Parallel Computing with GPUs

GPU Architectures
Part 1 - Introduction to GPUs

The University Of Sheffield.

Dr Paul Richmond
http://paulrichmond.shef.ac.uk/teaching/COM4521/

This Lecture (learning objectives)

- ❑Introduction to GPUs
    - ❑Compare latency with throughput and identify how this relates to CPU and GPU architectures
    - ❑Identify examples from Flynn’s taxonomy
    - ❑Classify the taxonomy of a GPU

Latency vs. Throughput

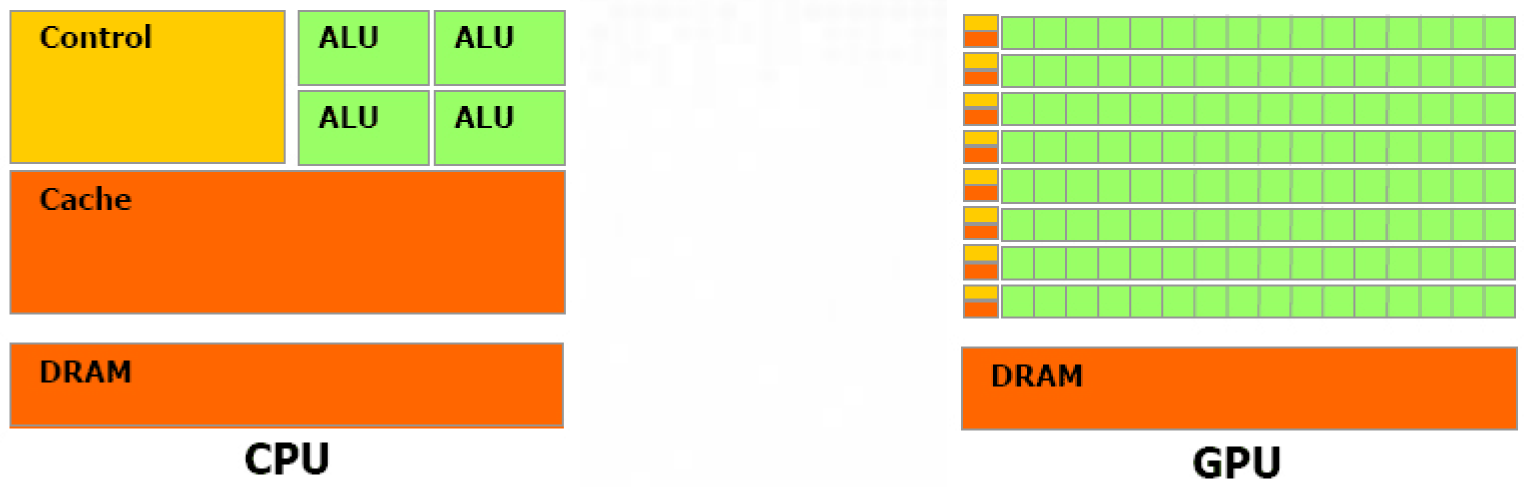
- Latency: The time required to perform some action
  - Measure in units of time
- Throughput: The number of actions executed per unit of time
  - Measured in units of what is produced

- E.g. *An assembly line manufactures GPUs. It takes* ***6 hours*** *to manufacture a GPU but the assembly line can manufacture* ***100 GPUs per day****.*

CPU vs GPU

- CPU
  - Latency oriented
  - Optimised for serial code performance
  - Good for single complex tasks
- GPU
  - Throughput oriented
  - Massively parallel architecture
  - Optimised for performing many similar tasks simultaneously (data parallel)

CPU vs GPU



- ❑Large Cache
  - ❑Hide long latency memory access
- ❑Powerful Arithmetic Logical Unit (ALU)
  - ❑Low Operation Latency
- ❑Complex Control mechanisms
  - ❑Branch prediction etc.

- ❑Small cache
  - ❑But faster memory throughput
- ❑Energy efficient ALUs
  - ❑Long latency but high throughput
- ❑Simple control
  - ❑No branch prediction

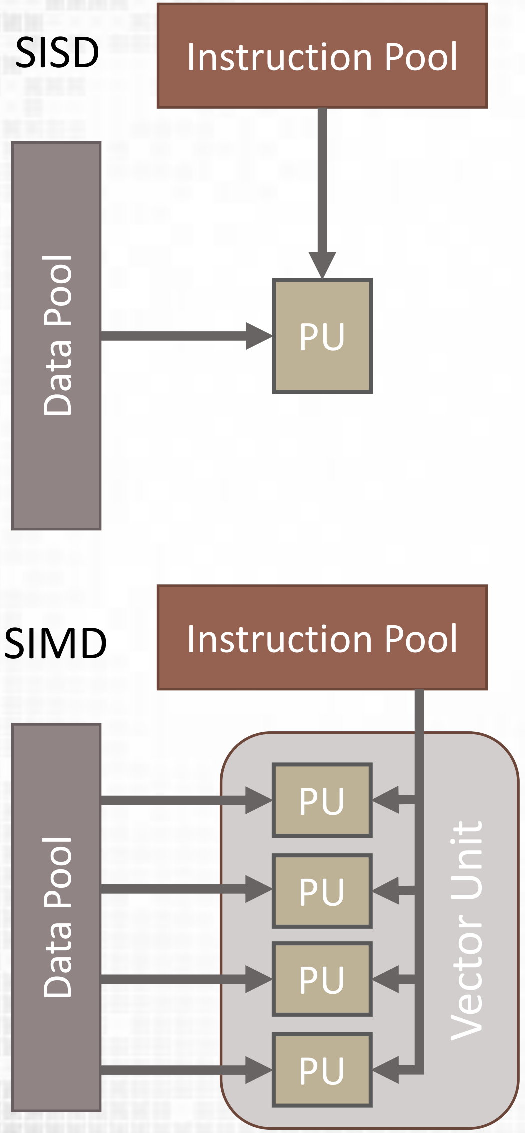
The University Of Sheffield.

