How to Write Fast Numerical Code

Spring 2012

Lecture 3

Instructor: Markus Püschel

TA: Georg Ofenbeck

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Technicalities

- **Research project: Let me know**
  - if you know with whom you will work
  - if you have already a project idea
  - current status: on the web
  - Deadline: March 7th
- **Finding partner: fastcode-forum@lists.inf.ethz.ch**
  - Recipients: TA Georg + all students that have no partner yet
- **Email for questions: fastcode@lists.inf.ethz.ch**
  - use for all technical questions
  - received by me and the TAs = ensures timely answer

Last Time

- **Asymptotic analysis versus cost analysis**

```
/* Multiply n x n matrices a and b  */
void mmm(double *a, double *b, double *c, int n) {
    int i, j, k;
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            for (k = 0; k < n; k++)
                c[i*n+j] += a[i*n + k]*b[k*n + j];
}
```

Asymptotic runtime: $O(n^3)$
Cost: (adds, mults) = (n^3, n^3)
Cost: flops = $2n^3$

- **Cost analysis enables performance plots**

Today

- Architecture/Microarchitecture
- Crucial microarchitectural parameters
- Peak performance

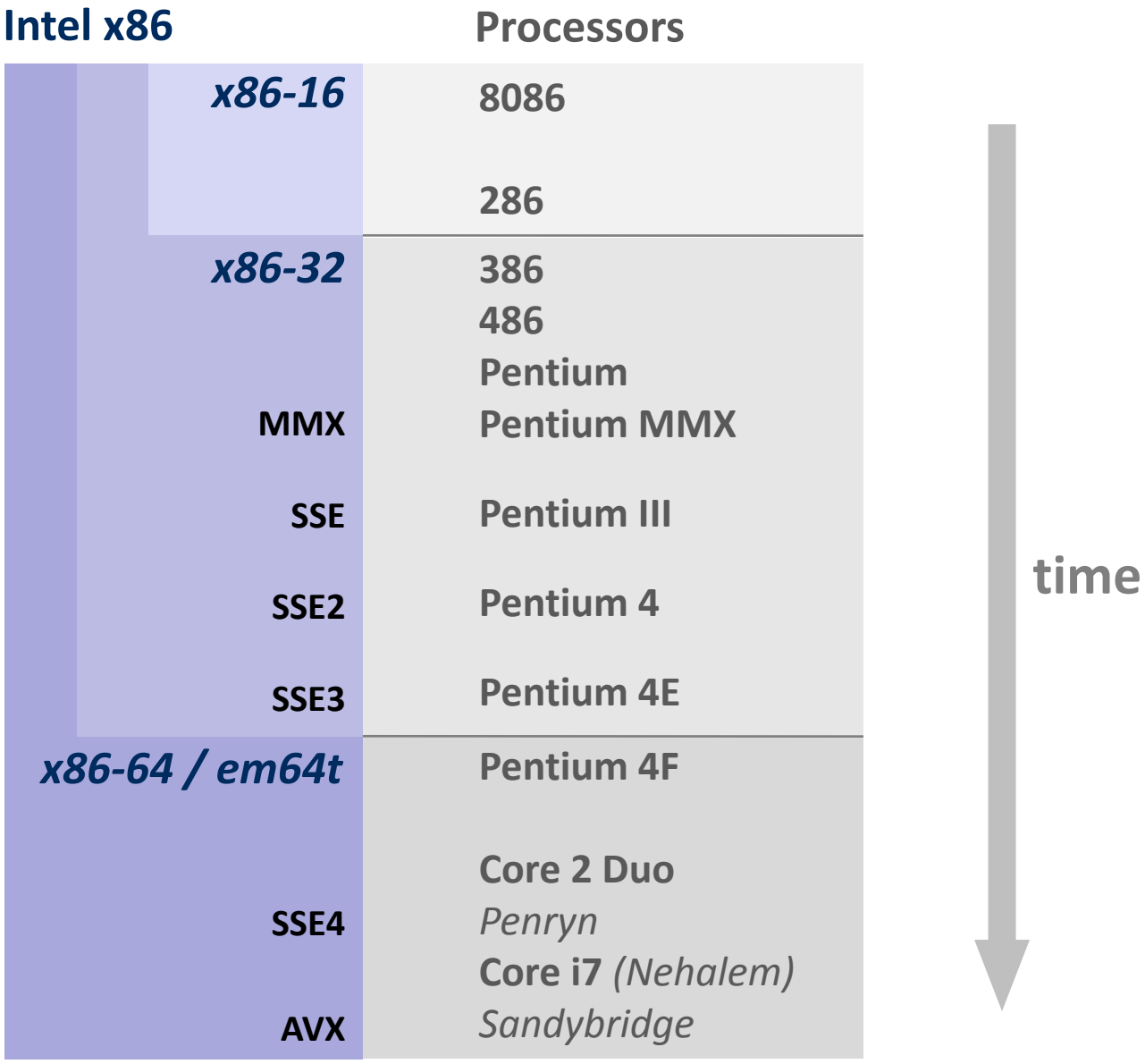
Definitions

- ***Architecture:*** **(also instruction set architecture = ISA) The parts of a processor design that one needs to understand to write assembly code.**
- ***Examples:*** **instruction set specification, registers**
- ***Counterexamples:*** **cache sizes and core frequency**
- **Example ISAs**
  - x86
  - ia
  - MIPS
  - POWER
  - SPARC
  - ARM

MMX:
Multimedia extension

SSE:
Streaming SIMD extension

AVX:
Advanced vector extensions



ISA SIMD (Single Instruction Multiple Data) Vector Extensions

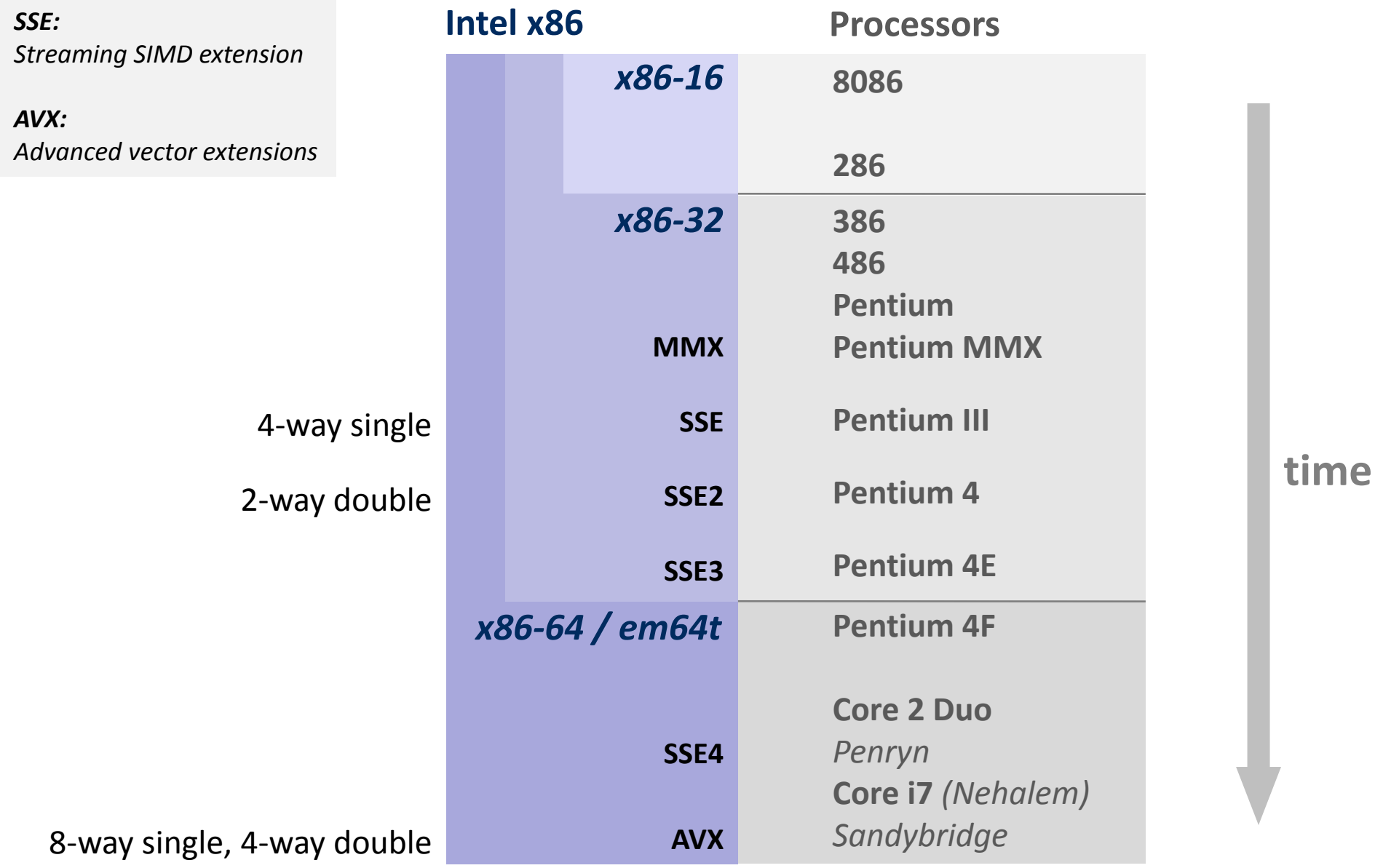
- **What is it?**
  - Extension of the ISA. Data types and instructions for the parallel computation on short (length 2-8) vectors of integers or floats

