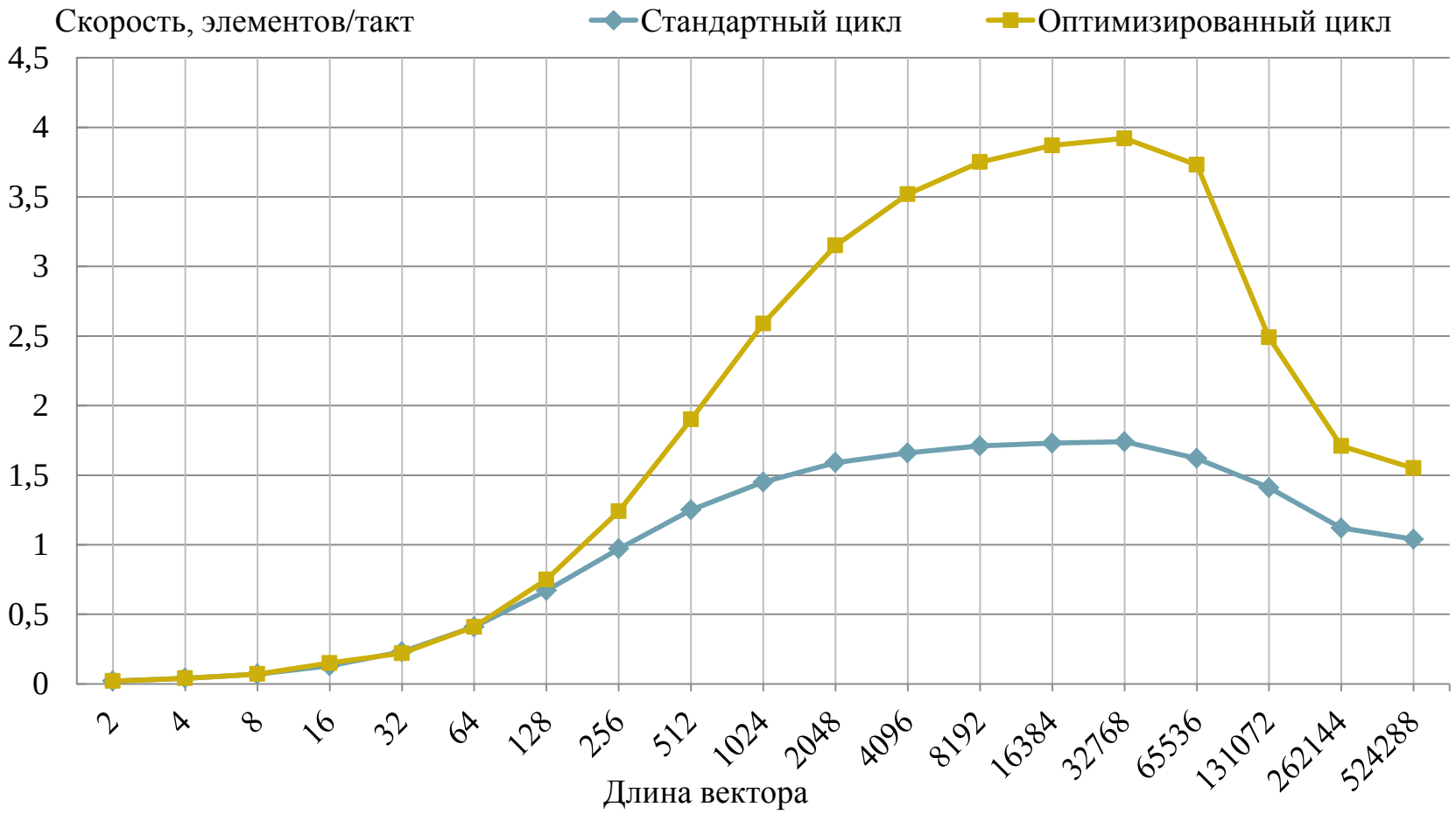
```
#pragma unroll(1)
for (i = 0; i < (len >> 2); i++) {
      tmp = e3m_pshufb (sp1[i], sp1[i + 1], align1);
      dp[2 * i]     = e3m_pmaddh  (tmp,mask0);
      dp[2 * i + 1] = e3m_pmaddh  (tmp,mask1);
   }
```

```
M_5480:loop_mode
ct %ctpr1 ? #NOT_LOOP_END
ipd 3
abn abnf=1, abnt=1
abp abpf=1, abpt=1
alc alcf=1, alct=1
pshufb,0,sm %db[13], %db[11], %dr1, %db[13]
pmaddh,1,sm %db[21], %dr5, %db[21]
std,2 %db[25], [ %dr0 + %db[24] ]
addd,3,sm 0x10, %db[22], %db[20] ? %pcnt1
pmaddh,4,sm %db[21], %dr3, %db[24]
std,5 %db[28], [ %dr6 + %db[24] ]

movad,1 area = 0, ind = 0, am = 1, be = 0, %db[1]
rlp,cd00 %pcnt1, >alc3
```

