Wigner GPU Lab

Modern Computing Hardware

Overview of modern CPU and GPU architectures

Lectures on Modern Scientific Programming
Wigner RCP
23-25 November 2015

D. Berényi – M. F. Nagy-Egri

Overview

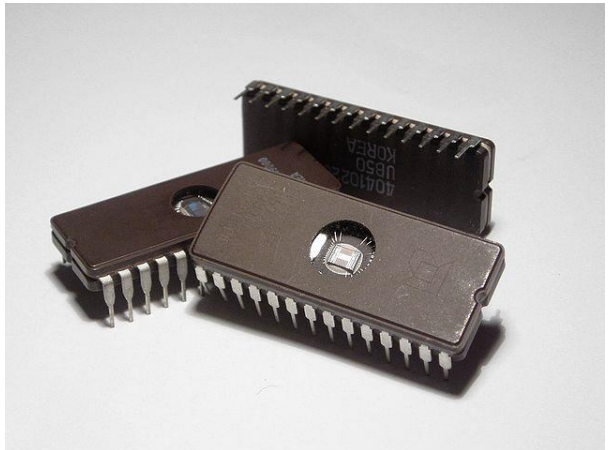
- Computing hardware architectures
- Strengths and weaknesses
- Instruction sets
- Timings

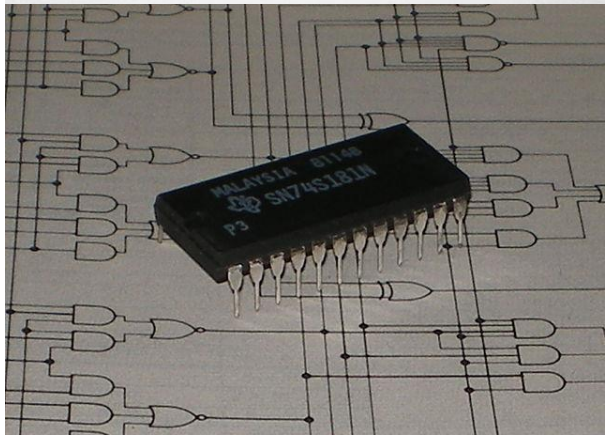
Brief History

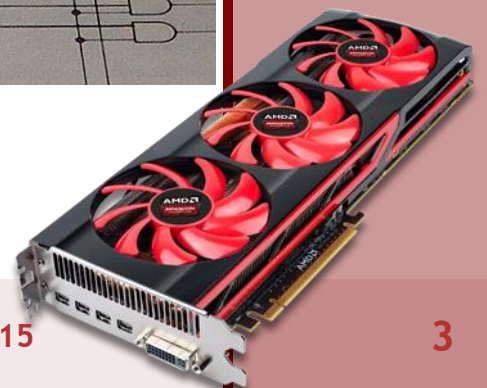
- With the advent of integrated circuits increasingly complex microelectronics become possible

- Production is only cheap in large quantities, but target applications are highly different…

- Programmable microprocessors appeared







Hardware architectures

Main architectures of microprocessors:

x86

- CPUs in desktop computers and notebooks

ARM

- Mobile, handheld devices
- Embedded systems

GPU

- Graphics accelerator cards

Operation

How does a microprocessor work?

- It is connected to a memory and peripherals via some bus
- Reads instructions from the memory sequentially
- Operates on data according to the instruction
- E.g. read, write, do arithmetic…

Instruction sets

A microprocessor has a fix instruction set

- Data movement between memory and internal registers
- Arithmetic (+, -, /, *), Logic (and, or, xor, neg)
- Branching (jump to a point in the program)

- Complex tasks should be composed from these instructions

- If there is a built-in instruction for a specific task in the processor, then that instruction is the most efficient way to do that task on that chip!