  - Names: MMX, SSE, SSE2, …
- **Why do they exist?**
  - **Useful:** Many applications have the necessary fine-grain parallelism
    Then: speedup by a factor close to vector length
  - **Doable:** Chip designers have enough transistors to play with
- **We will have an extra lecture on vector instructions**
  - What are the problems?
  - How to use them efficiently

MMX:
Multimedia extension

SSE:
Streaming SIMD extension

AVX:
Advanced vector extensions



Definitions

- ***Microarchitecture:*** **Implementation of the architecture.**

- ***Examples:*** **caches, cache structure, CPU frequency, details of the virtual memory system**

- **Examples**
  - Intel processors (Wikipedia)
  - AMD processors (Wikipedia)

Microarchitecture:
The View of the Computer Architect

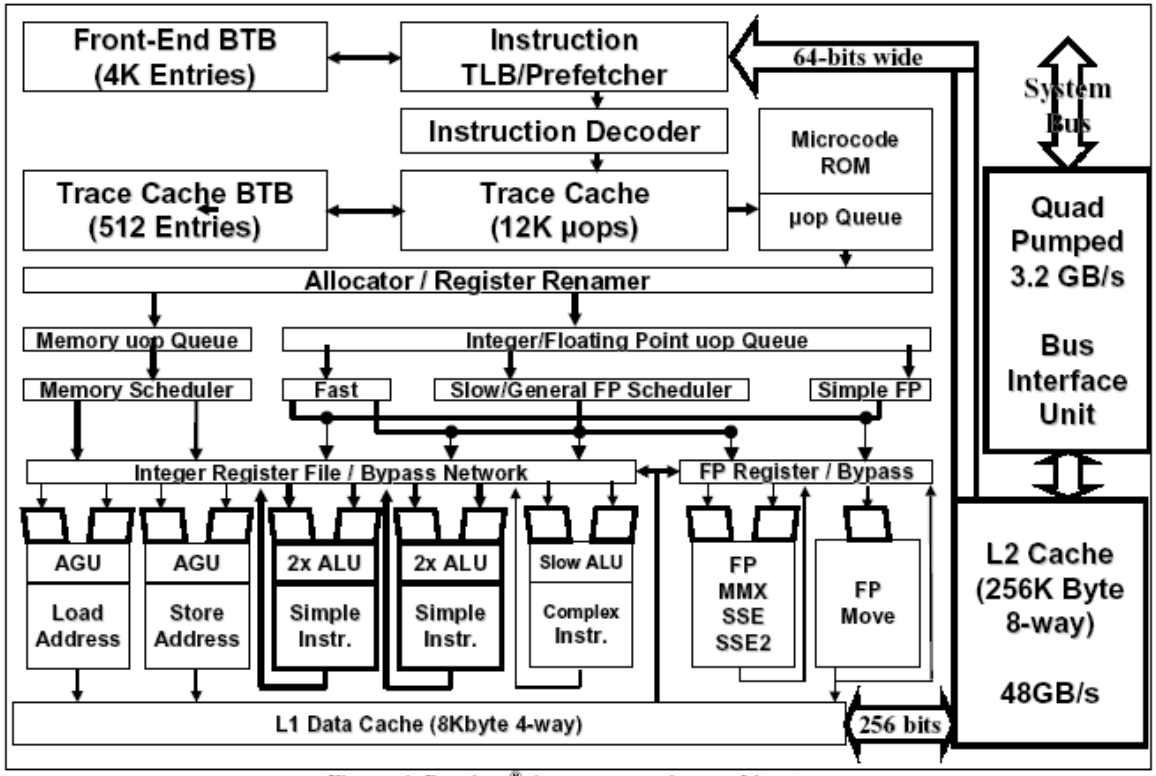


Figure 4: Pentium® 4 processor microarchitecture

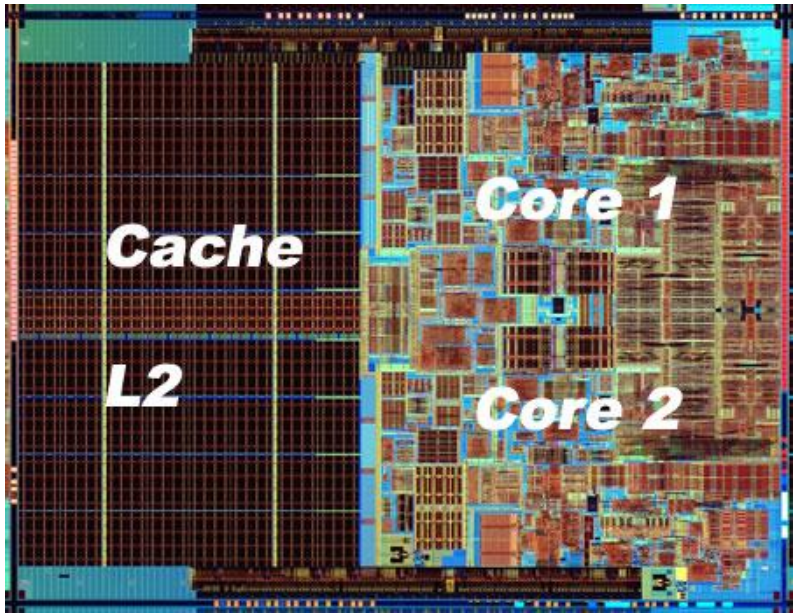
we take the software developers view …

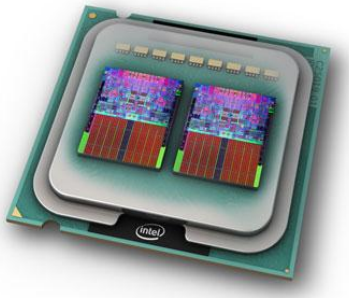
Source: "The Microarchitecture of the Pentium 4 Processor, "
Intel Technology Journal Q1 2001

