Project #1

Student Names and SFUIDs:

Zhixin Huang 301326521
Nontawat Janpongsri 301311427

Purpose: This pair-project focuses on documentation for a software package.

Tasks:

Phase One (Today – Wednesday January 29th):

- Find a partner for this week-long project. Please do not ask me to pair you up. If you cannot find a partner, then you will have to work alone.
- Become familiar with the concept of pair-programming – know the pros and cons and be able to classify it within the context of software engineering practices.
- Read and understand *everything* in the following blog post:

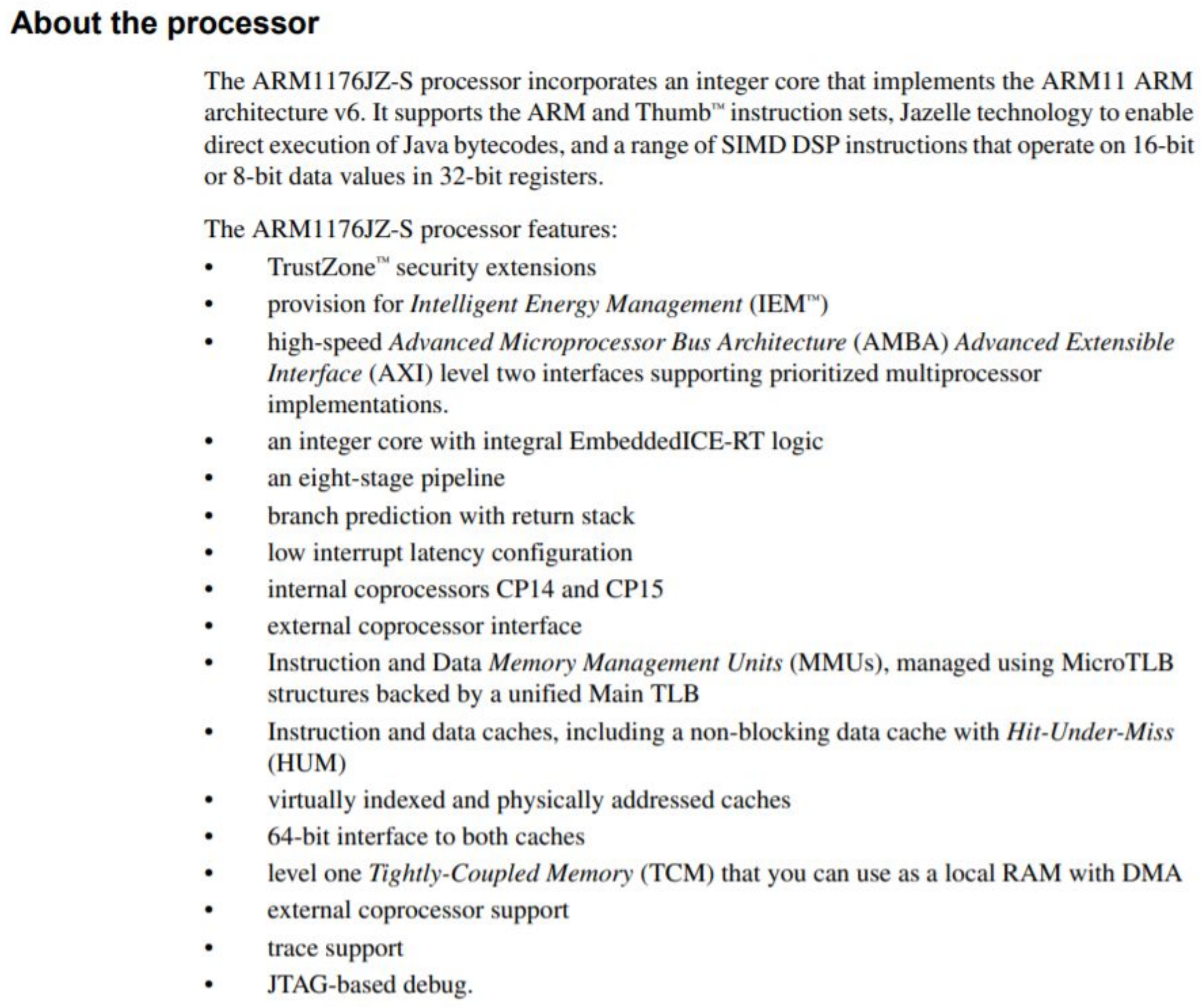
  https://www.ybrikman.com/writing/2014/05/05/you-are-what-you-document/

Phase Two (Due next Friday, February 7th):

- You will then be set upon a week-long task next class (Friday, June 3rd) involving the *three types of documentation* referred to in the post above.
- Submission instructions will be provided next week by the TA.
- You will have the rest of next week to work on this project outside of class in your pairs.
- Use this document as your cover page

IMPORTANT NOTE: I do not make attendance mandatory. However, it is the student's responsibility to find out what was discussed from your peers. Assignments and in-class activities are designed so that attendance is NOT mandatory. However, those that do not attend will find the assignment somewhat difficult.

