Спецификация вычислительной библиотеки с архитектурно-зависимым интерфейсом.

Версия 2.00

Оглавление

1. InitLibrary. Инициализация библиотеки.

- Инициализация библиотеки. Функция должна вызываться один раз при загрузке перед использованием библиотеки.
- Инициализирует внутренние переменные библиотеки.

В текущей реализации библиотеки, переменные, требующие инициализации отсутствуют. Функция зарезервирована для расширения функциональности библиотеки.

Входные параметры: нет

Выходные параметры:

Статус (Status):
Зарезервировано для расширения функциональности библиотеки.

*long uint Status (__stdcall *InitLibrary) ( );*

Здесь и далее:

Для чисел без знака, параметров типа *long uint* подразумевается разрядность 32 бита, *long long uint* - 64 бита.

Для чисел со знаком, параметров типа *long int* подразумевается разрядность 32 бита, *long long int* - 64 бита.

```
// Инициализация
FUNCTION InitLibrary()
    : Uint32;  // Возвращаемый статус
```

2. Группа функций для умножения матриц на основе базового цикла версии 1 (вариант для обработки больших матриц или классическая схема).

- Основная вычислительная процедура (ОВП) операции «Умножение матриц» → умножение одного макростолбца A на последовательность блоков матрицы B (иллюстрации в документах Mutr_Mult_a.GIF, Mutr_Mult_b.gif).
- Все указатели 64 битные, выровненные на 32 байта (5 младших бит указателей должны быть нулевые).
- Длина полосочки (Hb) определяется выбором одной из функций в пределах группы, счетчиком развернутого цикла и дополнительным слагаемым. См. формулы при перечислении функций.
- Количество полосочек в блоке матрицы B (Wb) определяется выбором одной из функций в пределах группы, поэтому не передается в виде отдельного параметра. См. расшифровку имен функций.

- Все счетчики - 32 битные целые значения.
- A-атом содержит 2 четверки чисел (64 байта).
- B-атом содержит 4 четверки чисел (128 байт).
- Сохранение результирующей матрицы C выполняется сложением с памятью (а не записью в память). Поэтому матрица C исходно должна быть обнулена, за исключением ситуаций, при которых необходимо сложение результирующей матрицы с некоторой исходной матрицей.
- При формировании входных параметров, следует учитывать, что для части функций (см. перечисление функций), пятый параметр (N_x3_x4) отсутствует, в этом случае параметры, ниже перечисленные под номерами 6 и 7, становятся соответственно параметрами 5 и 6.

Список параметров.

1. **PointerA.** Указатель на входной A-макростолбец.
2. **PointerB.** Указатель на начальный B-блок.
3. **PointerC.** Указатель на начальный макростолбец выходной матрицы C.
4. **PointerPB.**
   Указатель используемый для B-prefetch при обработке последнего B-блока На всех внутренних итерациях функция вычисляет адрес для prefetch автономно, данный указатель используется только для последней итерации, когда функция "не знает" адрес следующего B-блока.
5. **N_x3_x4.**

   Счетчик развернутого цикла для определения длины полосочки, см. формулы при описании функций.
6. **Ha.**

   Количество полосочек в A-макростолбце.
7. **Nb.**
   Количество B-блоков в серии, на которую умножается A-макростолбец.

```
// Умножение матриц, вариант без цикла для количества атомов не более 3(4)
PROCEDURE <Name>
      ( PointerA  : P_Float64;   // Указатель на матрицу A, выровнен на 32
        PointerB  : P_Float64;   // Указатель на матрицу B, выровнен на 32
        PointerC  : P_Float64;   // Указатель на матрицу C, выровнен на 32
        PointerPB : P_Float64;   // Указатель для B-prefetch
        Ha        : Uint32;      // Полосочек в A-макростолбце
        Nb        : Uint32);     // B-блоков в серии
```

<Name> имена функций:

MultiplyMatrixBig_Wb1_1 , MultiplyMatrixBig_Wb1_2 , MultiplyMatrixBig_Wb1_3, MultiplyMatrixBig_Wb1_4.

MultiplyMatrixBig_Wb2_1 , MultiplyMatrixBig_Wb2_2 , MultiplyMatrixBig_Wb2_3, MultiplyMatrixBig_Wb2_4.

MultiplyMatrixBig_Wb3_1 , MultiplyMatrixBig_Wb3_2 , MultiplyMatrixBig_Wb3_3

Расшифровка имен функций. *MultiplyMatrixBig_WbX_Y.*

X = количество полосочек в блоке матрицы B.

Y = длина полосочки в атомах.

```
// Умножение матриц, вариант с циклом для количества атомов более 3(4)
PROCEDURE <Name>
        ( PointerA  : P_Float64;    // Указатель на матрицу A, выровнен на 32
          PointerB  : P_Float64;    // Указатель на матрицу B, выровнен на 32
          PointerC  : P_Float64;    // Указатель на матрицу C, выровнен на 32
          PointerPB : P_Float64;    // Указатель для B-prefetch
          N_x3_x4   : Uint32;       // Счетчик для определения длины полосочки
          Ha        : Uint32;       // Полосочек в A-макростолбце
          Nb        : Uint32);      // B-блоков в серии
```

<Name> имена функций:

MultiplyMatrixBig_Wb1_4N1 , MultiplyMatrixBig_Wb1_4N2 , MultiplyMatrixBig_Wb1_4N3, MultiplyMatrixBig_Wb1_4N4.