Умножение матрицы на скаляр

Сравнение циклов умножения вектора на скаляр



Теоретическая скорость стандартного цикла – 1,75 элемента/такт, оптимизированного – 4 элемента/такт

Умножение матрицы на вектор

За основу для реализации взята функция умножения матрицы A (float_32) на вектор X (float_32) с вектором Y (float_32) в качестве результата.
Для умножения используются вложенные циклы. Внешний цикл выбирает строку матрицы A, внутренний цикл выбирает элемент строки, умножает на соответствующий элемент вектора X и прибавляет результат к элементу из Y.

Задачи:

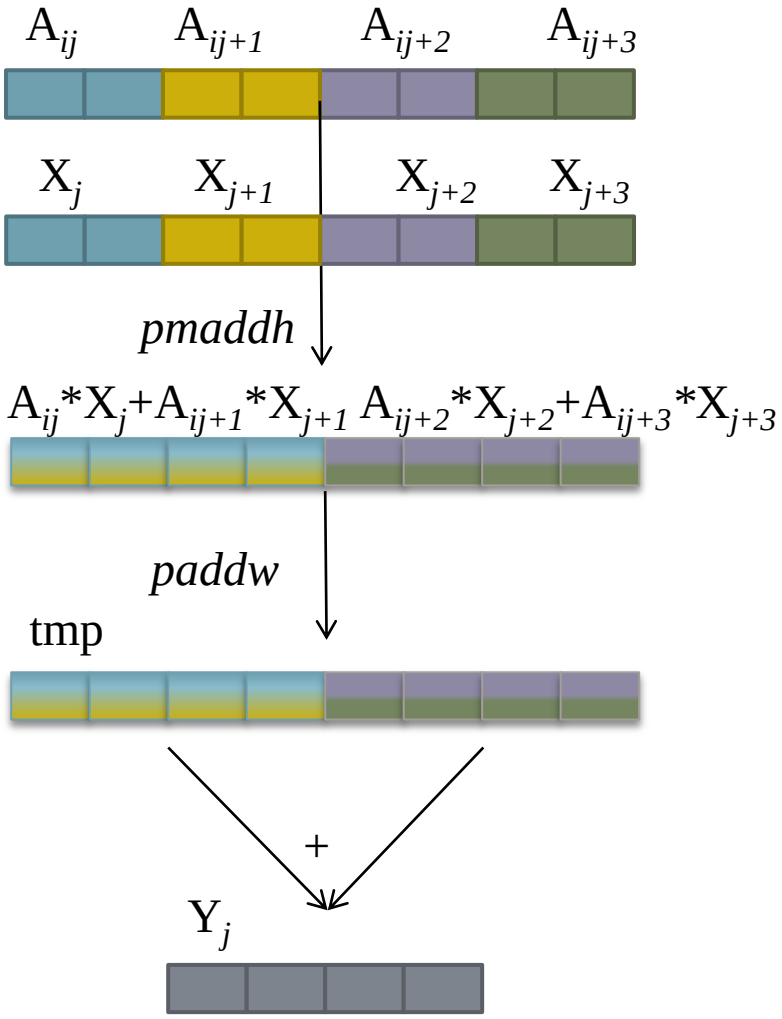
1. Подобрать состав упакованных операций для внутреннего цикла.
2. Решить проблему выравнивания при длине строки матрицы равной $4k+2$, не выровнена каждая вторая строка.
3. Решить проблему выравнивания при длине строки матрицы равной $4k+1$ и $4k+3$, не выровнены три из четырех строк.

