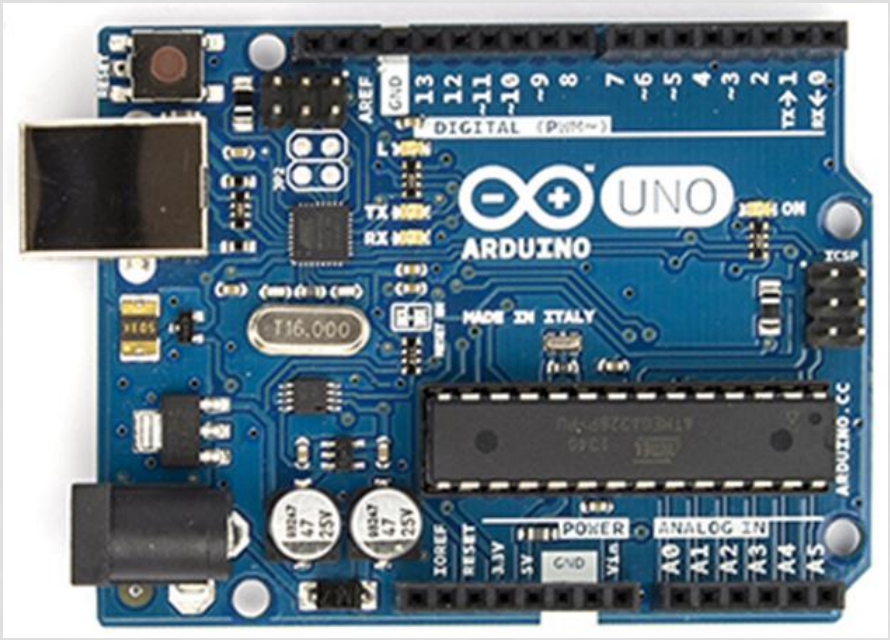

DIGITAL (PWM~)
UNO
ARDUINO
ON
ICSP
MADE IN ITALY
POWER
ANALOG IN
ARDUINO.CC

CS

WEEK 3-2

AGENDA

week	onderwerp	P&H	AT	Dijkstra
1	coderingen en talstelsels representatie van getallen optellen en aftrekken vermenigvuldigen en delen logische poorten schakelingen met poorten geheugen-elementen systeemklok & timers	App. B2, B3, B7, B8, B9 2.4 3.2 t/m 3.5 app B	App. A, B 3.1, 3.2, 3.3	H1 H2
2	typen computers 8 great ideas organisatie van de computer CPU intern, instructies uitvoeren geheugen systeem adres- en databus byte ordering pipelining de AVR MCU	1.1 t/m 1.4 2.12 4.1 t/m 4.5	1.3 2.1, 2.2 3.7	H3 6.1 en 6.2 7.1 en 7.2
3	typen geheugen caching opslag (ssd, harddisk) translating and starting a program parallelle architecturen - h/w multi-threading - multicore - GPU	5.2, 5.3 6.4 t/m 6.6	2.2, 2.3 7.3, 7.4 H8	4.1 7.3

AGENDA

- **parallelle architecturen**
- GPU

INTRODUCTION

- if we can't improve CPU performance, we have to do things parallel
- there are many options :
  - pipelining in the CPU
  - superscalar
  - threads in the CPU (logical cores)
  - multicore
  - multi-threading & multi-processing
  - grid computing

HARDWARE AND SOFTWARE

- hardware
  - serial: e.g., Pentium 4
  - parallel: e.g., quad-core Xeon e5345
- software
  - sequential: e.g., matrix multiplication
  - concurrent: e.g., operating system
- sequential/concurrent software can run on serial/parallel hardware
  - challenge: making effective use of parallel hardware

PARALLEL PROGRAMMING

- parallel software is the main problem
- difficulties
  - partitioning
  - coordination
  - communications overhead

PITFALL : AMDAHL'S LAW

- Amdahl's law is often used in parallel computing to predict the theoretical speedup when using multiple processors
- suppose a program needs 20 hours using a single core
- suppose a part of this program which takes one hour cannot be parallelized; the remaining 19 hours can be parallelized
- so the minimum execution time cannot be less than that critical one hour

PITFALL : AMDAHL'S LAW

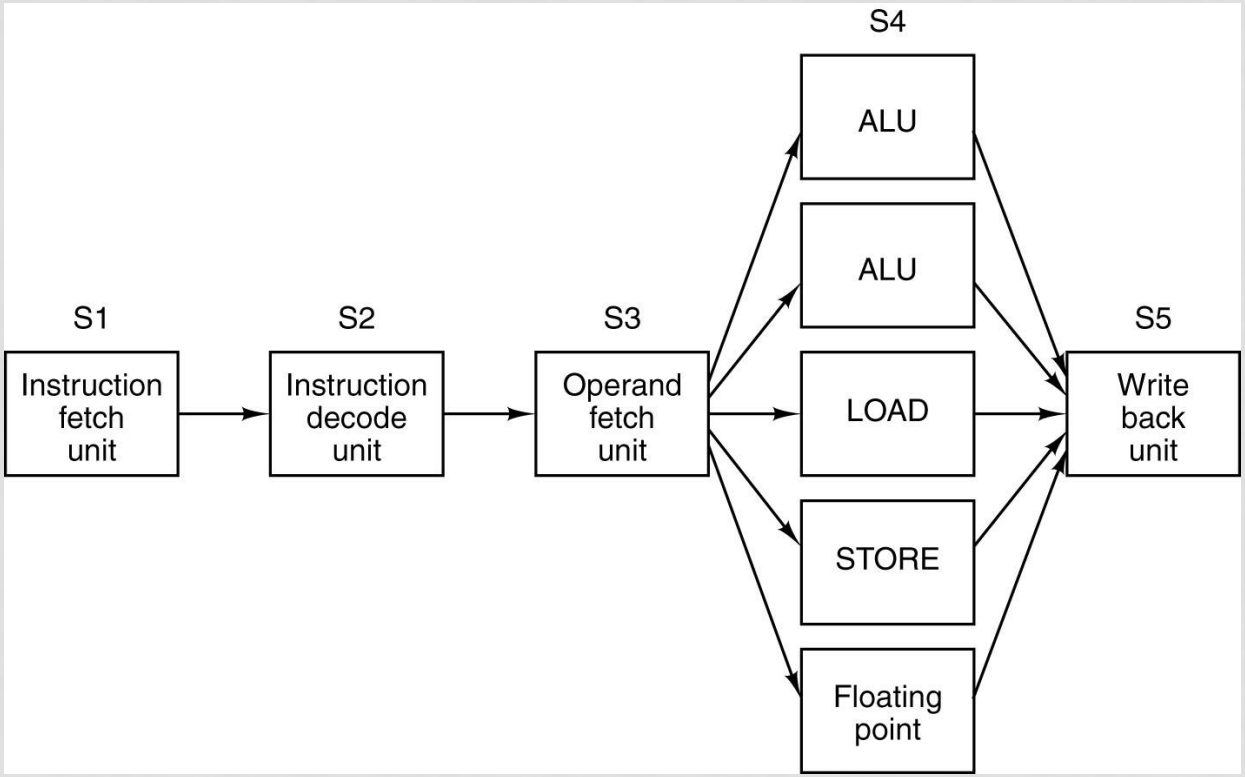
- let p = 19/20 = 0.95
- suppose there are 4 cores, and the 19 hours can be reduced to 19/4 hours, then s= 4
- the overall speedup will be a factor :