MultiplyMatrixBig_Wb2_4N1 , MultiplyMatrixBig_Wb2_4N2 , MultiplyMatrixBig_Wb2_4N3, MultiplyMatrixBig_Wb2_4N4.

MultiplyMatrixBig_Wb3_3N1 , MultiplyMatrixBig_Wb3_3N2 , MultiplyMatrixBig_Wb3_3N3

Расшифровка имен функций. *MultiplyMatrixBig_WbX_YNZ.*

X = количество полосочек в блоке матрицы B.

Y = длина одной итерации развернутого цикла в атомах.

Z = слагаемое для получения длины полосочки.

Длина полосочки определяется по формуле:

Hb = N_x3_X4 * Y + Z

Здесь N_x3_x4 - входной параметр функции, Y,Z - числа из имени функции.

Например, чтобы задать B-блок размером три полосочки при длине полосочки 48 атомов, требуется передать параметр N_x3_x4 = 15 и вызвать функцию *MultiplyMatrixBig_Wb3_3N3*.

Hb = 15*3+3 = 45+3 = 48.

Функции, использующие базовый цикл версии 1 (условное название - horizontal add method) построены в виде следующей системы вложенных циклов.

1) *Базовый цикл*. Умножение одной A-полосочки на одну B-полосочку. При этом формируется C-атом, записываемый путем сложений с памятью (а не сохранением в память), так как на каждом проходе формируется только одно из слагаемых. Данный метод требует серии "горизонтальных сложений" над содержимым AVX-регистров перед сохранением результатов. Базовый цикл состоит из "разгонного блока", цикла и "хвостика". Количество обработанных атомов равно количеству итераций цикла + 1, так как "разгонный блок" и "хвостик" обрабатывают один атом. Цикл развернут на 3 или 4 в зависимости от конкретной функции. Оптимизация обработки произвольной длины полосочки в развернутом цикле реализуется за счет использования нескольких вариантов функции, представленных для программы верхнего уровня как несколько подобных функций одной группы. Детальное описание базового цикла приведено в документе *Базовый_цикл_2x4_1-1б.doc*.

2) Цикл для умножения серии полосочек B-блока на одну A-полосочку *(L1 cycle)*, в этот цикл вложен базовый цикл (пункт 1). Для каждой следующей запускаемой серии итераций базового цикла, изменяется адрес B-полосочки. Адрес A-полосочки восстанавливается, так как в этом цикле базовый цикл требуется повторно запускать для одной A-полосочки. Адрес матрицы C увеличивается на один атом после каждой серии итераций базового цикла.

3) Цикл для серии A-полосочек *(L2 cycle)*. При входе в серию итераций восстанавливаем указатель B, для повторного прохода по B-блоку и модифицируем указатель A для перехода на следующую полосочку A-макростолбца.

4) Цикл для обработки серии из B-блоков. На каждой итерации этого цикла обрабатывается следующий B-блок, а указатель A на каждой итерации восстанавливается для повторного прохода по A-макростолбцу.

С целью минимизации задержек (латентности) при доступе к матрицам, выполняется предварительная загрузка данных в кэш-память с использованием специальных инструкций (Prefetch Hints), имеющихся в составе функциональных расширений SSE/AVX. Общий принцип расстановки таких инструкций состоит в том, чтобы частота их выполнения обеспечивала загрузку следующего блока данных за время обработки текущего блока. Выполнение этого условия зависит от соотношения между параметрами Hb (длина полосочки), Wb (количество полосочек в B-блоке) и Ha (количество полосочек в A-макростолбце). Это необходимо учитывать при задании размеров блоков. Наличие отдельных вариантов функций для различных значений Wb, связано с данным обстоятельством.

Логика формирования адресов для операции Prefetch учитывает специальные случаи, при которых следующий адрес не представим в виде " текущий адрес + шаг ". Сюда относится повторное прохождение по блоку данных и завершение серии последовательно расположенных блоков. В первом случае адрес для следующей операции Prefetch функция формирует самостоятельно (возвращаясь на начало блока), во втором случае функция должна получить его как входной параметр из процедуры верхнего уровня.

Для архитектуры Intel Sandy Bridge, одна операция Prefetch загружает в кэш-память 64 байта. Так как параллельно с программной пред-загрузкой функционирует аппаратная, в некоторых случаях размер автоматически загруженных данных может увеличиваться до 128 байт и более.