Умножение матрицы на вектор

Состав упакованных операций

1. С помощью двух операций pmaddh и paddw накапливать частичные суммы в старших и младших разрядах двойного слова временной переменной tmp. Характеристики:
   - две pmaddh и две paddw исполняются за один такт;
   - теоретическая скорость – 8 элементов/такт;
   - арифметико-логические каналы 2/5 остаются свободными.

Для наполнения широкой команды нужно умножать двойные слова из двух соседних строк матрицы A.

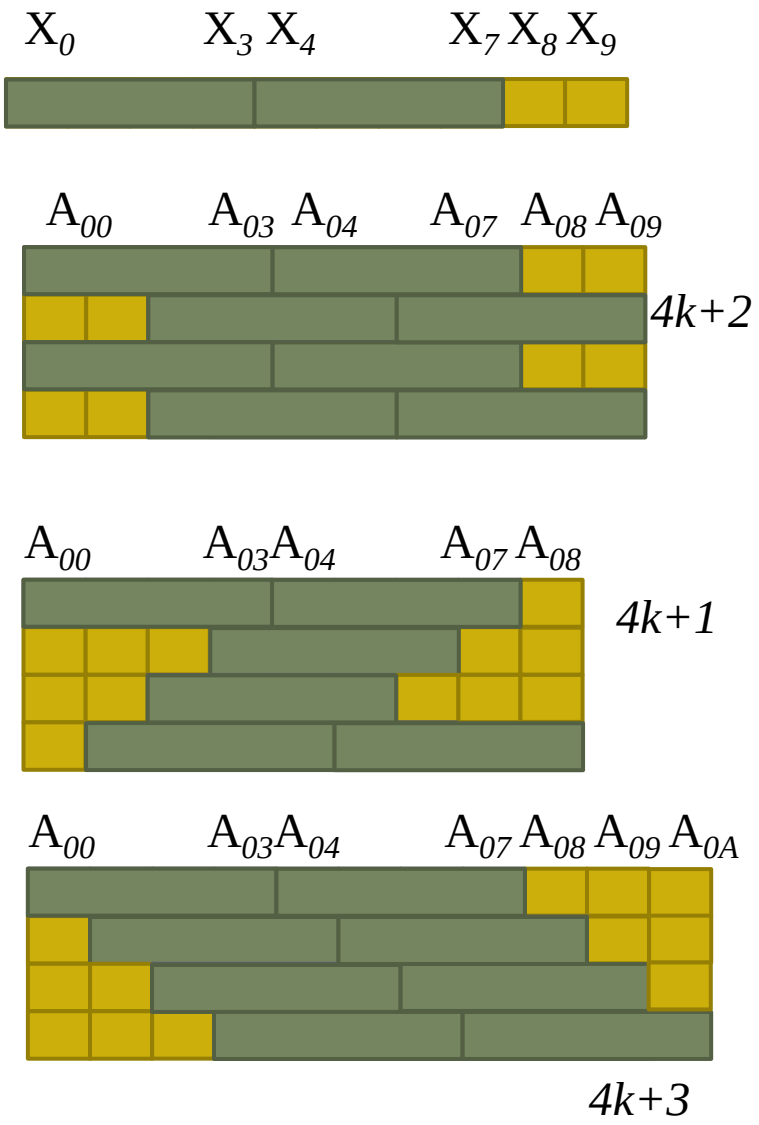


Умножение матрицы на вектор

Выравнивание

2. Выравнивать двойные слова вектора X с помощью трех (сдвиг влево, сдвиг вправо, побитовое или) логических операций. При длине строки *4k+2* скомпоновать четыре строки. Логические операции заполнят второй и пятый каналы двух широких команд, не конфликтуя с pmaddh и paddw.