Instruction sets

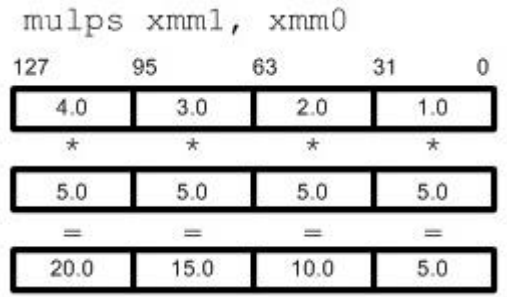
Examples, what can an x86 processor do:

- Data movement
  (memory, stack, registers, I/O)
- Arithmetic operations on (signed, unsigned) integers
  (+, -, *, /, % (modulo))
- Logic on integers (and, or, not, xor), shift, rotate
- Conditional / unconditional jump, subroutine call
- Arithmetic on floating point numbers
  (+, -, *, /, abs, sqrt, sin, cos, tan, atan,
  $2^x, 2^x - 1,\ y \cdot \log_2 x,\ y \cdot \log_2 x + 1$)

Instruction sets

Multimedia applications made necessary some composite operations that are also of scientific interest:

- Single Instruction Multiple Data (SIMD) instructions (SSE, AVX)
- These can carry out some subset of the earlier operations on multiple (typically 4, recently 8) data at the same time



Execution

Two strategies for task execution

Pizza delivery example:

- Do you want your pizza hot?
  - Low latency
- Do you want your pizza to be cheap?
  - High throughput (many pizzas / hour)

Execution

Two strategies for task execution

Pizza delivery example:

- Do you want your pizza hot?
  - Low latency → CPUs
- Do you want your pizza to be cheap?
  - High throughput (many pizzas / hour) → GPUs

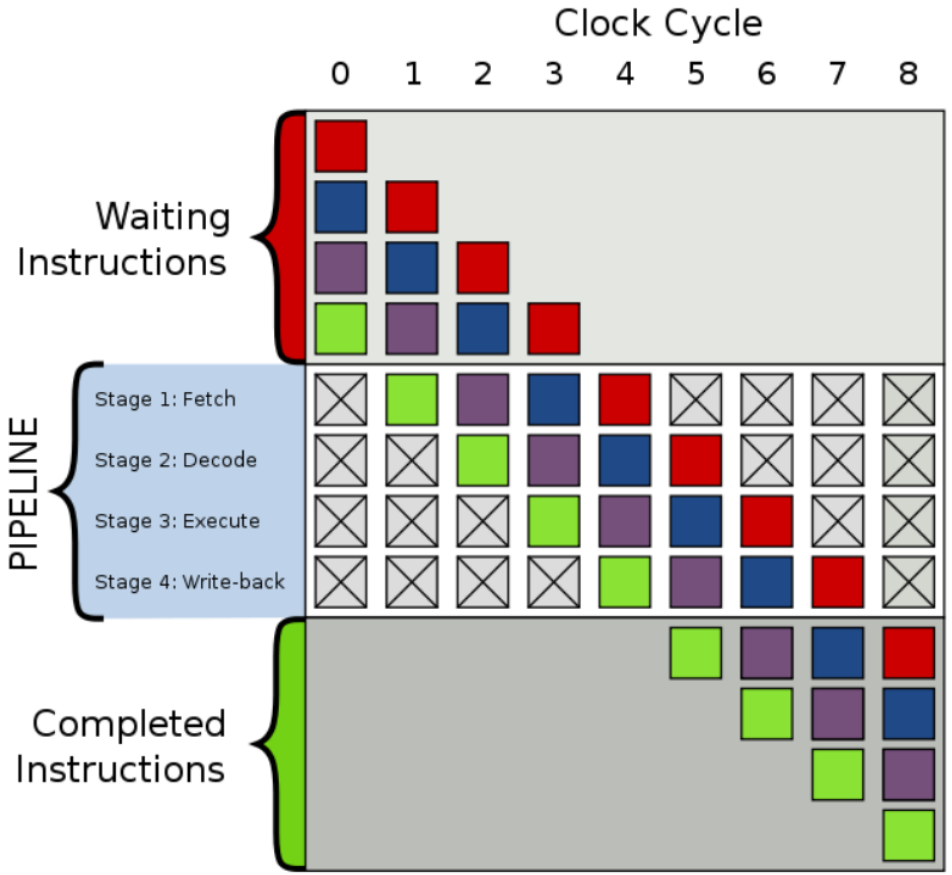
CPU execution

- CPUs are designed for generic tasks, especially running an operating system
- CPUs are fast relative to memory access: data availability is a bottle neck
- CPUs need to solve complex tasks
- Need to support an instruction set that is 30+ years old for compatibility

