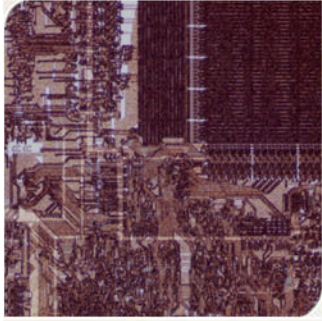


# CS4803DGC Design and Programming of Game Consoles

Spring 2011

Prof. Hyesoon Kim



**Georgia  
Tech**



College of  
Computing

Xbox 360 System Architecture, 'Anderews, Baker

# Xbox 360 System Block Diagram

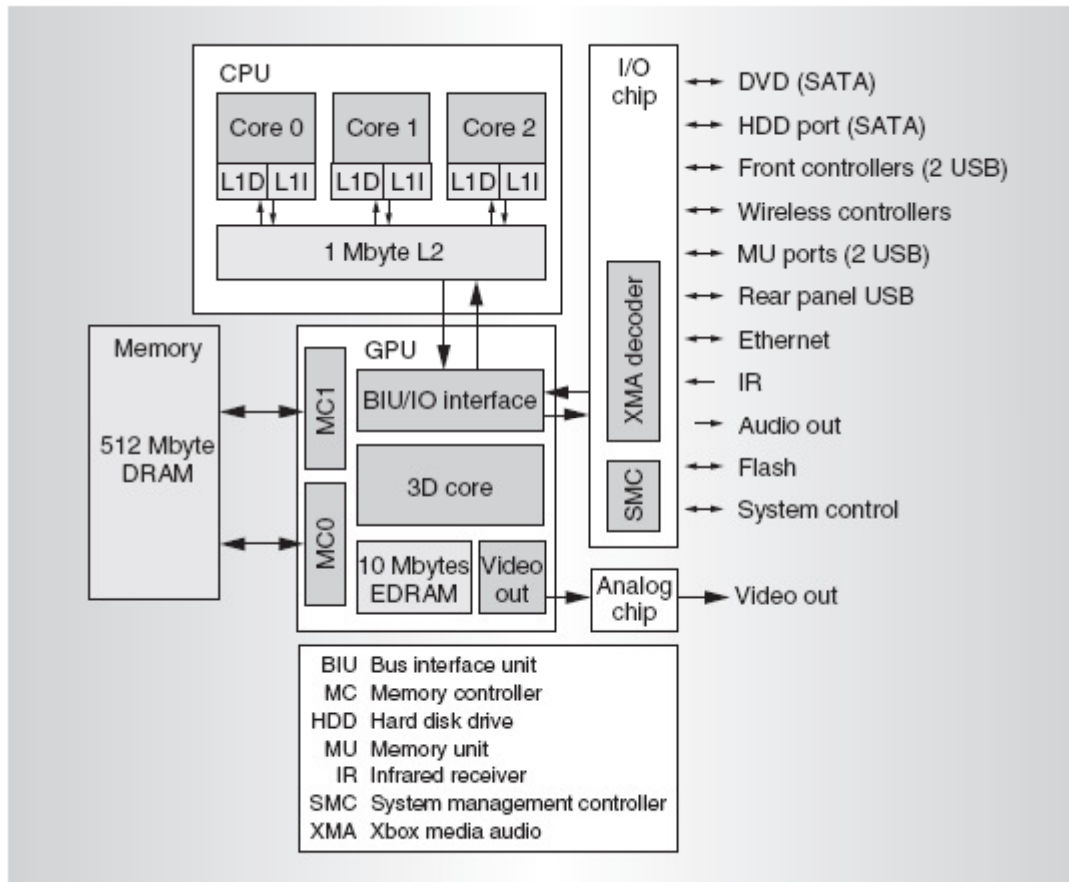


Figure 2. Xbox 360 system block diagram.



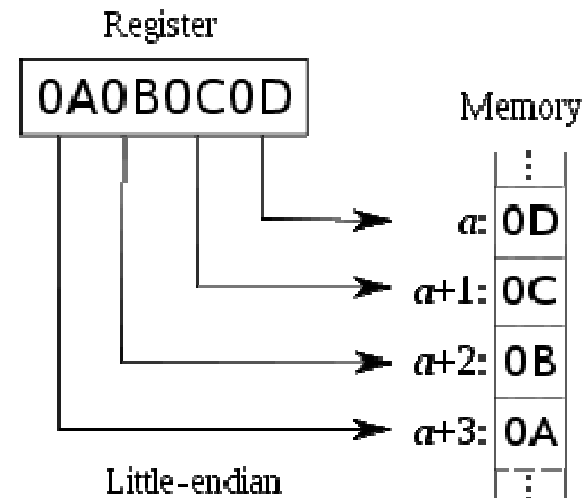
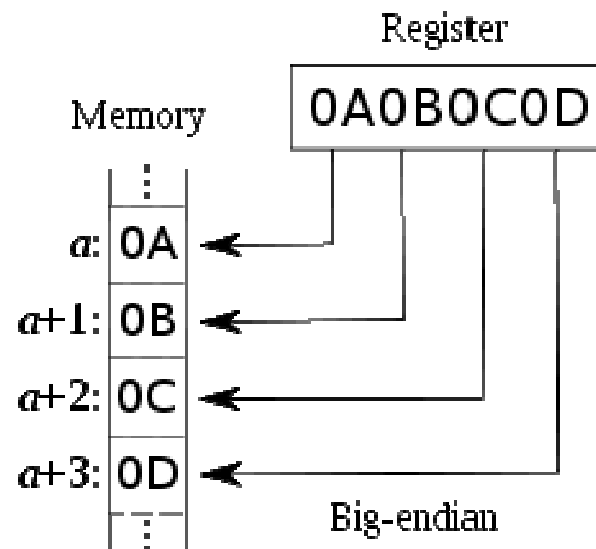
# Xbox 360 Architecture

- 3 CPU cores
  - 4-way SIMD vector units
  - 8-way 1MB L2 cache (3.2 GHz)
  - 2 way SMT
- 48 unified shaders
- 3D graphics units
- 512-Mbyte DRAM main memory
- FSB (Front-side bus): 5.4 Gbps/pin/s (16 pins)
- 10.8 Gbyte/s read and write



