
CS4803DGC Design Game Console
Spring 2009
Prof. Hyesoon Kim
Georgia Tech
College of Computing

Multiprocessing

- Flynn's Taxonomy of Parallel Machines
  - How many Instruction streams?
  - How many Data streams?
- SISD: Single I Stream, Single D Stream
  - A uniprocessor
- SIMD: Single I, Multiple D Streams
  - Each "processor" works on its own data
  - But all execute the same instrs in lockstep
  - E.g. a vector processor or MMX, CUDA

Georgia Tech College of Computing

Flynn's Taxonomy

- MISD: Multiple I, Single D Stream
  - Not used much
  - Stream processors are closest to MISD
- MIMD: Multiple I, Multiple D Streams
  - Each processor executes its own instructions and operates on its own data
  - This is your typical off-the-shelf multiprocessor (made using a bunch of "normal" processors)
  - Includes multi-core processors

Georgia Tech College of Computing

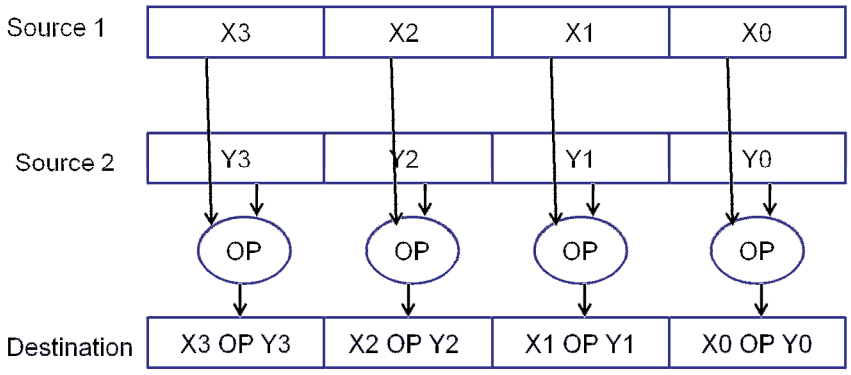

SIMD
Georgia Tech
College of Computing

SIMD Model

- Texas C62xx, IA32 (SSE), AMD K6, CUDA, Xbox..
- Early SIMD machines: e.g.) CM-2 (large distributed system)
  - Lack of vector register files and efficient transposition support in the memory system.
  - Lack of irregular indexed memory accesses
- Modern SIMD machines:
  - SIMD engine is in the same die

Georgia Tech College of Computing

SIMD Execution Model




```
for (ii = 0; ii < 4; ii++)
x[ii] = y[ii]+z[ii];
```

SIMD_ADD(X, Y, Z)

Georgia Tech College of Computing

MMX/SSE/3D Now!

- MMX :
  - Virtual registers. Use FP registers
  - Handles only integers. Designed for 2D graphics
- SSE: Introduce new registers (XMM0..XMM7, XMM15)
- 3DNow! (AMD)

Georgia Tech College of Computing

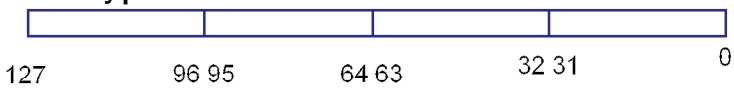
MMX Instruction Set

- Arithmetic
  - Addition, subtraction, multiplication, multiply and Add
- Comparison
  - Compare for equal, greater than
- Conversion
  - Pack
- Unpack
- Logical
  - And, And Not, Or, Exclusive OR
- Data Transfer
  - Register to register, load from memory

Georgia Tech College of Computing

SSE

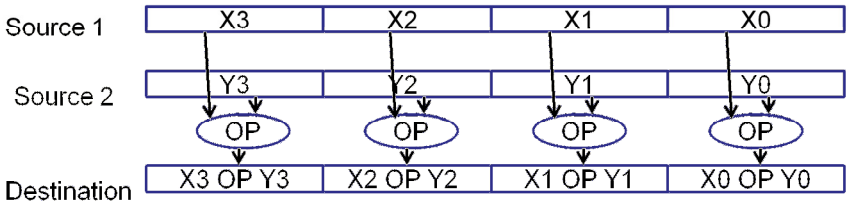
- New data type
  - 128-bit packed single-precision floating-point data type