CPU execution

Some consequences

- CPUs are using an instruction pipeline
- Different units for specific tasks
  integer arithm., floating arithm., branching, memory access



CPU execution

Some consequences

Stall of the pipeline should be avoided at all costs!

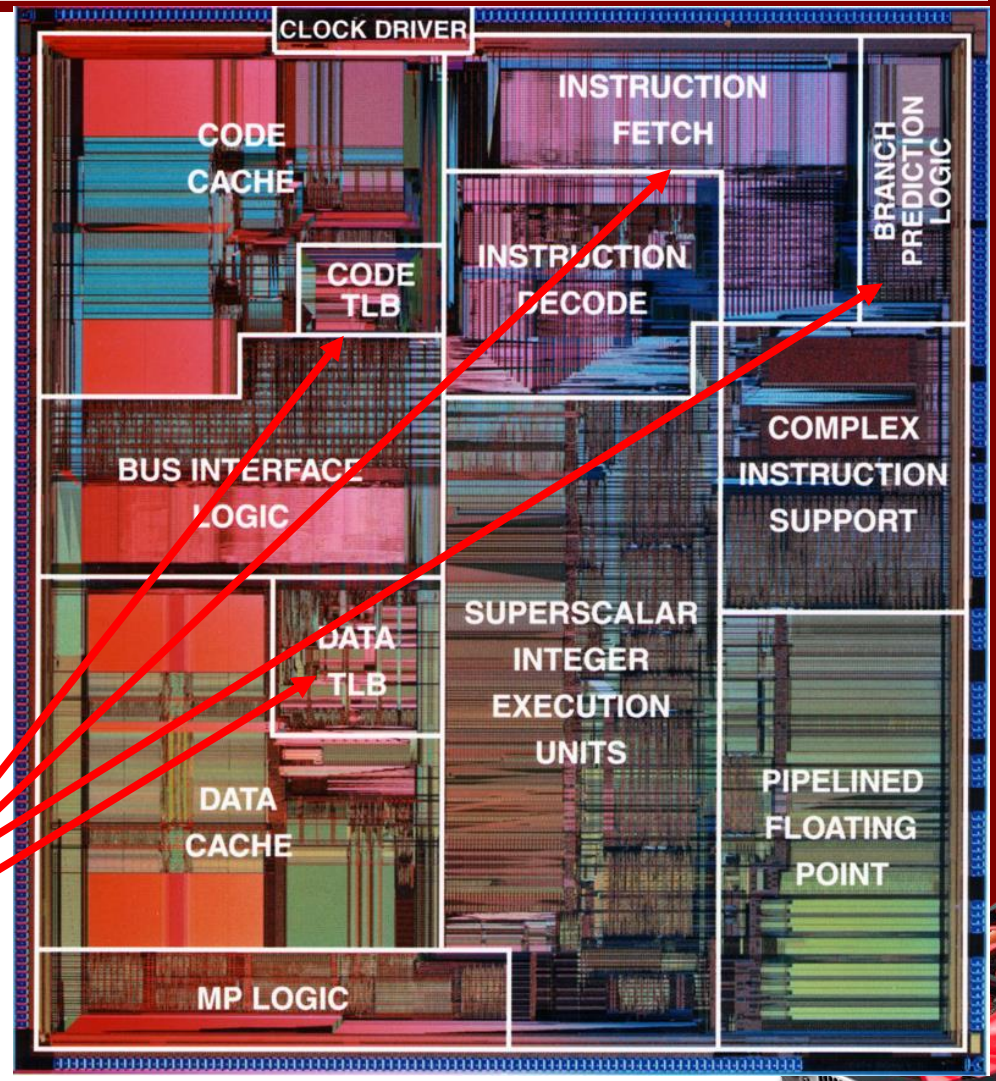
- Memory bottle neck necessities caching of instructions and data
- Read ahead when accessing, keep in cache temporary results

CPU execution

Some consequences

Stall of the pipeline should be avoided at all costs!