Data Parallelism

- Program has many similar threads of execution
  - Each thread performs the same behaviour on different data
  - Good for high throughput
- We can classify an architecture based on instructions and data (Flynn's Taxonomy)
  - Instructions:
    - Single instruction (SI)
    - Multiple Instruction (MI)
    - *Single Program (SP)*
    - *Multiple Program (MP)*

    } *Not part of the original taxonomy*
  - Data:
    - Single Data (SD) – w.r.t. *work item not necessarily single word*
    - Multiple Data (MD)
  - e.g. SIMD = Single Instruction and Multiple Data

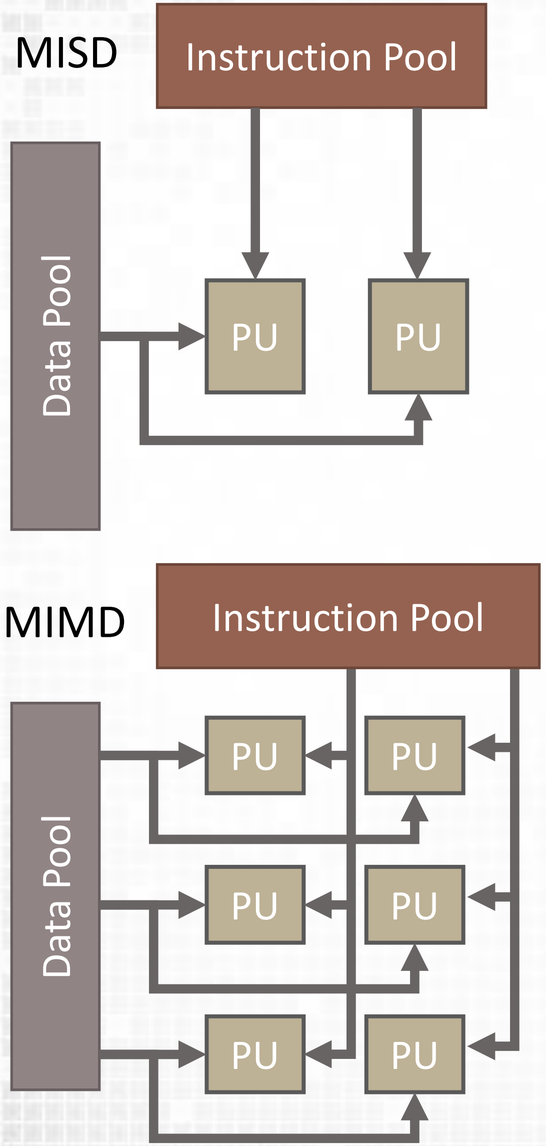
SISD and SIMD



- □SISD
  - □Classic von Neumann architecture
  - □PU = Processing Unit
- □SIMD
  - □Multiple processing elements performing the same operation simultaneously
  - □E.g. Early vector super computers
  - □Modern CPUs have SIMD instructions
    - □But are not SIMD in general

The University Of Sheffield.

MISD and MIMD



- ❑MISD
  - ❑E.g. Pipelined architectures
- ❑MIMD
  - ❑Processors as functionally asynchronous and independent
  - ❑Different processors may execute different instructions on different data
  - ❑E.g. Most parallel computers
  - ❑E.g. OpenMP programming model

The University Of Sheffield.

SPMD and MPMD

- SPMD
  - Multiple autonomous processors simultaneously executing a program on different data
  - Program execution can have an independent path for each data point
  - E.g. Message passing on distributed memory machines.
- MPMD
  - Multiple autonomous processors simultaneously executing at least two independent programs.
  - Typically client & host programming models fit this description.
  - E.g. Sony PlayStation 3 SPU/PPU combination, Some system on chip configurations with CPU and GPUs

Taxonomy of a GPU

❑What taxonomy best describes data parallelism with a GPU?

❑SISD?

❑SIMD?

❑MISD?

❑MIMD?

❑SPMD?

❑MPMD?

Taxonomy of a GPU

- ❑What taxonomy best describes data parallelism with a GPU?

- ❑**Obvious Answer**: SIMD
- ❑**Less Obvious answer**: SPMD
- ❑**Slightly confusing answer**: SIMT (Single Instruction Multiple Thread)
  - ❑This is a combination of both it differs from SIMD in that;
    1) Each thread has its own registers
    2) Each thread has multiple addresses
    3) Each thread has multiple flow paths
  - ❑We will explore this in more detail when we look at the hardware!
  - ❑http://yosefk.com/blog/simd-simt-smt-parallelism-in-nvidia-gpus.html

Summary

- ❑Introduction to GPUs
  - ❑Compare latency with throughput and identify how this relates to CPU and GPU architectures
  - ❑Identify examples from Flynn’s taxonomy
  - ❑Classify the taxonomy of a GPU

- ❑Next Lecture: Programming GPUs