- **Show microarchitecture slide**
- **Small runtime bound quiz**

Core Processor

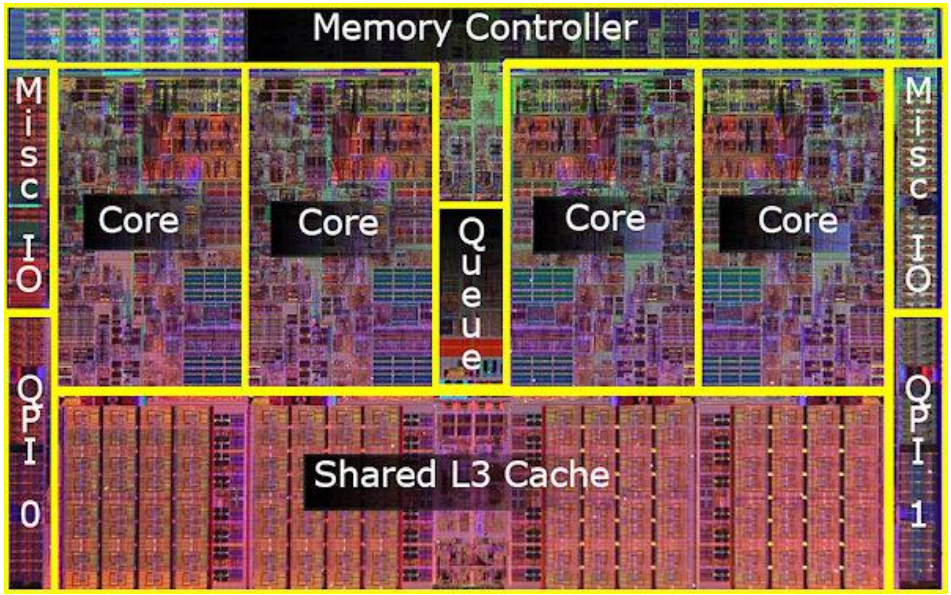
Core 2 Duo





2 x Core 2 Duo packaged

Core i7



Detailed information about Core processors

Pictures: Intel

Core Pipeline

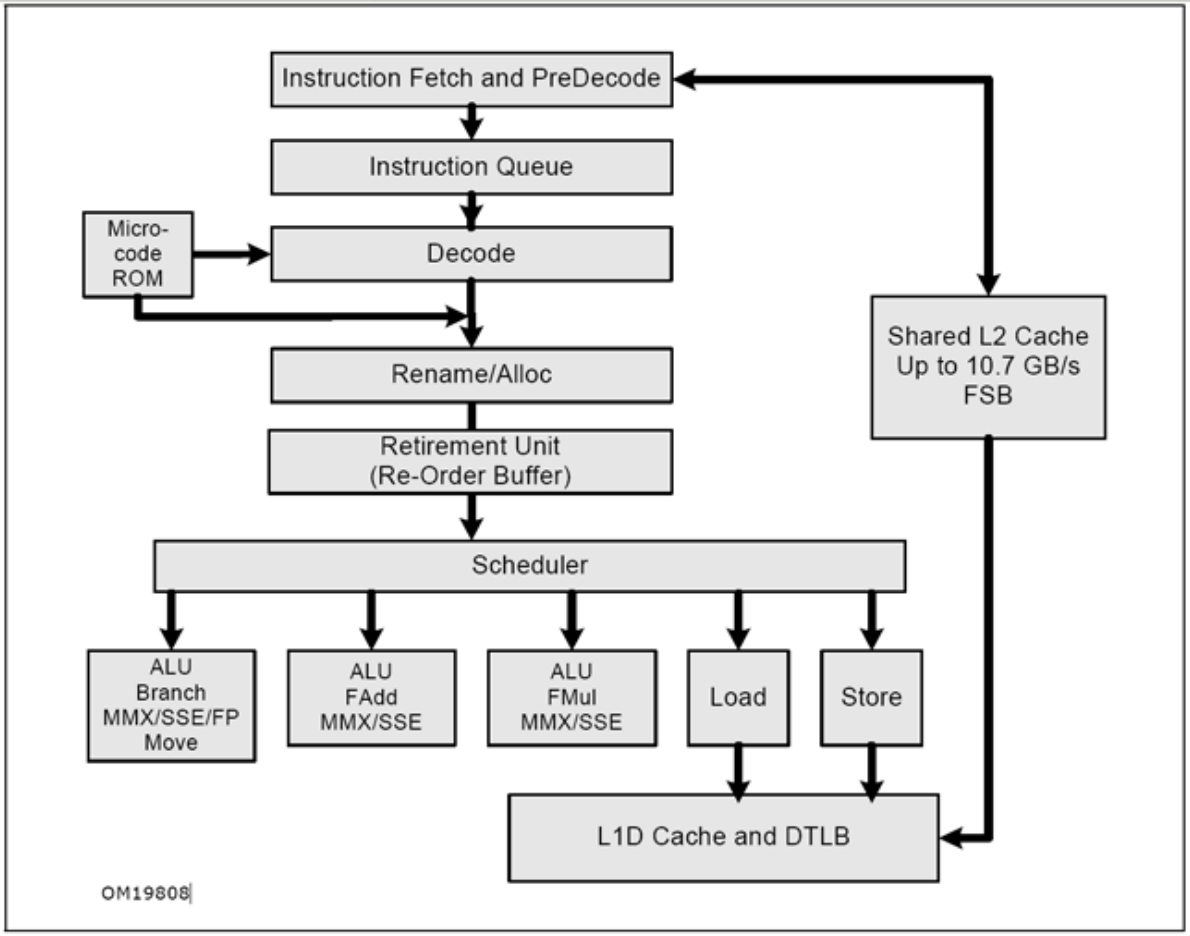


Figure 2-1. Intel Core Microarchitecture Pipeline Functionality

- **Floating point peak performance?**

Floating Point Peak Performance?

Table 2-15. Issue Ports of Intel Core Microarchitecture and Enhanced Intel Core Microarchitecture

Executable operations	Latency, Throughput		Comment[1]
	Signature = 06_0FH	Signature = 06_17H	
Integer ALU Integer SIMD ALU FP/SIMD/SSE2 Move and Logic	1, 1 1, 1 1, 1	1, 1 1, 1 1, 1	Includes 64-bit mode integer MUL; Issue port 0; Writeback port 0;
Single-precision (SP) FP MUL Double-precision FP MUL	4, 1 5, 1	4, 1 5, 1	Issue port 0; Writeback port 0
FP MUL (X87) FP Shuffle DIV/SQRT	5, 2 1, 1	5, 2 1, 1	Issue port 0; Writeback port 0 FP shuffle does not handle QW shuffle.
Integer ALU Integer SIMD ALU FP/SIMD/SSE2 Move and Logic	1, 1 1, 1 1, 1	1, 1 1, 1 1, 1	Excludes 64-bit mode integer MUL; Issue port 1; Writeback port 1;
FP ADD QW Shuffle	3, 1 1, 1[2]	3, 1 1, 1[3]	Issue port 1; Writeback port 1;
Integer loads FP loads	3, 1 4, 1	3, 1 4, 1	Issue port 2; Writeback port 2;
Store address[4]	3, 1	3, 1	Issue port 3;
Store data[5].			Issue Port 4;
Integer ALU Integer SIMD ALU FP/SIMD/SSE2 Move and Logic	1, 1 1, 1 1, 1	1, 1 1, 1 1, 1	Issue port 5; Writeback port 5;
QW shuffles 128-bit Shuffle/Pack/Unpack	1, 1[2] 2-4, 2-4[6]	1, 1[3] 1-3, 1[7]	Issue port 5; Writeback port 5;

Two different processor lines

1 or 2 mults/cycle?