- What to cache when a branch will occur?
  - Extremely complicated branch prediction logic, speculative execution, out-of-order execution, translation look-aside logic, etc...

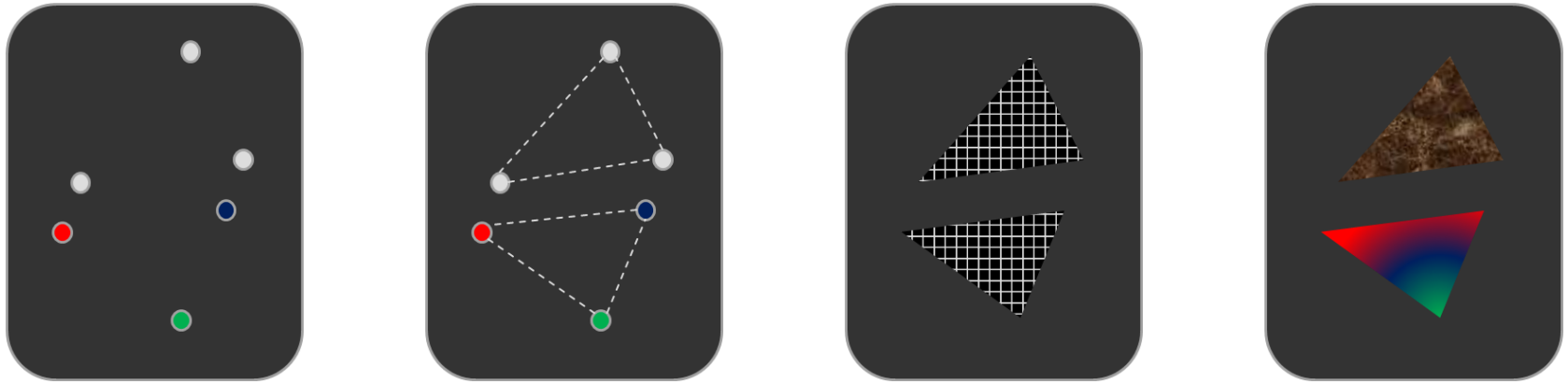


GPUs

GPUs are completely contrast to CPUs, they are designed for throughput of simple tasks!

- Minimal caching
- Virtual instruction set
- No complex branching logic
- Many simple, identical execution units
- Clever scheduler possible to avoid stall

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & b_{14} \\ a_{21} & a_{22} & a_{23} & b_{24} \\ a_{31} & a_{32} & a_{33} & b_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix}$$