Частота выполнения Prefetch следующей полосочки из макростолбца входной матрицы A выбирается по критерию обеспечения загрузки следующей A-полосочки за время использования текущей полосочки, то есть за время обработки серии полосочек B-блока. Это означает:
1 prefetch на обработку каждого атома при Wb=1
1 prefetch на обработку каждого второго атома при Wb=2
1 prefetch на обработку каждого третьего атома при Wb=3

Частота выполнения Prefetch следующего блока входной матрицы B фиксирована для всех функций - один Prefetch на серию итераций базового цикла. Следовательно, для того, чтобы за время использования текущего B-блока, то есть за время прохождения по A-макростолбцу, загружался следующий B-блок, количество полосочек A-макростолбца (Ha) должно быть равно количеству атомов в полосочке (Hb). Параметр Wb здесь учитывается автоматически, так как количество операций Prefetch возрастет пропорционально увеличению Wb.

Prefetch для выходной матрицы C был отключен по результатам испытания базового цикла с поддержкой отложенной записи результата. Детальное описание базового цикла приведено в документе *Базовый_цикл_2x4_1-1б.doc*.

3. Группа функций для умножения матриц на основе базового цикла версии 2 (вариант для обработки маленьких матриц или broadcast схема).

- Основная вычислительная процедура (ОВП) операции «Умножение матриц» → умножение одного этажа A на последовательность блоков одной вертикальной секции матрицы B (иллюстрации в документе Mutr_Mult_V2_algorithm.GIF).
- Все указатели 64 битные, выровненные на 32 байта (5 младших бит указателей должны быть нулевые).
- Длина полосочки (Wb) определяется выбором одной из функций в пределах группы, счетчиком развернутого цикла и дополнительным слагаемым. См. формулы при перечислении функций.
- Количество полосочек в блоке матрицы B (Hb) определяется выбором одной из функций в пределах группы, поэтому не передается в виде отдельного параметра. См. расшифровку имен функций.
- Все счетчики - 32 битные целые значения.
- A-атом содержит 2 четверки чисел (64 байта).
- B-атом содержит 4 четверки чисел (128 байт).
- Сохранение результирующей матрицы C выполняется сложением с памятью (а не записью в память). Поэтому матрица C исходно должна

быть обнулена, за исключением ситуаций, при которых необходимо сложение результирующей матрицы с некоторой исходной матрицей.
- При формировании входных параметров, следует учитывать, что для части функций (см. перечисление функций), пятый параметр (N_x3_x4) отсутствует, в этом случае параметры, ниже перечисленные под номерами 6 и 7, становятся соответственно параметрами 5 и 6.

Список параметров.

1. **PointerA.** Указатель на начальный A-макростолбец.
2. **PointerB.** Указатель на начальный B-блок.
3. **PointerC.** Указатель на формируемый блок выходной матрицы C.
4. **PointerPB.**
   Указатель используемый для B-prefetch при обработке последнего B-блока На всех внутренних итерациях функция вычисляет адрес для prefetch автономно, данный указатель используется только для последней итерации, когда функция "не знает" адрес следующего B-блока.
5. **N_x3_x4.**

   Счетчик развернутого цикла для определения длины полосочки, см. формулы при описании функций.
6. **Ha.**

   Количество полосочек в A-макростолбце.
7. **Nb.**
   Количество B-блоков в серии, на которую умножается A-макростолбец.

```
// Умножение матриц, вариант без цикла для количества атомов не более 3(4)
PROCEDURE <Name>
        ( PointerA   : P_Float64;   // Указатель на матрицу A, выровнен на 32
          PointerB   : P_Float64;   // Указатель на матрицу B, выровнен на 32
          PointerC   : P_Float64;   // Указатель на матрицу C, выровнен на 32
          PointerPB  : P_Float64;  // Указатель для B-prefetch
          Ha         : Uint32;        // Полосочек в C-блоке
          Nb         : Uint32);       // B-блоков в серии

<Name> имена функций:
MultiplyMatrixSmall_Hb1_1 , MultiplyMatrixSmall_Hb1_2 , MultiplyMatrixSmall_Hb1_3,
MultiplyMatrixSmall_Hb1_4.
MultiplyMatrixSmall_Hb2_1 , MultiplyMatrixSmall_Hb2_2 , MultiplyMatrixSmall_Hb2_3,
MultiplyMatrixSmall_Hb2_4.
MultiplyMatrixSmall_Hb3_1 , MultiplyMatrixSmall_Hb3_2 , MultiplyMatrixSmall_Hb3_3
```

Расшифровка имен функций. *MultiplyMatrixSmall_HbX_Y.*

X = количество полосочек в блоке матрицы B.

