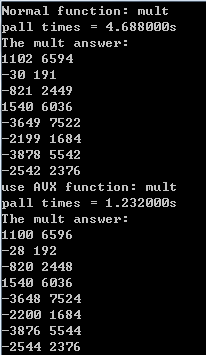
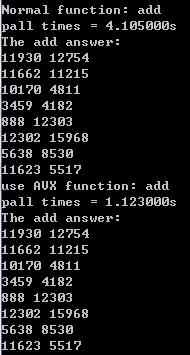
## 向量指令集测试：

乘法，使用\_\_m256i，总共一亿六千万次处理，在i3处理器VS2013下面测试结果：

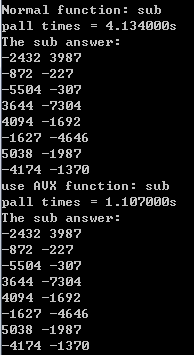


速度提升约3.8倍

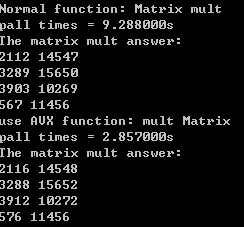
加法测试结果，速度提升3.7倍：



减法测试结果，速度提升3.7倍：



加上矩阵乘以向量的测试，这里是4维的，测试结果如下：



## 整形操作

### 1、8位整数相加

\_\_m256i \_mm256\_add\_epi8 (\_\_m256i a, \_\_m256i b)  
#include "immintrin.h"  
Instruction: vpaddb ymm, ymm, ymm  
CPUID Flags: AVX2

#### 1.1、Description

Add packed 8-bit integers in a and b, and store the results in dst.

#### 1.2、Operation

FOR j := 0 to 31

i := j\*8

dst[i+7:i] := a[i+7:i] + b[i+7:i]

ENDFOR

dst[MAX:256] := 0

### 2、16位整数相加

\_\_m256i \_mm256\_add\_epi16 (\_\_m256i a, \_\_m256i b)  
#include "immintrin.h"  
Instruction: vpaddw ymm, ymm, ymm  
CPUID Flags: AVX2

#### 2.1、Description

Add packed 16-bit integers in a and b, and store the results in dst.

#### 2.2、Operation

FOR j := 0 to 15

i := j\*16

dst[i+15:i] := a[i+15:i] + b[i+15:i]

ENDFOR

dst[MAX:256] := 0

### 3、32位整数相加

\_\_m256i \_mm256\_add\_epi32 (\_\_m256i a, \_\_m256i b)  
#include "immintrin.h"  
Instruction: vpaddd ymm, ymm, ymm  
CPUID Flags: AVX2

#### 3.1、Description

Add packed 32-bit integers in a and b, and store the results in dst.

#### 3.2、Operation

FOR j := 0 to 7

i := j\*32

dst[i+31:i] := a[i+31:i] + b[i+31:i]

ENDFOR

dst[MAX:256] := 0

### 4、64位整数相加

\_\_m256i \_mm256\_add\_epi64 (\_\_m256i a, \_\_m256i b)  
#include "immintrin.h"  
Instruction: vpaddq ymm, ymm, ymm  
CPUID Flags: AVX2

#### 4.1、Description

Add packed 64-bit integers in a and b, and store the results in dst.

#### 4.2、Operation

FOR j := 0 to 3

i := j\*64

dst[i+63:i] := a[i+63:i] + b[i+63:i]

ENDFOR

dst[MAX:256] := 0

### 5、8位整数相减

\_\_m256i \_mm256\_sub\_epi8 (\_\_m256i a, \_\_m256i b)  
#include "immintrin.h"  
Instruction: vpsubb ymm, ymm, ymm  
CPUID Flags: AVX2

#### 5.1、Description

Subtract packed 8-bit integers in b from packed 8-bit integers in a, and store the results in dst.

#### 5.2、Operation

FOR j := 0 to 31

i := j\*8

dst[i+7:i] := a[i+7:i] - b[i+7:i]

ENDFOR

dst[MAX:256] := 0

### 6、16位整数相减

\_\_m256i \_mm256\_sub\_epi16 (\_\_m256i a, \_\_m256i b)  
#include "immintrin.h"  
Instruction: vpsubw ymm, ymm, ymm  
CPUID Flags: AVX2

#### 6.1、Description

Subtract packed 16-bit integers in b from packed 16-bit integers in a, and store the results in dst.