1 add/cycle

The Two Floating Points

```
float ipf (float x[], float y[], int n) {
  int i;
  float result = 0.0;

  for (i = 0; i < n; i++)
    result += x[i]*y[i];
  return result;
}
```

standard compilation (SSE)

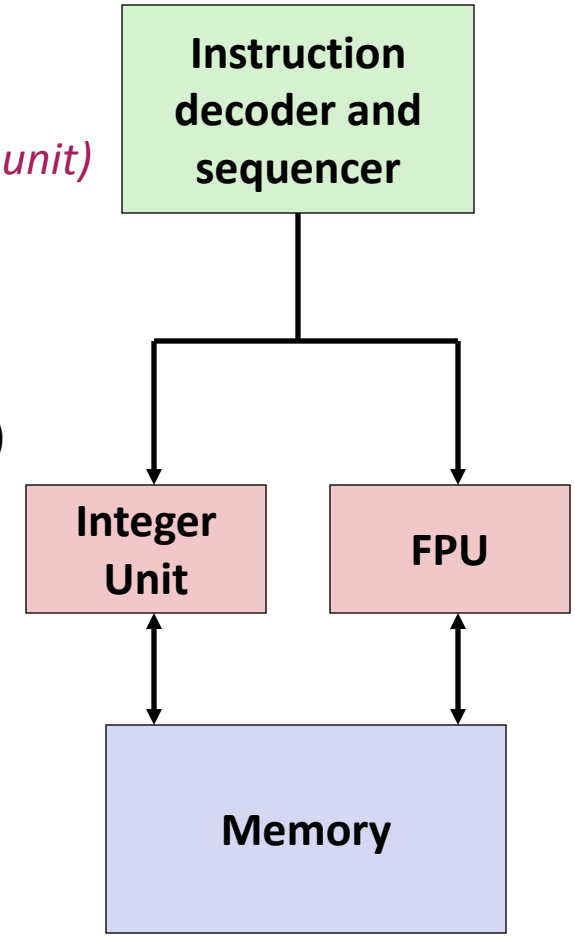
```
ipf:
   xorps   %xmm1, %xmm1
   xorl    %ecx, %ecx
   jmp     .L8
.L10:
   movslq  %ecx,%rax
   incl    %ecx
   movss (%rsi,%rax,4), %xmm0
   mulss (%rdi,%rax,4), %xmm0
   addss   %xmm0, %xmm1
.L8:
   cmpl    %edx, %ecx
   jl      .L10
   movaps  %xmm1, %xmm0
   ret
```

compilation for x87

```
…
cmpl %edx,%eax
   jge .L3
.L5:
   flds (%ebx,%eax,4)
   fmuls (%ecx,%eax,4)
   faddp
   incl %eax
   cmpl %edx,%eax
   jl .L5
.L3:
   movl -4(%ebp),%ebx
   movl %ebp, %esp
   popl %ebp
   ret
```

The Other Floating Point (x87)

