PROCESS SCHEDULING IN EMBEDDED XINU

Tom Lazar, Patrick J. McGee, Rade Latinovich,Priya Bansal, and Dennis Brylow

INTRODUCTION

Embedded Xinu is a lightweight operating system that is designed to assist in teaching high school and college-level students operating systems concepts[1]. The goal of our project was to port the code to the new Raspberry Pi 3 platform[2]. We chose to do this because:

• Raspberry Pi 1 are no longer being sold

• New hardware features allow Xinu to assist with a larger array of subjects

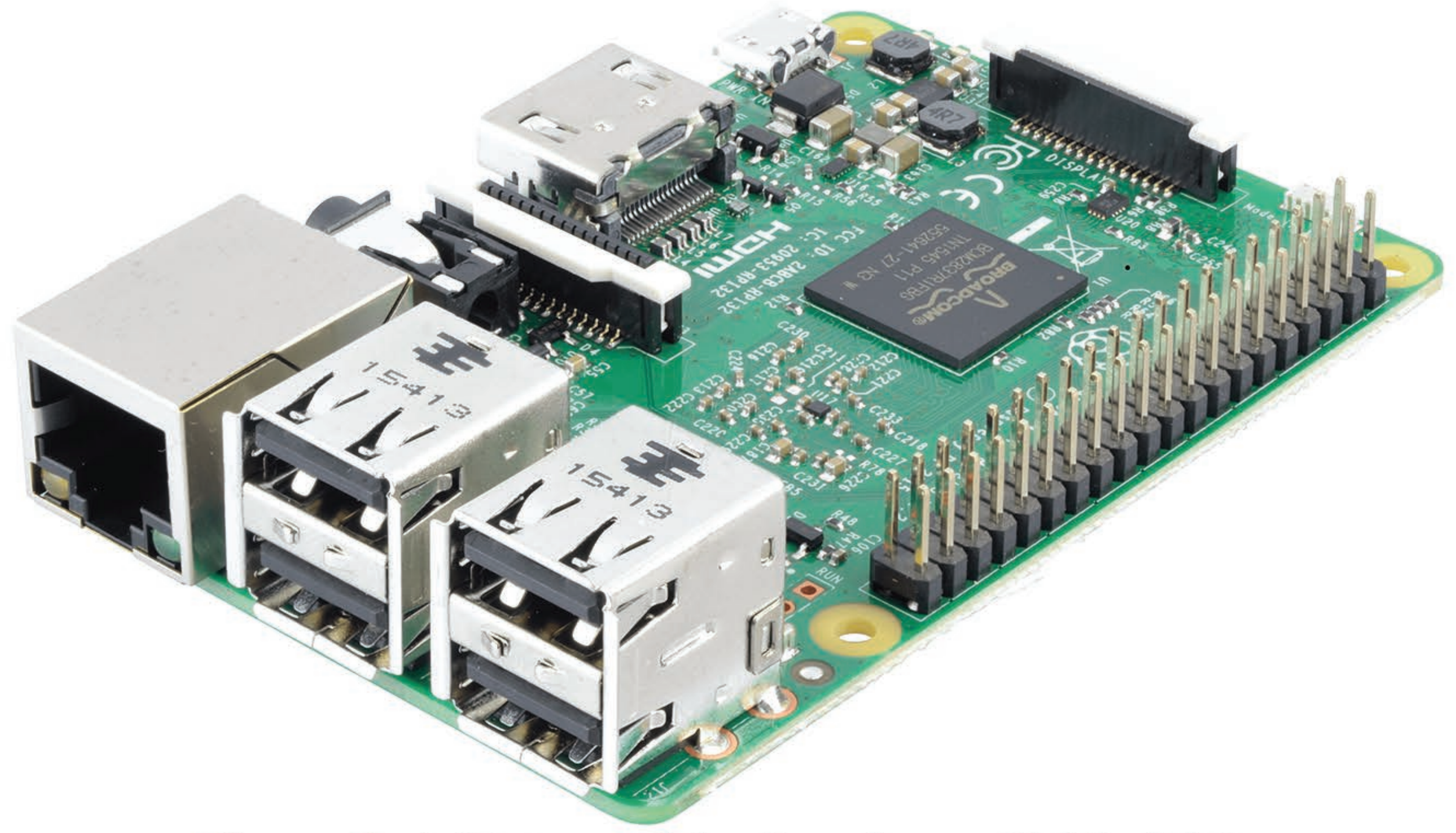

Figure 1: A Picture of the Raspberry Pi Model 3

SIMULTANEOUS MULTI-PROCESSING (SMP)

The Raspberry Pi 3 has an upgraded processor form the Original Raspberry Pi, the new Arm Cortex-A53 processor has 4 cores that are each faster than the core on the Pi 1.

What is Multi-Processing…

• Separate Processes operating Concurrently on each processor

• It is not Multitasking, Multitasking simulates Multi-Processing by switching the tasks run-ning on a single core rapidly.[4]

The Previous Versions of Xinu had a multitasking system, since none of the previous platforms could support anything else, but now we can expand it to a Multiprocessing System. Multiprocessing however means true parallel execution of multiple processes using more than one processor.

```
v3.14
Welcome to the wonderful word of Xinu!
xsh$
```

Figure 2: XinuPi Shell Welcome Screen

SCHEDULING PROCESSES

The process scheduler is the component of the operating system that is responsible for deciding whether the currently running process should continue running and, if not, which process should run next.