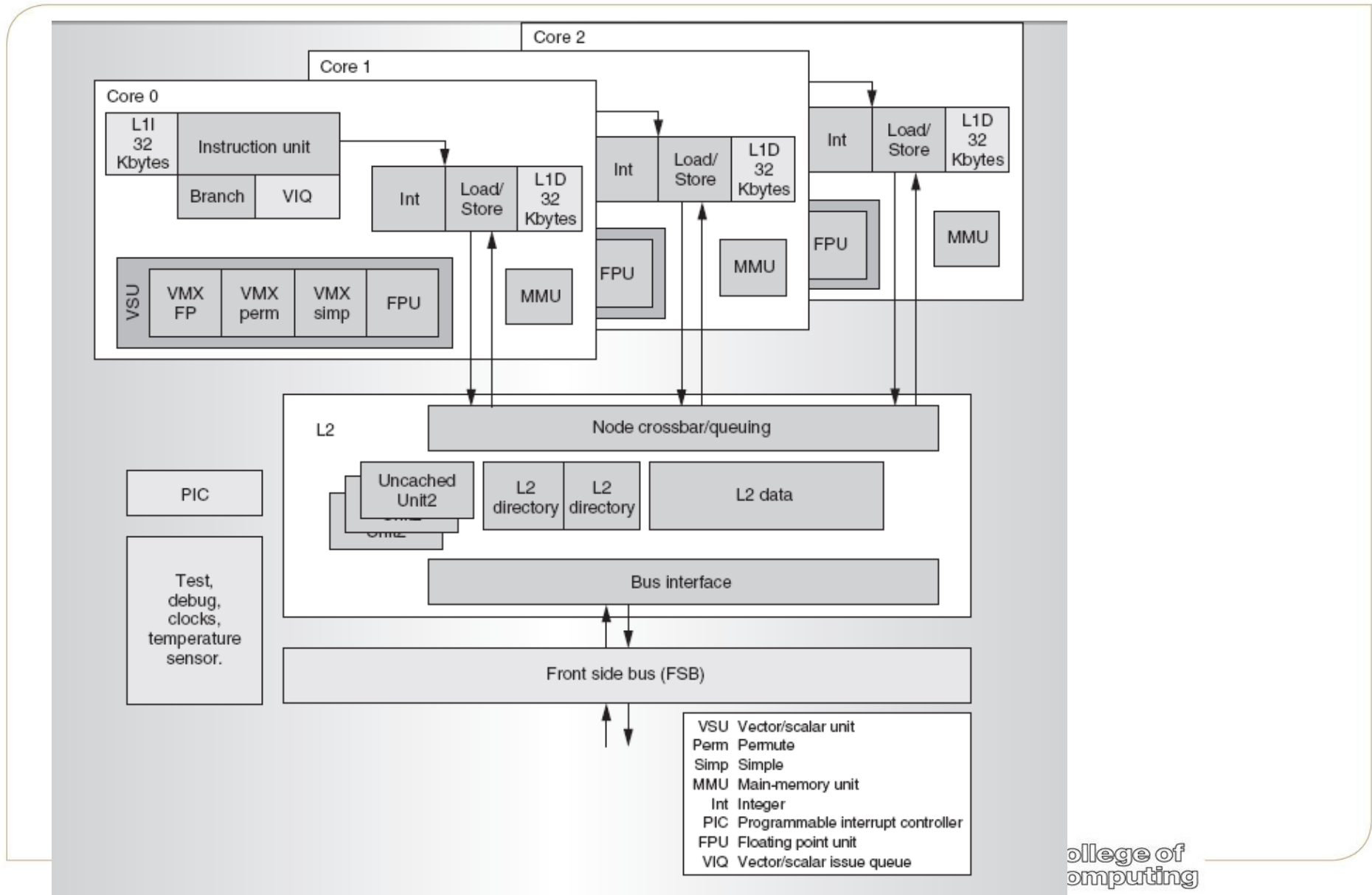
# Xbox 360 vs. Windows

- Xbox 360: Big endian
- Windows: Little endian





# Xbox 360 CPU Block Diagram





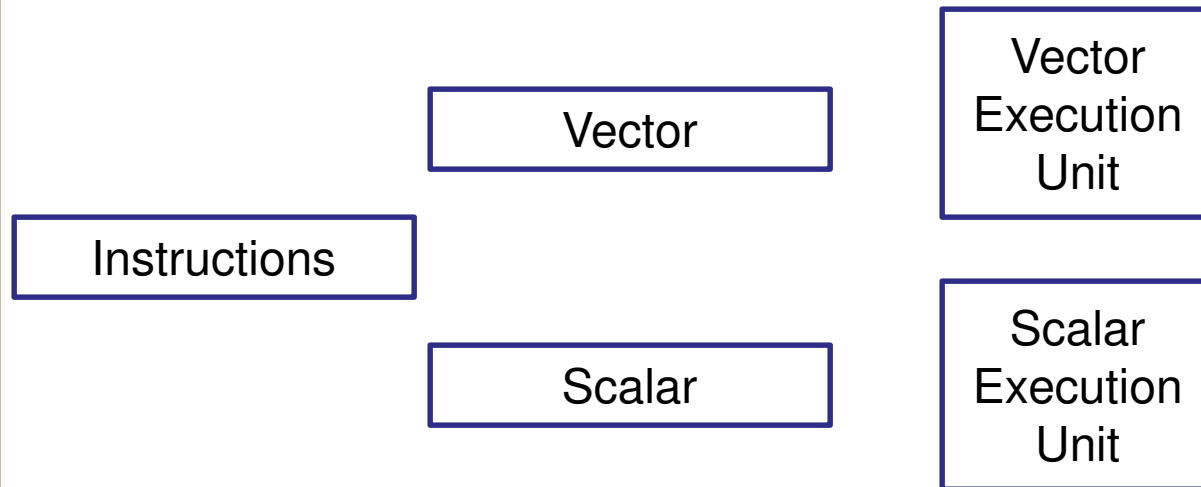
# On-chip caches

- L2 cache :
  - Greedy allocation algorithm
  - Different workloads have different working set sizes
- 2-way 32 Kbyte L1 I-cache
- 4-way 32 Kbyte L1 data cache
- Write through, no write allocation
- Cache block size :128B (high spatial locality)



# Core

- 2-way SMT,
- 2 insts/cycle,
- In-order issue
- Separate vector/scalar issue queue (VIQ)