3. При длине строки *4k+1* или *4k+3* необходимо выравнивать двойное слово X с тремя разными смещениями. Итого – 9 логических команд. Для их размещения во вторых и пятых каналах, необходимо в цикле компоновать 12 соседних строк. Планируется в шесть широких команд.



Умножение матрицы на вектор

Стандартный цикл умножения матрицы на вектор

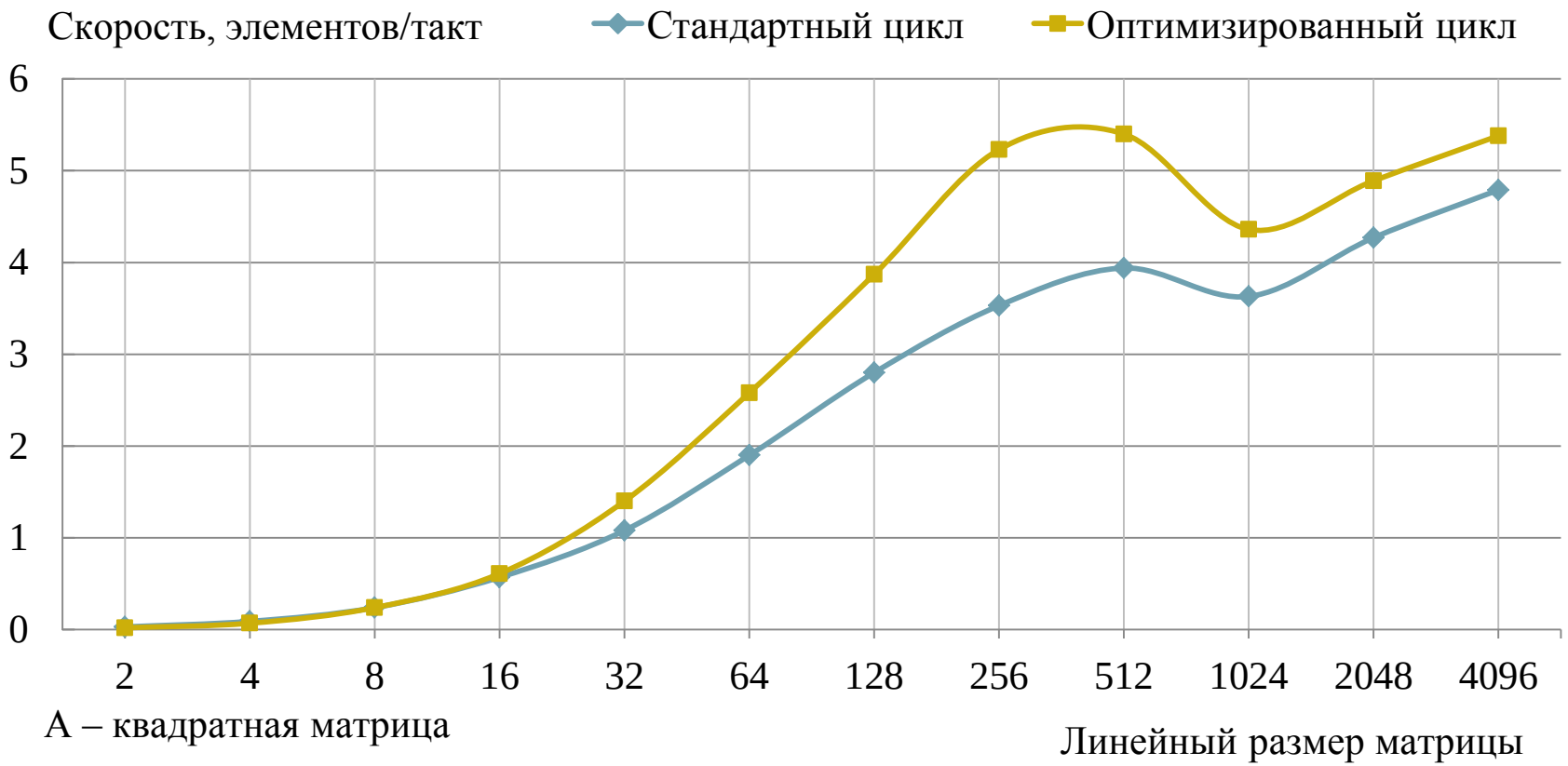
```
for(j = 0; j<lenY; j++)
      for(i=0; i<lenX; i++)
            Y[j] += ((eml_32s) X[i]) * A[j*lda+i];
```