Y = длина полосочки в атомах.

```
// Умножение матриц, вариант с циклом для количества атомов более 3(4)
PROCEDURE <Name>
```

```
( PointerA   : P_Float64;   // Указатель на матрицу A, выровнен на 32
  PointerB   : P_Float64;   // Указатель на матрицу B, выровнен на 32
  PointerC   : P_Float64;   // Указатель на матрицу C, выровнен на 32
  PointerPB  : P_Float64;  // Указатель для B-prefetch
  N_x3_x4    : Uint32;     // Счетчик для определения атомов в полосочке
  Ha         : Uint32;       // Полосочек в C-блоке
  Nb         : Uint32);      // B-блоков в серии
```

<Name> имена функций:
MultiplyMatrixSmall_Hb1_4N1 , MultiplyMatrixSmall_Hb1_4N2 ,
MultiplyMatrixSmall_Hb1_4N3, MultiplyMatrixSmall_Hb1_4N4.
MultiplyMatrixSmall_Hb2_4N1 , MultiplyMatrixSmall_Hb2_4N2 ,
MultiplyMatrixSmall_Hb2_4N3, MultiplyMatrixSmall_Hb2_4N4.
MultiplyMatrixSmall_Hb3_3N1 , MultiplyMatrixSmall_Hb3_3N2 ,
MultiplyMatrixSmall_Hb3_3N3

Расшифровка имен функций. *MultiplyMatrixSmall_HbX_YNZ.*

X = количество полосочек в блоке матрицы B.

Y = длина одной итерации развернутого цикла в атомах.

Z = слагаемое для получения длины полосочки.

Длина полосочки определяется по формуле:

Wb = N_x3_X4 * Y + Z

Здесь N_x3_x4 - входной параметр функции, Y,Z - числа из имени функции.

Например, чтобы задать B-блок размером в одну полосочку при длине полосочки 17 атомов, требуется передать параметр N_x3_x4 = 4 и вызвать функцию *MultiplyMatrixSmall_Hb1_4N1*.

Wb = 4*4+1 = 16+1 = 17.

Функции, использующие базовый цикл версии 2 (условное название - broadcast method) построены в виде следующей системы вложенных циклов.

1) *Базовый цикл*. Умножение одного A-атома (подвергнутого BroadCast в 8 AVX-регистров) на B-полосочку. При этом формируется C-полосочка, записываемая путем сложений с памятью (а не сохранением в память), так как на каждом проходе формируется только одно из слагаемых. Базовый цикл состоит из "разгонного блока", цикла и "хвостика". Количество обработанных атомов равно количеству итераций цикла + 1, так как "разгонный блок" и "хвостик" обрабатывают один атом. Цикл развернут на 3 или 4 в зависимости от конкретной функции. Оптимизация обработки произвольной длины полосочки в развернутом цикле реализуется за счет использования нескольких вариантов функции, представленных для программы верхнего уровня как несколько подобных функций одной группы. Детальное описание базового цикла приведено в документе *Базовый_цикл_2x4_2_LAST.doc*.

2) Цикл для серии полосочек B-блока *(L1 cycle)*, в этот цикл вложен базовый цикл (пункт 1). Для каждой следующей запускаемой серии итераций базового цикла, загружаются новые значения A-BroadCast и выполняется переход на следующую B-полосочку. Адрес C-полосочки восстанавливается,

так как в этом цикле базовый цикл требуется повторно запускать для одной С-полосочки.

3) Цикл для серии С-полосочек *(L2 cycle)*. При входе в серию итераций восстанавливаем указатель В, для повторного прохода по В-блоку и модифицируем С-указатель для перехода на следующую полосочку. Адресация матрицы А и загрузка BroadCast выполняется последовательно далее.

4) Цикл для обработки серии из В-блоков. На каждой итерации этого цикла обрабатывается следующий В-блок, а С-указатель на каждой итерации восстанавливается для повторного прохода по С-блоку. А-указатель модифицируется для дальнейшего продвижения по матрице. Данный цикл обеспечивает обработку полного этажа матрицы А и полной вертикальной секции матрицы В. Эта процедура формирует один С-блок и выполняется в виде одного вызова функции, в случае, если блоки в пределах обрабатываемого горизонтального А-этажа одинаковы и блоки в пределах обрабатываемой вертикальной В-секции одинаковы.

С целью минимизации задержек (латентности) при доступе к матрицам, выполняется предварительная загрузка данных в кэш-память с использованием специальных инструкций (Prefetch Hints), имеющихся в составе функциональных расширений SSE/AVX. Общий принцип расстановки таких инструкций состоит в том, чтобы частота их выполнения обеспечивала загрузку следующего блока данных за время обработки текущего блока. Выполнение этого условия зависит от соотношения между параметрами Wb (длина полосочки), Hb (количество полосочек в В-блоке) и Ha (количество полосочек в С-блоке). Это необходимо учитывать при задании размеров блоков. Наличие отдельных вариантов функций для различных значений Hb, связано с данным обстоятельством.
Логика формирования адресов для операции Prefetch учитывает специальные случаи, при которых следующий адрес не представим в виде " текущий адрес + шаг ". Сюда относится повторное прохождение по блоку данных и завершение серии последовательно расположенных блоков. В первом случае адрес для следующей операции Prefetch функция формирует самостоятельно (возвращаясь к началу блока), во втором случае функция должна получить его как входной параметр из процедуры верхнего уровня.

Для архитектуры Intel Sandy Bridge, одна операция Prefetch загружает в кэш-память 64 байта. Так как параллельно с программной пред-загрузкой функционирует аппаратная, в некоторых случаях размер автоматически загруженных данных может увеличиваться до 128 байт и более.

Для входной матрицы А, Prefetch следующего атома выполняется после загрузки очередного атома.