#### 6.2、Operation

FOR j := 0 to 15

i := j\*16

dst[i+15:i] := a[i+15:i] - b[i+15:i]

ENDFOR

dst[MAX:256] := 0

### 7、32位整数相减

\_\_m256i \_mm256\_sub\_epi32 (\_\_m256i a, \_\_m256i b)  
#include "immintrin.h"  
Instruction: vpsubd ymm, ymm, ymm  
CPUID Flags: AVX2

#### 7.1、Description

Subtract packed 32-bit integers in b from packed 32-bit integers in a, and store the results in dst.

#### 7.2、Operation

FOR j := 0 to 7

i := j\*32

dst[i+31:i] := a[i+31:i] - b[i+31:i]

ENDFOR

dst[MAX:256] := 0

### 8、64位整数相减

\_\_m256i \_mm256\_sub\_epi64 (\_\_m256i a, \_\_m256i b)  
#include "immintrin.h"  
Instruction: vpsubq ymm, ymm, ymm  
CPUID Flags: AVX2

#### 8.1、Description

Subtract packed 64-bit integers in b from packed 64-bit integers in a, and store the results in dst.

#### 8.2、Operation

FOR j := 0 to 3

i := j\*64

dst[i+63:i] := a[i+63:i] - b[i+63:i]

ENDFOR

dst[MAX:256] := 0

### 9、32位整数乘法

\_\_m256i \_mm256\_mul\_epi32 (\_\_m256i a, \_\_m256i b)  
#include "immintrin.h"  
Instruction: vpmuldq ymm, ymm, ymm  
CPUID Flags: AVX2

#### 9.1、Description

Multiply the low 32-bit integers from each packed 64-bit element in a and b, and store the signed 64-bit results in dst.

#### 9.2、Operation

FOR j := 0 to 3

i := j\*64

dst[i+63:i] := a[i+31:i] \* b[i+31:i]

ENDFOR

dst[MAX:256] := 0

### 10、16位整数相乘取高位

\_\_m256i \_mm256\_mulhi\_epi16 (\_\_m256i a, \_\_m256i b)  
#include "immintrin.h"  
Instruction: vpmulhw ymm, ymm, ymm  
CPUID Flags: AVX2

#### 10.1、Description

Multiply the packed 16-bit integers in a and b, producing intermediate 32-bit integers, and store the high 16 bits of the intermediate integers in dst.

#### 10.2、Operation

FOR j := 0 to 15

i := j\*16

tmp[31:0] := a[i+15:i] \* b[i+15:i]

dst[i+15:i] := tmp[31:16]

ENDFOR

dst[MAX:256] := 0

### 11、16位整数相乘移位后取低

\_\_m256i \_mm256\_mulhrs\_epi16 (\_\_m256i a, \_\_m256i b)  
#include "immintrin.h"  
Instruction: vpmulhrsw ymm, ymm, ymm  
CPUID Flags: AVX2

#### 11.1、Description

Multiply packed 16-bit integers in a and b, producing intermediate signed 32-bit integers. Truncate each intermediate integer to the 18 most significant bits, round by adding 1, and store bits [16:1] to dst.

#### 11.2、Operation

FOR j := 0 to 15

i := j\*16

tmp[31:0] := ((a[i+15:i] \* b[i+15:i]) >> 14) + 1

dst[i+15:i] := tmp[16:1]

ENDFOR

dst[MAX:256] := 0

### 12、16位整数相乘取低

\_\_m256i \_mm256\_mullo\_epi16 (\_\_m256i a, \_\_m256i b)  
#include "immintrin.h"  
Instruction: vpmullw ymm, ymm, ymm  
CPUID Flags: AVX2

#### 12.1、Description

Multiply the packed 16-bit integers in a and b, producing intermediate 32-bit integers, and store the low 16 bits of the intermediate integers in dst.

#### 12.2、Operation

FOR j := 0 to 15

i := j\*16

tmp[31:0] := a[i+15:i] \* b[i+15:i]

dst[i+15:i] := tmp[15:0]

ENDFOR

dst[MAX:256] := 0

### 12、32位整数相乘取低

\_\_m256i \_mm256\_mullo\_epi32 (\_\_m256i a, \_\_m256i b)  
#include "immintrin.h"  
Instruction: vpmulld ymm, ymm, ymm  
CPUID Flags: AVX2

#### 12.1、Description

Multiply the packed 32-bit integers in a and b, producing intermediate 64-bit integers, and store the low 32 bits of the intermediate integers in dst.