Компилятор делает векторизацию (использует операции pmaddh), производит 2 разных цикла в зависимости от выравнивания. Выравнивание компилятор обеспечивает с помощью операций insfd. Для тестирования в стандартном цикле применена оптимизация раскрутки внешнего цикла на 4.

Теоретическая скорость в выровненном случае – 5,33 элемента/такт.
Теоретическая скорость в невыровненном случае – 4 элемента/такт.

Умножение матрицы на вектор

Результаты тестирования. Выровненные строки.

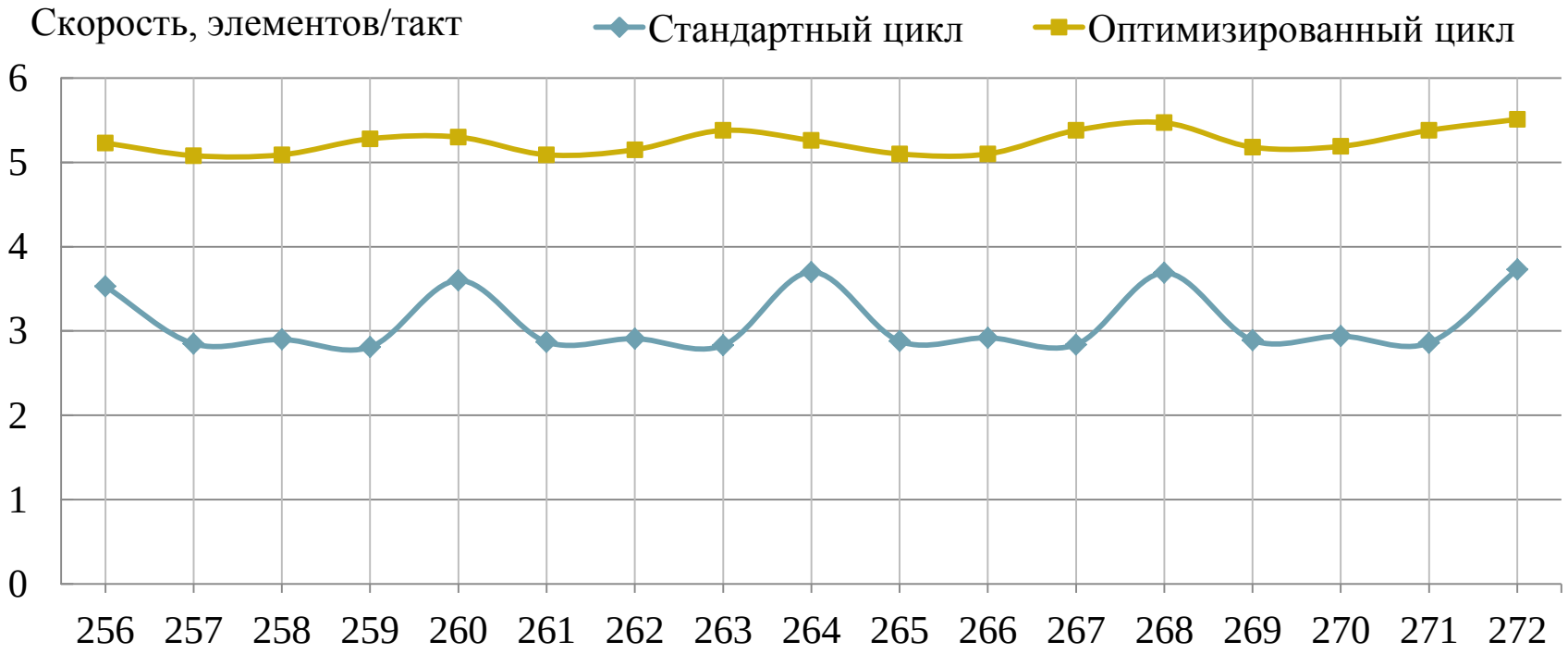


Теоретическая скорость стандартного цикла – 5,33 элемента/такт.