Частота выполнения Prefetch следующего блока входной матрицы В фиксирована для всех функций - один Prefetch на обработку С-полосочки. Этот параметр будет корректироваться индивидуально для функций по результатам расчетов и испытаний.

Частота выполнения Prefetch следующей полосочки из макростолбца выходной матрицы C выбирается по критерию обеспечения загрузки следующей C-полосочки за время использования текущей полосочки, то есть за время обработки серии полосочек B-блока. Это означает:
1 prefetch на обработку каждого атома при Hb=1
1 prefetch на обработку каждого второго атома при Hb=2
1 prefetch на обработку каждого третьего атома при Hb=3.

Примечание.

Для алгоритма, построенного на основе базового цикла V1 (horizontal add method), полосочка блока B расположена вертикально, в этом случае, параметр Hb определяет количество итераций базового цикла (длину полосочки), а Wb - количество полосочек (высоту блока B).

Для алгоритма, построенного на основе базового цикла V2 (broadcast method), полосочка блока B расположена горизонтально, в этом случае, параметр Wb определяет количество итераций базового цикла (длину полосочки), а Hb - количество полосочек (высоту блока B).

4. Упаковка макростолбца матрицы A (вариант для обработки больших матриц).

- Упаковка 1-го макростолбца матрицы A (иллюстрация в файле *Перепаковка_A.GIF*).
- A-атом содержит 2 четверки чисел из 2-х смежных строк, и упаковывается в линейную последовательность из 8 чисел.
- Все указатели 64 битные, выровнены на 32 байта (5 младших бит ==0).
- Все счетчики - 32 битные целые значения (беззнаковые, >=0).

Список параметров.

1. **PointerA**. Указатель на A-макростолбец.
2. **PointerBuffer**. Указатель на транзитный буфер.
3. **Xa**.
   Размер одной строки матрицы A в байтах. Это значение прибавляется к указателю для перехода на следующую строку, поэтому должно быть выровненным (кратным 32), аналогично указателям.
4. **Hb**. Количество атомов в полосочке.
5. **Ha**. Количество полосочек в A-макростолбце.
6. **FlagTransp.**
   Флаг транспонирования матрицы A. Нулевое значение (FALSE) означает, что исходная матрица A не транспонирована, ненулевое значение (TRUE) означает, что исходная матрица A транспонирована.

```
void ( __stdcall *PackA )
( double* PointerA,  double* PointerBuffer,
```

long uint Xa, long uint Hb, long uint Ha, long uint FlagTransp);

```
// Упаковка макростолбца матрицы A
PROCEDURE PackA
    ( PointerA       : P_Float64;    // Указатель на макрост. A, выровнен на 32
      PointerBuffer : P_Float64;     // Указатель на буфер, выровнен на 32
      Xa             :  Uint32;      // Размер строки матрицы A в байтах
      Hb             :  Uint32;      // Количество атомов в полосочке
      Ha             :  Uint32;      // Количество полосочек
      FlagTransp     : Uint32 );     // Флаг транспонирования
```

5. Упаковка блока матрицы B (вариант для обработки больших матриц).

- Упаковка 1-го блока матрицы B (иллюстрация в файле *Перепаковка_B.GIF*).
- B-атом содержит 4 четверки чисел из 4-х смежных столбцов, и преобразуется в линейную последовательность из 16 чисел.
- Все указатели 64 битные, выровненные на 32 байта (5 младших бит указателей должны быть нулевые).
- Все счетчики - 32 битные целые значения

Список параметров.

1. **PointerB**. Указатель на упаковываемый B-блок.
2. **PointerBuffer**. Указатель на транзитный буфер.
3. **Xb**.
   Размер одной строки матрицы B, байт. Функция использует это значение для вертикального перемещения по матрице. Для смещения на N строк вниз, к указателю прибавляется Xb*N. Поскольку это значение прибавляется к указателю оно также должно быть выровненным (кратным 32), аналогично указателям, чтобы в результате сложения указатель остался выровненным.
4. **Hb**. Количество атомов в полосочке.
5. **Wb**. Количество полосочек в B-блоке.

*void (__stdcall *PackB)*
(double PointerB, double* PointerBuffer,*
long uint Xb, long uint Hb, long uint Wb);

```
// Упаковка блока матрицы B
PROCEDURE PackB
    ( PointerB       : P_Float64;    // Указатель на блок B, выровнен на 32
      PointerBuffer  : P_Float64;    // Указатель на буфер, выровнен на 32
      Xb             :  Uint32;      // Размер строки матрицы B в байтах
      Hb             :  Uint32;      // Количество атомов в полосочке
```

```
    Wb          : Uint32 );     // Количество полосочек
```

6. Восстановление макростолбца матрицы A (вариант для обработки больших матриц).

- Восстановление 1-го макростолбца матрицы A.
- A-атом содержит 2 четверки чисел из 2-х смежных строк.
- Все указатели 64 битные, выровненные на 32 байта (5 младших бит указателей должны быть нулевые).
- Все счетчики - 32 битные целые значения

Процедура обеспечивает восстановление одного заданного макростолбца матрицы A, для приведения в исходное состояние после операции упаковки. Операция обратная по отношению к функции *PackA*.