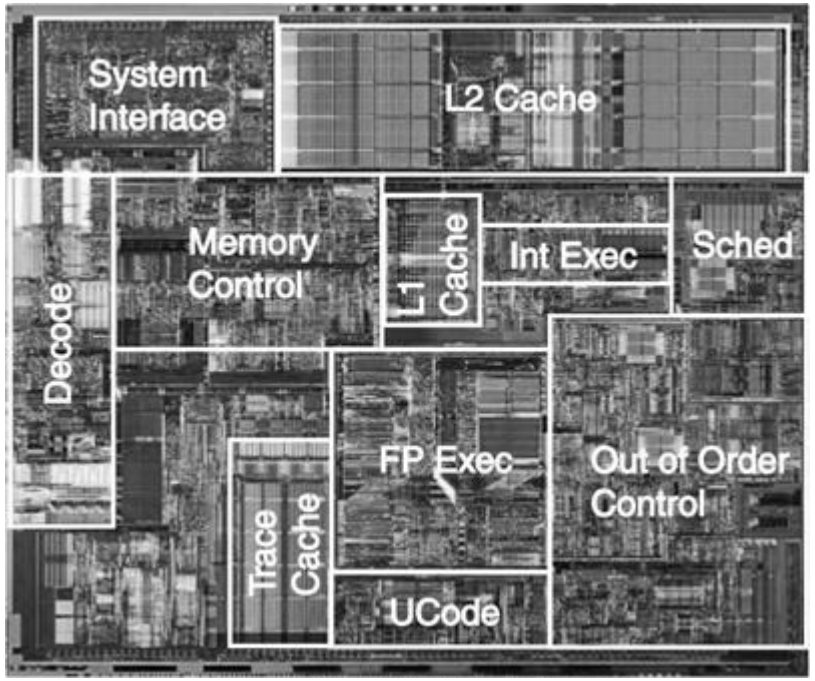


GPUs

Modern GPUs are capable of delivering:

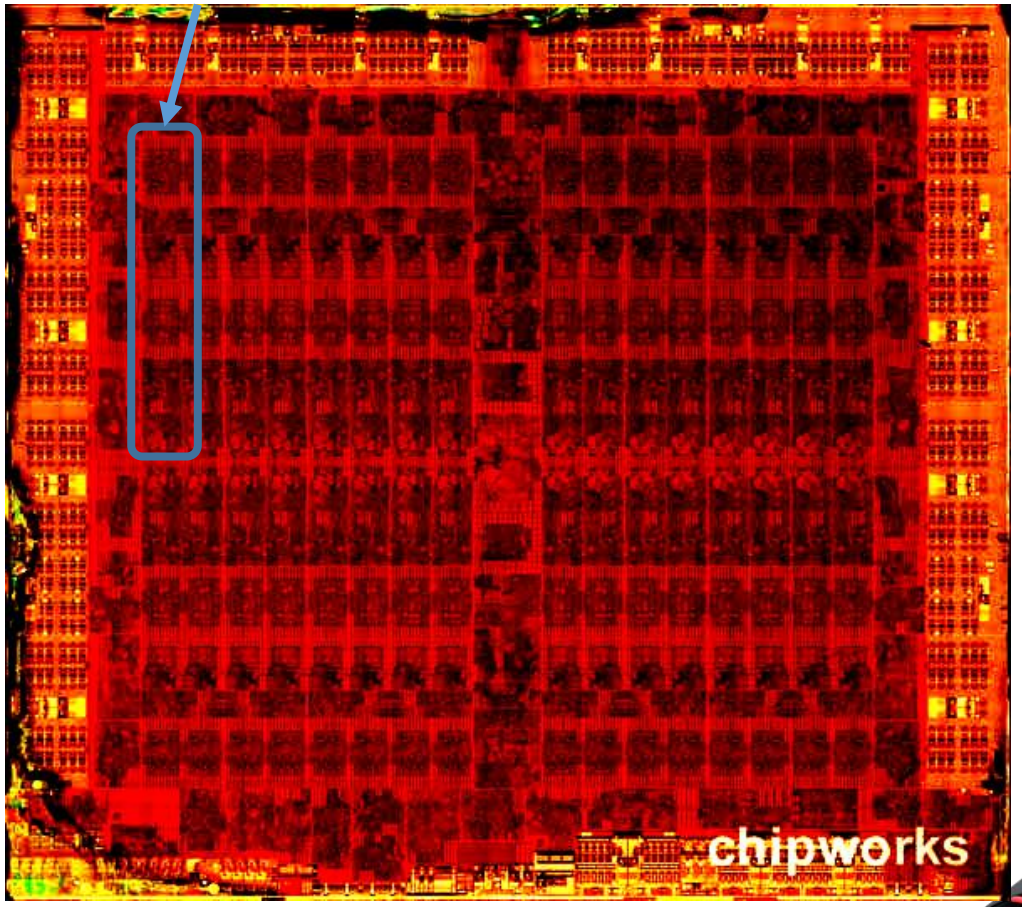
- Graphics rendering and Video encoding/decoding
- Especially effective operations on 4 vectors both integer and IEEE compatible floating point
- Thread synchronization and atomic operations