Теоретическая скорость оптимизированного цикла – 8 элемента/такт.

Умножение матрицы на вектор

Результаты тестирования. Зависимость от необходимости выравнивания.

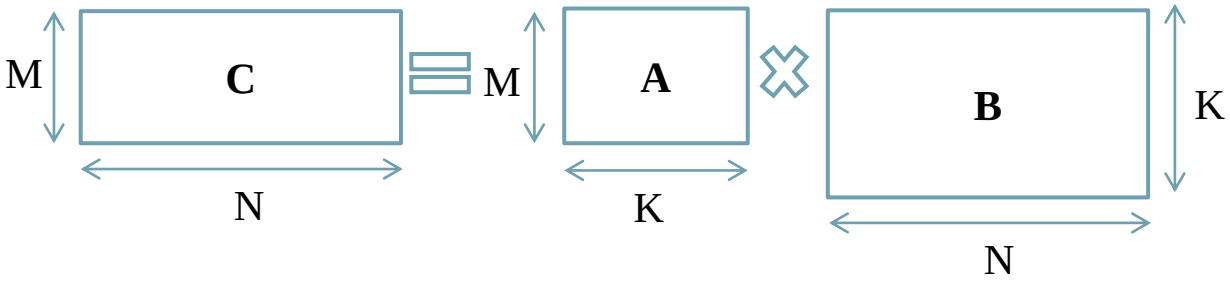


Теоретическая скорость стандартного цикла при выровненном случае – 5,33 элемента/такт; при выровненном случае – 4 элемента/такт.

Теоретическая скорость оптимизированного цикла – 8 элемента/такт.

Умножение матрицы на матрицу

За основу для реализации взята функция умножения двух матриц (float_32) с результатом в виде матрицы (float_32).



$$C_{ij} = \sum_{l=0}^{K} A_{il} B_{lj}$$

Для умножения используются три вложенных друг в друга цикла. Внутренний цикл “проходит” по строкам матрицы C и B. Загружается двойное слово $\{B_{l\ j},\ B_{l\ j+1},\ B_{l\ j+2},\ B_{l\ j+3}\}$. После применения pmaddh получаются суммы $\alpha*B_{l\,j} + \beta*B_{l\,j+1}$ и $\lambda*B_{l\,j+2} + \mu*B_{l\,j+3}$, не применимые в элементах $C_{i\,j}$, $C_{i\,j+1}$, $C_{i\,j+2}$, $C_{i\,j+3}$.

Задачи:

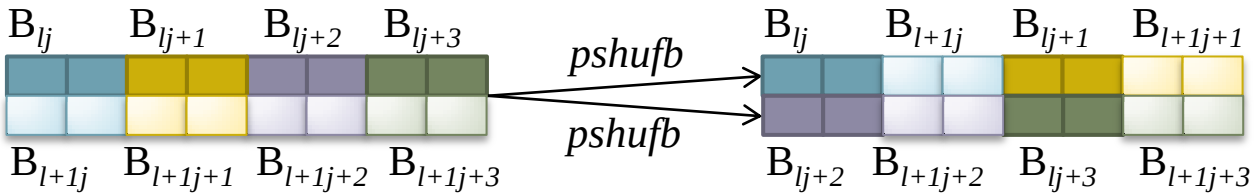
1. Решить проблему неприменимости pmaddh к двойному слову массива B.
2. Подобрать состав упакованных операций для внутреннего цикла.

Выравнивание массивов производится с помощью копирования матриц B и C до начала циклов умножения. Сложность копирования – $O(n^2)$, умножения – $O(n^3)$.