#### 12.2、Operation

FOR j := 0 to 7

i := j\*32

tmp[63:0] := a[i+31:i] \* b[i+31:i]

dst[i+31:i] := tmp[31:0]

ENDFOR

dst[MAX:256] := 0

### 13、逻辑与

\_\_m256i \_mm256\_and\_si256 (\_\_m256i a, \_\_m256i b)  
#include "immintrin.h"  
Instruction: vpand ymm, ymm, ymm  
CPUID Flags: AVX2

#### 13.1、Description

Compute the bitwise AND of 256 bits (representing integer data) in a and b, and store the result in dst.

#### 13.2、Operation

dst[255:0] := (a[255:0] AND b[255:0])

dst[MAX:256] := 0

### 14、逻辑异或

\_\_m256i \_mm256\_xor\_si256 (\_\_m256i a, \_\_m256i b)  
#include "immintrin.h"  
Instruction: vpxor ymm, ymm, ymm  
CPUID Flags: AVX2

#### 14.1、Description

Compute the bitwise XOR of 256 bits (representing integer data) in a and b, and store the result in dst.

#### 14.2、Operation

dst[255:0] := (a[255:0] XOR b[255:0])

dst[MAX:256] := 0

## 浮点类型

### 1、64位双精度浮点加法

\_\_m256d \_mm256\_add\_pd (\_\_m256d a, \_\_m256d b)  
#include "immintrin.h"  
Instruction: vaddpd ymm, ymm, ymm  
CPUID Flags: AVX

#### 1.1、Description

Add packed double-precision (64-bit) floating-point elements in a and b, and store the results in dst.

#### 1.2、Operation

FOR j := 0 to 3

i := j\*64

dst[i+63:i] := a[i+63:i] + b[i+63:i]

ENDFOR

dst[MAX:256] := 0

### 2、32位单精度浮点加法

\_\_m256 \_mm256\_add\_ps (\_\_m256 a, \_\_m256 b)  
#include "immintrin.h"  
Instruction: vaddps ymm, ymm, ymm  
CPUID Flags: AVX

#### 2.1、Description

Add packed single-precision (32-bit) floating-point elements in a and b, and store the results in dst.

#### 2.2、Operation

FOR j := 0 to 7

i := j\*32

dst[i+31:i] := a[i+31:i] + b[i+31:i]

ENDFOR

dst[MAX:256] := 0

### 3、64位双精度浮点减法

\_\_m256d \_mm256\_sub\_pd (\_\_m256d a, \_\_m256d b)  
#include "immintrin.h"  
Instruction: vsubpd ymm, ymm, ymm  
CPUID Flags: AVX

#### 3.1、Description

Subtract packed double-precision (64-bit) floating-point elements in b from packed double-precision (64-bit) floating-point elements in a, and store the results in dst.

#### 3.2、Operation

FOR j := 0 to 3

i := j\*64

dst[i+63:i] := a[i+63:i] - b[i+63:i]

ENDFOR

dst[MAX:256] := 0

### 4、32位单精度浮点减法

\_\_m256 \_mm256\_sub\_ps (\_\_m256 a, \_\_m256 b)  
#include "immintrin.h"  
Instruction: vsubps ymm, ymm, ymm  
CPUID Flags: AVX

#### 4.1、Description

Subtract packed single-precision (32-bit) floating-point elements in b from packed single-precision (32-bit) floating-point elements in a, and store the results in dst.

#### 4.2、Operation

FOR j := 0 to 7

i := j\*32

dst[i+31:i] := a[i+31:i] - b[i+31:i]

ENDFOR

dst[MAX:256] := 0

### 5、64位双精度浮点乘法

\_\_m256d \_mm256\_mul\_pd (\_\_m256d a, \_\_m256d b)  
#include "immintrin.h"  
Instruction: vmulpd ymm, ymm, ymm  
CPUID Flags: AVX

#### 5．1、Description

Multiply packed double-precision (64-bit) floating-point elements in a and b, and store the results in dst.

#### 5.2、Operation

FOR j := 0 to 3

i := j\*64

dst[i+63:i] := a[i+63:i] \* b[i+63:i]

ENDFOR

dst[MAX:256] := 0

### 6、32位单精度浮点乘法

\_\_m256 \_mm256\_mul\_ps (\_\_m256 a, \_\_m256 b)  
#include "immintrin.h"  
Instruction: vmulps ymm, ymm, ymm  
CPUID Flags: AVX

#### 6.1、Description

Multiply packed single-precision (32-bit) floating-point elements in a and b, and store the results in dst.