$$\frac{1}{1 - p + \frac{p}{s}}$$

= 1 / (0.05 + 0.24) = 3,5

SUPERSCALAR

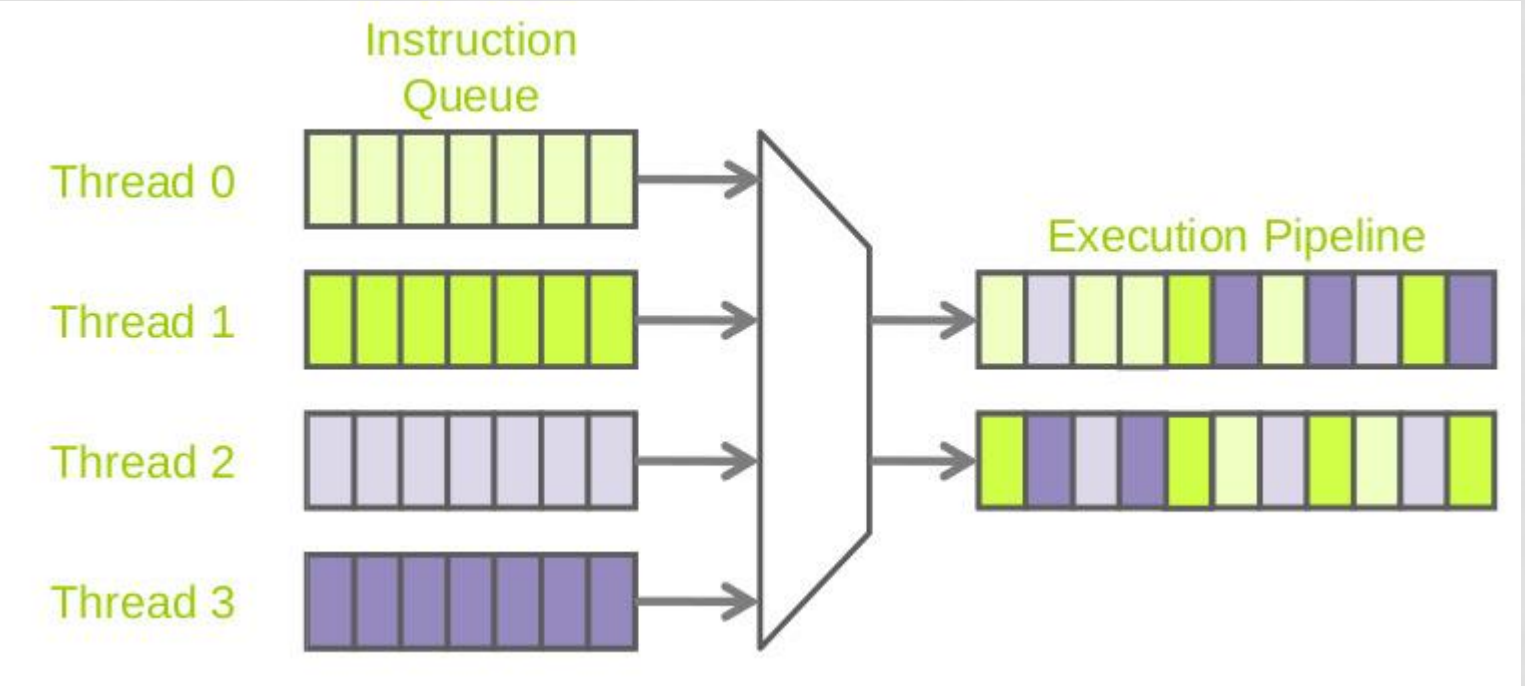
- superscalar : meerdere execution units
- voorbeeld : DSP, instructies met tegelijk ADD en MUL