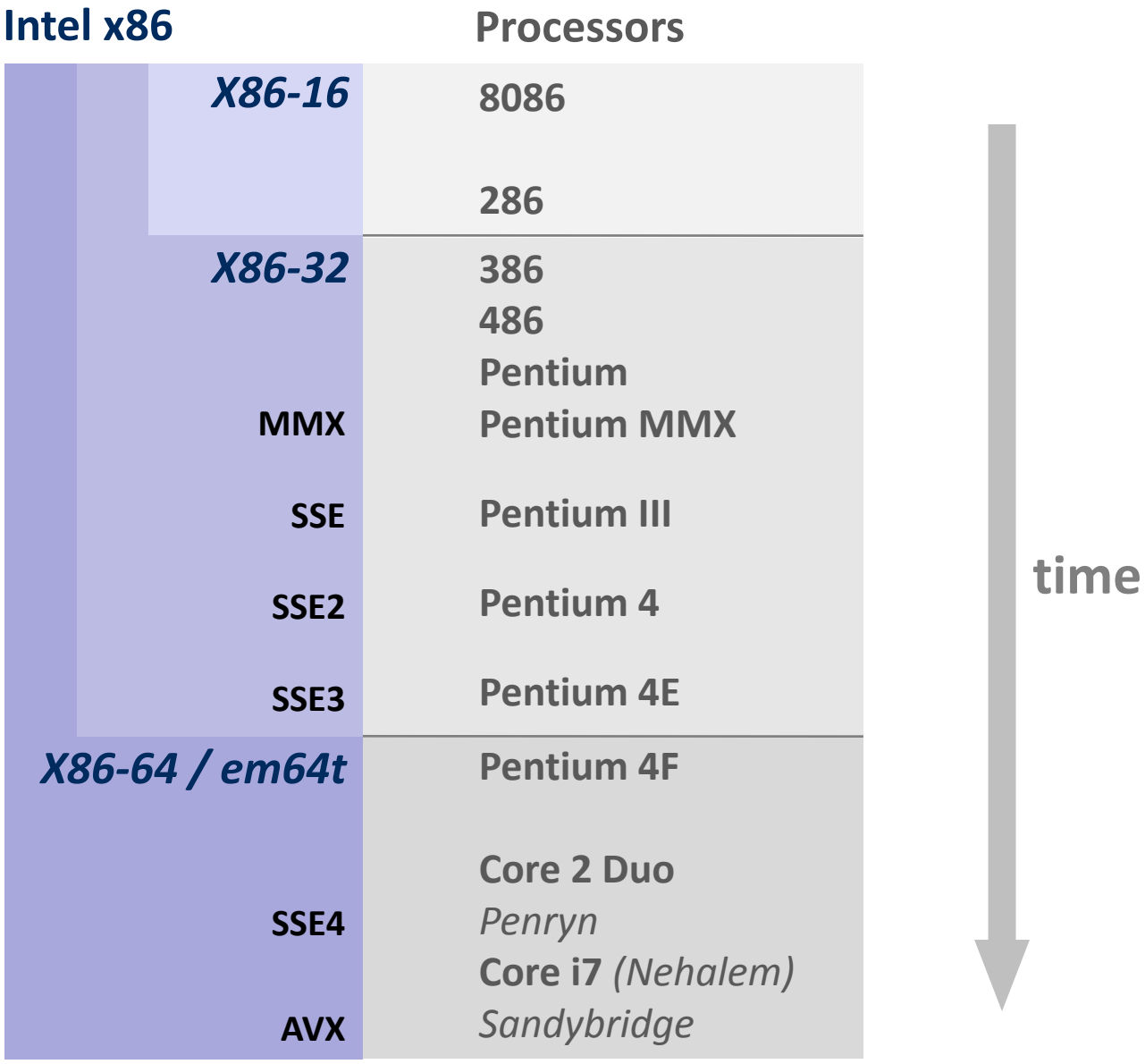
- **History**
  - 8086: first computer to implement IEEE FP
    (separate 8087 math coprocessor with floating point unit)
  - Logically stack based
  - 486: merged FPU and Integer Unit onto one chip
  - Once SSE came out, it was used for floating point
  - x87 is default on x86-32 (since SSE is not guaranteed)
  - Became obsolete with x86-64
- **Floating Point Formats**
  - single precision (C **`float`**): 32 bits
  - double precision (C **`double`**): 64 bits
  - extended precision (C **`long double`**): 80 bits



MMX:
Multimedia extension

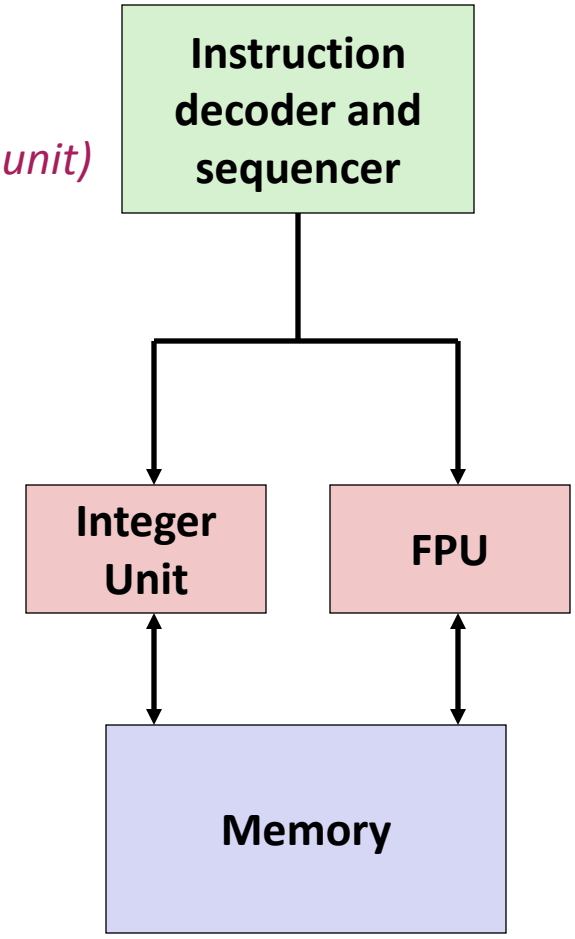
SSE:
Streaming SIMD extension

AVX:
Advanced vector extensions



The Other Floating Point (x87)