# A Brief History

- First game console by Microsoft, released in 2001, \$299
  - Glorified PC
    - 733 Mhz x86 Intel CPU, 64MB DRAM, NVIDIA GPU (graphics)
    - Ran modified version of Windows OS
    - ~25 million sold
- XBox 360
  - Second generation, released in 2005, \$299-\$399
  - All-new custom hardware
  - 3.2 Ghz PowerPC IBM processor (custom design for XBox 360)
  - ATI graphics chip (custom design for XBox 360)
  - 34+ million sold (as of 2009)
- Design principles of XBox 360 [Andrews & Baker]
  - Value for 5-7 years
  - lig performance increase over last generation
  - Support anti-aliased high-definition video (720\*1280\*4 @ 30+ fps)
  - extremely high pixel fill rate (goal: 100+ million pixels/s)
  - Flexible to suit dynamic range of games
  - balance hardware, homogenous resources
  - Programmability (easy to program)





# Xenon

- Code name of Xbox 360's core
- Shared cell (playstation processor) 's design philosophy.
- 2-way SMT
- Good: Procedural synthesis is highly multi-thread
- Bad: three types of game-oriented tasks are likely to suffer from the lack of high ILP support: game control, artificial intelligence (AI), and physics.

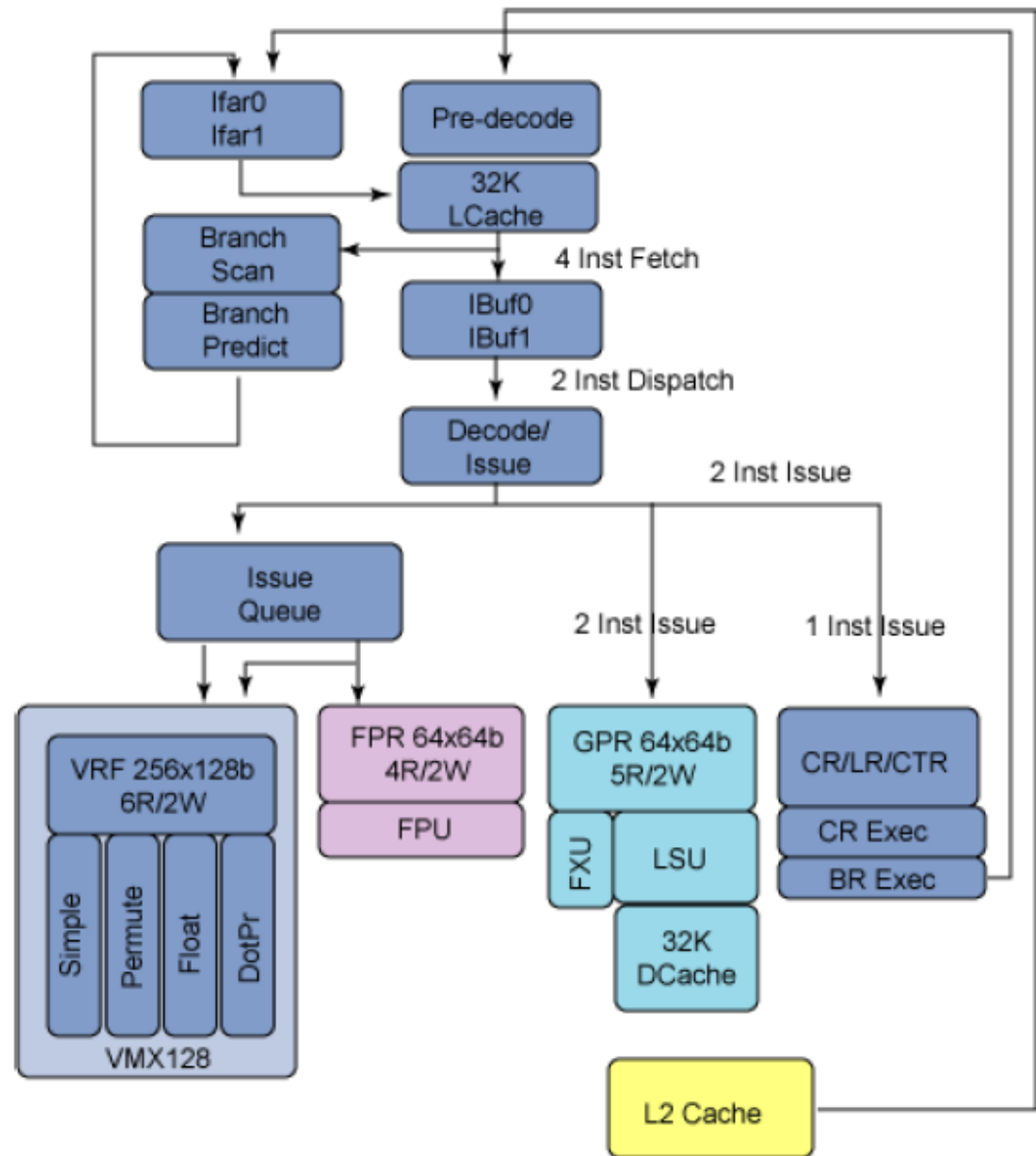


# Xenon Processor

- ISA: 64-bit PowerPC chip
  - RISC ISA
  - Like MIPS, but with condition codes
  - Fixed-length 32-bit instructions
  - 32 64-bit general purpose registers (GPRs)
- ISA++: Extended with VMX-128 operations
  - **128 registers, 128-bits each**
  - Packed “vector” operations
  - Example: four 32-bit floating point numbers
  - One instruction:  $VR1 * VR2 ! VR3$
  - Four single-precision operations
  - Also supports conversion to MS DirectX data formats
- Works great for 3D graphics kernels and compression
- 3.2 GHZ
- Peak performance Peak performance: ~75 gigaflops

