

# Building Blocks

---

CPUs, Memory and Accelerators

**EPSRC**

**NERC** SCIENCE OF THE ENVIRONMENT



**CRAY**  
THE SUPERCOMPUTER COMPANY

**epcc**



# Outline

- Computer layout
  - CPU and Memory
  - What does performance depend on?
  - Limits to performance
- Silicon-level parallelism
  - Single Instruction Multiple Data (SIMD/Vector)
  - Multicore
  - Symmetric Multi-threading (SMT)
- Accelerators (GPGPU and Xeon Phi)
  - What are they good for?

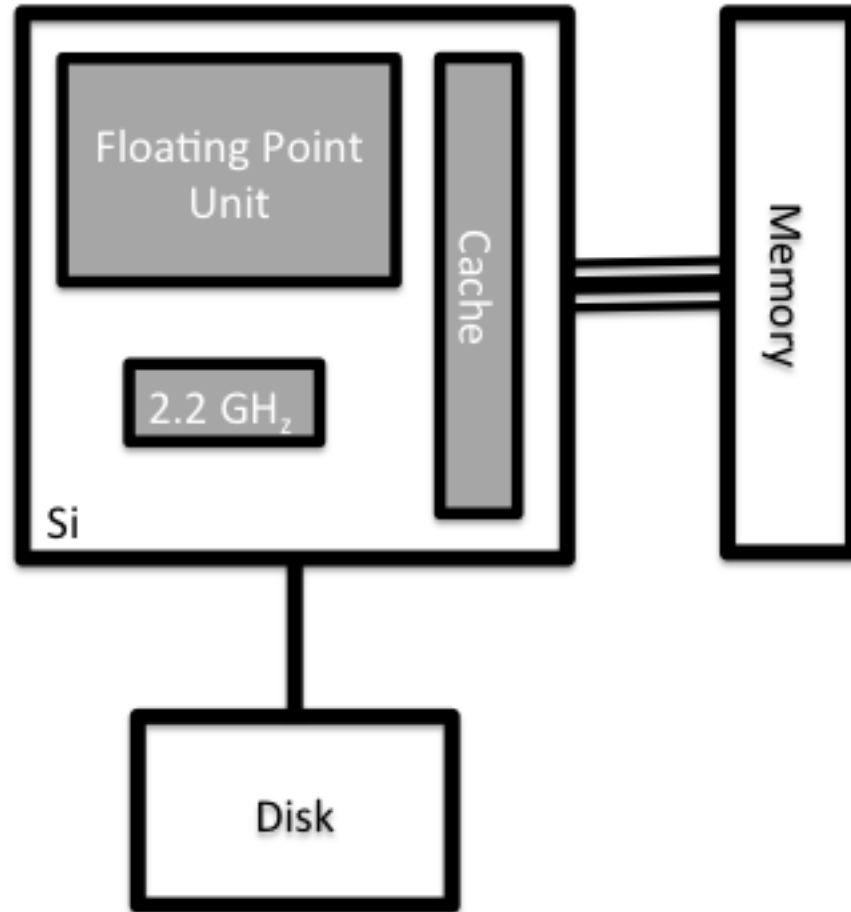


# Computer Layout

How do all the bits interact and which ones matter?



# Anatomy of a computer



# Data Access

- Disk access is slow
  - a few hundreds of Megabytes/second
- Large memory sizes allow us to keep data in memory
  - but memory access is slow
  - a few tens of Gigabytes/second
- Store data in fast cache memory
  - cache access much faster: hundreds of Gigabytes per second
  - limited size: a few Megabytes at most



# Performance

- The performance (time to solution) on a single computer can depend on:
  - Clock speed – how fast the processor is
  - Floating point unit – how many operands can be operated on and what operations can be performed?
  - Memory latency – what is the delay in accessing the data?
  - Memory bandwidth – how fast can we stream data from memory?
  - Input/Output (IO) to storage – how quickly can we access persistent data (files)?



# Performance (cont.)

- Application performance often described as:
  - Compute bound
  - Memory bound
  - IO bound
  - (Communication bound – more on this later...)
- For computational science
  - most calculations are limited by memory bandwidth
  - processor can calculate much faster than it can access data



# Silicon-level parallelism

What does Moore's Law mean anyway?