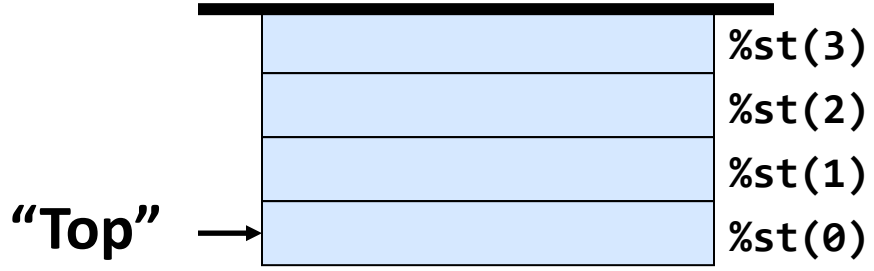
- **History**
  - 8086: first computer to implement IEEE FP
    (separate 8087 math coprocessor with floating point unit)
  - Logically stack based
  - 486: merged FPU and Integer Unit onto one chip
  - Default on x86-32 (since SSE is not guaranteed)
  - Became obsolete with x86-64
- **Floating Point Formats**
  - single precision (C **`float`**): 32 bits
  - double precision (C **`double`**): 64 bits
  - extended precision (C **`long double`**): 80 bits



x87 FPU Instructions and Register Stack

- **Sample instructions:**
  - flds (load single precision)
  - fmuls (mult single precision)
  - faddp (add and pop)

- **8 registers %st(0) - %st(7)**
- **Logically form stack**
- **Top: %st(0)**



- **Bottom disappears (drops out) after too many pushs**

- ***Obsolete***

FP Code Example (x87)

- **Inner product of two vectors**
  Single precision arithmetic

```
float ipf (float x[],
           float y[],
           int n) {
 int i;
 float result = 0.0;

 for (i = 0; i < n; i++)
   result += x[i]*y[i];
 return result;
}
```

```
   pushl %ebp              # setup
   movl %esp,%ebp
   pushl %ebx

   movl 8(%ebp),%ebx       # %ebx=&x
   movl 12(%ebp),%ecx      # %ecx=&y
   movl 16(%ebp),%edx      # %edx=n
   fldz                    # push +0.0
   xorl %eax,%eax          # i=0
   cmpl %edx,%eax          # if i>=n done
   jge .L3
.L5:
   flds (%ebx,%eax,4)      # push x[i]
   fmuls (%ecx,%eax,4)     # st(0)*=y[i]
   faddp                   # st(1)+=st(0); pop
   incl %eax               # i++
   cmpl %edx,%eax          # if i<n repeat
   jl .L5
.L3:
   movl -4(%ebp),%ebx      # finish
   movl %ebp, %esp
   popl %ebp
   ret                     # st(0) = result
```

Core: Floating Point Peak Performance

Single-precision (SP) FP MUL Double-precision FP MUL	4, 1 5, 1	4, 1 5, 1	Issue port 0; Writeback port 0
FP MUL (X87) FP Shuffle DIV/SQRT	5, 2 1, 1	5, 2 1, 1	Issue port 0; Writeback port 0 FP shuffle does not handle QW shuffle.

SSE based FP

x87 FP

- **Scalar:**
  - 1 add and 1 mult / cycle
  - Assume 3 GHz:
    6 Gflop/s scalar peak performance on one core
- **Vector double precision (SSE)**
  - 1 vadd and 1 vmult / cycle (2-way)
  - Assume 3 GHz:
    12 Gflop/s peak performance on one core
- **Vector single precision (SSE)**
  - 1 vadd and 1 vmult / cycle (4-way)
  - Assume 3 GHz:
    24 Gflop/s peak performance on one core

Core: Floating Point Peak Performance

- **Overall peak on 3 GHz Core 2 and Core i3/i5/i7 Nehalem: (2 cores, SSE)**
  - Double precision: 24 Gflop/s
  - Single precision: 48 Gflop/s

- **Overall peak on 3 GHz Core i3/i5/i7 Sandy Bridge: (4 cores, AVX)**
  - Double precision: 96 Gflop/s
  - Single precision: 192 Gflop/s

Performance in Numerical Computing

- **Numerical computing =**
  computing dominated by floating point operations
- **Cost measure: usually number of floating point ops**
- **Performance measure (in most cases) for a numerical function:**

$$\frac{\#\text{floating point operations}}{\text{runtime [s]}}$$

unit: flop/s

- **Makes it possible to show percentage of peak performance**