THE PERFECT PROCESSOR FOR EMBEDDED SYSTEMS
Embedded designers can’t afford “the blue screen of death.” Desktop operating systems can get rebooted every so often, but embedded systems—good ones, at least—often have to run for years without a single reboot or power cycle. Medical devices are one obvious example in which reliability is absolutely paramount. But industrial-automation systems, security systems, motion controllers, automotive systems, and most other embedded devices have to be just as reliable. And that means building on a reliable foundation.

Building the Best from the Ground Up

For more than 40 years, embedded engineers and programmers have been using Intel® architecture microprocessor chips for their most-demanding embedded systems. There’s good reason for that, and it has to do with Intel’s underlying design decisions. Back in 1971, the very first Intel® microprocessor chip was designed for an embedded system (there were no personal computers in those days), so it had to be reliable. Even then, the Intel architecture design philosophy combined high performance with high reliability. Today, Intel architecture chips embody dozens of different design techniques collected over the years. Every new generation of Intel architecture chips includes all the features acquired from previous years, plus new concepts intended to make embedded systems even more robust, reliable, and secure. The result is a solid foundation for even the most demanding real-time, always-on, hostile environments.

Privilege Has Its Advantages

There are so many robust features inside an Intel architecture processor that it’s hard to know where to start. Let’s begin with the built-in privilege hierarchy. Modern security systems often implement levels of trust, or levels of access security. Remarkably, Intel® microprocessors do this automatically, as a permanent, built-in feature of every single Intel architecture chip. Every bit of software running on an Intel chip is assigned one of four levels of trust, or privilege. Under no circumstances can a piece of software code exceed its privilege level; it’s simply not possible. No amount of hacking, spoofing, or accidental debugging can sidestep the built-in silicon protection circuitry that underlies this privilege mechanism. Many modern operating systems take advantage of Intel’s unique four-level “rings of privilege” to manage their own software tasks. Other embedded systems use the privilege levels to separate trusted software from outside or third-party programs. Some designers reserve the highest privilege level for their own diagnostics, security code, safety watchdog, or as a back-door “kill switch” in case the machine is stolen or compromised. The possibilities are endless—all because Intel processors have security built right in.

“Memory segmentation prevents one program from exceeding its boundaries and accidently (or maliciously) interfering with another program.”

Another security-related feature that all Intel architecture chips share is called memory segmentation. Segmentation has been a feature of Intel microprocessors since the beginning, and its use is twofold: It helps prevent software bugs, and it helps reduce system cost. “Segmenting” memory means putting invisible boundaries around each piece of software, each collection of data, and each internal software “stack” to hold parameters. Like all good fences, these boundaries prevent unwanted intrusion. For example, one program can’t exceed its boundaries and accidentally (or maliciously) interfere with another program. Similarly, one data stack can’t affect another data stack. No unintended sharing of data is allowed, and updates to one set of data can’t surreptitiously overwrite different data. The Intel chip makes sure that
each program, data, and stack stays within its own boundaries, even in the face of deliberate attempts to “jump the fence.”

This memory segmentation also helps to catch accidental program bugs, saving time in the early development stages. A common mistake is to let a program run “off into the weeds,” exceeding its memory boundaries and creating havoc within the system. Memory segmentation prevents any program from ever exceeding its defined boundaries, so even rough programs under development can’t run amok.

“No amount of hacking, spoofing, or accidental debugging can sidestep the built-in silicon protection circuitry that underlies this privilege mechanism.”

A more subtle and insidious bug is called a “stack overrun.” In this case, a program stores too many parameters and runs out of space to store them. Typically, the extra space is quietly borrowed from a different program, but the programmers are often unaware that this has happened. These are particularly difficult bugs to find, let alone fix, and programmers have wasted weeks (or even months) trying to track down mysterious stack overruns. Stack overruns are also a favorite attack vector used by viruses or malware, precisely because they’re so difficult to detect. Having a chip that can help prevent this type of bug from ever happening can be a huge time saver for engineering teams on a tight schedule.

Dense Is Good

Every embedded system requires memory, and memory costs money. So squeezing a program into less memory can translate into savings right on the bottom line. If the bill of materials can get smaller, the entire system might get smaller, too. That’s why Intel® processors are designed to be stingy with memory.