# What to do with all those transistors?

- For over 3 decades until early 2000's
  - more complicated processors
  - bigger caches
  - faster clock speeds
- Clock rate increases as inter-transistor distances decrease
  - so performance doubled every 18 months
- Came to a grinding halt about a decade ago
  - reached power and heat limitations
  - who wants a laptop that runs for an hour and scorches your trousers!



# Alternative approaches

- Introduce parallelism into the processor itself
  - vector instructions
  - simultaneous multi-threading
  - multicore



# Single Instruction Multiple Data (SIMD)

- For example, vector addition:

$$\begin{array}{|c|} \hline a_0 \\ \hline a_1 \\ \hline a_2 \\ \hline a_3 \\ \hline \end{array} + \begin{array}{|c|} \hline b_0 \\ \hline b_1 \\ \hline b_2 \\ \hline b_3 \\ \hline \end{array} = \begin{array}{|c|} \hline c_0 \\ \hline c_1 \\ \hline c_2 \\ \hline c_3 \\ \hline \end{array}$$

- single instruction adds 4 numbers
- potential for 4 times the performance

# Symmetric Multi-threading (SMT)

- Some hardware supports running multiple instruction streams simultaneously on the same processor, e.g.
  - stream 1: loading data from memory
  - stream 2: multiplying two floating-point numbers together
- Known as *Symmetric Multi-threading (SMT)* or *hyperthreading*
- Threading in this case can be a misnomer as it can refer to processes as well as threads
  - These are hardware threads, not software threads.
  - Intel Xeon supports 2-way SMT
  - IBM BlueGene/Q 4-way SMT

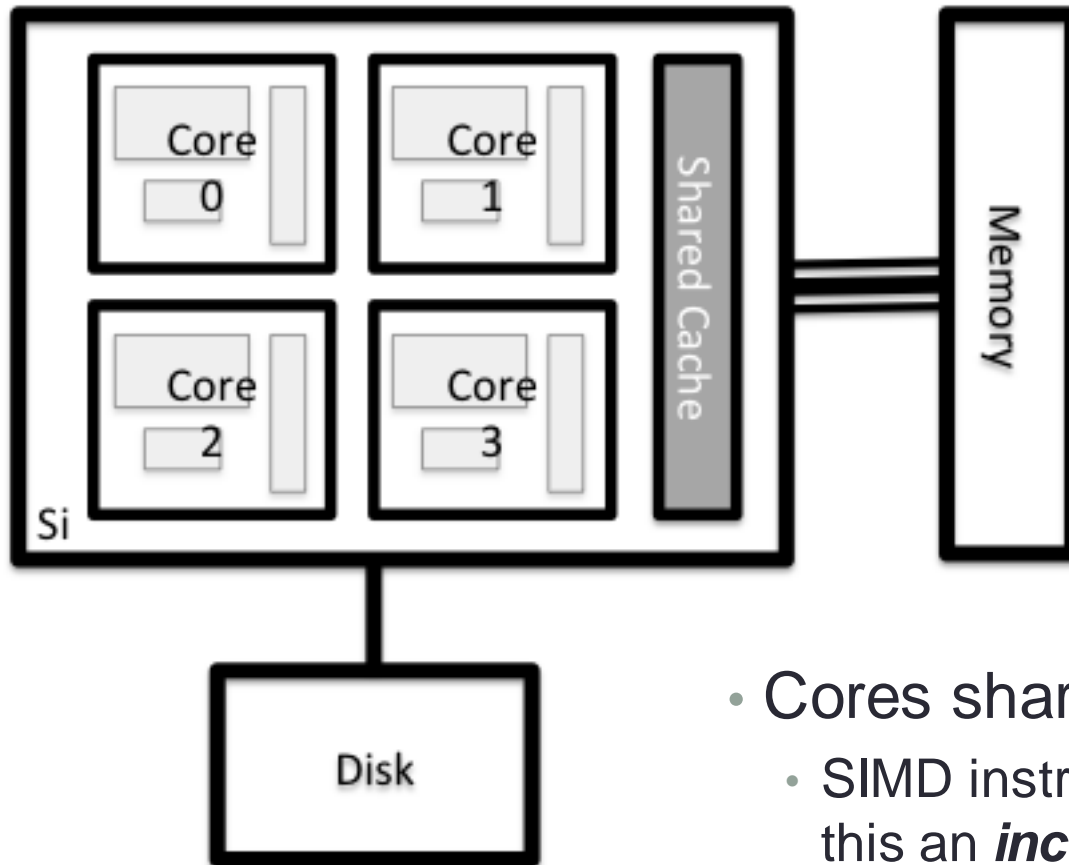


# Multicore

- Twice the number of transistors gives 2 choices
  - a new more complicated processor with twice the clock speed
  - two versions of the old processor with the same clock speed
- The second option is more power efficient
  - and now the only option as we have reached heat/power limits
- Effectively two independent processors
  - ... except they can share cache
  - commonly called “cores”