Все указатели - 64 битные, выровненные на 32 байта (5 младших бит указателей должны быть нулевые). Все счетчики - 32 битные целые значения, так как значения счетчиков более 2^32 не будут использоваться.

Список параметров.

1. PointerA. Указатель на A-макростолбец, ранее подвергнутый упаковке функцией *PackA*.

2. PointerBuffer. Указатель на транзитный буфер, используемый для восстановления макростолбцов.

3. Xa.

Размер одной строки матрицы A в байтах. Это значение прибавляется к указателю для перехода на следующую строку восстанавливаемой матрицы, поэтому должно быть выровненным (кратным 32), аналогично указателям.

4. Hb. Количество атомов в полосочке.

5. Ha. Количество полосочек в A-макростолбце.

```
void ( __stdcall *UnPackA )
( double* PointerA,  double* PointerBuffer,
long uint Xa, long uint Hb, long uint Ha, long int FlagTransp );
```

```
// Восстановление макростолбца матрицы A
PROCEDURE UnPackA
    ( PointerA      : P_Float64;    // Указатель на макрост. A, выровнен на 32
      PointerBuffer : P_Float64;    // Указатель на буфер, выровнен на 32
      Xa            : Uint32;       // Размер строки матрицы A в байтах
      Hb            : Uint32;       // Количество атомов в полосочке
      Ha            : Uint32 );     // Количество полосочек
```

7. Восстановление блока матрицы B (вариант для обработки больших матриц).

- Восстановление 1-го блока матрицы B
  B-атом содержит 4 четверки чисел из 4-х смежных столбцов.
- Все указатели 64 битные, выровненные на 32 байта (5 младших бит указателей должны быть нулевые).
- Все счетчики - 32 битные целые значения

Процедура обеспечивает восстановление одного заданного блока матрицы B для приведения в исходное состояние после операции упаковки. Операция, обратная по отношению к функции *PackB*.

Все указатели - 64 битные, выровненные на 32 байта (5 младших бит указателей должны быть нулевые). Все счетчики - 32 битные целые значения, так как значения счетчиков более 2^32 не будут использоваться.

Список параметров.

1. PointerB. Указатель на начальный B-блок, ранее подвергнутый упаковке функцией PackB.

2. PointerBuffer. Указатель на транзитный буфер, используемый для восстановления блоков.

3. Xb.
Размер одной строки матрицы B, байт. Функция использует это значение для вертикального перемещения по восстанавливаемой матрице. Для смещения на N строк вниз, к указателю прибавляется Xb*N. Поскольку это значение прибавляется к указателю оно также должно быть выровненным (кратным 32), аналогично указателям, чтобы в результате сложения указатель остался выровненным.

4. Hb. Количество атомов в полосочке.

5. Wb. Количество полосочек в B-блоке.

```
void ( __stdcall *UnPackB )
( double* PointerB,  double* PointerBuffer,
long uint Xb, long uint Hb, long uint Wb );
```

```
// Восстановление блока матрицы B
PROCEDURE UnPackB
    ( PointerB       : P_Float64;    // Указатель на блок B, выровнен на 32
      PointerBuffer  : P_Float64;    // Указатель на буфер, выровнен на 32
      Xb             :  Uint32;      // Размер строки матрицы B в байтах
      Hb             :  Uint32;      // Количество атомов в полосочке
      Wb             :  Uint32 );    // Количество полосочек
```

8. Распаковка макростолбца матрицы С (вариант для обработки больших матриц).

Для выполнения этой операции используется ранее описанная функция *UnPackA*, с указанием соответствующих размеров макростолбца (иллюстрация в файле *Перепаковка_П.GIF*). По этой причине функция *UnPackC* в библиотеке отсутствует.

1. Этаж заранее переносится в буфер. C_PackedMcrC^ находится в буфере, доступ к элементам которого линейный.
2. Атом С имеет размеры 2x4: "высота атома А" x "ширина атома В". При принятых размерах А-атома (2x4) и В-атома (4x4) размер С-атома совпал с размером А-атома. Вследствие принятого алгоритма расчета С-макростолбец формируется по ходу расчета таким образом, что его структура полностью аналогична структуре А-макростолбца.
3. ПРИМЕР: если бы мы использовали А-атом (4x4) и В-атом (4x2), то упаковка матрицы С оказалась бы иной.

9. Перепаковка входной матрицы А (вариант для обработки маленьких матриц).

- Функция обрабатывает один А-блок. Предполагается, что процедура верхнего уровня, используя данную функцию, обрабатывает серию А-блоков. Перепаковка осуществляется в транзитный буфер. (Описание и иллюстрации в документах MxM_BroadCast_перепаковка.DOC, Mutr_Mult_V2_algorithm.GIF, Перепаковка_2_A.GIF).
- Все указатели 64 битные, выровненные на 32 байта (5 младших бит указателей должны быть нулевые).
- Все счетчики - 32 битные целые значения.
- А-атом содержит 2 четверки чисел (64 байта).