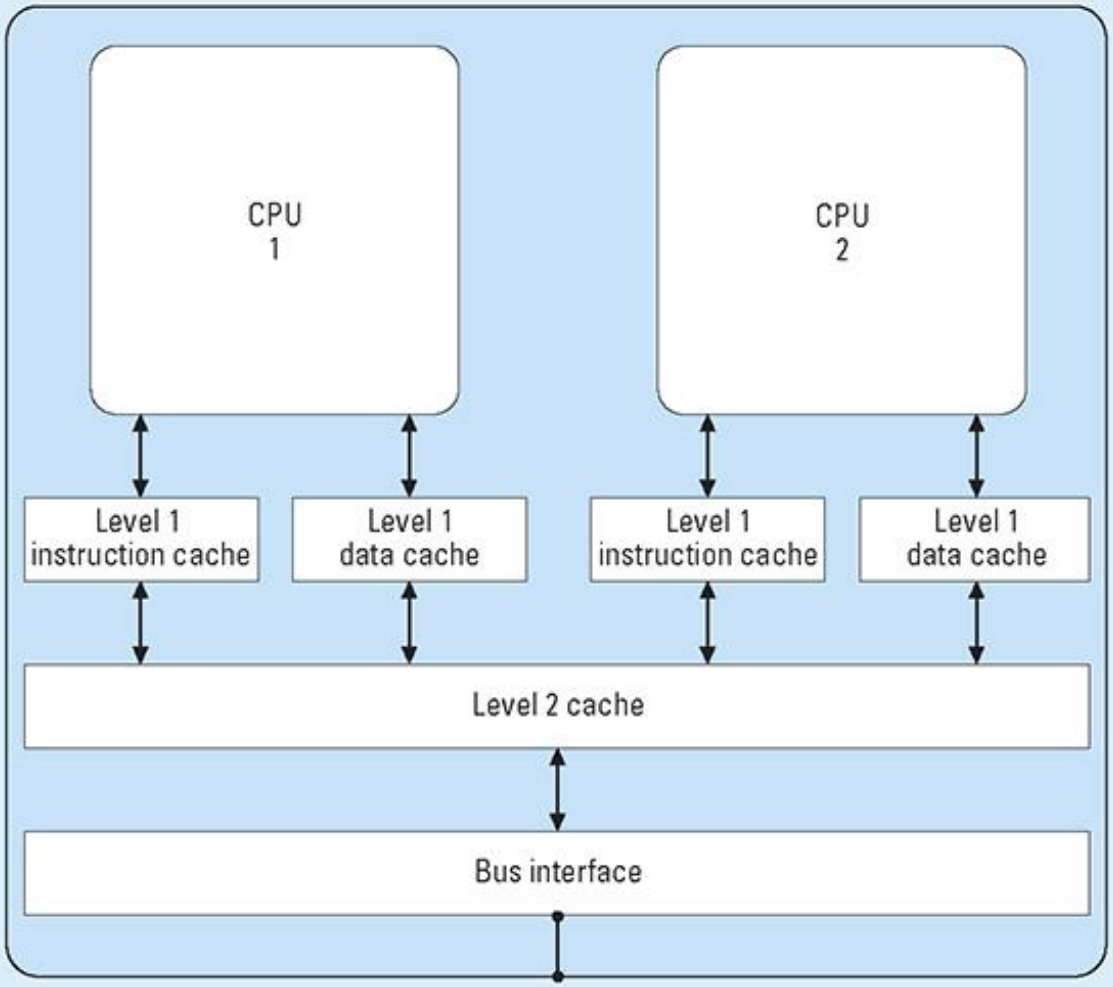
ON-CHIP MULTITHREADING

- als gegeven niet in L1/L2 cache is, dan staat de pipeline lang stil
  - sommige CPU units worden dan niet gebruikt
  - mogelijke oplossing : multithreading
  - als thread X moet wachten dan kan thread Y wel verder
- Intel name : hyperthreading (Pentium4)
- het OS 'ziet' twee cores, de kernel kan dan twee threads/processen parallel schedulen
- logische cores delen resources zoals execution engine, caches en system bus interface
- 20% to 40% beter dan CPU zonder hyperthreading