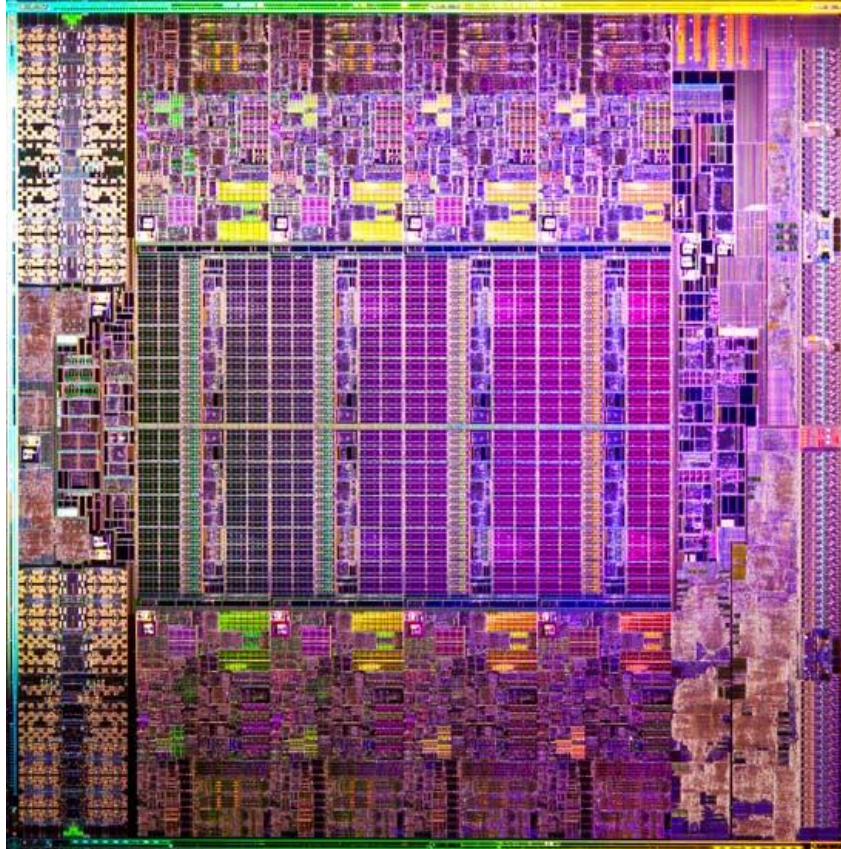


# Multicore



- Cores share path to memory
  - SIMD instructions + multicore make this an *increasing* bottleneck!

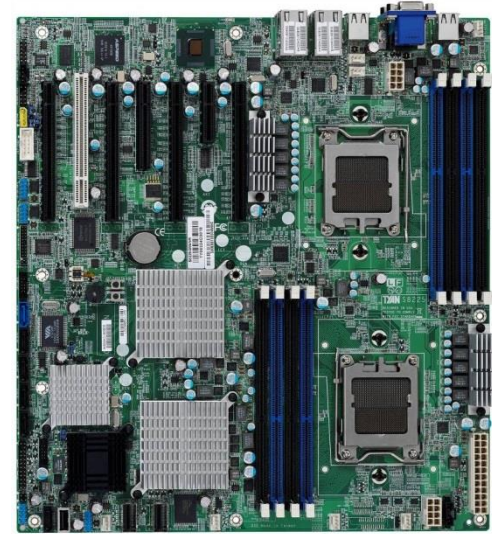
# Intel Xeon E5-2600 – 8 cores HT





# What is a processor?

- To a programmer
  - the thing that runs my program
  - i.e. a single core of a multicore processor
- To a hardware person
  - the thing you plug in to a socket on the motherboard
  - i.e. an entire multicore processor
- Some ambiguity
  - in this course we will talk about cores and sockets
  - try and avoid using “processor”