Writing Documentation



About the processor

The ARM1176JZ-S processor incorporates an integer core that implements the ARM11 ARM architecture v6. It supports the ARM and Thumb™ instruction sets, Jazelle technology to enable direct execution of Java bytecodes, and a range of SIMD DSP instructions that operate on 16-bit or 8-bit data values in 32-bit registers.

The ARM1176JZ-S processor features:

- TrustZone™ security extensions
- provision for *Intelligent Energy Management* (IEM™)
- high-speed *Advanced Microprocessor Bus Architecture* (AMBA) *Advanced Extensible Interface* (AXI) level two interfaces supporting prioritized multiprocessor implementations.
- an integer core with integral EmbeddedICE-RT logic
- an eight-stage pipeline
- branch prediction with return stack
- low interrupt latency configuration
- internal coprocessors CP14 and CP15
- external coprocessor interface
- Instruction and Data *Memory Management Units* (MMUs), managed using MicroTLB structures backed by a unified Main TLB
- Instruction and data caches, including a non-blocking data cache with *Hit-Under-Miss* (HUM)
- virtually indexed and physically addressed caches
- 64-bit interface to both caches
- level one *Tightly-Coupled Memory* (TCM) that you can use as a local RAM with DMA
- external coprocessor support
- trace support
- JTAG-based debug.

About the coprocessor interface

The processor supports the connection of on-chip coprocessors through an external coprocessor interface. All types of coprocessor instruction are supported.

The ARM instruction set supports the connection of 16 coprocessors, numbered 0-15, to an ARM processor. In the processor, the following coprocessor numbers are reserved:

CP10	VFP control
CP11	VFP control
CP14	Debug and ETM control
CP15	System control.

You can use CP0-9, CP12, and CP13 for your own external coprocessors.

The processor is designed to pass instructions to several coprocessors and exchange data with them. These coprocessors are intended to run in step with the core and are pipelined in a similar way to the core. Instructions are passed out of the Fetch stage of the core pipeline to the coprocessor and decoded. The decoded instruction is passed down its own pipeline. Coprocessor instructions can be canceled by the core if a condition code fails, or the entire coprocessor pipeline can be flushed in the event of a mispredicted branch. Load and store data are also required to pass between the core *Logic Store Unit* (LSU) and the coprocessor pipeline.

The coprocessor interface operates over a two-cycle delay. Any signal passing from the core to the coprocessor, or from the coprocessor to the core, is given a whole clock cycle to propagate from one to the other. This means that a signal crossing the interface is clocked out of a register on one side of the interface and clocked directly into another register on the other side. No combinatorial process must intervene. This constraint exists because the core and coprocessor can be placed a considerable distance apart and generous timing margins are necessary to cover signal propagation times. This delay in signal propagation makes it difficult to maintain pipeline synchronization, ruling out a tightly-coupled synchronization method.

The processor implements a token-based pipeline synchronization method that enables some slack between the two pipelines, while ensuring that the pipelines are correctly aligned for crucial transfers of information.

Code Documentation

Coprocessor instructions

Each coprocessor might only execute a subset of all possible coprocessor instructions. Coprocessors reject those instructions they cannot handle. Table 11-1 lists all the coprocessor instructions supported by the processor and gives a brief description of each. For more details of coprocessor instructions, see the *ARM Architecture Reference Manual*.

Table 11-1 Coprocessor instructions

Instruction	Data transfer	Vectored	Description
CDP	None	No	Processes information already held within the coprocessor
MRC	Store	No	Transfers information from the coprocessor to the core registers
MCR	Load	No	Transfers information from the core registers to the coprocessor
MRRC	Store	No	Transfers information from the coprocessor to a pair of registers in the core
MCRR	Load	No	Transfers information from a pair of registers in the core to the coprocessor
STC	Store	Yes	Transfers information from the coprocessor to memory and might be iterated to transfer a vector
LDC	Load	Yes	Transfers information from memory to the coprocessor and might be iterated to transfer a vector

The coprocessor instructions fall into three groups:

- loads
- stores
- processing instructions.