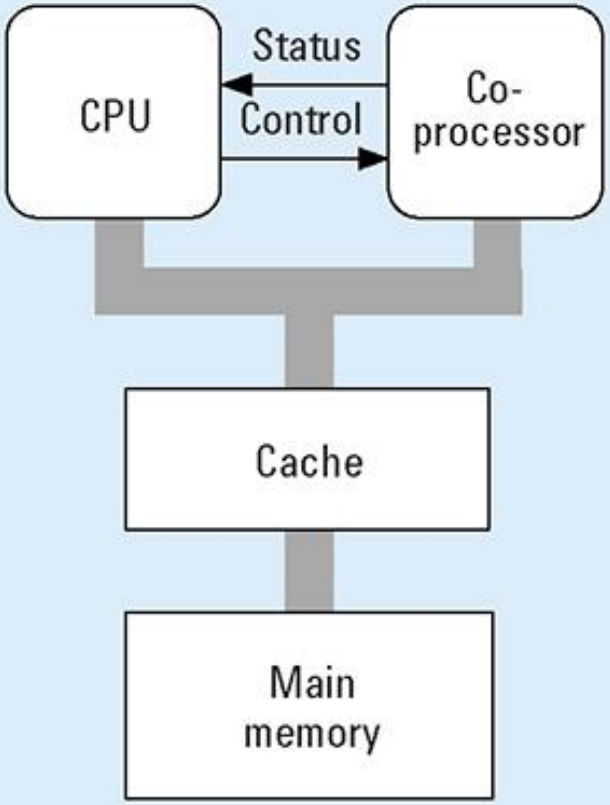
MULTITHREADING



Figuur 7.15 **Multicore processor chip**

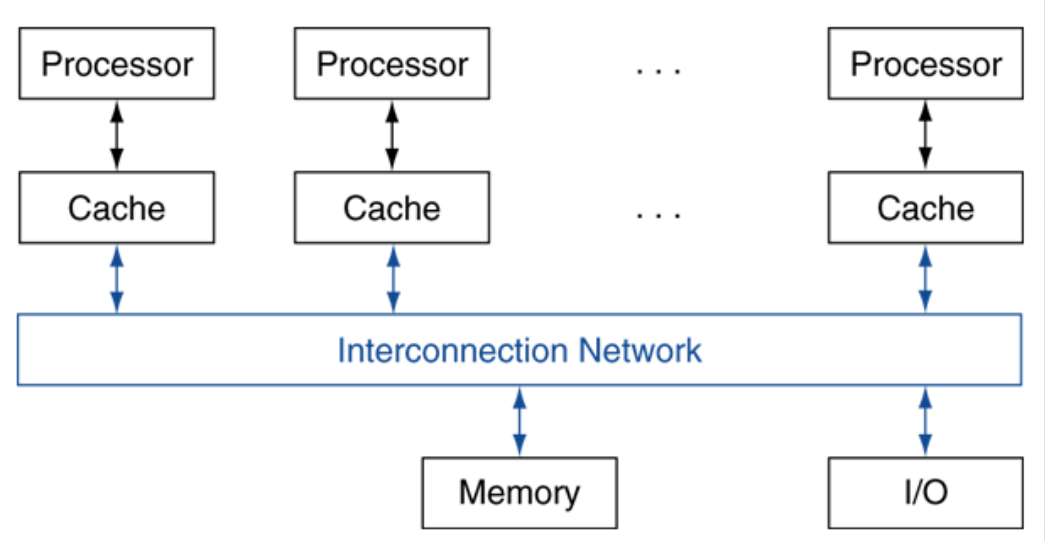


Figuur 7.13 **Coprocessor**

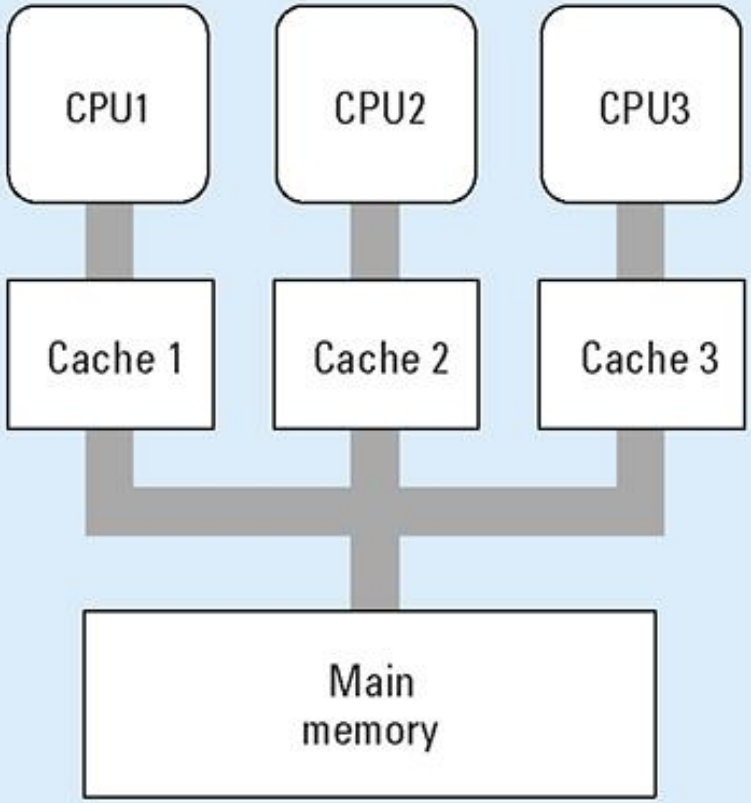


SHARED MEMORY

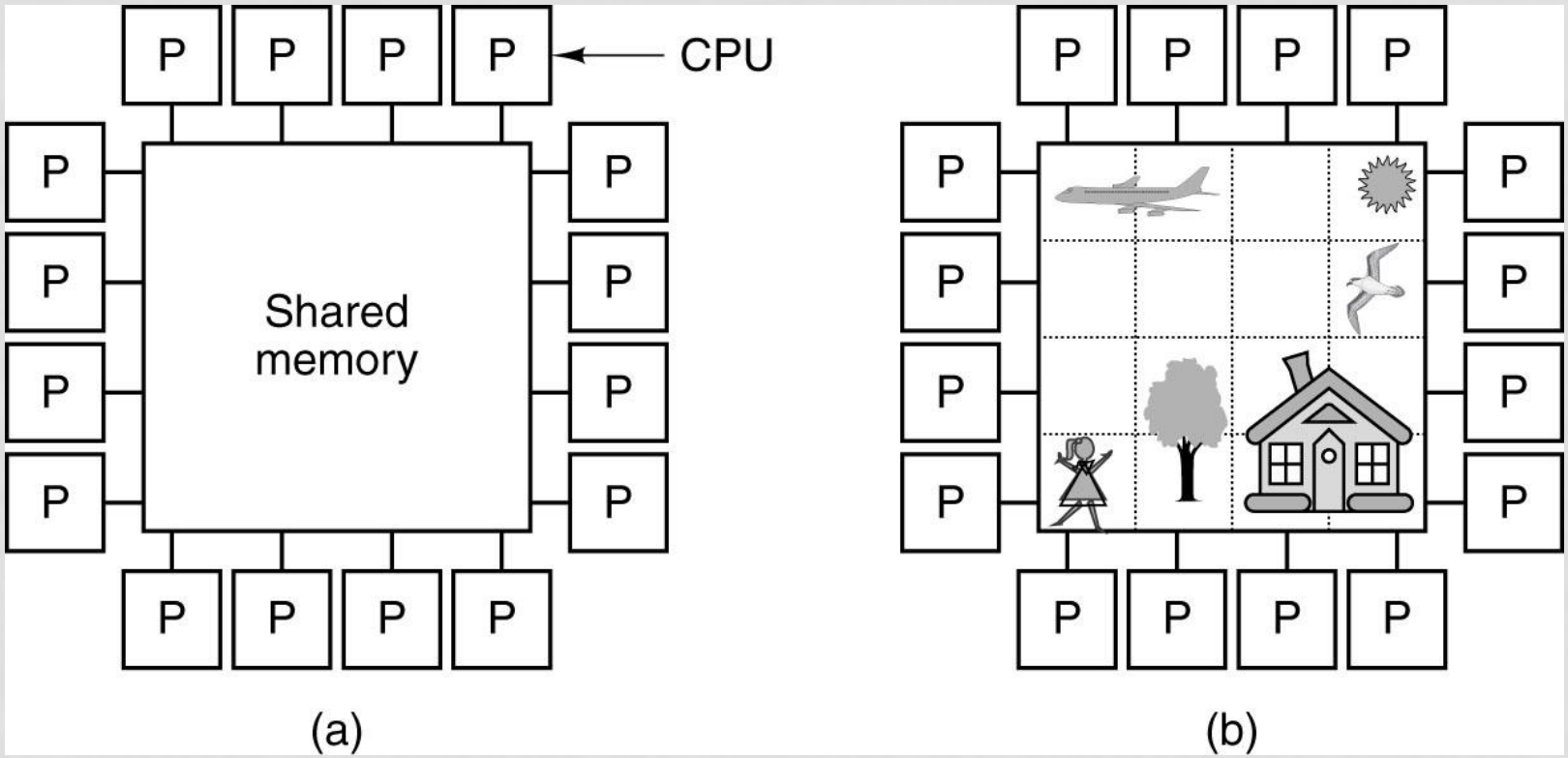
- SMP: shared memory multiprocessor
  - hardware provides single physical address space for all processors
  - synchronize shared variables using locks
  - memory access time
    - UMA (uniform) vs. NUMA (no-nuniform)



