COMP 8551 Advanced Games Programming Techniques

Borna Noureddin, Ph.D.

British Columbia Institute of Technology

Hardware and Assembly Language

Overview

Hardware concepts

- Definitions

- Design and architectures

Hardware Concepts

CPU, GPU, GPGPU, FPGA

- All integrated circuits
- FPGA: program arrays of transistors
- CPU/GPU/GPGPU/FPU: general purpose or specialized microprocessors
- CPU: most general, FPU/GPU: most specific
- Traditionally: processing (physics, AI, etc.) on CPU, rendering (geometry, shading, lighting, etc.) on GPU

Hardware Concepts

CPU, GPU, GPGPU, FPGA

- Game engines used to process some graphics (e.g., lighting) on CPU, but now moving away from that: very efficient, empty refrigerator is useless
- CPU still important: concurrent tasks (firewall, networking, I/O, etc.), physics, AI – very efficient, full refrigerator still means bigger refrigerator may be needed

Hardware Concepts

CPU, GPU, GPGPU, FPGA

- Over past decade, graphics and gaming drove Moore's Law
- Demand for better GPUs increased dramatically: GPUs became arguably more powerful than CPUs
- GPGPUs attempt to combine both, but require radical change in how one approaches programming

RISC, CISC, SIMD

CISC

- Complex instruction set computer
- CPU design strategy
- Single instructions execute several low-level operations

http://en.wikipedia.org/wiki/Complex_instruction_set_computer

RISC

- Reduced instruction set computing
- CPU design strategy
- Small set of simple instructions that can run very fast

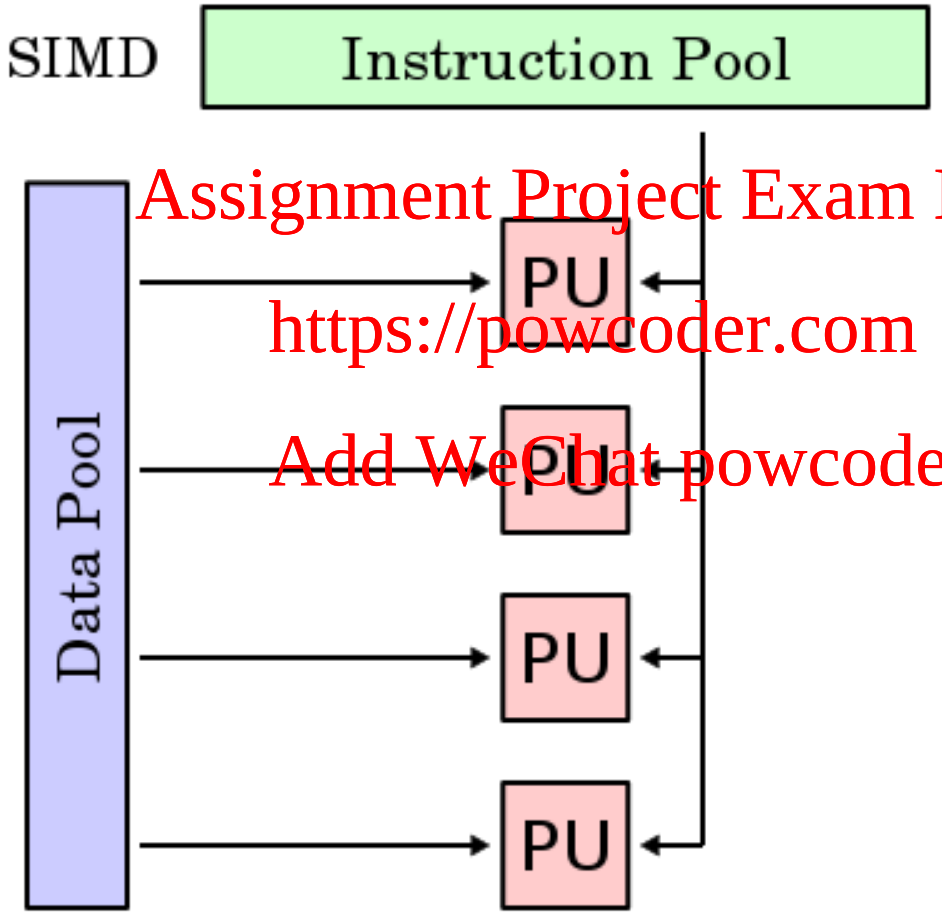
http://en.wikipedia.org/wiki/Reduced_instruction_set_computer

