Machine Architecture - Lecture 10

Ioannis Ivrissimtzis

ioannis.ivrissimtzis@durham.ac.uk

Multiprocessors

Slide material acknowledgements: Magnus Bordewich

Extra functionality

In MIPS the Float Processing Unit (FPU) was originally a separate chip.

This approach of adding additional chips to handle specific tasks has been used for other purposes too:

Graphics 'cards' or GPUs

FPGA

DSP (digital signal processors) in smartphones

Apple M7 chip

Crypto chips

And as with FPU with time these additional functionalities have been incorporated into the CPU chip.

GPUs

Many graphics tasks involve applying the same transformation to a large number of pixels, either on screen or in memory.

Adjusting brightness on a 12 megapixel image:

12,000,000 pixels

4 bytes of memory per pixel (RGBA).

Wastes a lot of time fetching and decoding the same instruction 48,000,000 times.

At 30fps you would save 1,440,000,000 fetch-and-decodes per second if you could apply one instruction to all the data at once.

GPUs

Special massively parallel computing units.

Holds two buffers for the screen/image pixels: a live and a working buffer.

Do updates on the working buffer then swap the display to that (making it the live buffer) and start working on the now unused previous live buffer.

Can be very fast for naturally SIMD operations, e.g. updating brightness, or computing shading or texture of each pixel given shared access to the data.

Not very flexible and not able to cope with branching.

SIMD

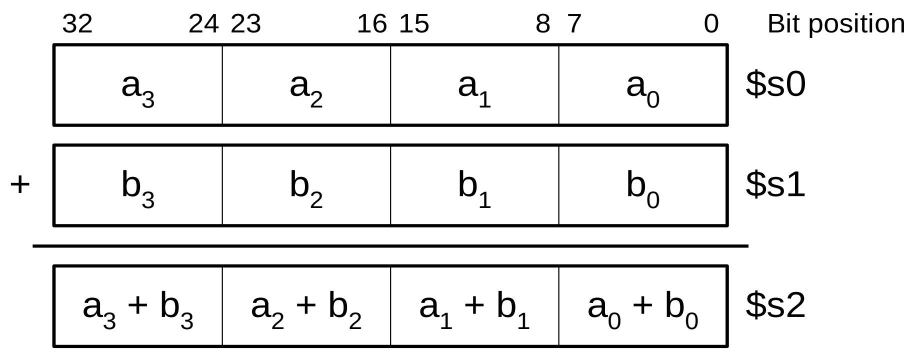
Single Instruction Multiple Data (SIMD).

Single instruction acts on multiple pieces of data at once.

Perform short arithmetic operations (also called packed arithmetic).

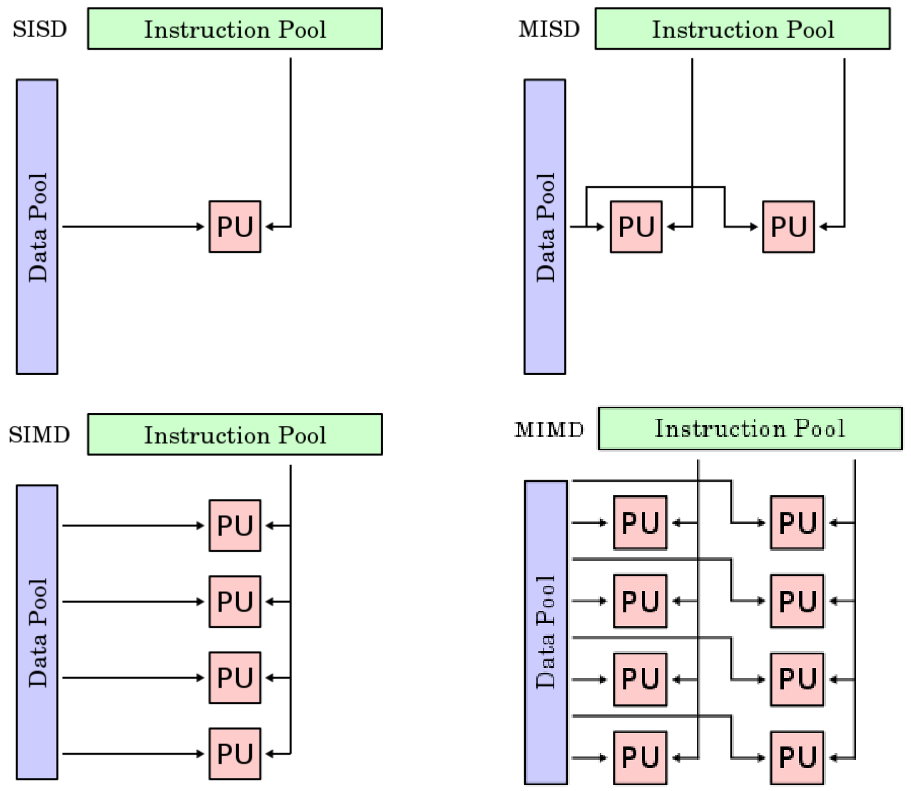
Can do some SIMD in CISC CPUs (e.g. Intel MMX), for example: add four 8-bit elements.

```
padd8 $s2, $s0, $s1
```