CPUs and GPUs



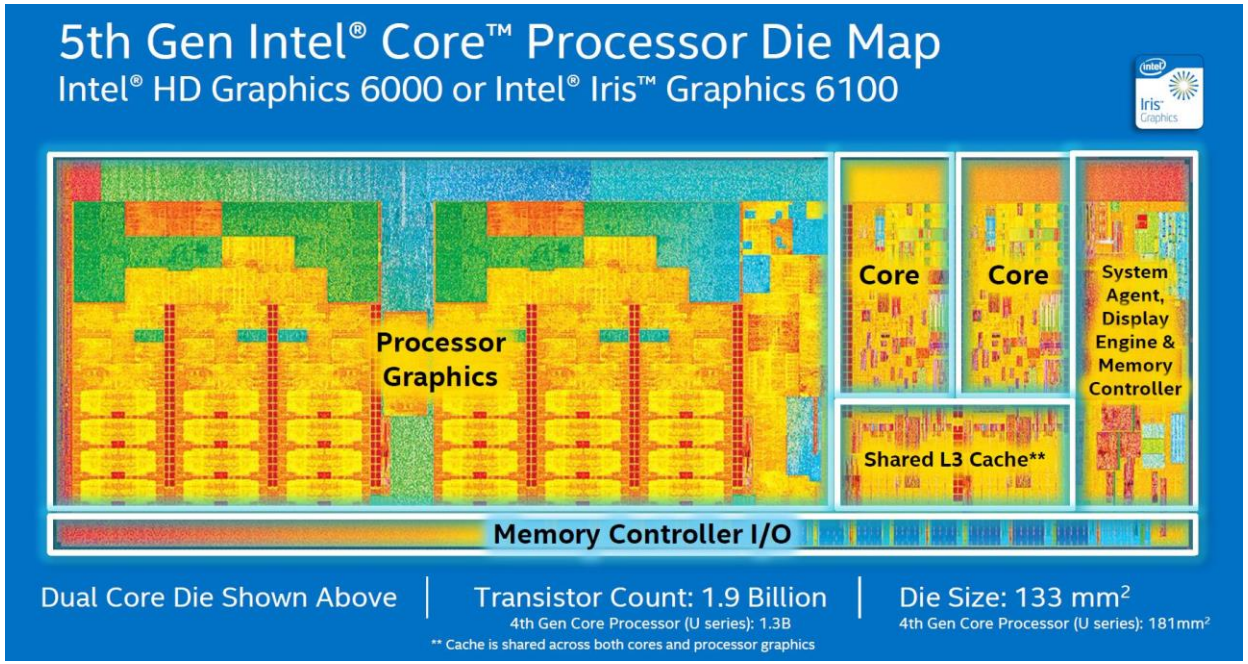
Pentium 4 chip die shot
(roughly what is in one core today)