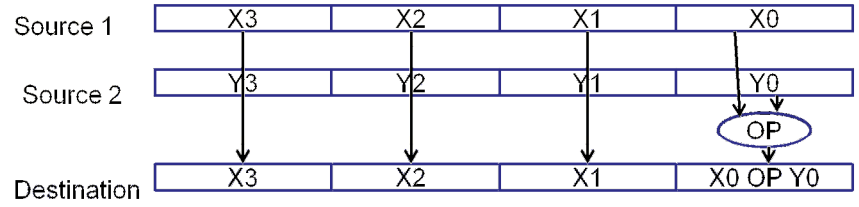
- Packed/Scalar singe-precision floating-point instruction
- 64-bit SIMD integer instruction
- State management instructions
- Cacheability control, prefetch, and memory ordering instructions

Georgia Tech College of Computing

SSE Packed and Scalar Floating-Point Instructions



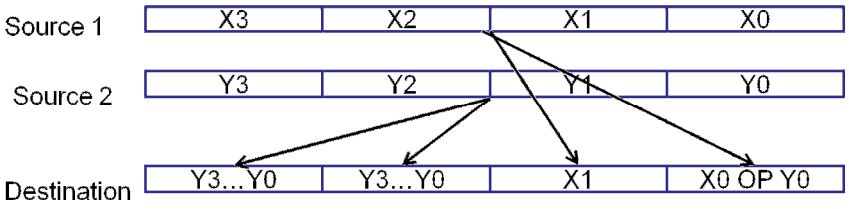
Packed single-precision floating-point operation



Scalar single-precision floating-point operation

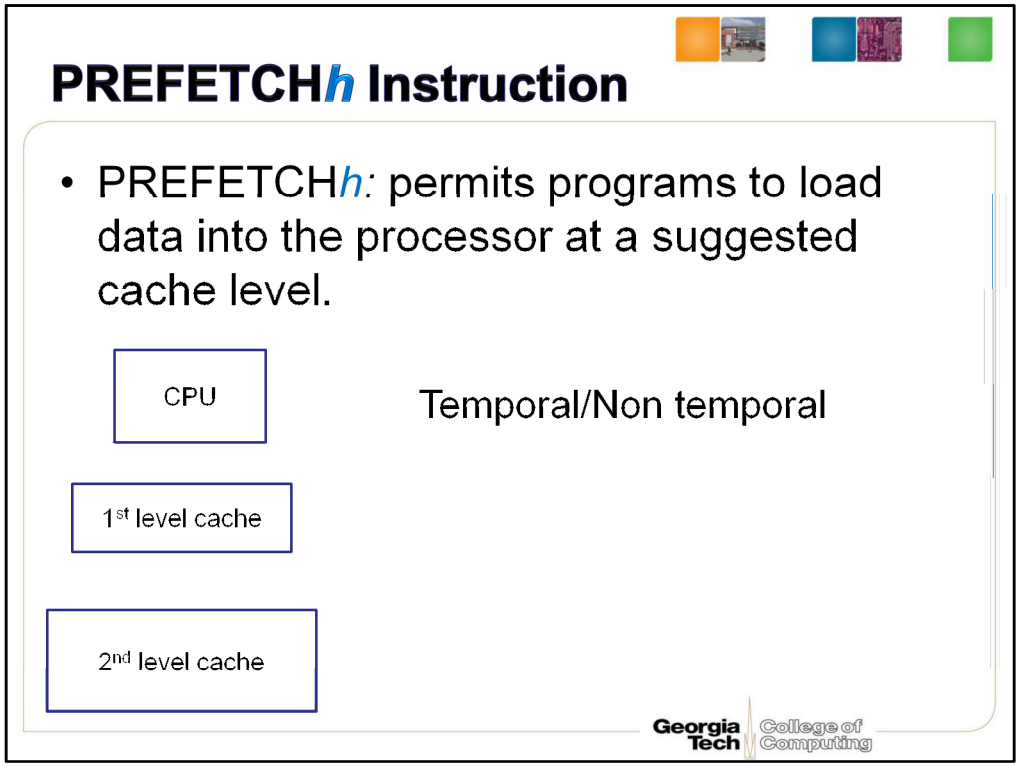
Georgia Tech College of Computing

Shuffle and Unpack Instructions



Scalar single-precision floating-point operation

Georgia Tech College of Computing


PREFETCHh Instruction
• PREFETCHh: permits programs to load data into the processor at a suggested cache level.
CPU
Temporal/Non temporal
1st level cache
2nd level cache
Georgia Tech College of Computing