# Data path

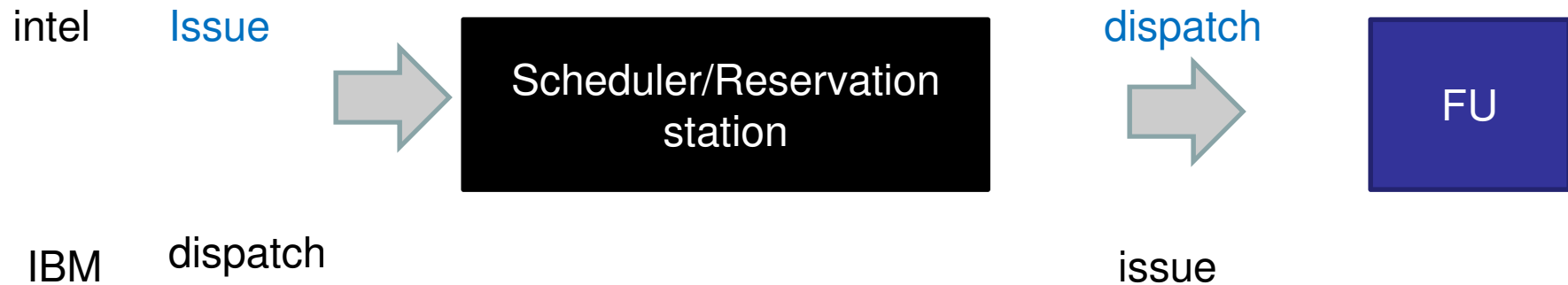
- Four-instruction fetch
- Two-instruction “dispatch”
- Five functional units
- “VMX128” execution “decoupled” from other units
- 14-cycle VMX dot-product
- Branch predictor:
- “4K” G-share predictor
- Unclear if 4KB or 4K 2-bit counters
- Per thread





# Issue and Dispatch

- Issue and Dispatch mean differently depending on companies, academia etc.





# Ann.

- Make-up class Wed 6-7 pm. The same classroom
- By Friday:
  - Student's information sheet
  - Presentation partner and topic information



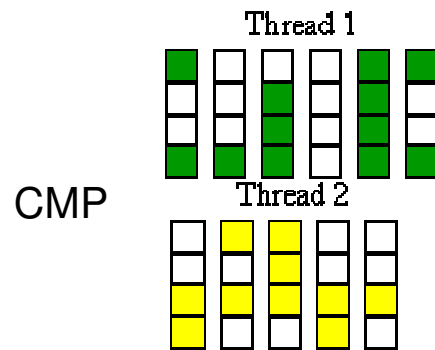
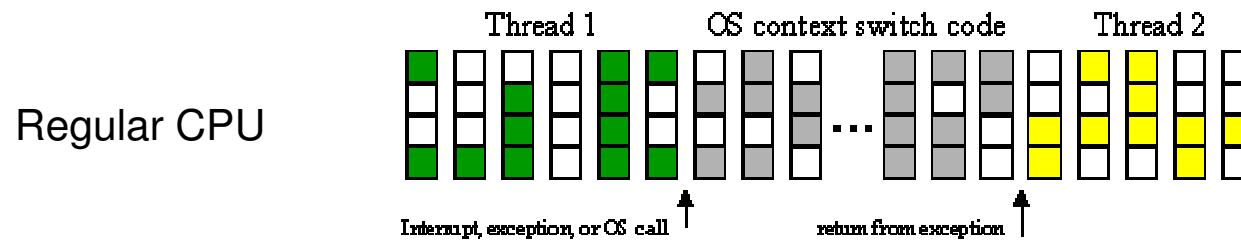
# BACKGROUND:SMT



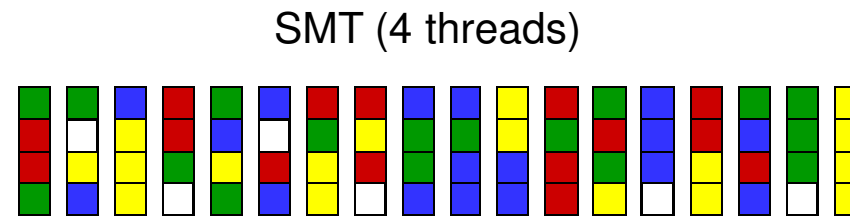
# Simultaneous Multi-Threading

- SMT: Idea is to use a single large uni-processor as a multi-processor

# SMT (2)



2x HW Cost



Approx 1x HW Cost





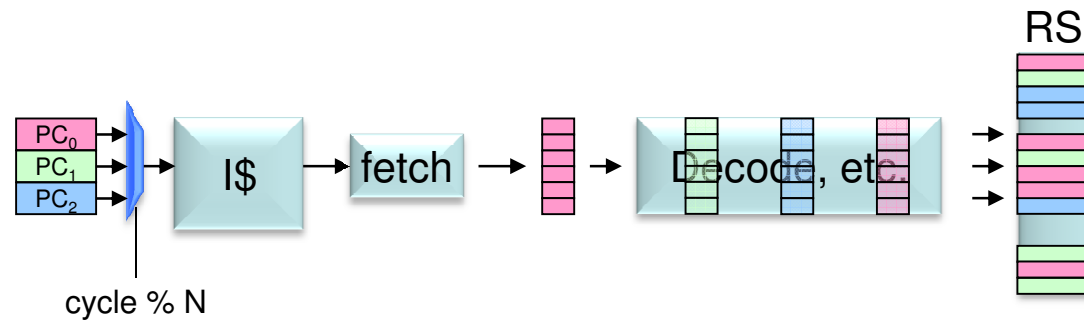
# Overview of SMT Hardware Changes

- For an N-way (N threads) SMT, we need:
  - Ability to fetch from N threads
  - N sets of architectural registers (including PCs)
  - N rename tables (RATs)
  - N virtual memory spaces
  - Front-end: branch predictor?: no, RAS? :yes
- But we don't need to replicate the entire OOO execution engine (schedulers, execution units, bypass networks, ROBs, etc.)



# SMT Fetch

- Multiplex the Fetch Logic

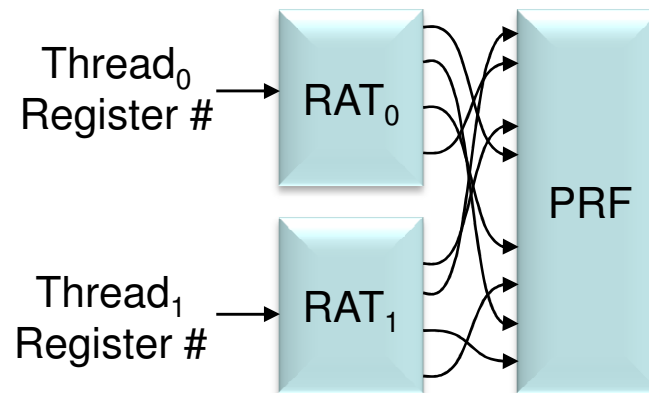


Can do simple round-robin between active threads, or favor some over the others based on how much each is stalling relative to the others



# SMT Rename

- Thread #1's R12  $\neq$  Thread #2's R12
  - separate name spaces
  - need to disambiguate