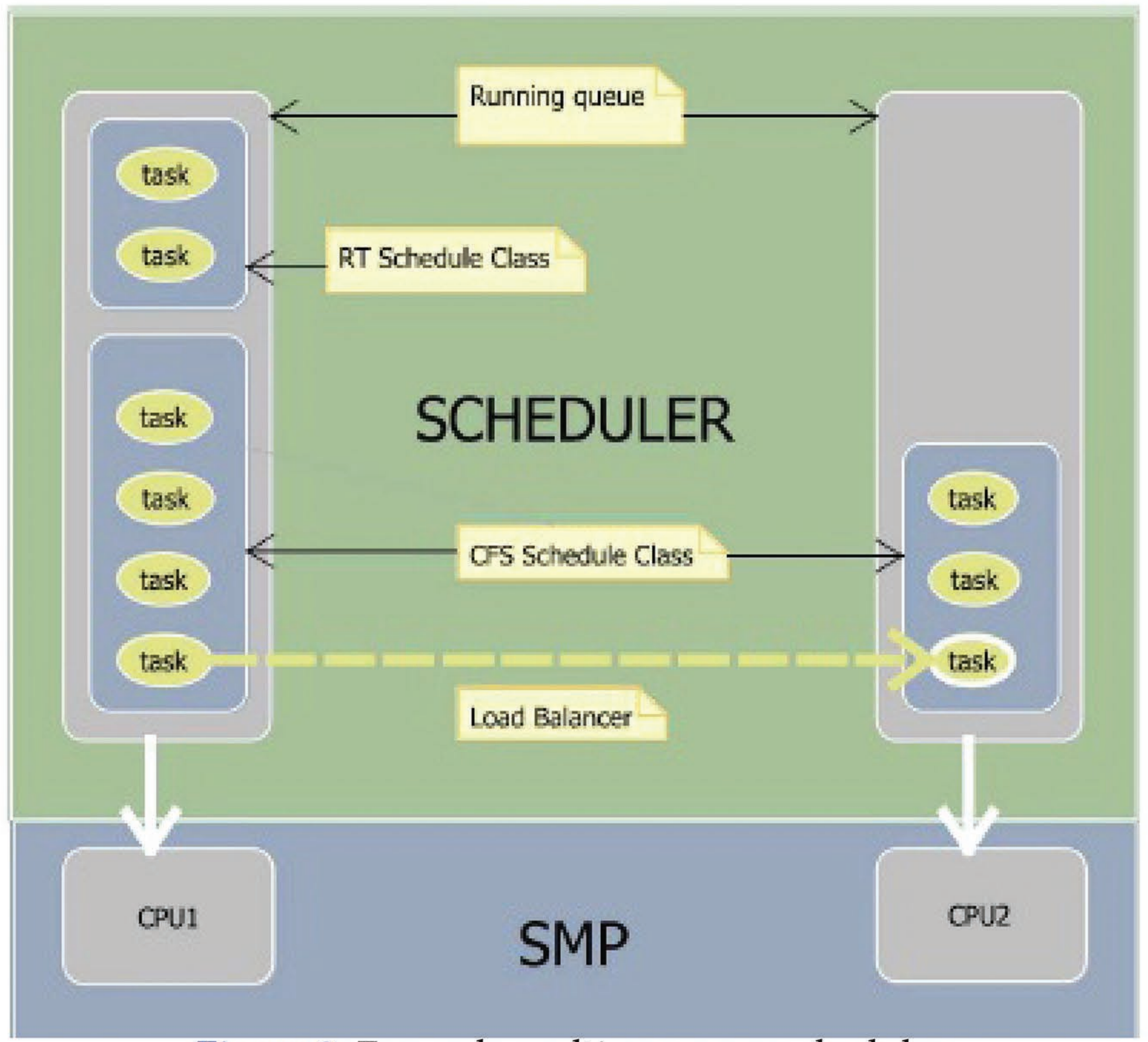

Figure 3: Example multiprocessor scheduler.

A process must on every core at all times since there might not always be a process that is running in xinu, we have a Null Process, a process that takes up any extra time the processor might have.

THE ctxsw() FUNCTION

In Xinu `ctxsw()` or context switch controls the low level mechanics of how a processor stops working one task and begins working on another. Every time the scheduler decides that a waiting process needs to be run, the context switch is called to exchange the waiting process with the active one. In the Operating Systems Course at Marquette students are assigned to implement this as an assignment. Due to the fact this function operates directly on the memory stored in each core this operating is specific to each implementation of Xinu.

ACKNOWLEDGEMENTS

All work completed was with the mentorship of Dr. Dennis Brylow at Marquette University. The National Science Foundation made this relationship possible. Without the support and guidance from Dr. Dennis Brylow, Marquette University, all past Xinu researchers[3] and the National Science Foundation, the opportunity to port embedded Xinu during the summer of 2017 would have never been possible. NSF CE21 award #CNS-1339392.

REFERENCES

[1] http://xinu.mscs.mu.edu/Main_Page

[2] https://www.raspberrypi.org/help/what-is-a-raspberry-pi

[3] https://github.com/tomlazar/xinu/blob/master/AUTHORS

[4] Shibu K. V. Introduction to Embedded Systems. Tata McGraw-Hill Education. p. 402. ISBN 978-0-07-014589-4.