ECE/CS 5780-6780 : Embedded Systems Design
Lect. 04: ARM Toolchain - Memory manipulation

Pierre-Emmanuel Gaillardon
Department of Electrical and Computer Engineering – University of Utah

Spring 2017
Salt Lake City, UT, USA

Objectives for Today!

- **Get familiar with the ARM M processor architecture**
- **Understand the architecture of a microcontroller**
- Acquire the fundamentals of the hardware/software interface
- Acquire the fundamentals for sensing and controlling the physical world
- Learn modeling techniques for embedded system design
- Understand the usage of several peripherals through labs
- Design a complete embedded system – from specs to realization
  - Realization of a PCB
  - Behavioral modeling of the system
  - Complete SW realization

Cross-compilation Toolchain

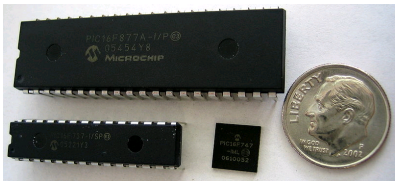
Programming Microcontrollers

- Multiple levels of abstraction possible, the choice depends on required I/O efficiency and complexity in general of embedded system validation

Abstraction level

MCS-48 (8048, 8035, 8748) [Intel Corp., 1976]
(8-bit architecture, 64-128B RAM, 1MHz)

\- Low level = *Assembly Language,* each line is a minimum fundamental operation to be done by microcontroller/microprocessor



PIC16x84 families [Microchip, 1993]
(14-bit architectures, 512B-2.5KB RAM, up to 20MHz)

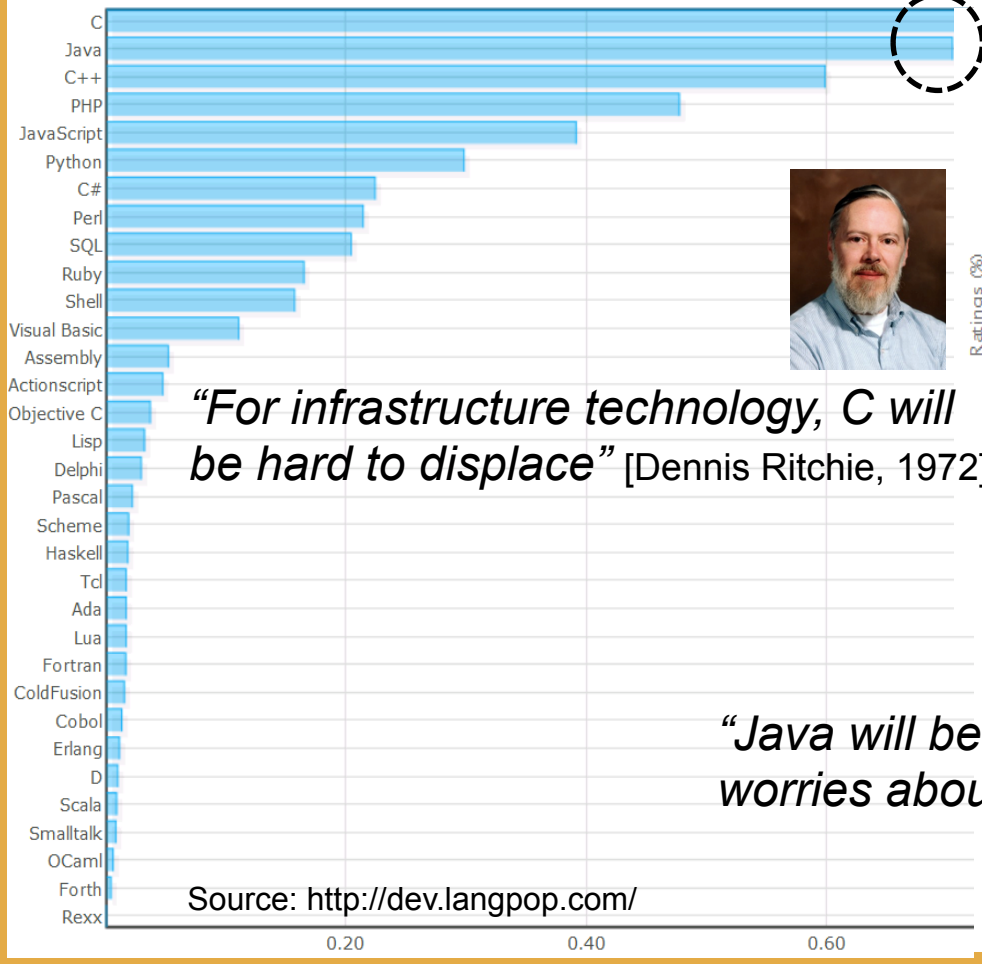
High-level abstraction languages

1967	BCPL	Martin Richards	Cambridge
1969	B	Ken Thomson	Bell Labs
1972	**C**	**Dennis Ritchie**	**Bell Labs**
1983	C++	Bjarne Stroustrup	Bell Labs
1995	Java	James Gosling	Sun