# SMT Issue, Exec, Bypass, ...

- No change needed

After Renaming

Thread 0:

Add R1 = R2 + R3  
 Sub R4 = R1 - R5  
 Xor R3 = R1 ^ R4  
 Load R2 = 0[R3]

Thread 0:

Add T12 = T20 + T8  
 Sub T19 = T12 - T16  
 Xor T14 = T12 ^ T19  
 Load T23 = 0[T14]

Shared RS Entries

Sub T5 = T17 - T2
Add T12 = T20 + T8
Load T25 = 0[T31]
Xor T14 = T12 ^ T19
Load T23 = 0[T14]
Sub T19 = T12 - T16
Xor T31 = T17 ^ T5
Add T17 = T29 + T3

Thread 1:

Add R1 = R2 + R3  
 Sub R4 = R1 - R5  
 Xor R3 = R1 ^ R4  
 Load R2 = 0[R3]

Thread 1:

Add T17 = T29 + T3  
 Sub T5 = T17 - T2  
 Xor T31 = T17 ^ T5  
 Load T25 = 0[T31]



# SMT Cache

- Each process has own virtual address space
  - TLB must be thread-aware
    - translate (thread-id, virtual page) → physical page
  - Virtual portion of caches must also be thread-aware
    - VIVT cache must now be (virtual addr, thread-id)-indexed, (virtual addr, thread-id)-tagged
    - Similar for VIPT cache
    - No changes needed if using PIPT cache (like L2)



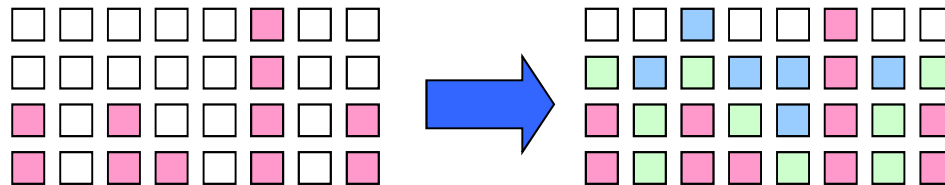
# SMT Commit

- Register File Management
  - ARF/PRF organization
    - need one ARF per thread
- Need to maintain interrupts, exceptions, faults on a per-thread basis
  - like OOO needs to appear to outside world that it is in-order, SMT needs to appear as if it is actually N CPUs

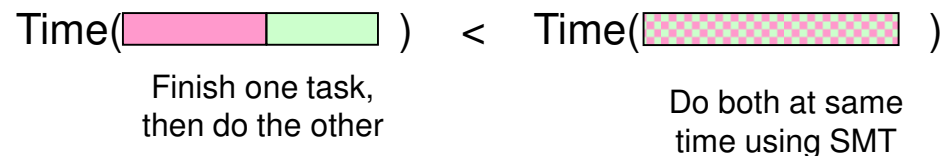


# SMT Performance

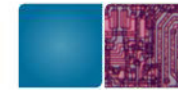
- When it works, it fills idle “issue slots” with work from other threads; throughput improves



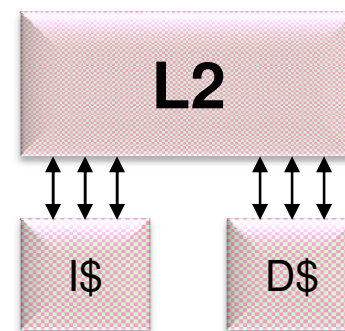
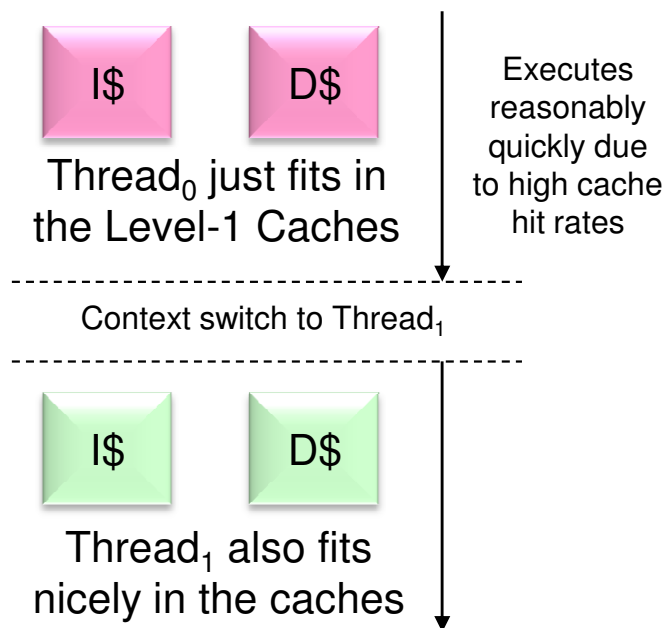
- But sometimes it can cause performance degradation!



# How?



- Cache thrashing



Caches were just big enough to hold one thread's data, but not two thread's worth