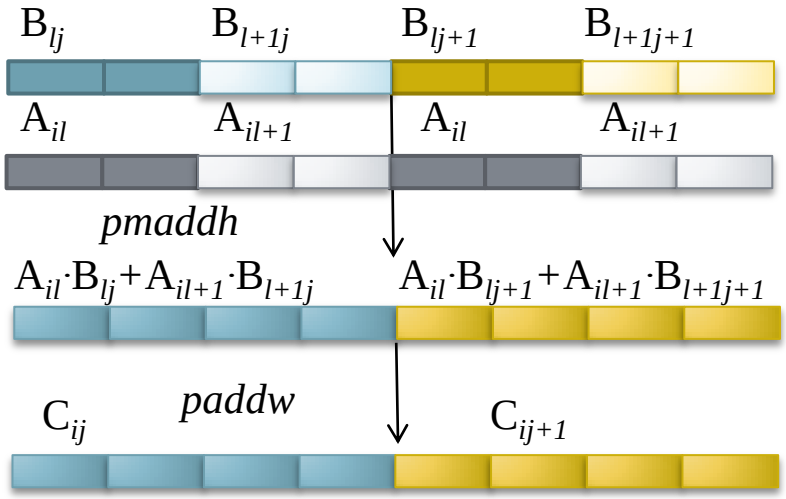
Умножение матрицы на матрицу

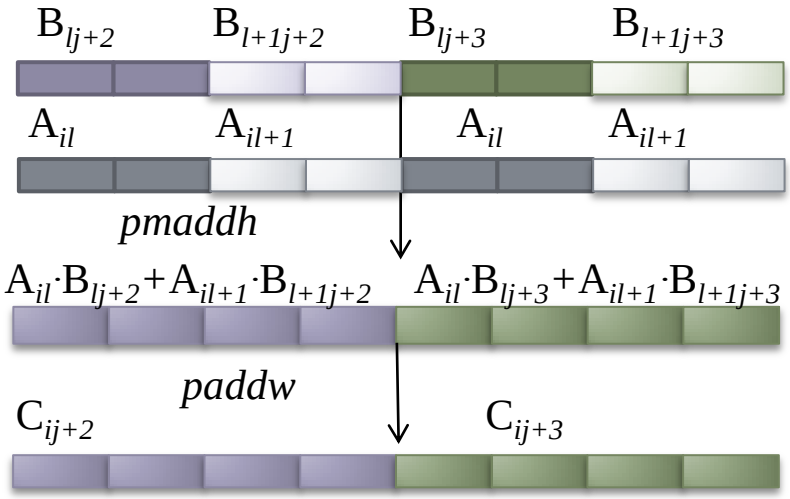
Решения поставленных задач

1. Сделать матрицу B` из B с помощью перестановки:



2. До начала внутреннего цикла необходимо подготовить операнд с числами матрицы A. Во внутреннем цикле используются операции pmaddh и paddw. Теоретическая скорость – 8 элементов/такт.





