

یک.

مزایای UMA: یکسان بودن زمان دسترسی بین پردازنده‌ها، مموری کنترلر کم‌تر (= قیمت کم‌تر)
 معایب UMA: زمان دسترسی کندتر (به علت رنج حافظه بزرگ‌تر) – پهنای باند محدودتر
 مزایای NUMA: زمان دسترسی سریع‌تر – پهنای باند بیش‌تر
 معایب NUMA: متغیر بودن زمان دسترسی پردازنده‌ها – مموری کنترلر بیش‌تر (= قیمت بیش‌تر)

معماری hUMA: مخفف heterogeneous Uniform Memory Access است.

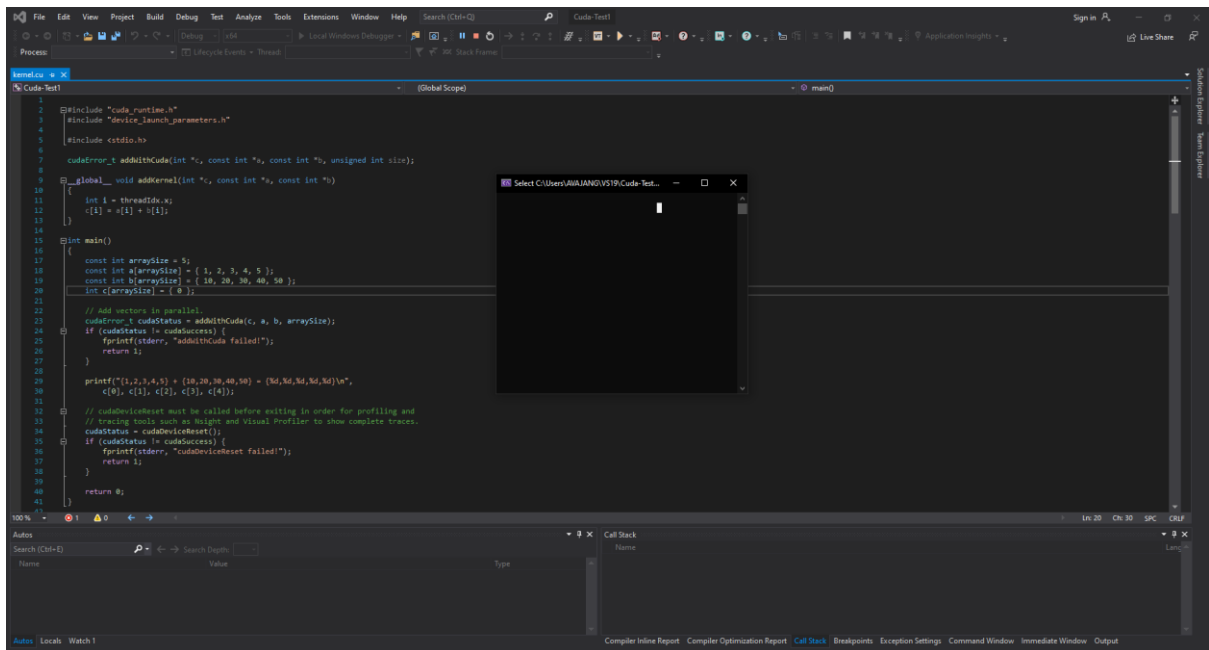
CPU های فعلی در انجام عملیات‌های موازی عملکرد ضعیفی از خود نشان می‌دهند و GPU ها در انجام عملیات‌های سری. برنامه‌های سیستم ترکیبی از عملیات‌های موازی و سری (بدون ترتیب خاصی و حتی سری و موازی تودرتو) است، در معماری‌های فعلی فضای حافظه‌ی CPU و GPU متفاوت است و برای جابه‌جایی بین عملیات‌های سری و موازی، داده باید بین فضاهای CPU و GPU (با استفاده از DMA) کپی شود. به علت استفاده‌ی بسیار زیاد CPU از پوینترها، این حقیقت که پوینترها قابل کپی‌شدن مستقیم در GPU را ندارند و page out شدن برخی خانه‌های حافظه به دیسک سخت سیستم، عملکرد برنامه هنگام این جابه‌جایی‌ها دچار penalty قابل توجهی می‌شود. معماری hUMA پیشنهاد شرکت AMD برای حل این مساله است. در این معماری، CPU و GPU از memory space یکسانی استفاده می‌کنند و نحوه‌ی آدرس‌دهی در هر دو به یک صورت در می‌آید. این معماری یک سیستم cache-coherent است، یعنی CPU و GPU همیشه حافظه‌ی یکسانی را مشاهده می‌کنند. این امر کار برنامه‌نویسان را آسان می‌کند چون در غیر این صورت، برنامه‌ها باید هنگام تغییر دیتا سیگنالی را به main memory بفرستد. این کار سخت‌افزار را ساده‌تر می‌کند اما نرم‌افزار را نسبت به باگ‌های سخت‌تشخیص آسیب‌پذیر می‌کند. برای رفع مشکل آدرس‌های page-out شده نیز، GPU علاوه بر استفاده از سیستم CPU برای آدرس‌دهی main memory، به demand-paged virtual memory پردازنده نیز دسترسی دارد، اگر GPU درخواستی برای یکی از این خانه‌های حافظه دهد، CPU وارد عمل شده و آن خانه را وارد حافظه‌ی اصلی می‌کند. این معماری در بعضی از سیستم‌های فعلی همانند پلی‌استیشن 4 پیاده‌سازی شده‌است.