Figuur 7.14 **Shared memory multiprocessing**



SHARED MEMORY MULTIPROCESSORS



(a) A multiprocessor with 16 CPUs sharing a common memory
(b) An image partitioned into 16 sections, each being analyzed by a different CPU

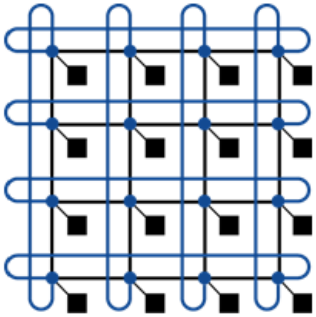
GRID COMPUTING

- separate computers interconnected by long-haul networks
  - e.g., Internet connections
  - work units farmed out, results sent back
- can make use of idle time on PCs
  - e.g., SETI@home, World Community Grid

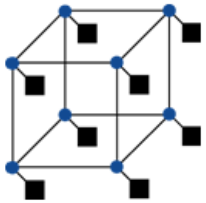
NETWORK TOPOLOGIES

Bus

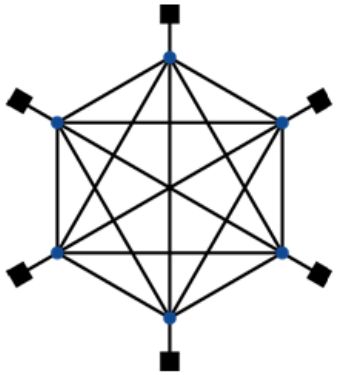
Ring



2D Mesh



N-cube (N = 3)



Fully connected

AGENDA

- parallelle architecturen
- **GPU**

HISTORY OF GPUS

- 3D graphics processing
  - originally high-end computers (e.g., SGI)
  - Moore's Law ⇒ lower cost, higher density
  - 3D graphics cards for PCs and game consoles
- video gaming
  - many pixels have to be handled independently and simultaneously
  - latency is not important, but throughputs

HISTORY OF GPUS

- Graphics Processing Units
  - processors oriented to 3d graphics tasks
  - optimized for
    - 3-d matrix calculations
    - floating point arithmetic
    - operations on buffers with pixels
  - optimized for parallel processing

3D-RENDERING

- designed to do efficient processing of 3-d images
- things like e.g. geometry, colors, light, point-of-view (camera)
- project 3-d scene on a 2-d screen
- then move the camera around

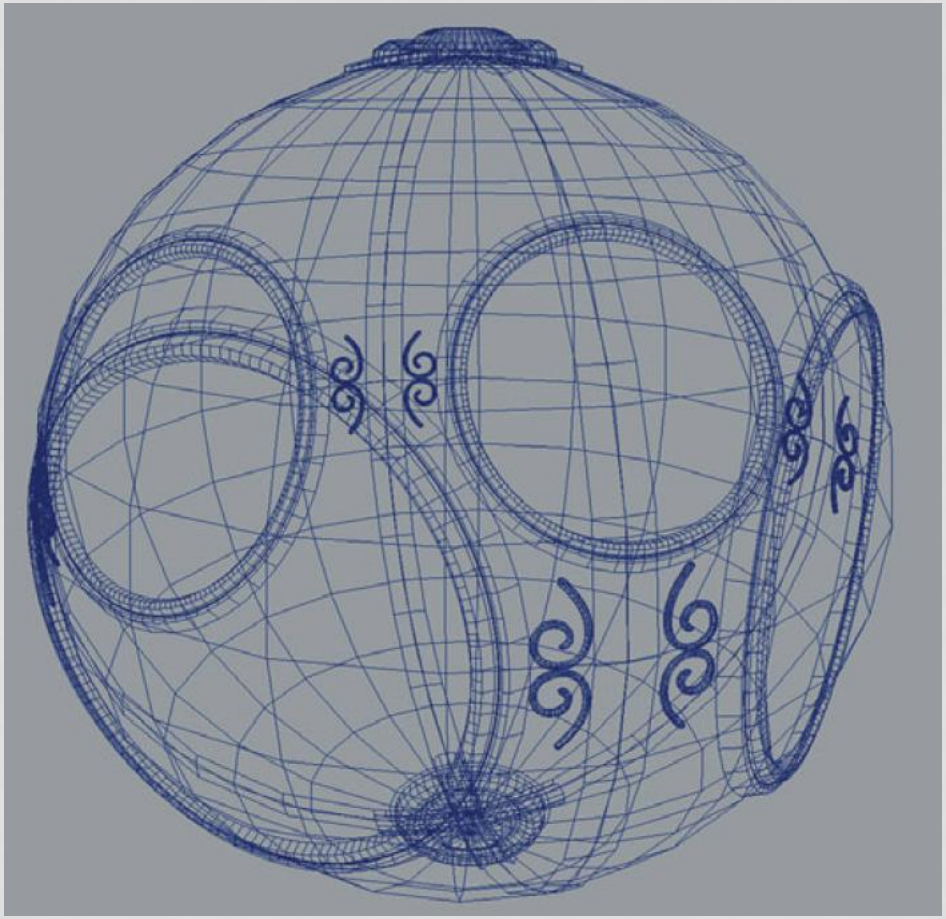
3D-RENDERING

- rendering = generating an image from a 3d-model
- build a wireframe model : from a small set of geometric primitives—points, lines, and polygons construct shapes
  - by creating mathematical descriptions of objects
- arrange the objects in 3d-space and select the desired view point for viewing the composed scene
- calculate the colors of all the objects.
  - colors might be explicitly assigned by the application, determined from specified lighting or obtained by pasting textures onto the objects