#### 6.2、Operation

FOR j := 0 to 7

i := j\*32

dst[i+31:i] := a[i+31:i] \* b[i+31:i]

ENDFOR

dst[MAX:256] := 0

### 7、64位双精度浮点除法

\_\_m256d \_mm256\_div\_pd (\_\_m256d a, \_\_m256d b)  
#include "immintrin.h"  
Instruction: vdivpd ymm, ymm, ymm  
CPUID Flags: AVX

#### 7.1、Description

Divide packed double-precision (64-bit) floating-point elements in a by packed elements in b, and store the results in dst.

#### 7.2、Operation

FOR j := 0 to 3

i := 64\*j

dst[i+63:i] := a[i+63:i] / b[i+63:i]

ENDFOR

dst[MAX:256] := 0

### 8、32位单精度浮点除法

\_\_m256 \_mm256\_div\_ps (\_\_m256 a, \_\_m256 b)  
#include "immintrin.h"  
Instruction: vdivps ymm, ymm, ymm  
CPUID Flags: AVX

#### 8.1、Description

Divide packed single-precision (32-bit) floating-point elements in a by packed elements in b, and store the results in dst.

#### 8.2、Operation

FOR j := 0 to 7

i := 32\*j

dst[i+31:i] := a[i+31:i] / b[i+31:i]

ENDFOR

dst[MAX:256] := 0

## 512位接口

### 1、load

\_\_m512i \_mm512\_load\_si512 (void const\* mem\_addr)  
#include "immintrin.h"  
Instruction: vmovdqa32 zmm {k}, m512  
CPUID Flags: AVX512F for AVX-512, KNCNI for KNC

#### 1.1 Description

Load 512-bits of integer data from memory into dst. mem\_addr must be aligned on a 64-byte boundary or a general-protection exception may be generated.

#### 1.2 Operation

dst[511:0] := MEM[mem\_addr+511:mem\_addr]

dst[MAX:512] := 0

### 2、add

\_\_m512i \_mm512\_add\_epi16 (\_\_m512i a, \_\_m512i b)  
#include "immintrin.h"  
Instruction: vpaddw  
CPUID Flags: AVX512BW

#### 2.1 Description

Add packed 16-bit integers in a and b, and store the results in dst.

#### 2.2 Operation

FOR j := 0 to 31

i := j\*16

dst[i+15:i] := a[i+15:i] + b[i+15:i]

ENDFOR

dst[MAX:512] := 0

### 3、sub

\_\_m512i \_mm512\_sub\_epi16 (\_\_m512i a, \_\_m512i b)  
#include "immintrin.h"  
Instruction: vpsubw  
CPUID Flags: AVX512BW

#### 3.1 Description

Subtract packed 16-bit integers in b from packed 16-bit integers in a, and store the results in dst.

#### 3.2 Operation

FOR j := 0 to 31

i := j\*16

dst[i+15:i] := a[i+15:i] - b[i+15:i]

ENDFOR

dst[MAX:512] := 0

### 4、mult

\_\_m512i \_mm512\_mulhrs\_epi16 (\_\_m512i a, \_\_m512i b)  
#include "immintrin.h"  
Instruction: vpmulhrsw  
CPUID Flags: AVX512BW

#### 4.1 Description

Multiply packed 16-bit integers in a and b, producing intermediate signed 32-bit integers. Truncate each intermediate integer to the 18 most significant bits, round by adding 1, and store bits [16:1] to dst.

#### 4.2 Operation

FOR j := 0 to 31

i := j\*16

tmp[31:0] := ((a[i+15:i] \* b[i+15:i]) >> 14) + 1

dst[i+15:i] := tmp[16:1]

ENDFOR

dst[MAX:512] := 0

### 5、store

void \_mm512\_store\_si512 (void\* mem\_addr, \_\_m512i a)  
#include "immintrin.h"  
Instruction: vmovdqa32 m512 {k}, zmm  
CPUID Flags: AVX512F for AVX-512, KNCNI for KNC

#### 5.1 Description

Store 512-bits of integer data from a into memory. mem\_addr must be aligned on a 64-byte boundary or a general-protection exception may be generated.

#### 5.2 Operation

MEM[mem\_addr+511:mem\_addr] := a[511:0]