RISC, CISC, SIMD

SIMD

- Multiple processing elements that perform the same operation on multiple data simultaneously
- Examples:
  - Change image brightness: each pixel (RGB) read from memory, value added/subtracted, result written back to memory
  - Can process multiple pixels simultaneously: single instruction to fetch multiple memory locations
  - Value added to all locations referenced by instruction at the same time (high level of parallelism)

RISC, CISC, SIMD



Multiprocessor vs Multicore

- Traditionally, for true parallelism, needed multiple processors (CPUs)
- Allows each CPU to work independently
- Data synchronization was always challenging
- Bottlenecks included bus speeds and lack of parallel code (or good compilers that balance loads)
- But, more CPUs always improves speed if they are utilized at all (even if not in the most efficient way): this is why multi-threading is so important!
- Main obstacle was cost

Multiprocessor vs Multicore

- Multicore means multiple processing units on a single chip
- In practice, lower the clock speed to save power consumption and heat, but much lower cost, making it now affordable for consumer-level products to have multiple "processors"
- Turns out multicore is often more efficient (e.g., share data bus)
- Newer game consoles (Xbox360, PS3) are multicore

Note: sometimes, bottleneck is not processing power, but bus speed/architecture and/or I/O

Hardware Definitions

Instruction set

- Instructions: mnemonics (aka machine language or opcodes) used to identify commands to CPU to carry out operations using transistors
- Native data types: depends on system architecture, bus width, etc.
- Registers: on-chip memory reserved for operands of instructions
- Addressing modes: methods of accessing instructions and other memory (e.g., direct, indirect, offset, etc.)

Hardware Definitions

Instruction set (cont'd)

- Interrupts and exceptions: mechanisms provided to allow change of regular sequential flow of operations
- All of this is paced by the system clock and hardware logic
- Micro-architecture: microprocessor design techniques used to implement instruction set (different micro-architectures can share common instruction set).
- Micro-code: instructions broken down into sub-operations that can be pipelined

Hardware Definitions

Pipelining

- Set of data processing elements connected in series, but often executed in parallel (which requires some buffering of shared data)
- Instruction pipeline: overlapping execution of multiple instructions with shared circuitry (e.g., units for decoding used for one instruction while units for arithmetic or register fetch used for another).

Hardware Definitions

Microcode

- Hardware instructions or data structures used to implement higher level machine code instructions
- Resides in specialized high-speed memory (not necessarily cache)
- Provides layer of abstraction so instructions can be designed independently from underlying electronics
- Related terms: microprogramming, microprogram
- Can also be used for hardware emulation or support for legacy hardware without having to include old circuitry
- Sometimes used as synonym for firmware

Hardware Definitions

Instruction set extensions

- MMX, SSE, SSE1-4, etc.
- With SIMD came various attempts to extend existing instruction sets
- They added
  - longer registers (e.g., 128-bit on 32-bit systems)
  - instructions to perform single operation on multiple memory locations simultaneously (vector operations)
  - more complex math operations at machine code level
  - DSP and thread management instructions
  - geometry instructions
  - complex integer arithmetic operations

Hardware Definitions

CPU cache

- Cache: special, more expensive but much faster access memory
- Used by CPU to reduce average time to access memory (which can be very expensive, depending on the type of memory and bus architecture)
- Stores copies of data from most frequently used memory locations
- Various sophisticated heuristics and algorithms used to optimize cache performance (trade-off between keeping things around too long or not long enough)

Hardware Definitions

CPU cache (cont'd)

- Instruction cache: fetching instructions stored in main memory
- Data cache: transfer of data between main memory and internal registers
- Translation lookaside buffer: virtual-physical address translation

Review

Hardware concepts

- Definitions

- Design and architectures

END