Список параметров.

1. **PointerA.** Указатель на начальный А-блок.
2. **PointerBuffer.** Указатель на транзитный буфер.
3. **Xa.**
   Размер одной строки матрицы А в байтах. Это значение прибавляется к указателю для перехода на следующую строку, поэтому должно быть выровненным (кратным 32), аналогично указателям.
4. **Hb.** Количество атомов в полосочке.
5. **Ha.** Количество полосочек в А-блоке.
6. **FlagTransp.**
   Флаг транспонирования матрицы А. Нулевое значение (FALSE)

означает, что исходная матрица A не транспонирована, ненулевое значение (TRUE) означает, что исходная матрица A транспонирована.

```
// Перепаковка матрицы A
PROCEDURE PackASmall
        ( PointerA       : P_Float64;    // Указатель на матрицу A, выровнен на 32
          PointerBuffer  : P_Float64;    // Указатель на буфер, выровнен на 32
          Xa             : Uint32;       // Байт на одну полную строку матрицы A
          Hb             : Uint32;       // Атомов в полосочке
          Ha             : Uint32;       // Полосочек в A-блоке
          FlagTransp     : Uint32);      // Флаг транспонирования матрицы A
```

10. Перепаковка входной матрицы B (вариант для обработки маленьких матриц).

- Функция обрабатывает один B-блок. Предполагается, что процедура верхнего уровня, используя данную функцию, обрабатывает серию B-блоков. Перепаковка осуществляется в транзитный буфер. (Описание и иллюстрации в документах MxM_BroadCast_перепаковка.DOC, Mutr_Mult_V2_algorithm.GIF, Перепаковка_2_B.GIF).
- Все указатели 64 битные, выровненные на 32 байта (5 младших бит указателей должны быть нулевые).
- Все счетчики - 32 битные целые значения.
- B-атом содержит 4 четверки чисел (128 байт).

Список параметров.

1. **PointerB.** Указатель на начальный B-блок.
2. **PointerBuffer.** Указатель на транзитный буфер.
3. **Xb.**
   Размер одной строки матрицы B в байтах. Это значение прибавляется к указателю для перехода на следующую строку, поэтому должно быть выровненным (кратным 32), аналогично указателям.
4. **Wb.** Количество атомов в полосочке.
5. **Hb.** Количество полосочек в B-блоке.

```
// Перепаковка матрицы B
PROCEDURE PackBSmall
        ( PointerB       : P_Float64;    // Указатель на матрицу B, выровнен на 32
          PointerBuffer  : P_Float64;    // Указатель на буфер, выровнен на 32
          Xb             : Uint32;       // Байт на одну полную строку матрицы B
          Wb             : Uint32;       // Атомов в полосочке
          Hb             : Uint32);      // Полосочек в B-блоке
```

11. Восстановление входной матрицы A (вариант для обработки маленьких матриц).

- Функция восстанавливает один A-блок. Предполагается, что процедура верхнего уровня, используя данную функцию, восстанавливает серию A-блоков. Распаковка осуществляется в транзитный буфер. (Описание и иллюстрации в документах MxM_BroadCast_перепаковка.DOC, Mutr_Mult_V2_algorithm.GIF, данная операция обратная по отношению к операции, показанной на диаграмме Перепаковка_2_A.GIF).
- Все указатели 64 битные, выровненные на 32 байта (5 младших бит указателей должны быть нулевые).
- Все счетчики - 32 битные целые значения.
- A-атом содержит 2 четверки чисел (64 байта).

Список параметров.

1. **PointerA.** Указатель на начальный A-блок.
2. **PointerBuffer.** Указатель на транзитный буфер.
3. **Xa.**
   Размер одной строки матрицы A в байтах. Это значение прибавляется к указателю для перехода на следующую строку, поэтому должно быть выровненным (кратным 32), аналогично указателям.
4. **Hb.** Количество атомов в полосочке.
5. **Ha.** Количество полосочек в A-блоке.

```
// Восстановление (распаковка) матрицы A
PROCEDURE UnPackASmall
        ( PointerA       : P_Float64;    // Указатель на матрицу A, выровнен на 32
          PointerBuffer  : P_Float64;    // Указатель на буфер, выровнен на 32
          Xa             : Uint32;       // Байт на одну полную строку матрицы A
          Hb             : Uint32;       // Атомов в полосочке
          Ha             : Uint32);      // Полосочек в A-блоке
```

12. Восстановление входной матрицы B (вариант для обработки маленьких матриц).

- Функция обрабатывает один B-блок. Предполагается, что процедура верхнего уровня, используя данную функцию, обрабатывает серию B-блоков. Распаковка осуществляется в транзитный буфер. (Описание и иллюстрации в документах MxM_BroadCast_перепаковка.DOC, Mutr_Mult_V2_algorithm.GIF, данная операция обратная по отношению к операции, показанной на диаграмме Перепаковка_2_B.GIF).
- Все указатели 64 битные, выровненные на 32 байта (5 младших бит указателей должны быть нулевые).
- Все счетчики - 32 битные целые значения.
- B-атом содержит 4 четверки чисел (128 байт).