SSE2/SSE3/SSE4

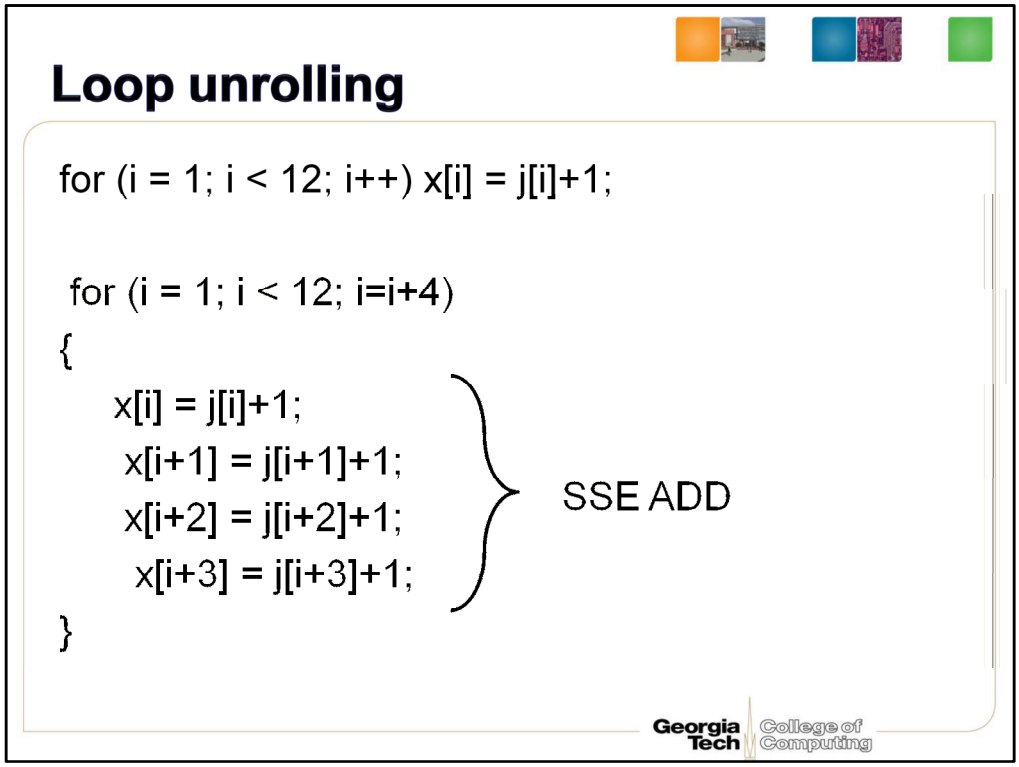
- Add new data types
- Add more complex SIMD instructions
- Additional vector registers
- Additional cacheability-control and instruction-ordering instructions.