Now both threads have significantly higher cache miss rates





**XBOX 360 ...**



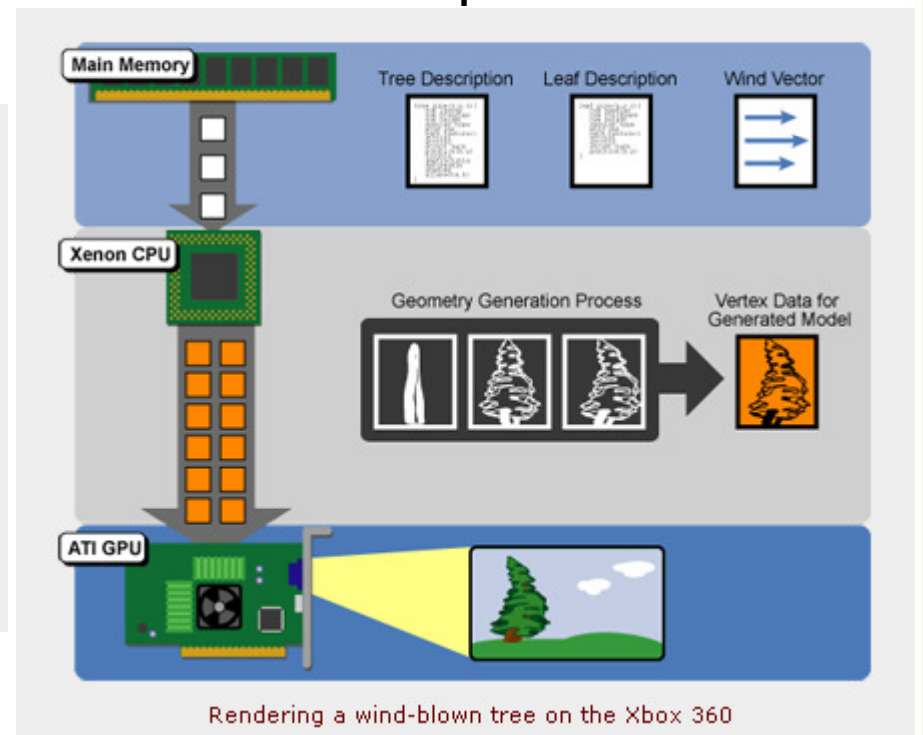
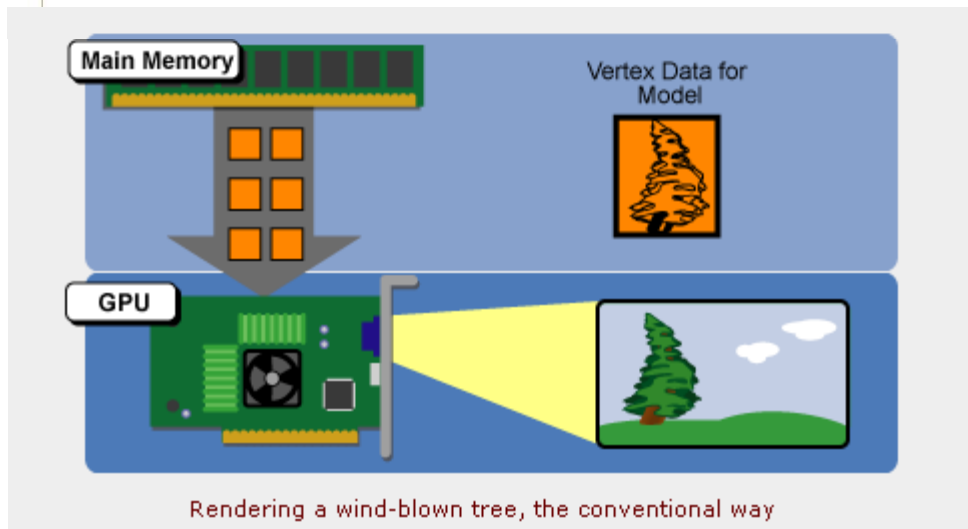
# VMX 128

- Four-way SIMD VMX 128 units:
  - FP, permute, and simple
- 128 registers of 128 bits each per hardware thread
- Added dot product instruction (simplifying the rounding of intermediate multiply results)
- 3D compressed data formats . Use compressed format to store at L2 or memory. 50% of space saving.
  - DXT1, DXT2/DXT3, and DXT4/DXT5
  - CTX1



# Procedural Synthesis

- Microsoft refers to this ratio of stored scene data to rendered vertex data as a **compression ratio**, the idea being that main memory stores a "compressed" version of the scene, while the GPU renders a "decompressed" version of the scene.





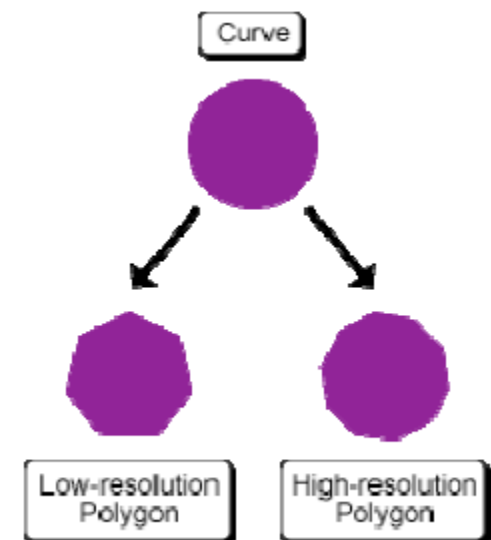
# The Benefits of Procedure Synthesis

- Scalable “virtual” artists
- Reduction of bandwidth from main memory to GPUs



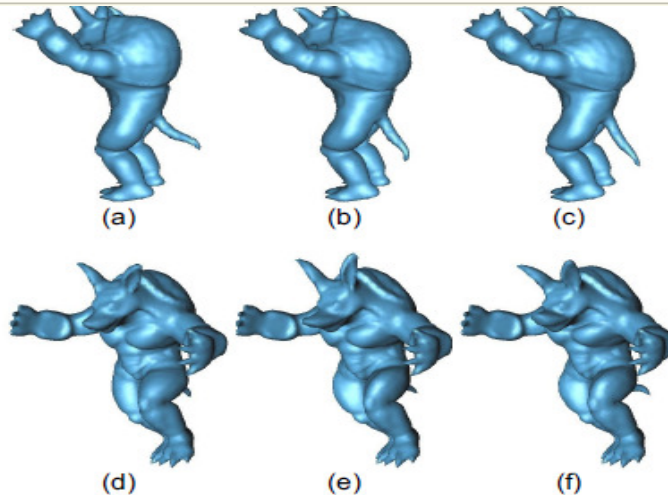
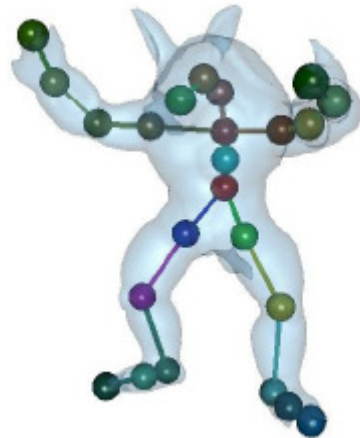
# Real-time Tessellation

- Tessellation: The process of taking a higher order curve and approximating it with a network of small flat surfaces is called tessellation.
- Traditional GPU: Artist
- Xbox 360: using Xeon
- Real time tessellation
  - Another form of data compression
  - Instead of list of vertex, stores them as higher order of curves
  - Dynamic Level of Detail (LOD)
    - Keep the total number of polygons in a scene under control



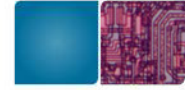


# Real-time Skinning



Images are from shi et al.'s "Example-based Dynamic Skinning in Real Time"