# Chip types and manufacturers

- x86 – Intel and AMD
  - “PC” commodity processors, SIMD (SSE, AVX) FPU, multicore, SMT (Intel); Intel currently dominates the HPC space.
- Power – IBM
  - Used in high-end HPC, high clock speed (direct water cooled), SIMD FPU, multicore, SMT; not widespread anymore.
- PowerPC – IBM BlueGene
  - Low clock speed, SIMD FPU, multicore, high level of SMT.
- SPARC – Fujitsu
- ARM – Lots of manufacturers
  - Not yet relevant to HPC (weak FP Unit)



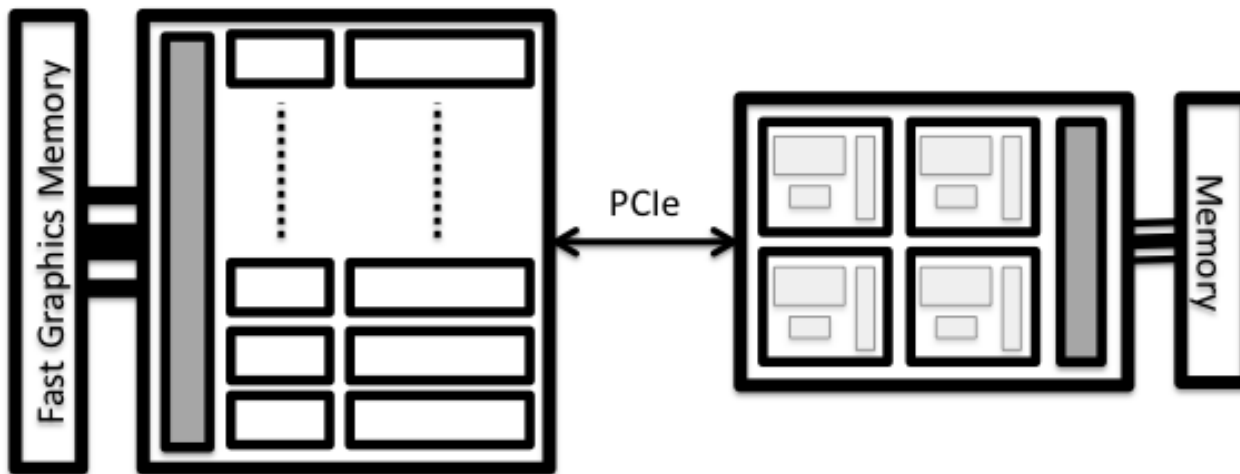
# Accelerators

Go-faster stripes



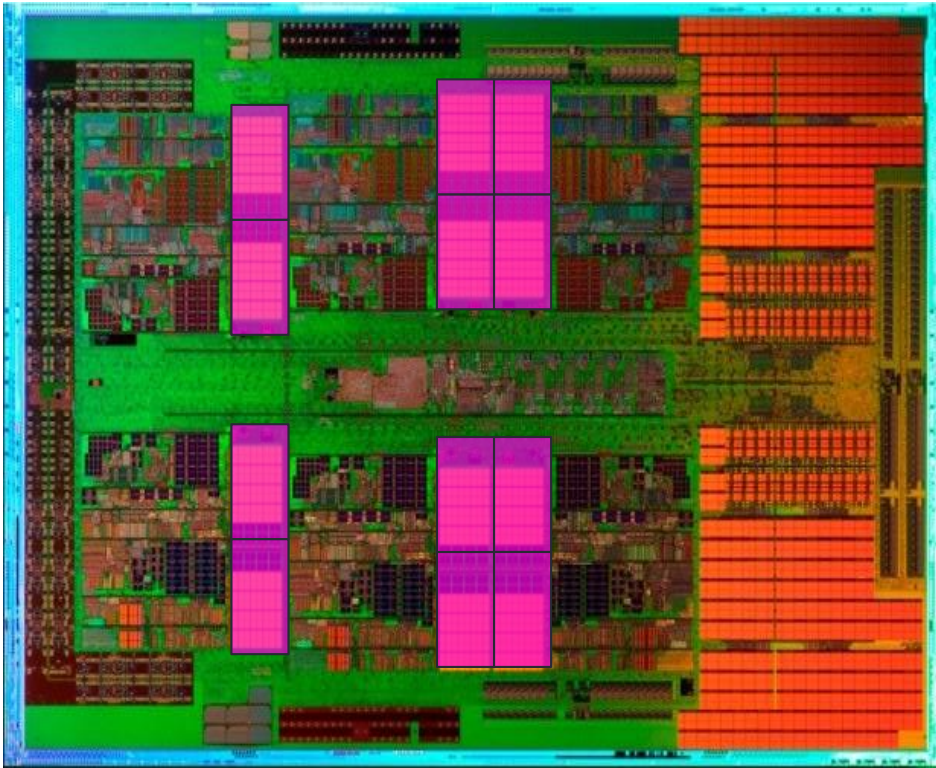
# Anatomy


- An *Accelerator* is an additional resource that can be used to off-load heavy floating-point calculation
  - additional processing engine attached to the standard processor
  - has its own floating point units and memory



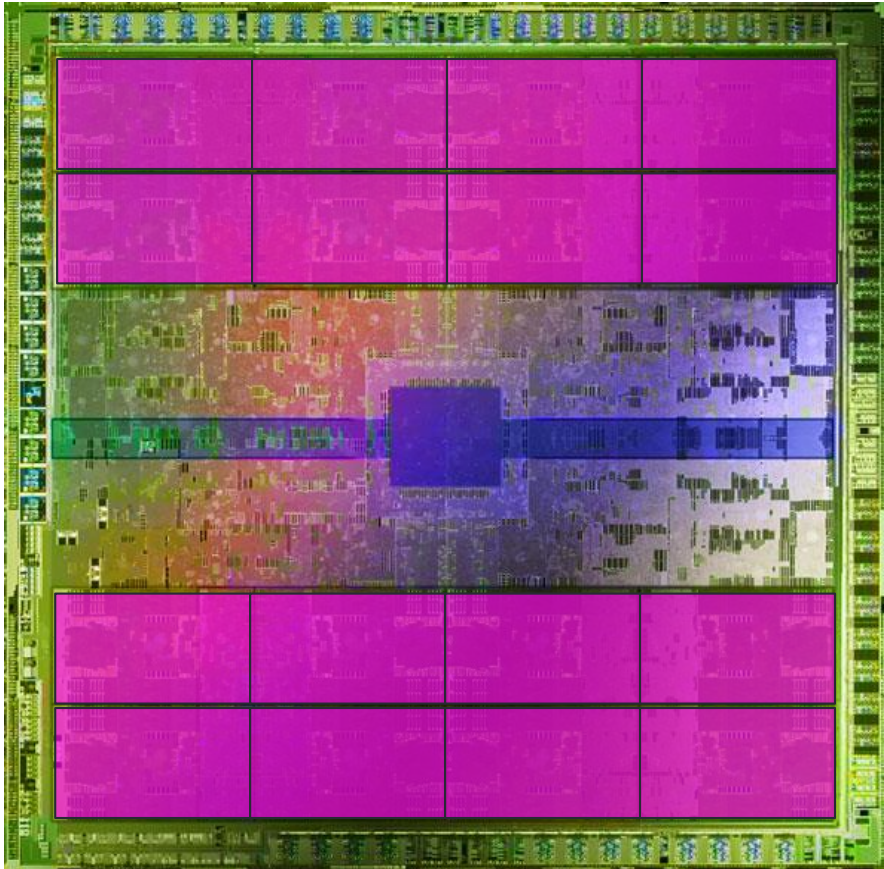
# AMD 12-core CPU

- Not much space on CPU is dedicated to computation




 = compute unit  
(= core)