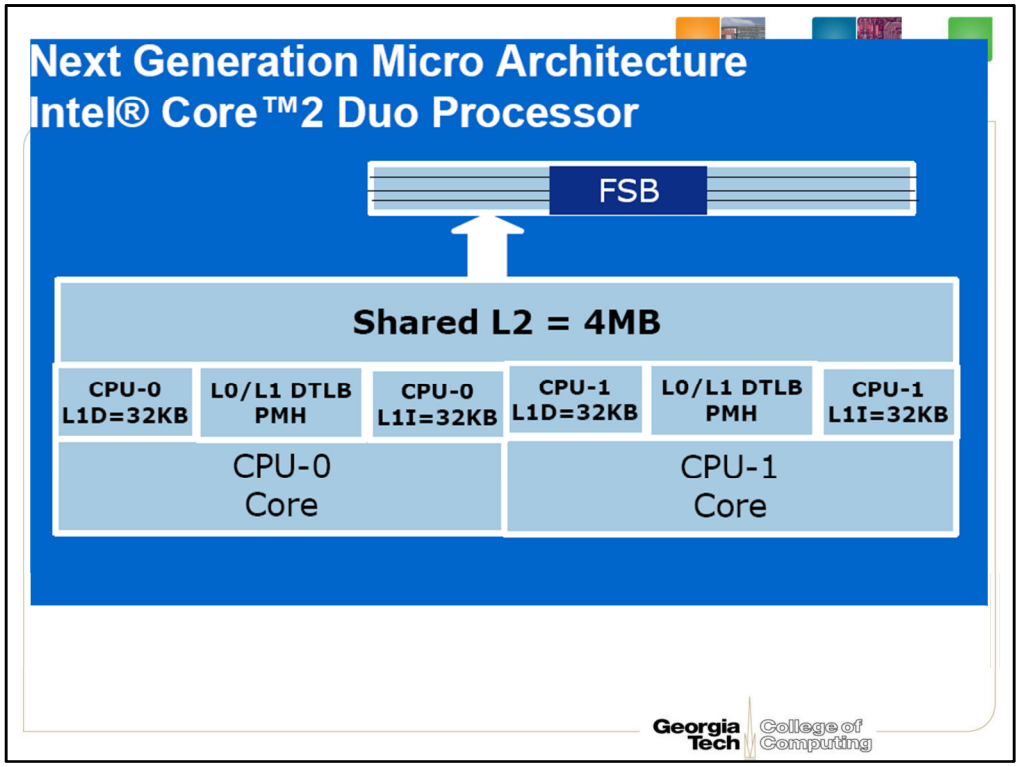
Georgia Tech College of Computing

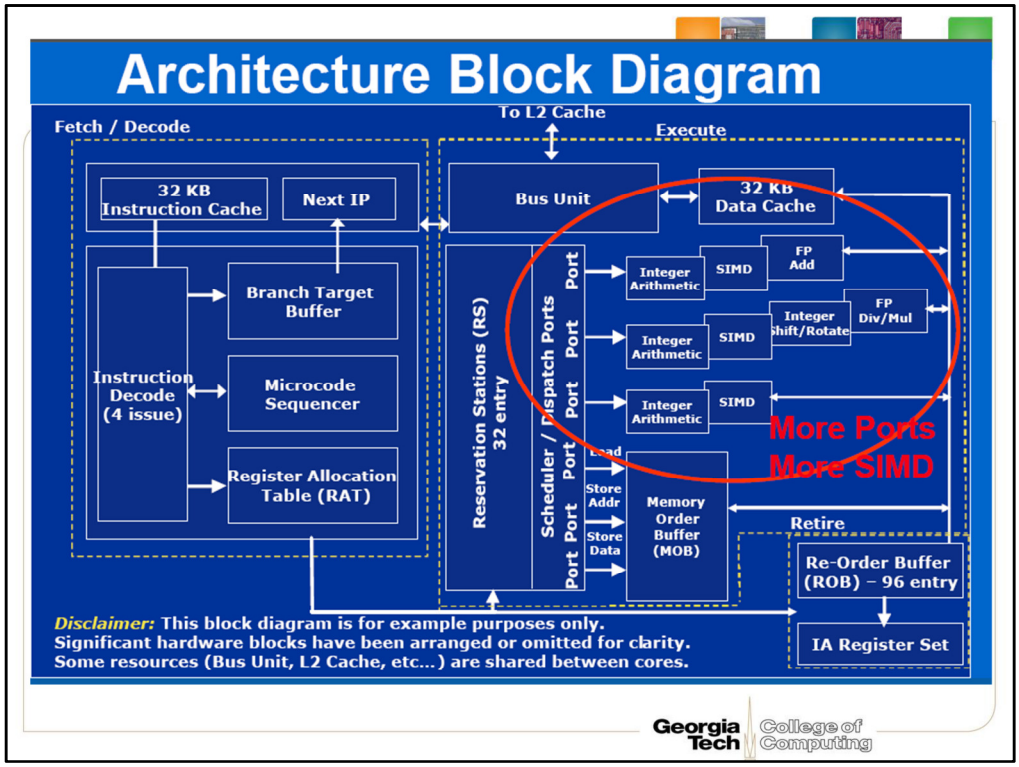
SIMD Programming:

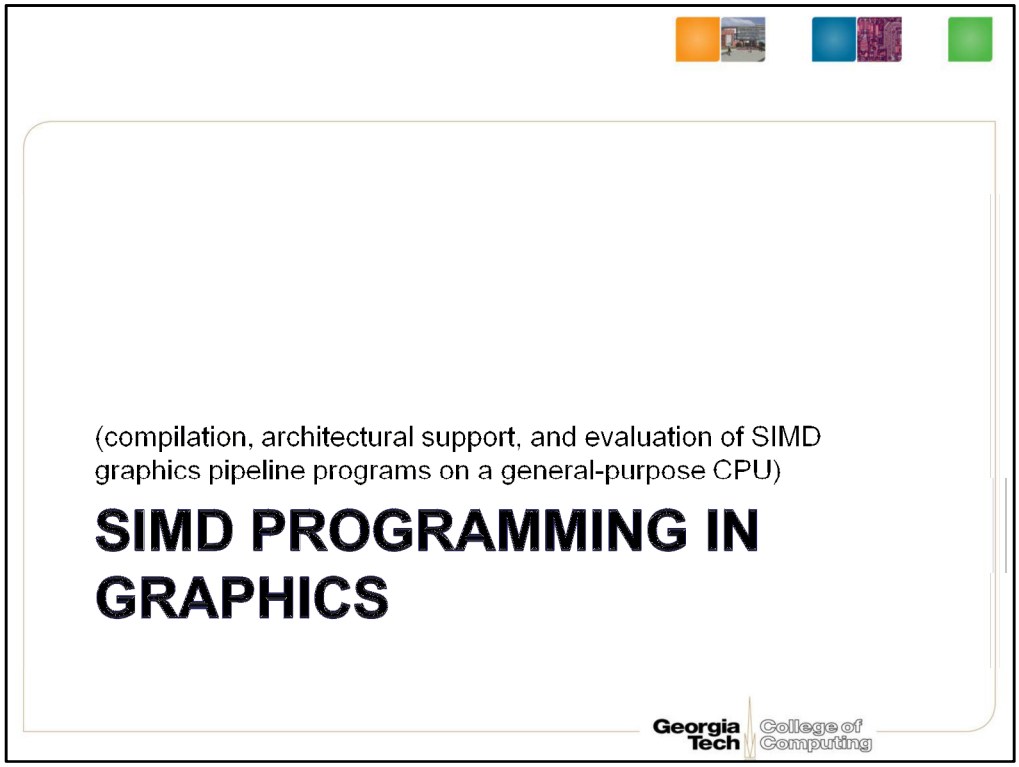
- Using compiler: Vectorising using compiler
- Assembler intrinsic:
  - E.g.) __m128_mm_add_ps (__m128 a, __m128b)
  - Translated one for one into equivalent assembler instructions
  - Pros: No need to worry about C optimization
  - Cons:
    - Not portable between processors
    - Requires the programmer to know the underlying machine architecture
    - Longer programming time

Georgia Tech College of Computing

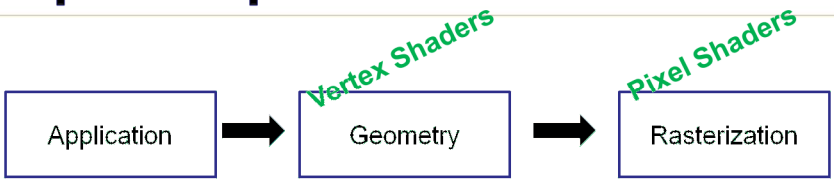

Loop unrolling
for (i = 1; i < 12; i++) x[i] = j[i]+1;
for (i = 1; i < 12; i=i+4)
{
x[i] = j[i]+1;
x[i+1] = j[i+1]+1;
x[i+2] = j[i+2]+1;
x[i+3] = j[i+3]+1;
}
SSE ADD
Georgia Tech College of Computing