معماری COMA: مخفف Cache Only Memory Architecture است و برخلاف معماری NUMA که در آن هر پردازنده آدرس‌دهی خودش را دارد، در این پردازنده همه‌ی آدرس‌دهی‌ها یکسانند، همه‌ی ماژول‌های حافظه در آن به عنوان کش‌هایی از نوع DRAM عمل می‌کنند. و دیگر مکان مرکزی‌ای برای یک داده وجود ندارد.

دو. قانون گوستافسون برخلاف قانون آمدال بر این فرض بنا شده است که در عمل، میزان ریسورس‌های برنامه با اندازه‌ی مساله ارتباط مستقیم دارد. یعنی بالفرض اگر سیستم ۵۰ هسته‌ی پردازشی دارد، احتمالاً برای حل مساله‌هایی با $n=100$ استفاده نمی‌شود و مساله در اردر بالاتری قرار دارد و نتیجتاً هدف این است که با افزایش منابع سیستم، بتوان مساله‌ی بزرگ‌تری را در همان ابعاد زمانی حل کرد. قانون سان-نی از حیث فرضیات مشابه قانون گوستافسون است اما با این تفاوت که هدف این است که مقدار Main Memory مساله‌ی scale شده در محدوده‌ی اندازه‌ی Main Memory سیستم بماند.

سه. به علت وجود قسمت‌های مختلف تسک‌های مختلف در یک زمان در پایپ‌لاین و علت تعداد زیاد برنامه‌ها، حالت‌هایی رخ می‌دهد که قسمت‌های زیادی از تسک‌ها کار خود را به اتمام می‌رسانند اما به علت ماندن در پشت یک تسک بسیار زمان‌گیر در پایپ‌لاین، توانایی پیشروی ندارند و اجباراً مجبور به توقف هستند. هرچه تعداد بیش‌تری تسک وارد سیستم شود احتمال رخ‌داد این اتفاق بیش‌تر است فلذا کارآیی کم می‌شود.

چهار.



پنج.

2Cores, 2Threads/Core => 4Threads.

Max frequency: 3.70GHz

3 Levels of cache:


L1 Data cache = 32 KB, 64 B/line, 8-WAY.

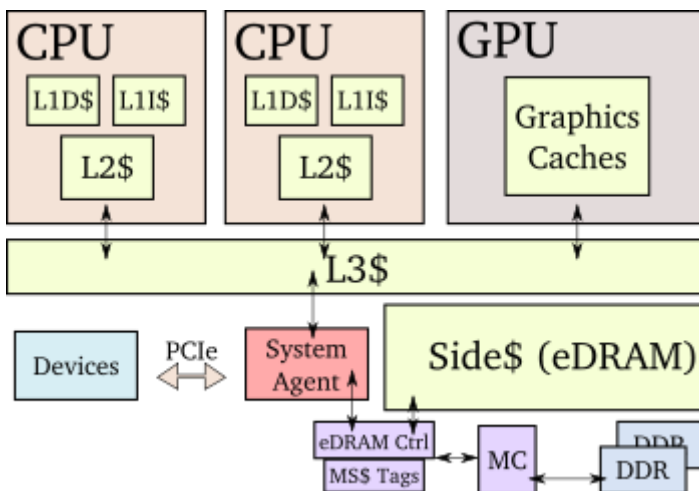
L1 Instruction cache = 32 KB, 64 B/line, 8-WAY.

L2 cache = 256 KB, 64 B/line, 4-WAY

L3 cache = 3 MB, 64 B/line, 12-WAY

Main Memory: 4GB

Processor					
Name	Intel Core i3 6100				
Code Name	Skylake	Max TDP	65.0 W		
Package	Socket 1151 LGA				
Technology	14 nm	Core Voltage	0.720 V		
Specification	Intel® Core™ i3-6100 CPU @ 3.70GHz				
Family	6	Model	E	Stepping	3
Ext. Family	6	Ext. Model	5E	Revision	R0
Instructions	MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T, VT-x, AES, AVX, AVX2, FMA3				
Clocks (Core #0)					
Core Speed	798.44 MHz				
Multiplier	x 8.0 (8 - 37)				
Bus Speed	99.81 MHz				
Rated FSB					
Cache					
L1 Data	2 x 32 KBytes		8-way		
L1 Inst.	2 x 32 KBytes		8-way		
Level 2	2 x 256 KBytes		4-way		
Level 3	3 MBytes		12-way		
Selection Socket #1 Cores 2 Threads 4					



هسته‌های این سیستم با استفاده از توپولوژی ring به هم متصل شده‌اند که این اتصال gpu را نیز شامل می‌شود.