Flynn's taxonomy

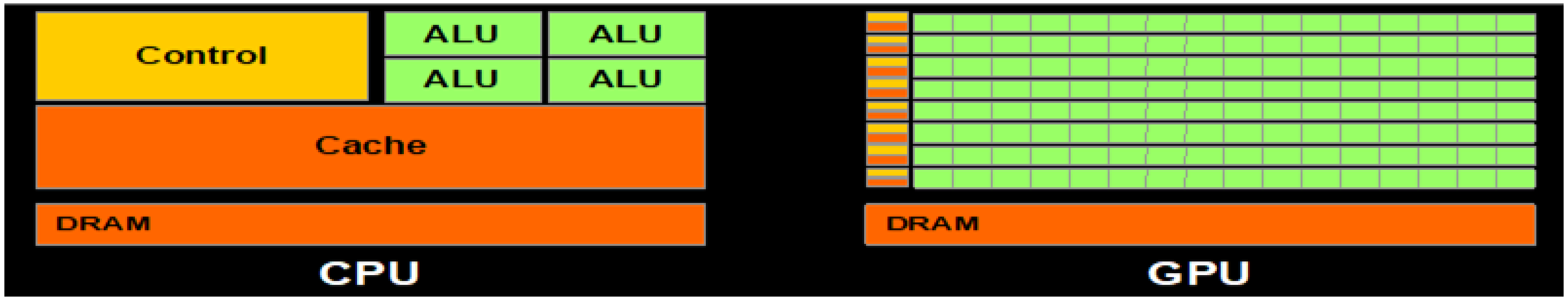
Flynn's taxonomy distinguishes between 4 classes of computer architectures.



CPUs vs GPUs

CPUs use task parallelism.	GPUs use data parallelism.
Multiple tasks running different instructions organized in threads.	SIMD model.
Relatively few but heavyweight threads.	Lightweight threads.
Each thread managed and scheduled explicitly.	Threads managed and scheduled by hardware.
Each thread is individually programmed.	Programming done for batches of threads (e.g. one pixel shader per group of pixels, or draw call).