Список параметров.

1. **PointerB.** Указатель на начальный B-блок.
2. **PointerBuffer.** Указатель на транзитный буфер.
3. **Xb.**
   Размер одной строки матрицы B в байтах. Это значение прибавляется к указателю для перехода на следующую строку, поэтому должно быть выровненным (кратным 32), аналогично указателям.
4. **Wb.** Количество атомов в полосочке.
5. **Hb.** Количество полосочек в B-блоке.

```
// Восстановление (распаковка) матрицы B
PROCEDURE UnPackBSmall
        ( PointerB       : P_Float64;    // Указатель на матрицу B, выровнен на 32
          PointerBuffer  : P_Float64;    // Указатель на буфер, выровнен на 32
          Xb             : Uint32;       // Байт на одну полную строку матрицы B
          Wb             : Uint32;       // Атомов в полосочке
          Hb             : Uint32);      // Полосочек в B-блоке
```

13. Восстановление выходной матрицы C (вариант для обработки маленьких матриц).

Поскольку выходная матрица C восстанавливается (распаковывается) «поэтажно», алгоритмически данная функция полностью соответствует распаковке входной матрицы A с отличием в ширине полосочки, задаваемой входным параметром. Для выполнения операции может быть использована функция UnPackASmall.

Описание и иллюстрации в документах MxM_BroadCast_перепаковка.DOC, Mutr_Mult_V2_algorithm.GIF, Перепаковка_2_A.GIF.

Приложение 1. Примечания по совместимости.

Библиотека требует наличия процессора с поддержкой набора инструкций AVX. Для сохранения и восстановления 256-битного контекста требуется поддержка AVX со стороны операционной системы.

Перед использованием вычислительных функций, приложению верхнего уровня необходимо убедиться в поддержке AVX, используя соответствующие функции библиотеки *AVX_OS.DLL*. Выбор оптимальных размеров обрабатываемых блоков, приложение верхнего уровня может осуществлять на основе параметров, возвращаемых функциями библиотеки *AVX_OS.DLL*.

При использовании виртуализации, виртуальная машина может поддерживать не все технологии, поддерживаемые физической машиной. Теоретически возможна и обратная ситуация - виртуальная машина может

эмулировать наборы инструкций, не поддерживаемые физической машиной. Очевидно, во втором случае производительность будет низкой.

Приложение 2. Примечания по интерфейсу.

Для вычислительных функций (но не для функций проверки системной конфигурации в соотв. библиотеках), все указатели должны быть выровнены на длину векторного регистра. Длина массивов указывается в единицах измерения, равных количеству чисел двойной точности в одном векторном регистре. Например, для AVX256 это означает кратность адресов 32 байтам и единицу измерения длины блоков - 4 числа двойной точности.

Для передачи параметров используется *Microsoft x64 calling convention.*

Целочисленные 64-битные параметры передаются следующим образом:

1 RCX
2 RDX
3 R8
4 R9
5 stack [RSP+40]
6 stack [RSP+48]
7 stack [RSP+56]
и так далее, используется стек

Числа с плавающей точкой передаются следующим образом (по одному числу в младших битах каждого регистра):

1 XMM0
2 XMM1
3 XMM2
4 XMM3
5 stack [RSP+40]
6 stack [RSP+48]
7 stack [RSP+56]
и так далее, используется стек

Для регистров процессора действует следующее правило. Если, например первый параметр целочисленный, а второй - число с плавающей точкой, то первый параметр передается в RCX, второй в XMM1. При этом XMM0 пропускается. Если, например первый параметр - число с плавающей точкой, второй параметр целочисленный, то первый параметр передается в XMM0, второй в RDX. При этом RCX пропускается.

Для параметров (1-4) в стеке всегда резервируется пространство (parameters shadow), несмотря на то, что данные параметры передаются через регистры, а не через стек.

Адреса параметров в стеке [RSP+40], [RSP+48], [RSP+56] вычислены с учетом того факта, что подпрограмма осуществляет доступ к параметрам после того, как вызов этой подпрограммы с сохранением в стеке 64-битного указателя инструкций *RIP* вызвал уменьшение указателя стека *RSP* на 8.

Код статуса или *return code* (если используется) возвращается в регистре RAX.

Четыре параметра с плавающей точкой возвращаются в регистрах XMM0-XMM3 (по одному числу в младших битах каждого регистра).

Функция имеет право не сохранять следующие регистры:
RAX, RCX, RDX, R8-R11, XMM0-XMM5.

Функция обязана сохранять следующие регистры:
RBX, RBP, RDI, RSI, R12-R15, XMM6-XMM15
Старшие 128-битные половины всех 256-битных регистров YMM0-YMM15 функция имеет право не сохранять. "Хорошим тоном" считается их обнуление инструкцией *VZEROUPPER* для минимизации потери производительности при переключении от AVX к SSE коду.

Значение регистра указателя стека (RSP) должно быть выровнено (кратно 16) на момент вызова подпрограммы, реализующей функцию, до выполнения инструкции *call*.