One compute unit

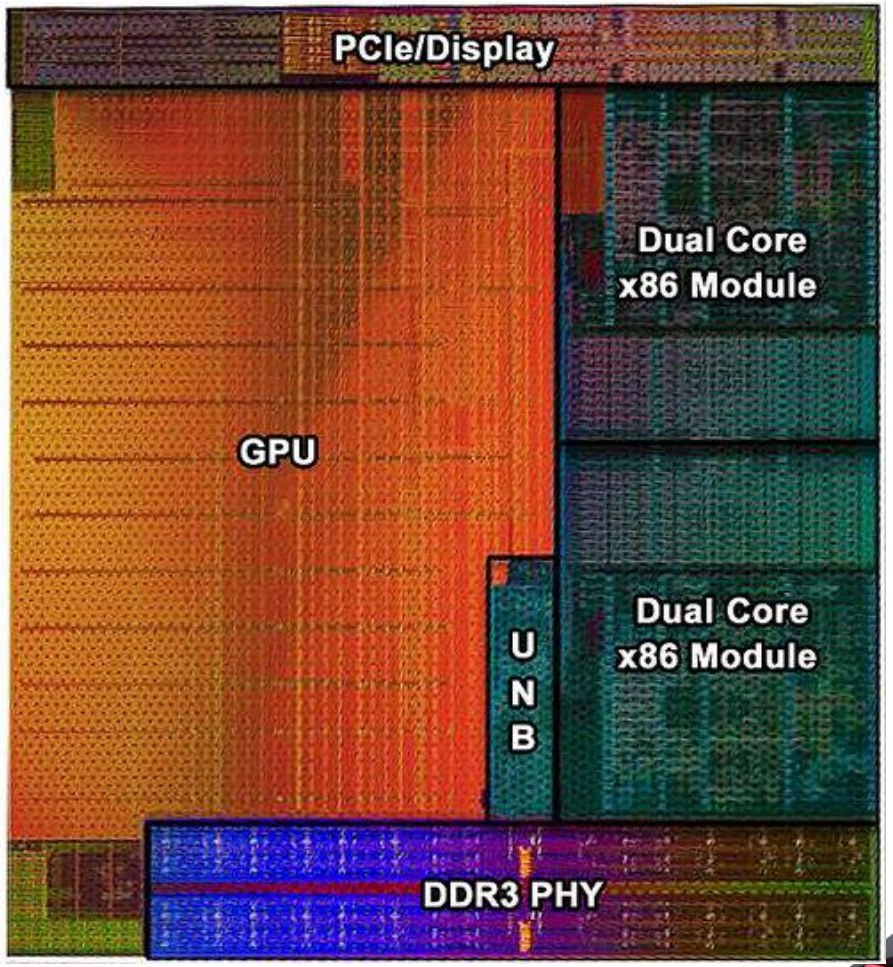


AMD Tahiti GPU chip die shot

Today: CPUs and GPUs on a single chip



Intel Broadwell (2015)



AMD Kaveri (2014)

Software design consequences

To maximize performance:

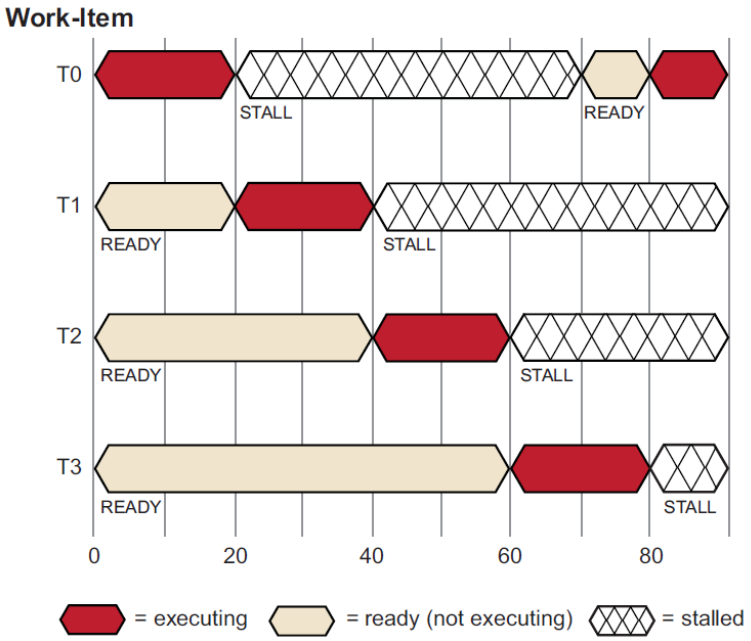
- Take memory bottleneck into account:
  - Long continuous data layout preferred
  - Long continuous block access preferred (coalesced r/w)
- Branching should be avoided
  - Hurts CPUs, but can live with that
  - Kills GPU performance

