Embedded Xinu's Utilization of Multi-core Threading on the Raspberry Pi 3

Patrick J. McGee and Dr. Dennis Brylow
Marquette University, Department of Mathematics, Statistics, and Computer Science

MOTIVATION

Embedded Xinu is an ongoing research project in the Systems Laboratory at Marquette University's Mathematics, Statistics, and Computer Science department. Facilitating an environment for students to test operating system components on real hardware such as the credit card-sized Raspberry Pi 3, the Embedded Xinu laboratory serves as a hands-on tool for concepts such as concurrency on a multi-core platform. By adapting this small form factor, educational operating system to the advanced Pi 3, this project aims to pave the roads for educators to demonstrate essential aspects of modern computing.

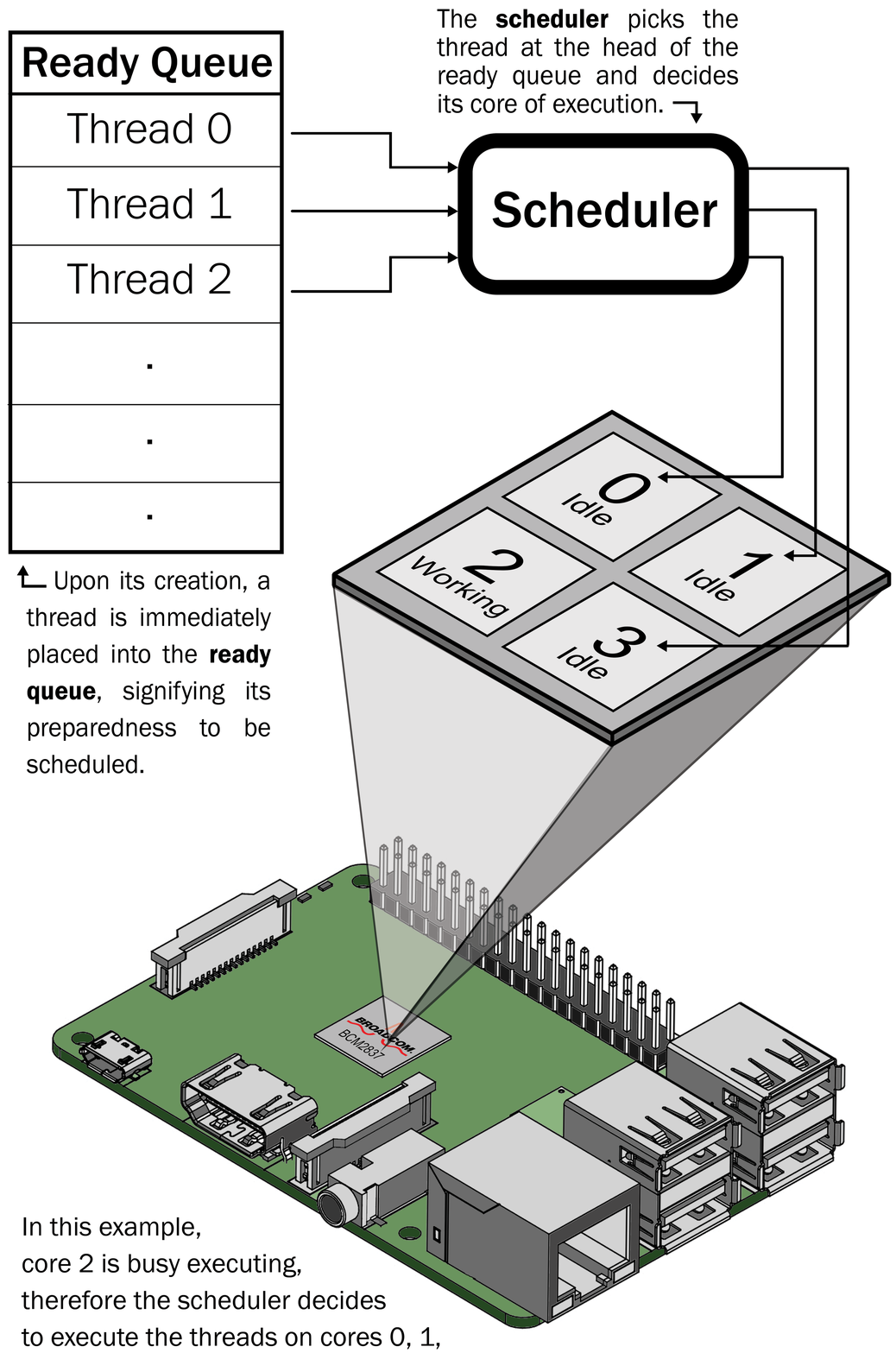
PREVIOUS WORK

Before this writing, Embedded Xinu had been ported onto the single-core CPU (Central Processing Unit) structure of the Raspberry Pi 1. On this single-core platform, depending on the operating system's scheduler, multiple threads may execute one-after-the-other or wait until one thread finishes or is preempted by the operating system.

MULTI-CORE THREADING

The CPU of the Raspberry Pi 3 is a chip made up of four cores, each of which use its own registers to perform arithmetic and logical operations at a high clock speed of 1.2GHz. Embedded Xinu takes advantage of this CPU's extensive hardware by utilizing concurrent multi-core threading. A thread is a set of values controlling the processor registers for a single core, thus commanding the order of operation execution. It is the job of the operating system to place newly-created threads into a ready queue, while the scheduler determines the core on which the thread executes.

MULTI-CORE SCHEDULING



In this example, core 2 is busy executing, therefore the scheduler decides to execute the threads on cores 0, 1, and 3, because they are idle (performing no vital activity).

METHODS

When the Pi 3 powers on, Embedded Xinu uses core 0 to set up the starting environment by performing the following tasks:

- Passing ATAGs for device memory information
- Initialize variables for C environment
- Start serial driver for console output
- Allocate all available memory for the OS (1GB RAM total)

Upon boot, cores 1-3 of the CPU are in a "parked" state in which they wait in an infinite loop until they are interrupted. In order to interrupt -- or "unpark" -- a core, there is a mailbox system in which, upon direct access to its memory location, the core runs that thread at that location immediately. With this multi-core mailbox system, Embedded Xinu now supports genuine concurrency of threads, thus providing a fourfold increase in execution speed compared to its parallel, single-core predecessor.

SINGLE-CORE vs. CONCURRENT MULTI-CORE

	Single-core	Four-core
Number of threads	4	4
Execution time per thread	5μs	5μs
Net execution time	20μs	5μs

REFERENCES

Arnold, Jason, et al. "Xinu-os/xinu." GitHub, 2015
Doeppner, Thomas W. "Operating Systems In Depth." Wiley, 2011.
Gagne, Abraham Silberschatz, Peter B. GalvinGreg. "Operating System Concepts." Wiley, 2012
Ras Pi World. All Raspberry Pi Products Dimension Drawings. Ras Pi World, 2016. Web.

ACKNOWLEDGEMENTS

Special thanks to Dr. Dennis Brylow, the National Science Foundation, the Raspberry Pi developers, and my research team.