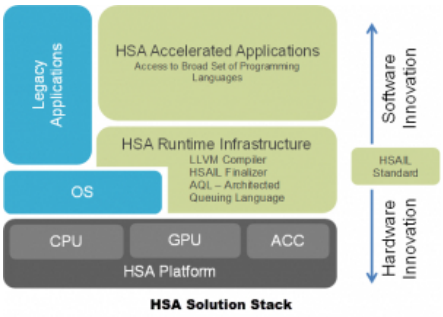
Heterogeneous Systems Architecture (HSA) - the TL;DR



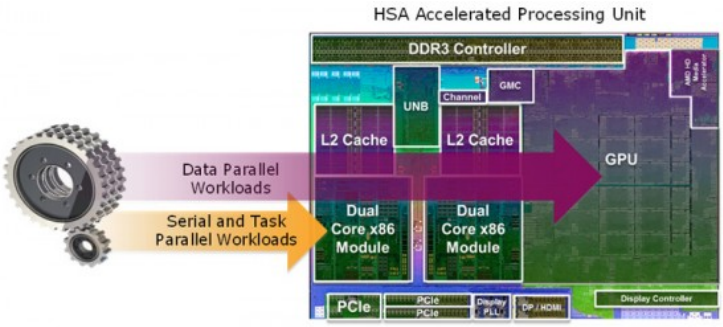
Legacy-apps run on HSA-hardware, but less optimal.

The main problem of discrete GPUs is that memory needs to be transferred from CPU-memory to GPU-memory. Luckily we have SoCs (GPU and CPU in one die), but still you need to do in-memory transfers as the two processors cannot access memory outside their own dedicated memory-regions. This is due the general architecture of computers, which did not take accelerators into account. Von Neumann, thanks!

HSA tries to solve this, by redefining the computer-architecture as we know it. AMD founded the HSA-foundation to share the research with other designers of SoCs, as this big change simply cannot be a one-company effort. Starting with 7 founders, it has now been extended to a long list of members.

Here I try to give an overview of what HSA is, not getting into much detail. It's a TL;DR.
What is Heterogeneous Systems Architecture (HSA)?
It consists mainly of **three parts**:

new **memory-architecture: hUMA**,
new **task-queueing: hQ**, and
an **intermediate language: HSAIL**.

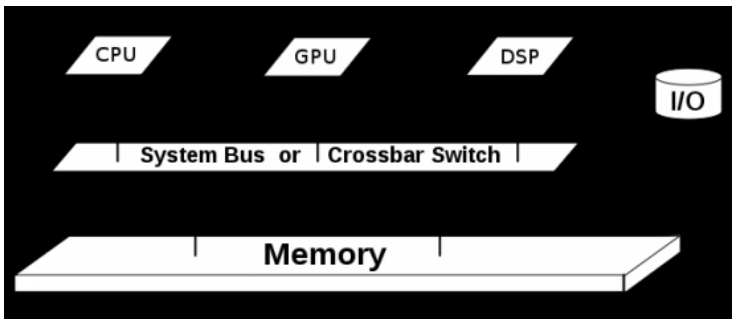


HSA enables tasks being sent to CPU, GPU or DSP without bugging the CPU.

The basic idea is to give GPUs and DSPs about the same rights as a CPU in a computer, to enable true heterogeneous computing.

hUMA (Heterogeneous Uniform Memory Access)

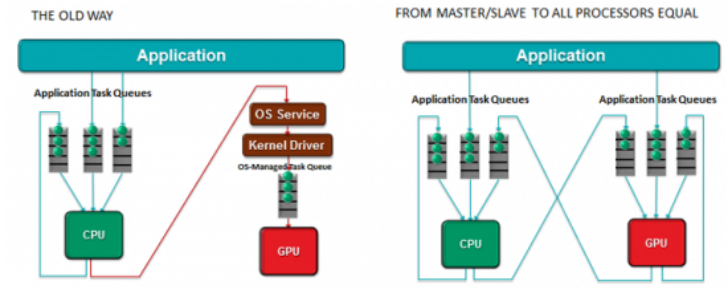
HSA changes the way memory is handled by eliminating a hierarchy in processing-units. In a hUMA architecture, the CPU and the GPU (inside the APU) have full access to the entire system memory. This makes it a **shared memory system** as we know it from multi-core and multi-CPU systems.



This is the super-simplified version of hUMA: a shared memory system with CPU, GPU and DSP having equal rights to the shared memory.

hQ (Heterogeneous Queuing)

HSA gives more rights to GPUs and DSPs, leveraging work from the CPU. Compared to the Von Neumann architecture, the CPU is not the Central Processing Unit anymore - each processor can be in control and create tasks for itself and the other processors.



HSA-processors have control over their own and other application task queues.

HSAIL (HSA Intermediate Language)

HSAIL is a sort of virtual target for HSA-hardware. Hardware-vendors focus on getting HSAIL compiled to their processor instruction sets, and developers of high-level languages target HSAIL in their compilers. This is a proven concept of evolving complex hardware-software projects.

It is pretty close to OpenCL SPIR, which has comparable goals. Don't see them as competitors, but two projects which both need different freedoms and will work along.

What is in it for OpenCL?

OpenCL 2.0 has support for Shared Virtual Memory, Generic Address Space and Recursive Functions. All supported by HSA-hardware.

OpenCL-code can be compiled to SPIR, which compiles to HSAIL, which compiles to HSA-hardware. When the time comes that HSAIL starts supporting legacy hardware, SPIR can be skipped.

HSA is going to be supported in OpenCL 1.2 via new flags - watch this thread.
Final words
Two companies not there: Intel and Nvidia. Why? Because they want to do it themselves. The good news is that HSA is large enough to define the new architecture, making sure we get a standard. The bad news is that the two outsiders will come up with an exception for whatever reason, which gives a need for exceptions in compilers.

You can read more on the website of the HSA-foundation or ask me in the comments below.