Next Generation Micro Architecture
Intel® Core™2 Duo Processor
FSB
Shared L2 = 4MB
CPU-0 L1D=32KB
L0/L1 DTLB PMH
CPU-0 L1I=32KB
CPU-1 L1D=32KB
L0/L1 DTLB PMH
CPU-1 L1I=32KB
CPU-0 Core
CPU-1 Core
Georgia Tech College of Computing


Architecture Block Diagram
To L2 Cache
Fetch / Decode
Execute
32 KB Instruction Cache
Next IP
Bus Unit
32 KB Data Cache
Branch Target Buffer
Instruction Decode (4 issue)
Microcode Sequencer
Register Allocation Table (RAT)
Reservation Stations (RS) 32 entry
Scheduler / Dispatch Ports
Port
Integer Arithmetic
SIMD
FP Add
Integer Shift/Rotate
FP Div/Mul
Load
Store Addr
Store Data
Memory Order Buffer (MOB)
More Ports
More SIMD
Retire
Re-Order Buffer (ROB) – 96 entry
IA Register Set
Disclaimer: This block diagram is for example purposes only.
Significant hardware blocks have been arranged or omitted for clarity.
Some resources (Bus Unit, L2 Cache, etc...) are shared between cores.
Georgia Tech College of Computing


(compilation, architectural support, and evaluation of SIMD graphics pipeline programs on a general-purpose CPU)
SIMD PROGRAMMING IN GRAPHICS
Georgia Tech College of Computing

Graphics Pipeline



- Vertex shader: operates on a vortex.
  - Transform step: converts element coordinates from one frame of reference to another e.g., between world space and eye space
  - Lighting,…
- Pixel shader(fragment shader): rasterization phase. Processes a pixel
  - Texturing, filtering

Georgia Tech College of Computing

Example: Vertex Shader

```
dp4 oPos.x, v0, c[0]
dp4 oPos.y, v0, c[1]
dp4 oPos.z, v0, c[2]
dp4 oPos.w, v0, c[3]
```

Apply a constant color to each vertex

dp4: a four-component dot-product

```
mov oD0,  c[4]
```

Mask: selects components of a vertex that are not affected by the operation.
Swizzle: reorders and/or replicates components of the vector operands to the instruction.

Georgia Tech College of Computing

Vertex Shader Program Using SSE

- Shader instruction: dp4 oPos, v0, v1
  - A dot prodcut of vo and v1 and stores the result in each of the four words of oPos

SSE2

Input: %xmm0, %xmm1, %xmm2 output

```
movaps %xmm5, %xmm0
mulps %xmm5, %xmm1
pshufd %xmm6, %xmm5, $14
addps %xmm6, %xmm5
pshufd %xmm5, %xmm6, $1
addps %xmm5, %xmm6
pshufd %xmm2, %xmm5, $0
```

Georgia Tech College of Computing

Vertex Shader Program Using SSE-II

- shader instruction: mul r0.xz, v0.xyz, v1.w

multipflies ([v0.x, v0.y, v0.z, v0.z]) , ([v1.w, v1.w, v1.w, v1.w]) stores the x and z components of the result into r0.

= mul r0.x_z_, v0.xyzz, v1.wwww

SSE:

```
pshufd %xmm5, %xmm0, $164
pshufd %xmm6, %xmm1, $255
mulps %xmm6. %xmm5
andps %xmm6, _Mask+80
andps %xmm2, _Mask+160
orps %xmm2, %xmm6
```

Georgia Tech College of Computing

SOA & AOS

- Array of structures (AOS)
  - {x1,y1, z1,w1} , {x2,y2, z2,w2} , {x3,y3, z3,w3} , {x4,y4, z4,w4} ….
  - Intuitive but less efficient
  - What if we want to perform only x axis?
- Structure of array (SOA)
  - {x1,x2,x3,x4}, …,{y1,y2,y3,y4}, …{z1,z2,z3,z4}, … {w1,w2,w3,w4}…

Georgia Tech College of Computing