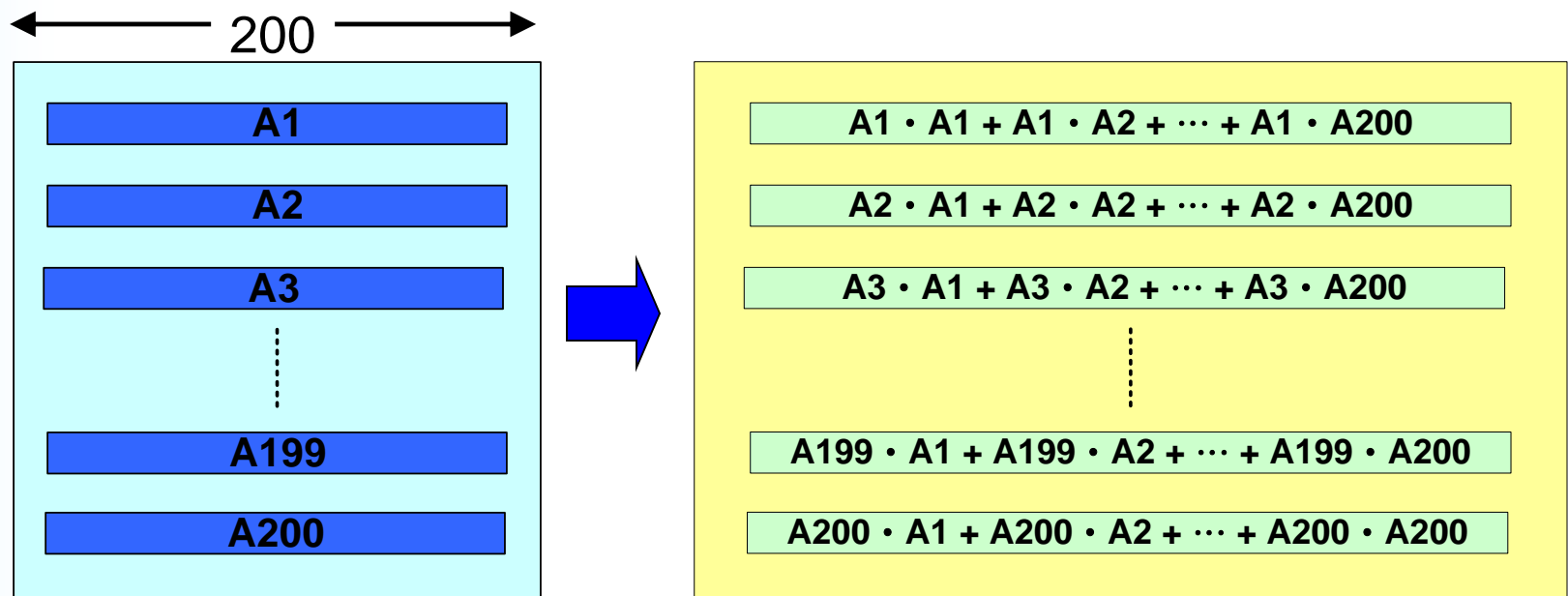


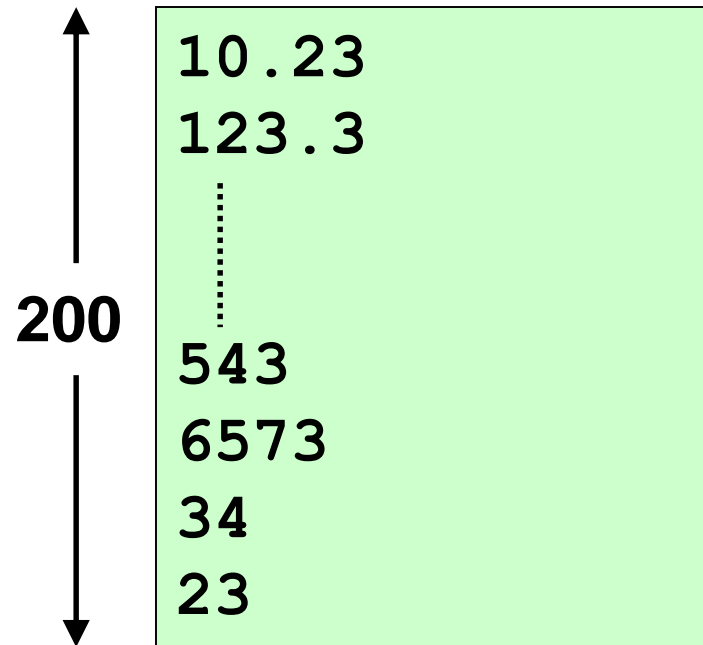
Homework #8 (1)

- Write a C program to perform:
 - Read a text file named “**data.txt**” (200x200 matrix, each element is a **float**)
 - Do following computation



Homework #8 (2)

- Output the result to a file named “output.txt”
- Output file needs to follow the form:



10.23
123.3
...
543
6573
34
23

The diagram illustrates the required output format. A green rectangular box contains a vertical list of numbers. To the left of the box, a vertical double-headed arrow spans the height of the box and is labeled with the number 200, indicating that the output file must contain 200 lines of data. The numbers inside the box are: 10.23, 123.3, followed by a vertical ellipsis (three dots) to represent intermediate lines, then 543, 6573, 34, and 23.