3D-RENDERING

- convert the mathematical description of objects and their associated color information to pixels on the screen (rasterization)
- frameworks : OpenGL (non-profit) and Direct3d (MS)
  - software interface (API) to graphics hardware (=GPU)

WIREFRAME MODEL



3D-RENDERING OF THE MODEL

GPU ARCHITECTURES

- processing is highly data-parallel
  - GPUs are highly multithreaded
  - use thread switching to hide memory latency
    - less reliance on multi-level caches
  - graphics memory is wide and high-bandwidth
- trend toward general purpose GPUs
  - heterogeneous CPU/GPU systems
  - CPU for sequential code, GPU for parallel code
- programming languages/APIs
  - DirectX, OpenGL
  - C for Graphics (Cg), High Level Shader Language (HLSL)
  - Compute Unified Device Architecture (CUDA) : gives direct access to the GPU's virtual instruction set and parallel computational elements

GPU PIPELINE

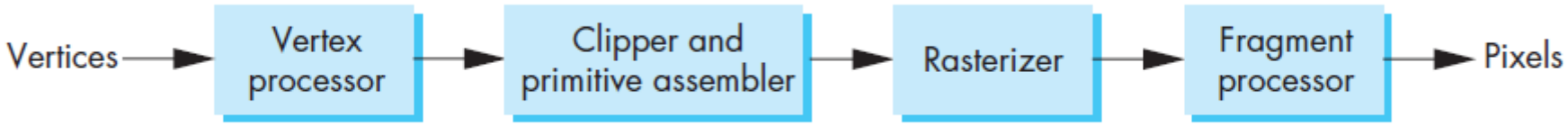


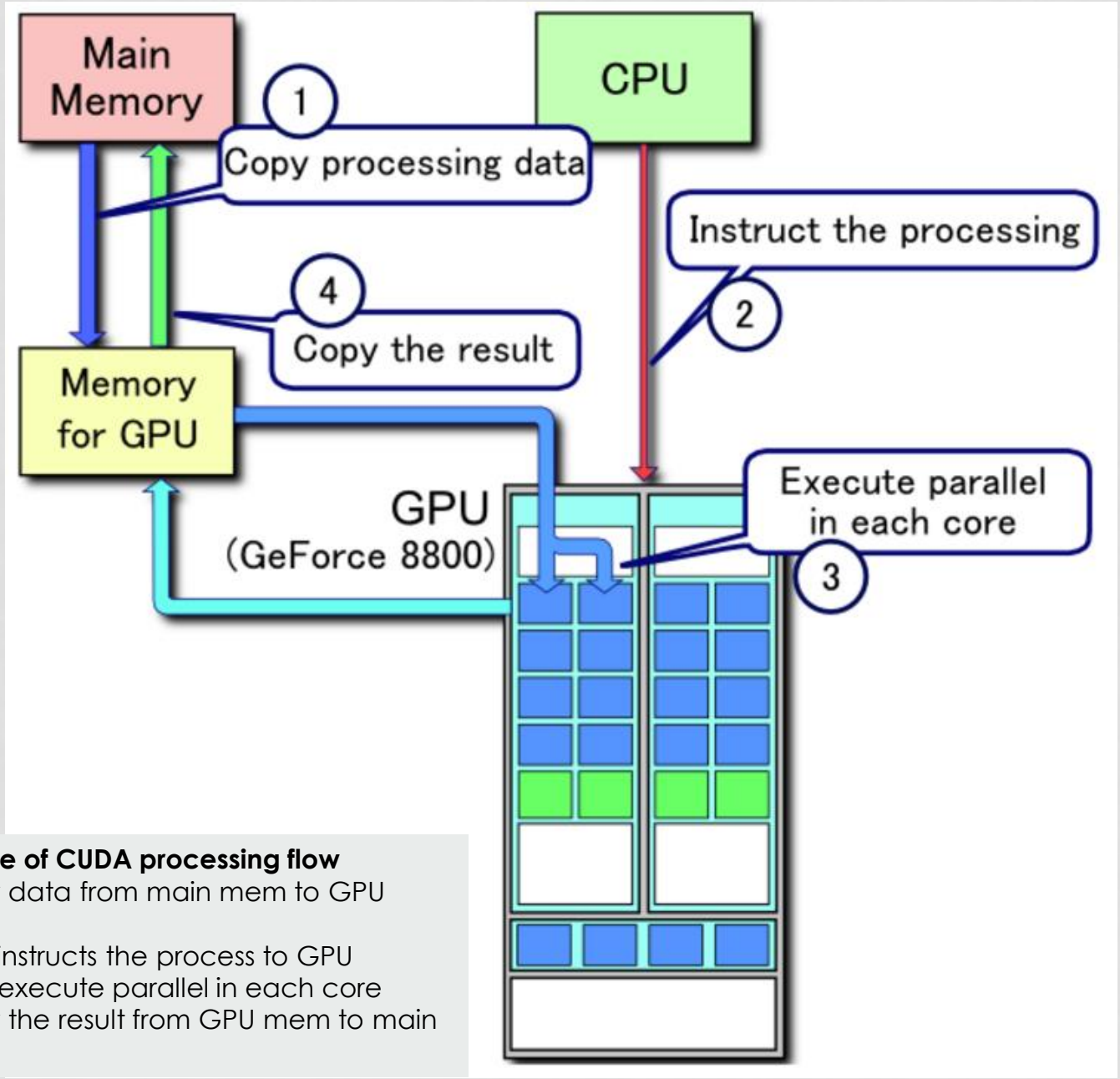
FIGURE 1.38 Geometric pipeline.