Умножение матрицы на матрицу

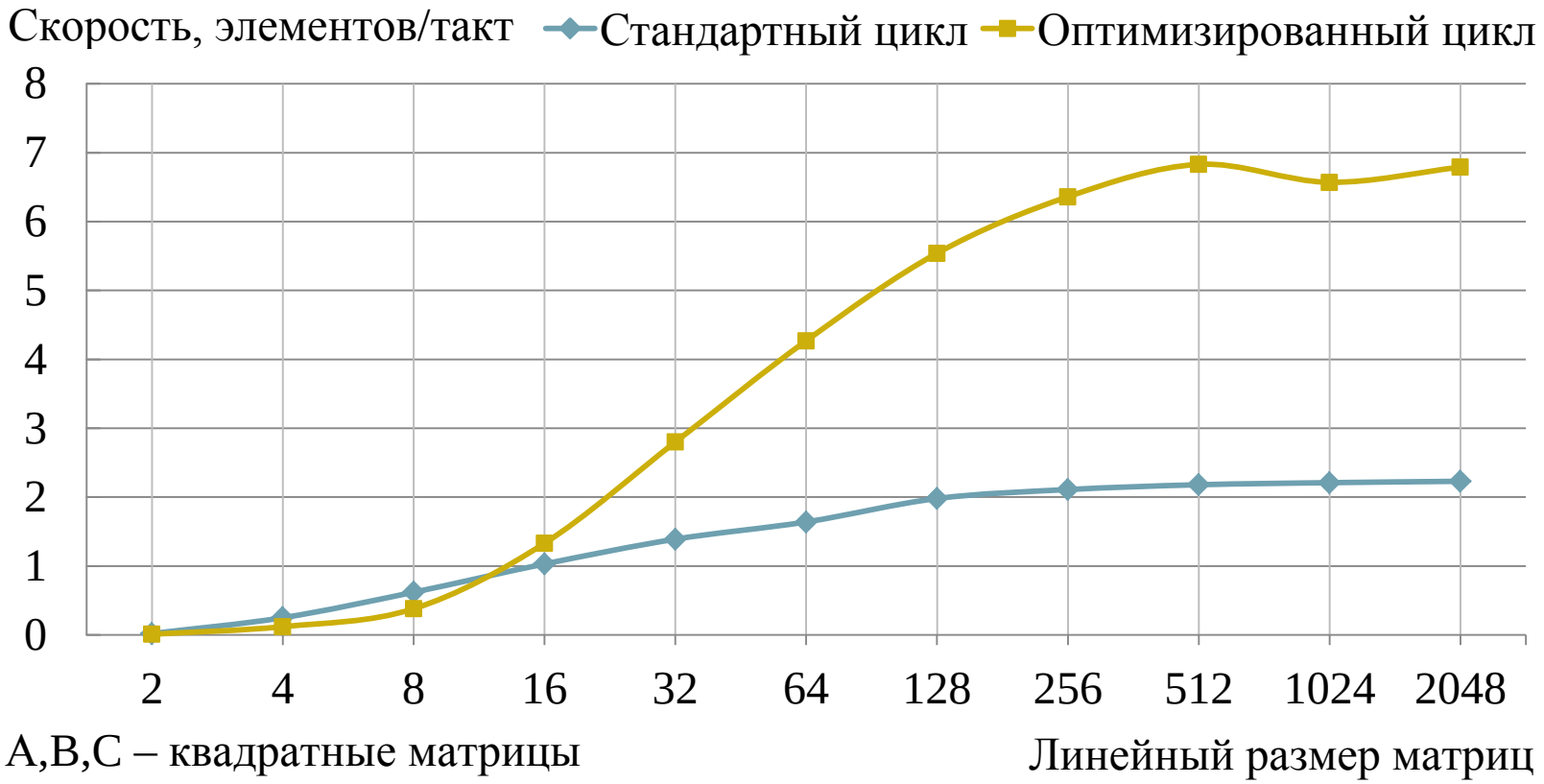
Стандартный цикл умножения матрицы на матрицу

```
for (i=0 ; i < M; i++ ) {
    for( l = 0; l < K; l++ ) {
        t00 = (eml_32s) A[(i + 0) * lda + l];
        for (j = 0; j < N; j++) {
            b0 = (eml_32s) B[l * ldb + j];
            C[i * ldc + j] += t00 * b0; } } }
```

При стандартном умножении циклы расположены в том же порядке – внутренний “проходит” по строкам матриц C и B. Применена раскрутка обоих внешних циклов на 4. Теоретическая скорость – 2,66 элемента/такт.

Умножение матрицы на матрицу

Результаты тестирования. Квадратные матрицы.



Теоретическая скорость стандартного цикла – 2,66 элемента/такт.

Теоретическая скорость оптимизированного цикла – 8 элементов/такт.

Результаты

- Проведена адаптация существующих архитектурно-оптимизированных функций библиотеки Elbrus Mathematical Library умножения матрицы на скаляр, вектор, матрицу для использования в Coremark на архитектуре Эльбрус-2С+.

- Решены проблемы, вызванные выравниванием данных в памяти.

- Реализованные функции протестированы на моноблочном компьютере КМ4-Эльбрус, построенном на базе процессора Эльбрус-2С+.

- Использование оптимизированных функций увеличивает скорость умножения матриц в тестовой задаче Coremark в 1,7 раза, что позволяет улучшить производительность тестовой задачи Coremark на 8%.

Спасибо за внимание!