- Artists use standard tools to generate a character model a long with a series of key poses
- Model: a set of bones + deformable skins
- Xenon interpolate new poses as needed
- Skins are generated on the fly
- Xenon only sends the vertices that have changed to save bandwidth



# MEMORY SYSTEM: STREAM OPTIMIZATIONS



# Xbox 360 Memory Hierarchy

- 128B cache blocks throughout
- 32KB 2-way set-associative instruction cache (per core)
- 32KB 4-way set-associative data cache (per core)
- Write-through, lots of store buffering
- Parity
- 1MB 8-way set-associative second-level cache (per chip)
- Special “skip L2” prefetch instruction
- MESI cache coherence
- ECC
- 512MB GDDR3 DRAM, dual memory controllers
- Total of 22.4 GB/s of memory bandwidth
- Direct path to GPU (not supported in current PCs)



# Background: Prefetch



- Software Prefetch

- Non-binding prefetch instructions

```
for(ii=0; ii < 100; ii++){  
    Y[ii]=X[ii]+1  
}
```

```
for(ii=0; ii < 100; ii++){
```

```
    pref(X[ii+10]);
```

```
    Y[ii]=X[ii]+1
```

```
}
```

10 can vary depending on memory latency

- Hardware Prefetch

- Hardware detect memory streams and generate memory requests before demand requests



# xDCBT

- Extended data cache block touch
- Prefetch data but do not put L2
- Directly put data into L1
- Stream behavior applications
- Reducing L2 cache pollution



# Block Compression

- a texture compression technique for reducing texture size.

1	2	3
4	5	6
7	8	9

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

1	2	9	info
---	---	---	------



# Stream Optimizations

- 128B cache line size
- Write streaming:
  - L1s are write through, write misses do not allocate in L1
  - 4 uncacheable write gathering buffers per core
  - 8 cacheable, non-sequential write gathering buffers per core
- Read streaming:
  - 8 outstanding loads/prefetches.
  - xDCBT: Extended data cache block touch, bringing data directly to L1 , never store L2
  - Useful for non-shared data



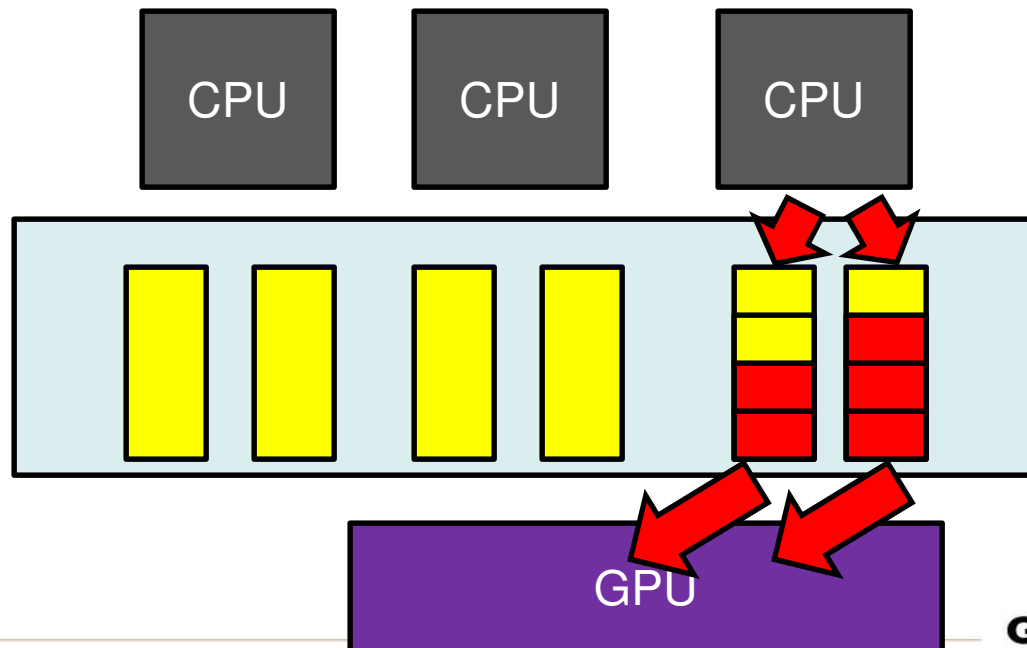
# CPU/GPU

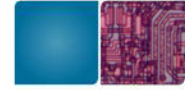
- CPU can send 3D compressed data directly to the GPU w/o cache
- Geometry data
- XPS support:
  - (1): GPU and the FSB for a 128-byte GPU read from the CPU
  - (2) From GPU to the CPU by extending the GPU's tail pointer write-back feature.



# Cache-set-locking

- Threads owns a cache sets until the instructions retires.
- Reduce cache contention.
- Common in Embedded systems
- Use L2 cache as a FIFO buffer: sending the data stream into the GPU





# Tail Pointer write-back

- Tail pointer write-back: method of controlling communication from the GPU to the CPU by having the CPU poll on a cacheable location, which is updated when a GPU instruction writes an updated to the pointer.
- Free FIFO entry
- System coherency system supports this.
- Reduce latency compared to interrupts.
- Tail pointer backing-store target

