

Today

■ Arrays

- One-dimensional
- Multi-dimensional (nested)
- Multi-level

■ Structures

- Allocation
- Access
- Alignment

■ Floating Point

Background

■ History

- x87 FP
 - Legacy, very ugly
- SSE FP
 - Supported by Shark machines
 - Special case use of vector instructions
- AVX FP
 - Newest version
 - Similar to SSE
 - Documented in book

Programming with SSE3

XMM Registers

■ 16 total, each 16 bytes

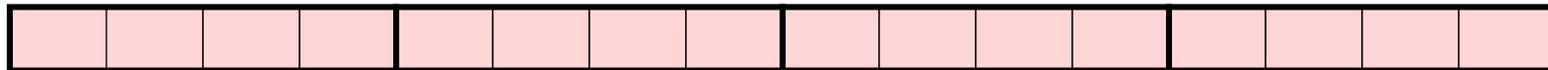
■ 16 single-byte integers



■ 8 16-bit integers



■ 4 32-bit integers



■ 4 single-precision floats



■ 2 double-precision floats



■ 1 single-precision float



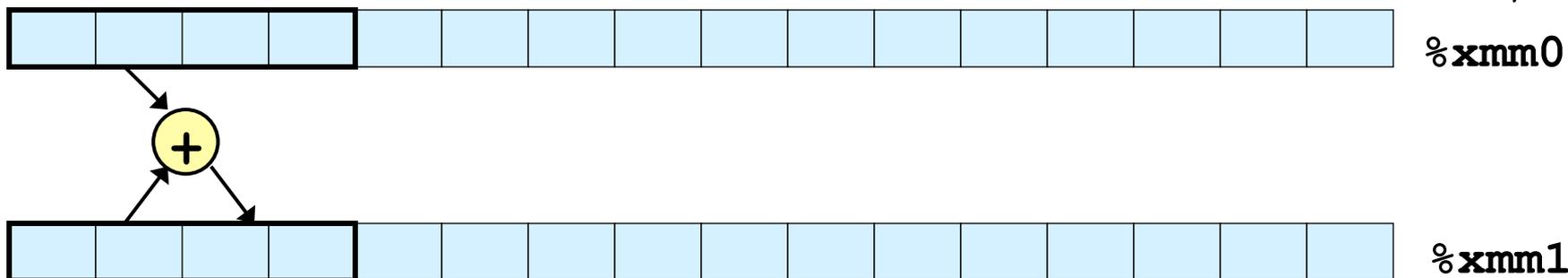
■ 1 double-precision float



Scalar & SIMD Operations

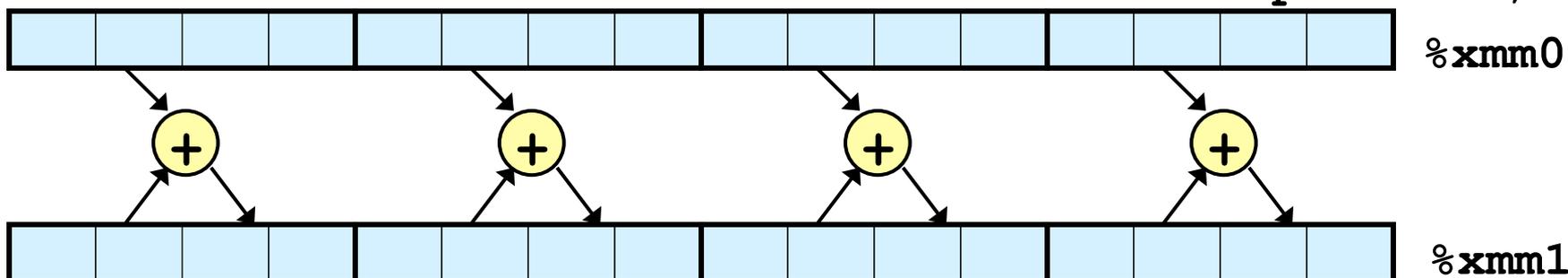
■ Scalar Operations: Single Precision

`addss %xmm0, %xmm1`



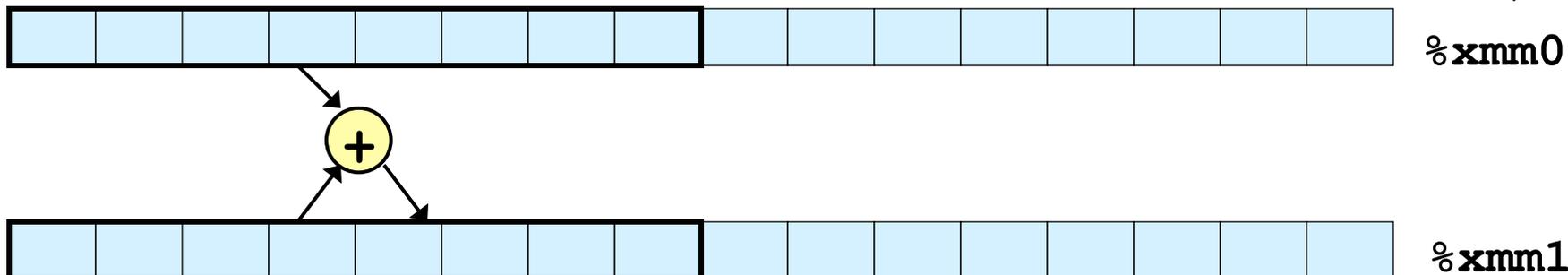
■ SIMD Operations: Single Precision

`addps %xmm0, %xmm1`



■ Scalar Operations: Double Precision

`addsd %xmm0, %xmm1`



FP Basics

- Arguments passed in `%xmm0`, `%xmm1`, ...
- Result returned in `%xmm0`
- All XMM registers caller-saved

```
float fadd(float x, float y)
{
    return x + y;
}
```

```
double dadd(double x, double y)
{
    return x + y;
}
```

```
# x in %xmm0, y in %xmm1
addss    %xmm1, %xmm0
ret
```

```
# x in %xmm0, y in %xmm1
addsd    %xmm1, %xmm0
ret
```

FP Memory Referencing

- Integer (and pointer) arguments passed in regular registers
- FP values passed in XMM registers
- Different mov instructions to move between XMM registers, and between memory and XMM registers

```
double dincr(double *p, double v)
{
    double x = *p;
    *p = x + v;
    return x;
}
```

```
# p in %rdi, v in %xmm0
movapd  %xmm0, %xmm1    # Copy v
movsd   (%rdi), %xmm0   # x = *p
addsd   %xmm0, %xmm1    # t = x + v
movsd   %xmm1, (%rdi)   # *p = t
ret
```

Other Aspects of FP Code

■ *Lots of instructions*

- Different operations, different formats, ...

■ **Floating-point comparisons**

- Instructions `ucomiss` and `ucomisd`
- Set condition codes CF, ZF, and PF

■ **Using constant values**

- Set XMM0 register to 0 with instruction `xorpd %xmm0, %xmm0`
- Others loaded from memory