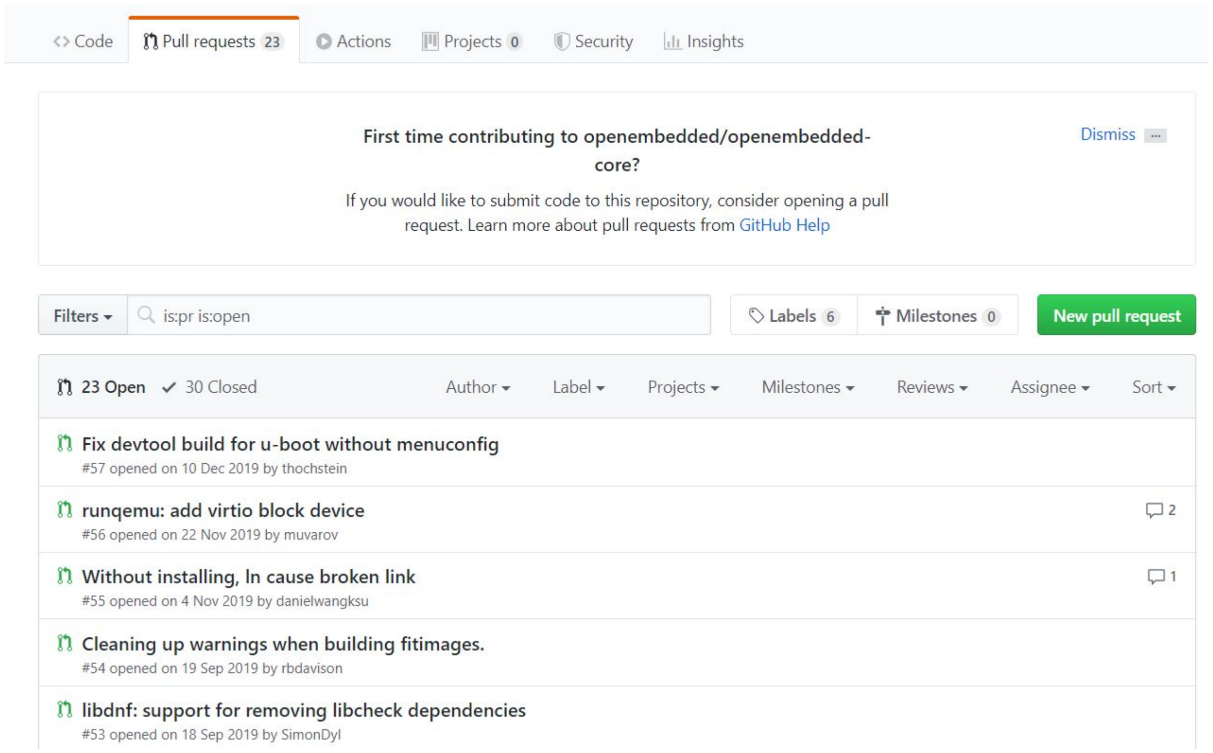
Coprocessor control

The coprocessor communicates with the core using several signals. Most of these signals control the synchronizing queues that connect the coprocessor pipeline to the core pipeline. Table 11-2 lists the signals used for general coprocessor control.

Table 11-2 Coprocessor control signals

Signal	Description
CLKIN	This is the clock signal from the core.
nRESETIN	This is the reset signal from the core.
ACPNUM[3:0]	This is the fixed number assigned to the coprocessor, and is in the range 0-13. Coprocessor numbers 10, 11, 14, and 15 are reserved for system control coprocessors.
ACPENABLE	When set, enables the coprocessor to respond to signals from the core.
ACPPRIV	When asserted, indicates that the core is in privileged mode. This might affect the execution of certain coprocessor instructions.

Community Documentation



Code | Pull requests 23 | Actions | Projects 0 | Security | Insights

First time contributing to openembedded/openembedded-core?

If you would like to submit code to this repository, consider opening a pull request. Learn more about pull requests from GitHub Help