Homework #8 (3)

1	2	3	4
4	1	3	1
8	9	2	1
5	6	7	8

30

$$1 \times 1 + 2 \times 2 + 3 \times 3 + 4 \times 4 = 1 + 4 + 9 + 16 = 30$$

Homework #8 (3)

1	2	3	4
4	1	3	1
8	9	2	1
5	6	7	8

30 + 19

$$1 \times 4 + 2 \times 1 + 3 \times 3 + 4 \times 1 = 4 + 2 + 9 + 4 = 19$$

Homework #9 (3)

1	2	3	4
4	1	3	1
8	9	2	1
5	6	7	8

$$30 + 19 + 36$$

$$1 \times 8 + 2 \times 9 + 3 \times 2 + 4 \times 1 = 8 + 18 + 6 + 4 = 36$$

Homework #8 (3)

1	2	3	4
4	1	3	1
8	9	2	1
5	6	7	8

$$30 + 19 + 36 + 70$$

$$1 \times 5 + 2 \times 6 + 3 \times 7 + 4 \times 8 = 5 + 12 + 21 + 32 = 70$$

Homework #8 (3)

1	2	3	4
4	1	3	1
8	9	2	1
5	6	7	8

$$30 + 19 + 36 + 70$$

155
?
?
?

Homework #8 (3)

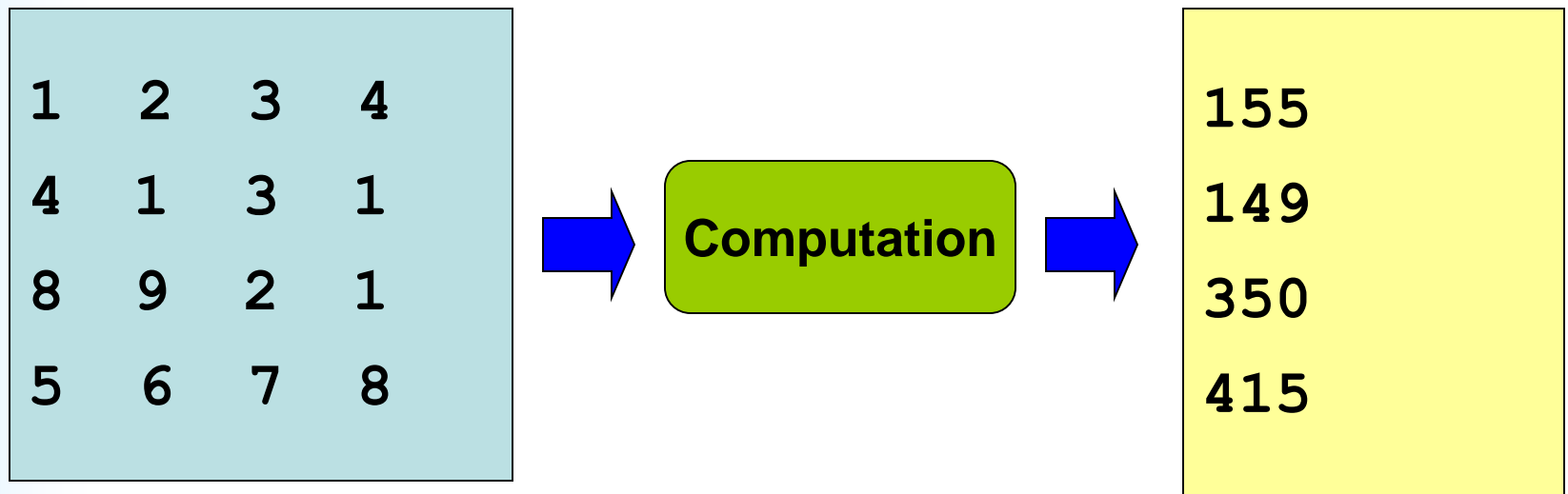
1	2	3	4
4	1	3	1
8	9	2	1
5	6	7	8

19

155
?
?
?

$$4 \times 1 + 1 \times 2 + 3 \times 3 + 1 \times 4 = 4 + 2 + 9 + 4 = 19$$

Homework #8 (3)



Homework #8 (4)

- 輸入檔檔名 **data.txt** (自行產生)
 - 包含200 row的資料
 - 每一row有200個floating point數字，數字與數字間用一個空白隔開
- 輸出檔檔名 **output.txt** (計算結果)
 - 包含200 row的資料