The technical term for this is “code density,” and Intel processors are the industry’s gold standard for code density. Simply put, code density measures the amount of memory that a program (code) requires to run. The denser the code, the less memory it requires.

This is not just an abstract engineering concept, either. Code density can vary quite a bit from one chip family to another, sometimes by as much as two-to-one, meaning some chips need twice as much memory to do the same work as Intel’s chips. That’s a cost hit directly to the bottom line, not to mention a complete waste of memory.

Intel’s exceptional code density isn’t just an accident, it’s by design. Each Intel architecture-family chip mixes 8-bit, 16-bit, 32-bit, and even 64-bit instructions in the most efficient manner possible. Whereas other microprocessor families use only 32-bit instructions, Intel architecture chips easily mix the longer and shorter instructions to pack the most code (software) into the least amount of space. In effect, other chips package everything from oranges to orangutans into a single, one-size-fits-all shipping container, while Intel fits each item into small, medium, or large boxes as required. No wasted space means no wasted memory and no additional cost.

The Taskmaster

One of Intel architecture’s most remarkable features is its built-in task management. Every modern embedded system juggles multiple tasks. Whether it’s real-time kinematics, streaming media, handling security packets, or monitoring multiple processes, the system has to keep tabs on multiple tasks at once. The juggling trick, for engineers and programmers, is to keep all those balls in the air without ever dropping one.

Intel’s processors give embedded developers a big leg up in that area by supporting task management right in the hardware. In effect, Intel architecture does the juggling for you. This allows programmers to avoid a lot of the complex task-management software they’d
normally have to write themselves, or would have to buy from a third-party software company. In some cases, Intel’s built-in task management can entirely replace a simple real-time kernel or task switcher. Imagine: a chip that comes with its own operating system.

The built-in task management is so advanced that it can prevent one task from interfering with a more-important task, so for example, an LED update doesn’t interfere with a critical motor-control loop. It can also allow one task to “chain” to another task, so that the two always run consecutively. Tasks can share information if that’s important, or they can be prevented from sharing information—all automatically. Tasks can start and stop automatically at regular intervals, or based on a timer, or on the occurrence of an interrupt, or when another task completes—there are many possibilities. And again, all this functionality comes with every Intel architecture microprocessor.

**Summary**

There’s still more to explore with Intel’s microprocessor. For instance, every Intel architecture chip comes with support for ASCII arithmetic, a useful feature for many industrial displays. Every Intel architecture chip also supports floating-point arithmetic, a vitally important accelerator for motion control, robotics, kinematics, and other embedded systems. Software loops can be compressed and accelerated through Intel architecture’s built-in loop primitives, a feature programmers will appreciate. And Intel’s media-processing extensions have set the standard for media processing, bringing exciting visual displays within reach of any embedded design team. All in all, Intel’s popular Intel architecture family of microprocessor chips have earned their reputation as the world’s best-known microprocessors, and the solid foundation for thousands of different embedded systems.

© 2013 UBM Tech, a UBM company. All Rights Reserved.
© 2013 Intel Corporation. All Rights Reserved. Intel and the Intel logo are trademarks of Intel Corporation in the United States and/or other countries.
*Other names and brands may be claimed as the property of others.
Information in this document is provided in connection with Intel® products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. Except as provided in Intel’s terms and conditions of sale for such products, Intel assumes no liability whatsoever, and Intel disclaims any express or implied warranty, relating to sale and/or use of Intel products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right. A “Mission critical application” is any application in which failure of the Intel product could result, directly or indirectly, in personal injury or death. Should you purchase or use Intel’s products for any such mission critical application, you shall indemnify and hold Intel and its subsidiaries, subcontractors and affiliates, and the directors, officers, and employees of each, harmless against all claims costs, damages, and expenses and reasonable attorneys’ fees arising out of, directly or indirectly, any claim of product liability, personal injury, or death arising in any way out of such mission critical application, whether or not Intel or its subcontractor was negligent in the design, manufacture, or warning of the Intel product or any of its parts. Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked “reserved” or “undefined.” Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information herein is subject to change without notice. Do not finalize a design with this information. The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request. Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order. Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel’s Web site at www.intel.com.

**ABOUT INTEL**

Intel (NASDAQ: INTC) is a world leader in computing innovation. The company designs and builds the essential technologies that serve as the foundation for the world’s computing devices. Additional information about Intel is available at www.intel.com/pressroom and blogs.intel.com.