Dismiss

Filters | is:pr is:open | Labels 6 | Milestones 0 | New pull request

23 Open ✓ 30 Closed | Author | Label | Projects | Milestones | Reviews | Assignee | Sort

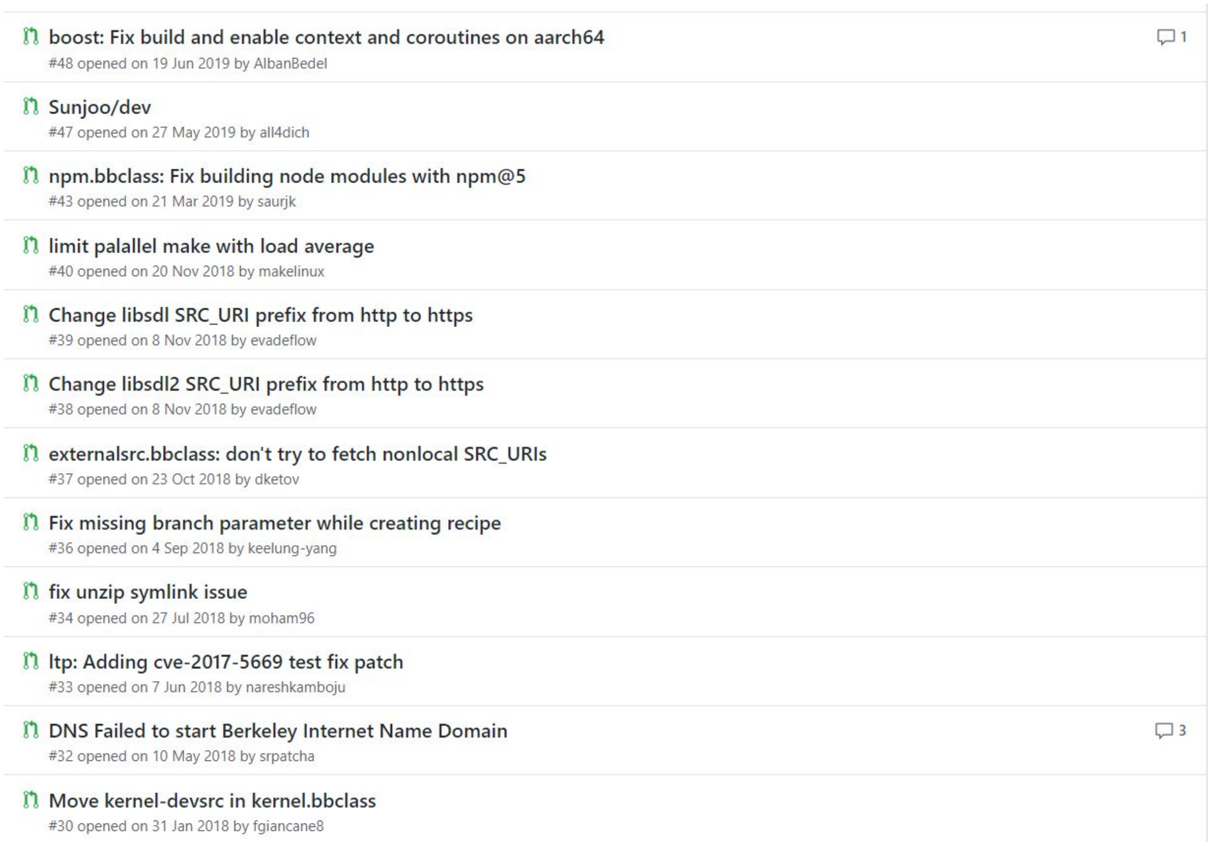
Fix devtool build for u-boot without menuconfig
#57 opened on 10 Dec 2019 by thochstein

runqemu: add virtio block device 2
#56 opened on 22 Nov 2019 by muvarov

Without installing, ln cause broken link 1
#55 opened on 4 Nov 2019 by danielwangksu

Cleaning up warnings when building fitimages.
#54 opened on 19 Sep 2019 by rbdavison

libdnf: support for removing libcheck dependencies
#53 opened on 18 Sep 2019 by SimonDyl



boost: Fix build and enable context and coroutines on aarch64 1
#48 opened on 19 Jun 2019 by AlbanBedel

Sunjoo/dev
#47 opened on 27 May 2019 by all4dich

npm.bbclass: Fix building node modules with npm@5
#43 opened on 21 Mar 2019 by saurjk

limit palallel make with load average
#40 opened on 20 Nov 2018 by makelinux

Change libsdl SRC_URI prefix from http to https
#39 opened on 8 Nov 2018 by evadeflow

Change libsdl2 SRC_URI prefix from http to https
#38 opened on 8 Nov 2018 by evadeflow

externalsrc.bbclass: don't try to fetch nonlocal SRC_URIs
#37 opened on 23 Oct 2018 by dketov

Fix missing branch parameter while creating recipe
#36 opened on 4 Sep 2018 by keelung-yang

fix unzip symlink issue
#34 opened on 27 Jul 2018 by moham96

ltp: Adding cve-2017-5669 test fix patch
#33 opened on 7 Jun 2018 by nareshkamboju

DNS Failed to start Berkeley Internet Name Domain 3
#32 opened on 10 May 2018 by srpatcha

Move kernel-devsrc in kernel.bbclass
#30 opened on 31 Jan 2018 by fgiancane8

Reference

ARM Information Center. [Online]. Available: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0333h/Bgbhbiah.html.

Openembedded, “openembedded/openembedded-core,” *GitHub*. [Online]. Available: https://github.com/openembedded/openembedded-core/pulls.