Example

```
#include <mmintrin.h>
#include <xmmintrin.h>
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
float A[4] __attribute__((aligned(16)));
float B[4] __attribute__((aligned(16)));
float C[4] __attribute__((aligned(16)));
__m128 *a, *b, *c;
```

```
a = (__m128*) A;
```

```
b = (__m128*) B;
```

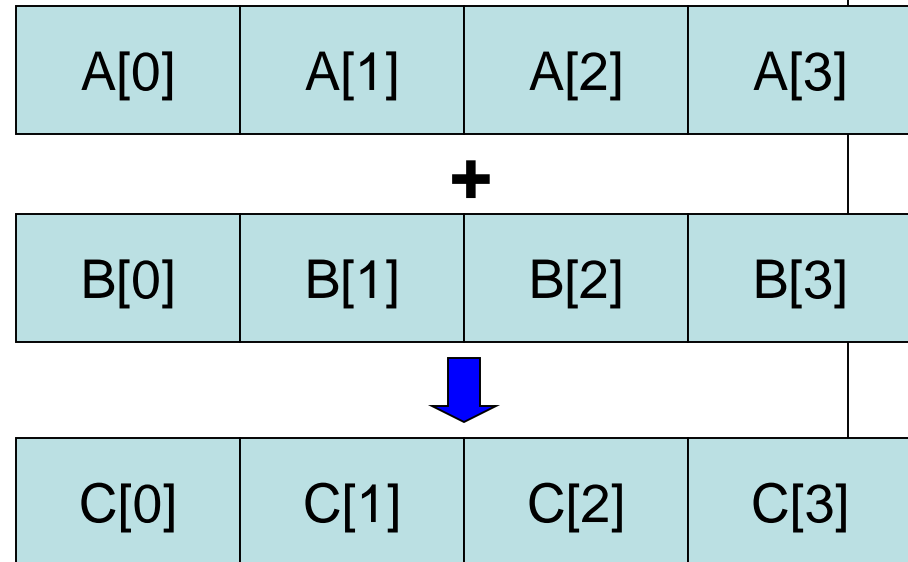
```
c = (__m128*) C;
```

```
*c = _mm_add_ps(*a, *b);
```

```
printf("%f %f %f %f\n", C[0], C[1], C[2], C[3]);
```

```
return 0;
```

```
}
```



GCC Options

- These **switches** enable or disable **the use of built-in functions** that allow direct access to the MMX, SSE, SSE2, SSE3, SSE4, AVX, and 3Dnow extensions of the instruction set
 - -mavx
 - -msse
 - -msse2
 - -msse3
 - -msse4
 - -m3dnow

```
gcc -msse4 test.c
```

Intrinsic Functions

- 你可以使用 SSE, SSE2, SSE3, SSE4 相關的 intrinsic functions (請參閱 [cref_cls.pdf](#) 文件裡 page 88, 124, 168 裡面的函式說明)
- 或請至下面的網站查詢有哪些 intrinsic functions
- <https://software.intel.com/sites/landingpage/IntrinsicsGuide/#>

Homework #8 (5)

- 使用SIMD intrinsic function來做計算
 - 請使用**GCC 3.4**以上的版本編譯你的程式
 - 主要評分標準：
 - 是否使用大量的**SIMD intrinsic function**？
 - 程式執行速度？
- 在Linux上進行編譯與測試
 - 請多利用系上工作站
- 程式中應有適當的說明（註解）

- 關於作業8的誤差問題: 在這個作業裡，因為浮點數運算順序的緣故，很小的誤差是可以接受的。
- 作業也讓我們意外了解到浮點數的運算與精確度的問題，不同的領域對於精確度的要求各有不同。

- 我建議有幾個方式驗證程式的正確性：
 - 縮小**200x200**矩陣裡每個元素的值，使他們在**1~10**之間，如此，應該可以大量減輕因為幅點數極大、極小值**運算順序不同**造成的誤差，看看結果是否很接近。
 - **2.** 嘗試改成整數的版本，每個元素的值也不要太大，避免**overflow**，如此應該**SIMD version**與一般的版本答案是一樣的。

Homework #8 (6)

- You should turn in to **ECOURSE**
 - “**README.txt**” file: 文字檔，描述你程式的內容、如何編譯程式、如何執行你的程式、**在哪個型號的CPU上執行成功**等等。
 - 一個可以執行成功的input檔案“**data.txt**”與相對應的結果檔案“**output.txt**”
 - A C program without SIMD intrinsic functions: **hw8.c**
 - A C program with SIMD intrinsic functions: **hw8simd.c**
 - Any file needed in your work (ex: Makefile)
- **Deadline: January 12, 24:00, 2019**

這時間之後，將無法補交這次作業