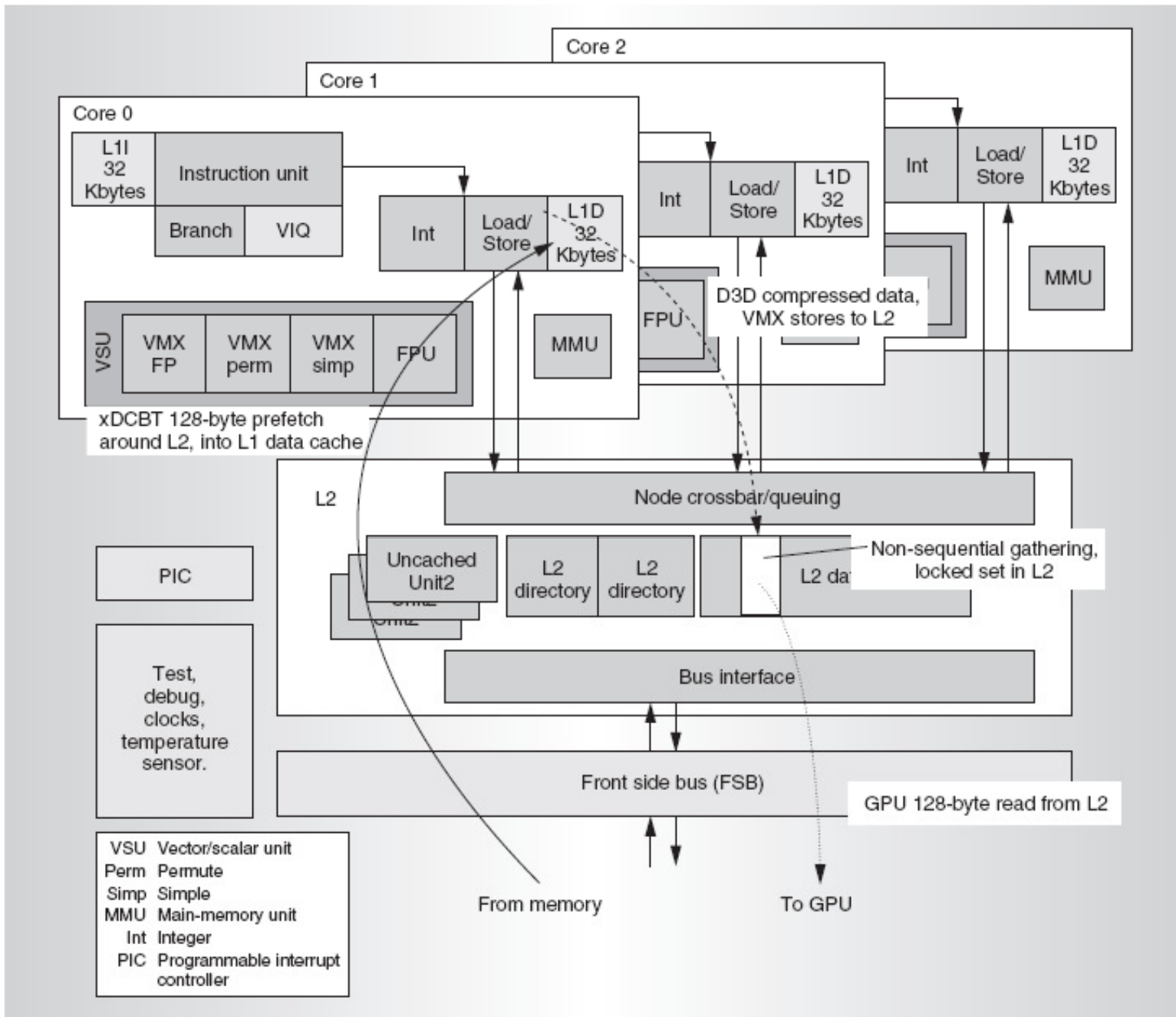
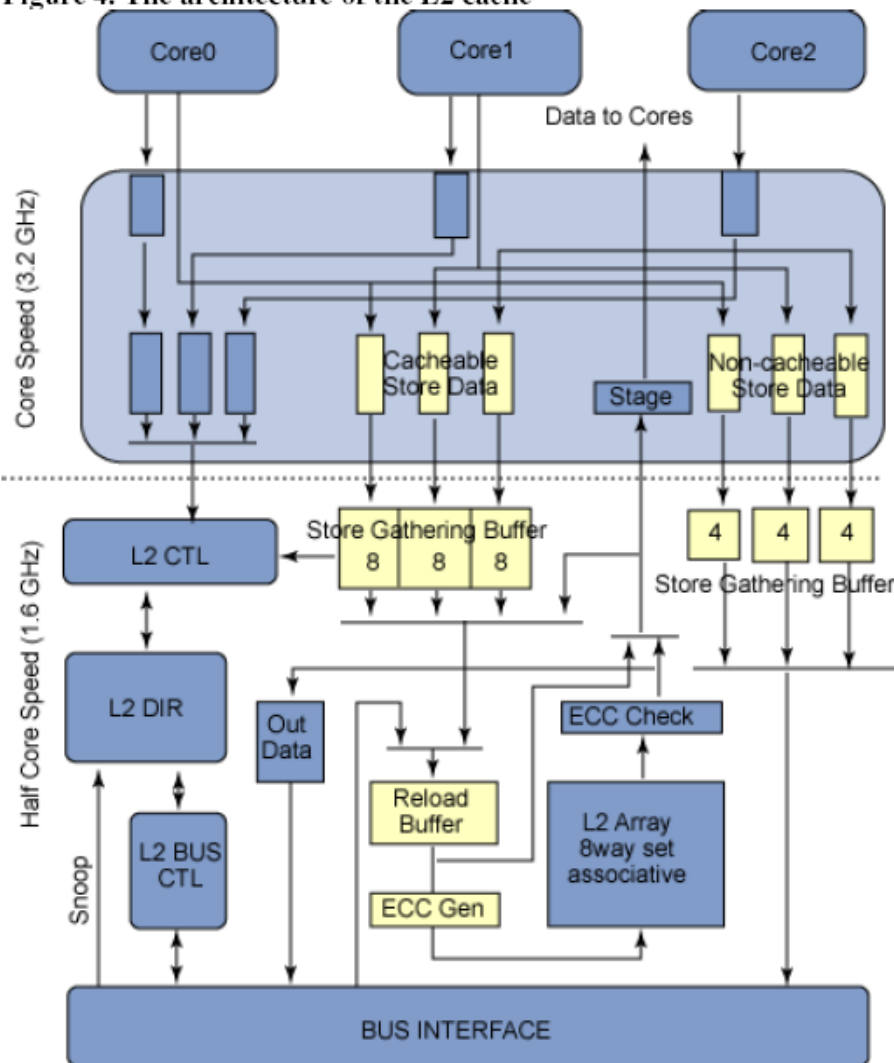


Figure 4. CPU cached data-streaming example.



# Memory systems

Figure 4. The architecture of the L2 cache





# Non-Blocking Caches

- Hit Under Miss
  - Allow cache hits while one miss in progress
  - But another miss has to wait
- Miss Under Miss, Hit Under Multiple Misses
  - Allow hits and misses when other misses in progress
  - Memory system must allow multiple pending requests
- MSHR (Miss Information/Status Holding Register):  
Stores unresolved miss information for each miss that will be handled concurrently.



# Ann.

- Make-up class, today 6 pm, the classroom
  - CUDA Programming
- By Friday
  - Student's information sheet
  - Presentation:
    - 20 min about any game consoles (hardware focus, architecture focus)
    - Partner's name and topics (more than one)
    - Email addresses