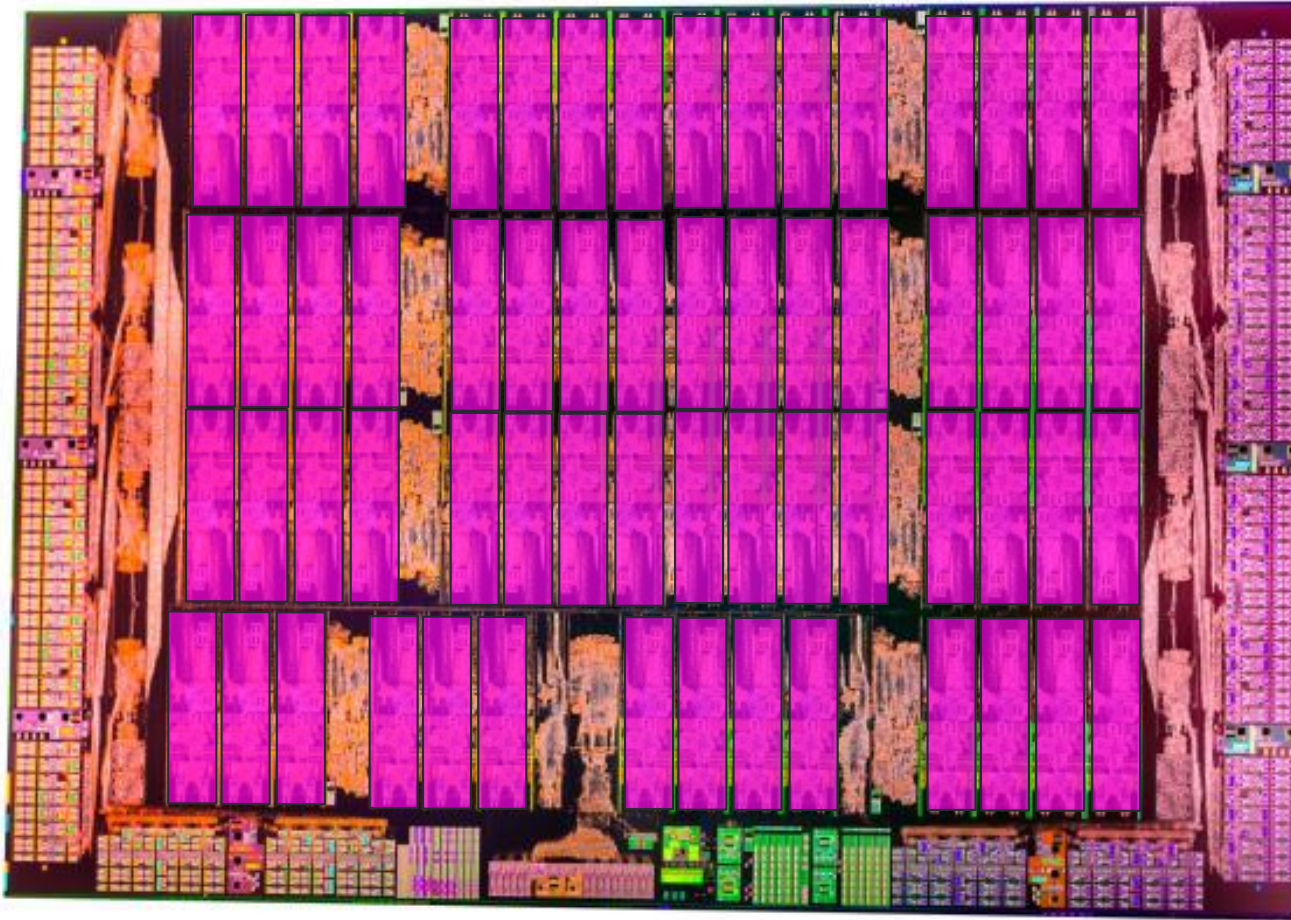
# NVIDIA Fermi GPU




- GPU dedicates much more space to computation
  - At expense of caches, controllers, sophistication etc

 = compute unit  
(= SM  
= 32 CUDA cores)

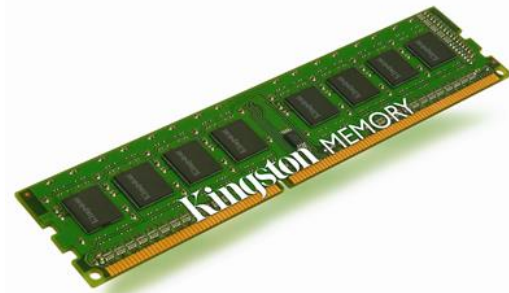
# Intel Xeon Phi



 = compute unit  
(= core)

# Memory

- For most HPC applications, performance is very sensitive to memory bandwidth
- GPUs and Intel Phi both use Graphics memory: much higher bandwidth than standard CPU memory



CPUs use DRAM



GPUs and Xeon Phi use Graphics DRAM



# Summary - What is automatic?

- Which features are managed by hardware/software and which does the user/programmer control?
  - Cache and memory – automatically managed
  - SIMD/Vector parallelism – automatically produced by compiler
  - SMT – automatically managed by operating system
  - Multicore parallelism – manually specified by the user
  - Use of accelerators – manually specified by the user

