

Re: a dozen cpu's on a chip

Re: a dozen cpu's on a chip

Source: <http://sci.tech--archive.net/Archive/sci.electronics.design/2008-05/msg01118.html>

- *From:* John Larkin <jjlarkin@xx>
 - *Date:* Fri, 09 May 2008 08:49:29 -0700
-

On Fri, 9 May 2008 07:42:29 -0700 (PDT), MooseFET <kensmith@xxxxxxxxxx> wrote:

On May 8, 8:27 am, John Larkin
<jjlar...@xx> wrote:

On Thu, 8 May 2008 07:42:04 -0700 (PDT), MooseFET
<kensm...@xxxxxxxxxx>
wrote:

On May 7, 7:48 pm, John Larkin
<jjlar...@xx> wrote:

<http://www.eetimes.com/news/latest/showArticle.jhtml;jsessionid=CESEX...>

I bet we'll see 256 one of these days.

When you get to large numbers of CPUs it seems to make sense to stop making them identical. For servers this would be doubly so. Many of the CPUs won't need to do floating point operations.

Right. Amybe a few cpu's would have serious floating point power, or a few separate fp engines could be assigned to cpu's as needed. Lots of cpu's, doing stuff like file i/o or serial stuff, could be less powerful. I suppose we'll always need special graphics hardware, but just a few of those per chip.

It could go even further. You could have a situation where the "boss" integer only CPU does this:

Re: a dozen cpu's on a chip

Dear Mr Floating processor #1: Please go perform the code at the following address.

Hey Byte slinger processor #7: Go make this memory move.

Hey I/O processor #3: go do this work.

... etc

Yes. But I think that stuff like FPU sharing could be done in hardware, with the OS cpu just setting up the rules. Really scary stuff like memory sharing/moves, disk writes, and DMA – things that work across all physical memory – could only be done by a manager cpu, never by an application. VMS had some protective features like that, and ran dozens of hostile users for months at a whack.

Early in the era of the 8086 there was an 8089 which was called a DMA processor even though it really was programmed I/O in its own instruction set. It could do I/O way faster than the 8086. If a CPU is intended to be part of a server, it could have parts like that in it for doing the things needed for fast disk operations.

It also would make sense to do things like memory moves in the "Memory Mismanagement Unit" since the values don't need to be modified on the way through.

This will make it a lot harder to say how many CPUs are in a chip. If there is only as much hardware as 200 full CPUs but 500 threads can be running at the same time, do you call it 200 or 500 CPUs.

Next step is to get rid of task swapping and threads altogether. One CPU is the OS, and one cpu gets assigned per process.

Actually I can see multiple CPUs being the OS. When you have a lot of tasks to manage, it takes some processing power just to manage them.

Re: a dozen cpu's on a chip

Re: a dozen cpu's on a chip

Future machines with thousands of CPUs may need tens of managers.

But only one executive manager boss. It could be a structured heirarchy: one cpu runs the top-level OS, the thing that gets booted and sets up/supervises everybody else; a few file managers; a bunch of hardware device drivers; a user/GUI interface cpu per screen; some TCP/IP managers, one per port maybe; finally at the bottom of the stack would be applications, which would be rigidly hardware-constrained from doing harm.

The only thing wrong with running all this one a single shared cpu is that nobody seems to be able to do it right. The multi-cpu thing isn't about performance or even architecture: it's a way to force programmers into a new set of rules, because they sure can't manage the ones they have now.

Oh, yeah, dump virtual memory. It was a bad idea when memory was scarce, and a worse one now that ram is bucks per gigabyte.

John

.