CPUs vs GPUs

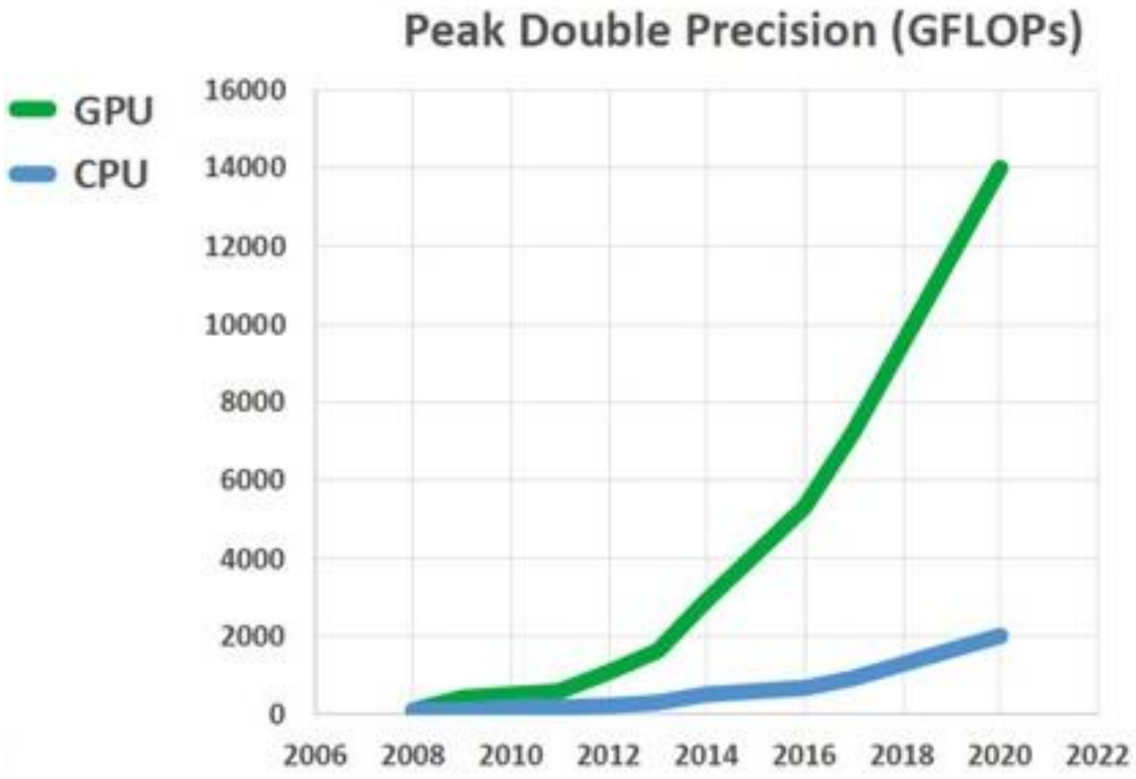


To reduce latency, CPUs use lots of cache.

To enable complex operations, CPUs have large ALUs.

To enact complex flow control CPUs have large areas of control circuitry.
GPUs are high latency, so can have many more (smaller) ALUs.

CPUs vs GPUs



nextplatform.com

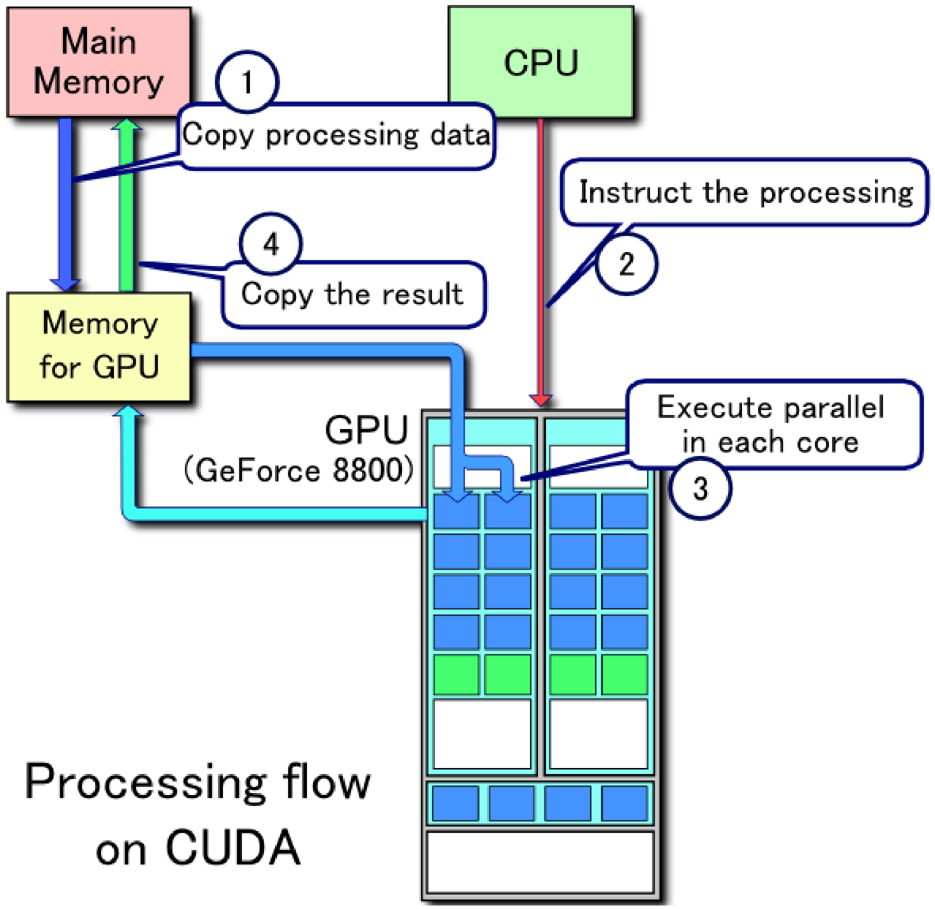
General purpose GPU computing

Recent drive to make use of the massive parallel capabilities of graphics cards for general purpose computation.

Known as GPGPU computing.

Implemented primarily as CUDA (Compute Unified Device Architecture) by NVIDIA.

Also OpenCL.



Threading

Process: program running on a computer.

Multiple processes can run at once: e.g., surfing the Web, playing music, writing a paper.

Thread: part of a program.

Each process has multiple threads: e.g., a word processor may have threads for typing, spell checking, printing.

Multithreading

Multiple copies of architectural state in the CPU.

Multiple threads active at once: when one thread stalls, another runs immediately.

Can save significant time in saving and restoring architectural states.

If one thread can’t keep all execution units busy, another thread can use them.

Does not increase instruction-level parallelism (ILP) of single thread, but increases throughput.