Software design consequences

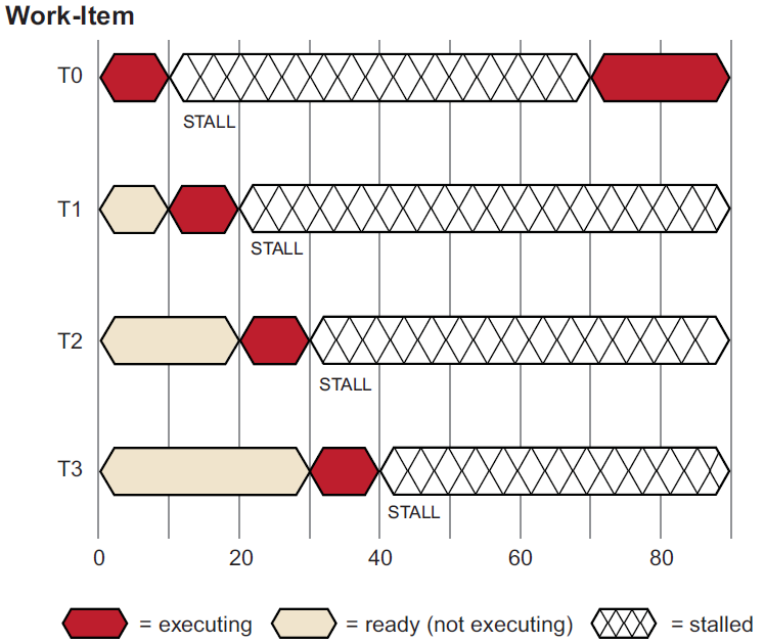
To maximize performance:
Supply enough work always

- CPUs: to feed the pipeline (marginal)
- GPUs: essential to avoid stalls:

Good:



Bad:



Timings

It is instructive to have a feeling about how long things take!

x86:

(ratios similar elsewhere)

Instruction	Latency
Shift / Rot	1-4
AND / OR / XOR	1-4
Compare/test	1-4
Call (Ret)	5 (8)
Integer add/sub	1-8
Integer mul	3-18
Integer div	32-103

Instruction	Latency
MMX	1-9
SSE Integer	1-9
SSE single simpl.	1-12
SSE single mul	3-7
SSE single div/sqrt	10-40
SSE2 double simpl.	1-12
SSE2 double mul	3-7
SSE2 double div/sqrt	14-70
SSE2 128bit int	1-10
SSE3 / SSE4	1-14

Instruction	Latency
FADD/FSUB/FABS	2-6
FMUL	7-8
FDIV	23-44
FSQRT	23-44
FSIN, FCOS	160-280
FSINCOS	160-250
FPTAN	225-300
FPATAN	150-300
FSCALE	60
FYL2X/FYL2XP1	100-250

Timings

Operation	Time [ns]	Time [ms]
1 cycle on a 3 GHz processor	1	1e-6
L1 cache access	0.5	5e-7
Cost of branch misprediction	5	5e-6
L2 cache access	7	7e-6
Mutex lock/unlock	25	2,5e-5
RAM access	100	0.0001
Compression of 1kB data with Snappy	3000	0.003
Transfer of 1kB data through a 1 Gbps network	10000	0.01
Random access read of 4 kB data from an SSD	150000	0.15
Contiguous read of 1 MB data from RAM	250000	0.25
Roundtrip inside a data center	500000	0.5
Contiguous read of 1 MB data from an SSD	1e6	1
Hard disk ‚seek' time	1e7	10
Contiguous read of 1 MB data from the hard disk	2e7	20
TCP/IP packet between continents	1.5e8	150

Source and Video to watch

Summary

Take home message for CPUs:

- Genericity, extremely complex, designed for ever changing tasks, backwards compatibility

Take home message for GPUs:

- Specialized, simple, suited for many identical computation heavy tasks