- vertex processing
  - coordinate transformations and color computations
- clipping and primitive assembly
  - assemble vertices into primitives before clipping
- rasterization
  - convert to pixels in the frame buffer
- fragment processing
  - update pixels in the frame buffer

EEN VOORBEELD

- https://en.wikipedia.org/wiki/Nvidia_Tesla
  (check aantal cores)

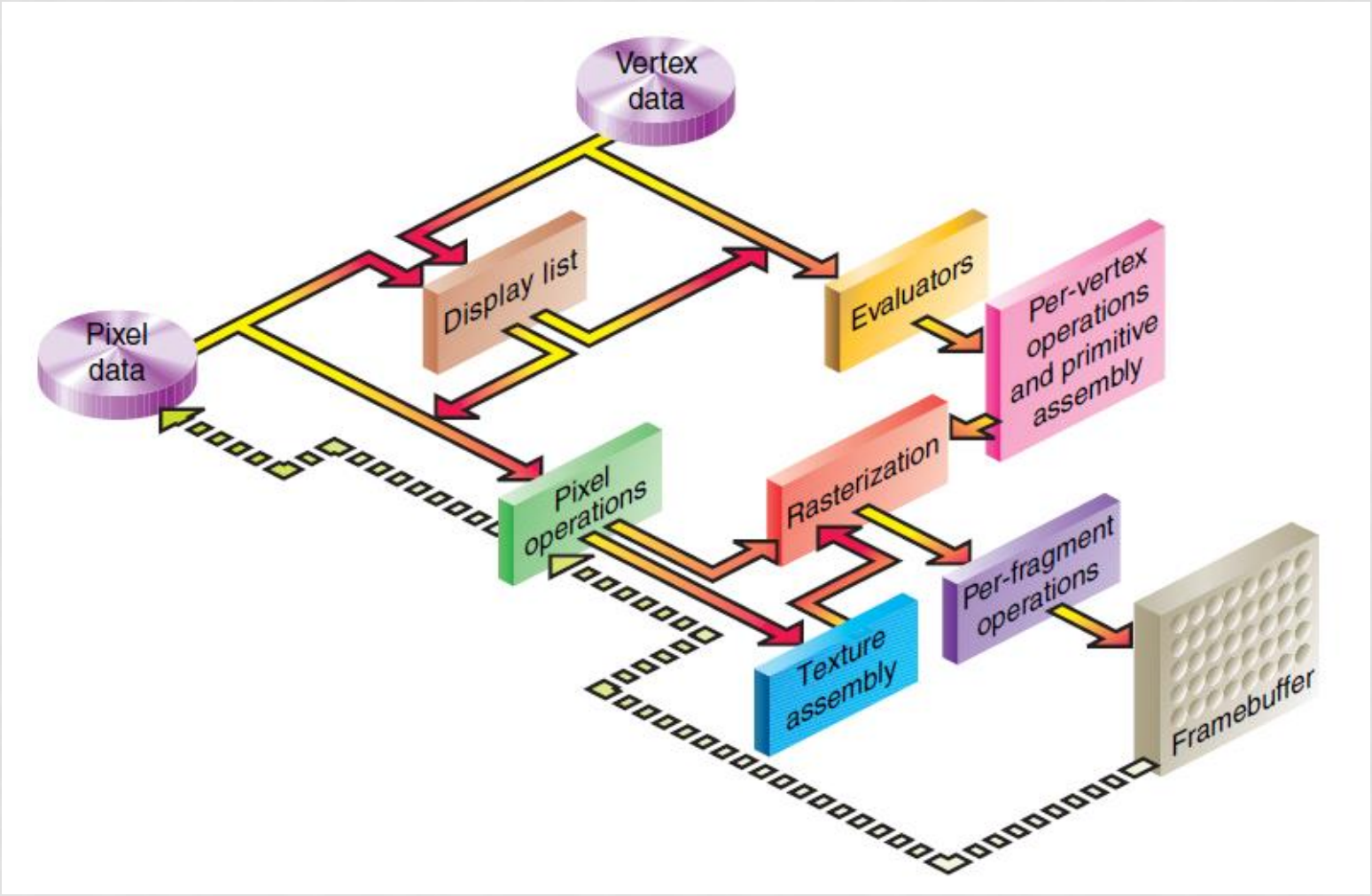
- https://www.youtube.com/watch?v=mwDPb3T8bOQ



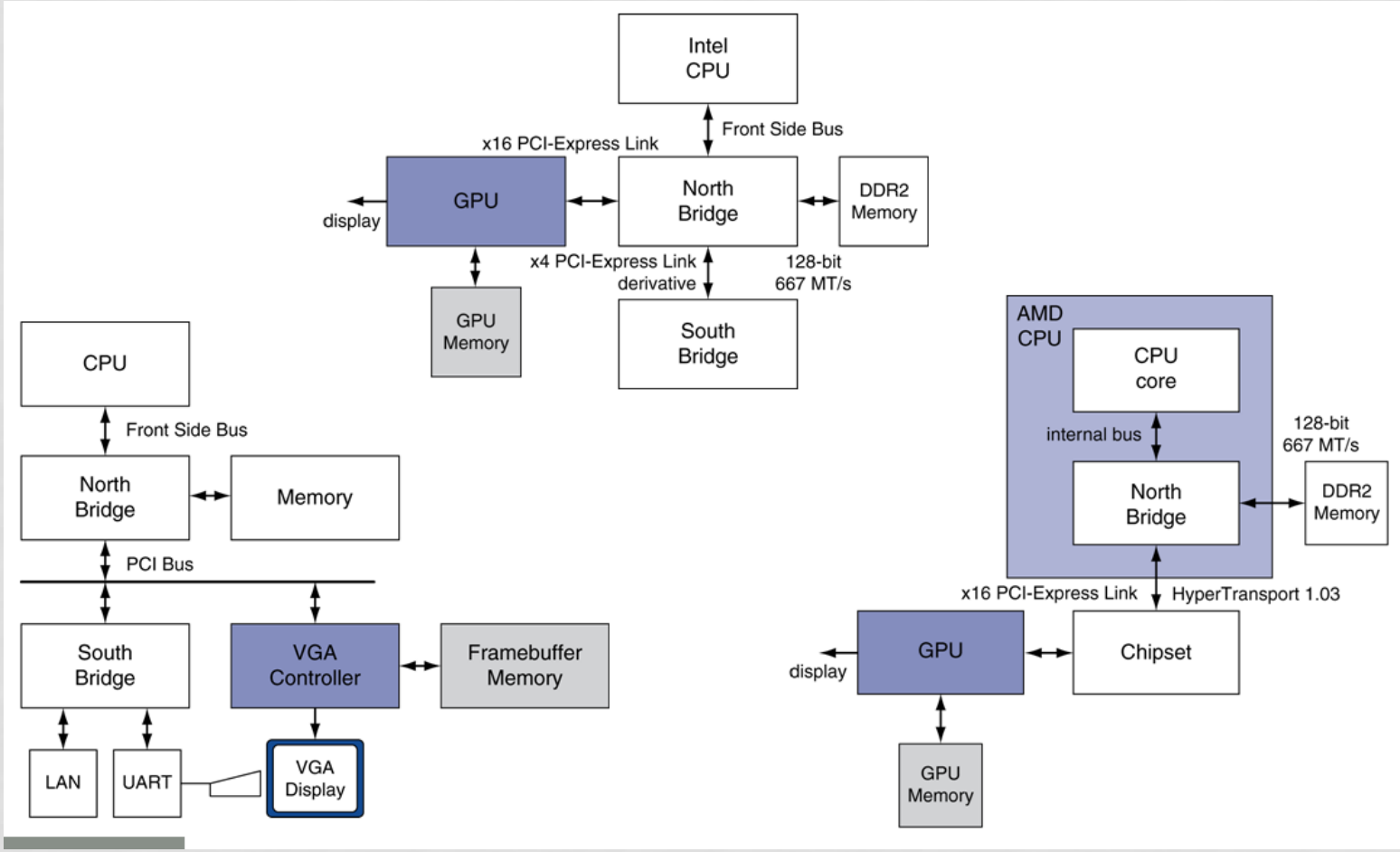
Example of CUDA processing flow

1. copy data from main mem to GPU mem
2. CPU instructs the process to GPU
3. GPU execute parallel in each core
4. copy the result from GPU mem to main mem

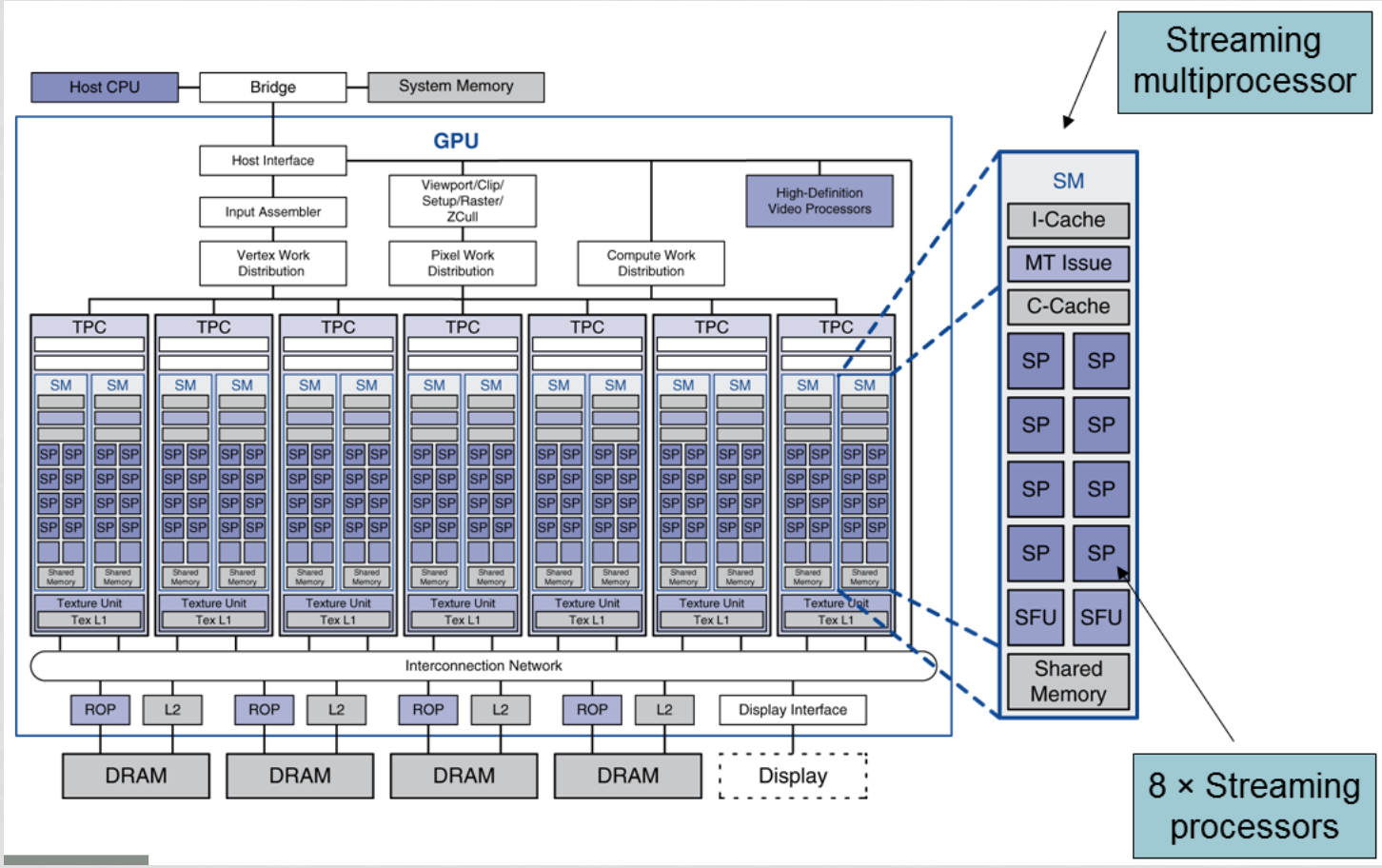
OPENGL PIPELINE



GRAPHICS IN THE SYSTEM



EXAMPLE: NVIDIA TESLA



GPU

- Nvidia GeForce
- AMD Radeon