\+

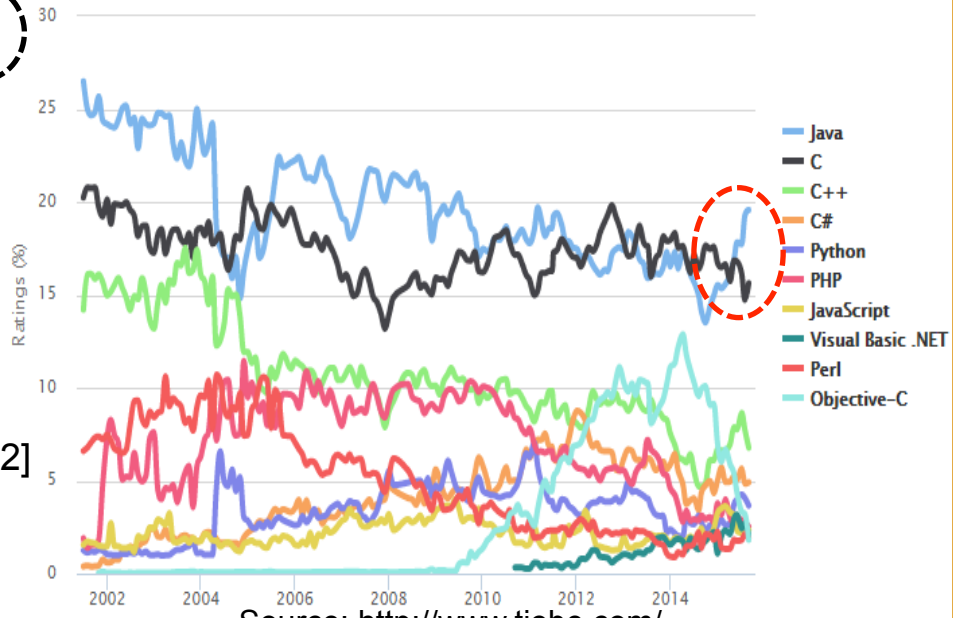
Java and C: Trendiest languages for ES

- Very different objectives, but both have found their reason to be there
  - Java: most retargettable solution (Virtual machine), interpreted language
  - C: closest to HW language, very optimized compilation technology



Source: http://dev.langpop.com/

"For infrastructure technology, C will be hard to displace" [Dennis Ritchie, 1972]



Source: http://www.tiobe.com/

"Java will be the next wave in computing: no worries about portability" [James A. Gosling, 1995]

The C language

- Nowadays, the most commonly-used programming language for embedded systems
- Powerful and easy-to-use to express algorithmic steps
  - Only 32 reserved words
- Considered "high-level" assembly …
  - Low-level control of what the processor does
  - Efficient compilers available almost for every existing architecture
  - High efficiency of the produced code
- … but highly portable
  - From mainframes to microprocessors and micro-controllers

Cross-development Platform

- Microprogrammed embedded systems are devices with limited resources, and are typically not powerful enough to run a compiler, a file system or a development environment

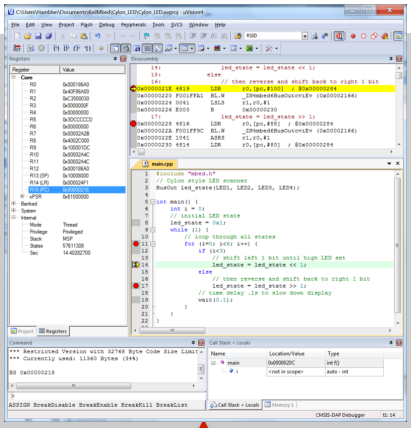
- Cross development is the separation of the system build environment from the target environment

- Benefits
  - Faster compilation of applications
  - Debugging and testing with more resources than available on target embedded system

Cross-toolchain for C language

- The complete flow to generate the final binary for the target platform and its validation for the target micro-programmed embedded systems requires a cross-compilation toolchain
- It is a set of tools running on a host machine, used to
  1. Pre-process header files (*#include*) and macros (*#define*) and Compile high-level source code (*.c*) to the target object code (*.o*)
     - Possible to get assembly output as intermediate step (*.s*)
  2. Link pre-existing collections or libraries of object files (*.o*) to obtain a final executable object (.elf, *.axf*) with all the necessary object code
  3. Pack and format the executable object into a format that can run with the target memory hierarchy and I/O subsystem

System co-design framework



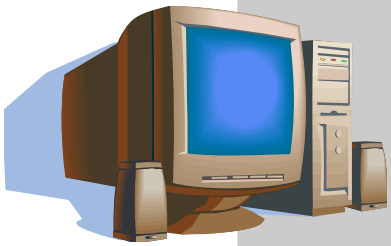
.c

Target architecture information

Tool Chain

.o

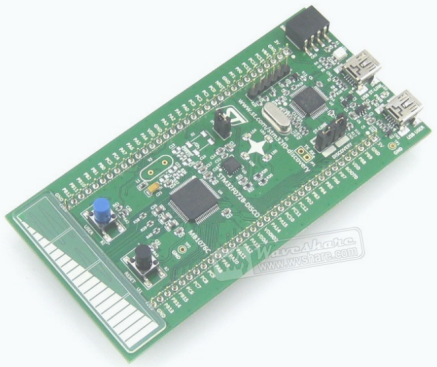
Favourite editor (our source code)



Development PC

Debugger

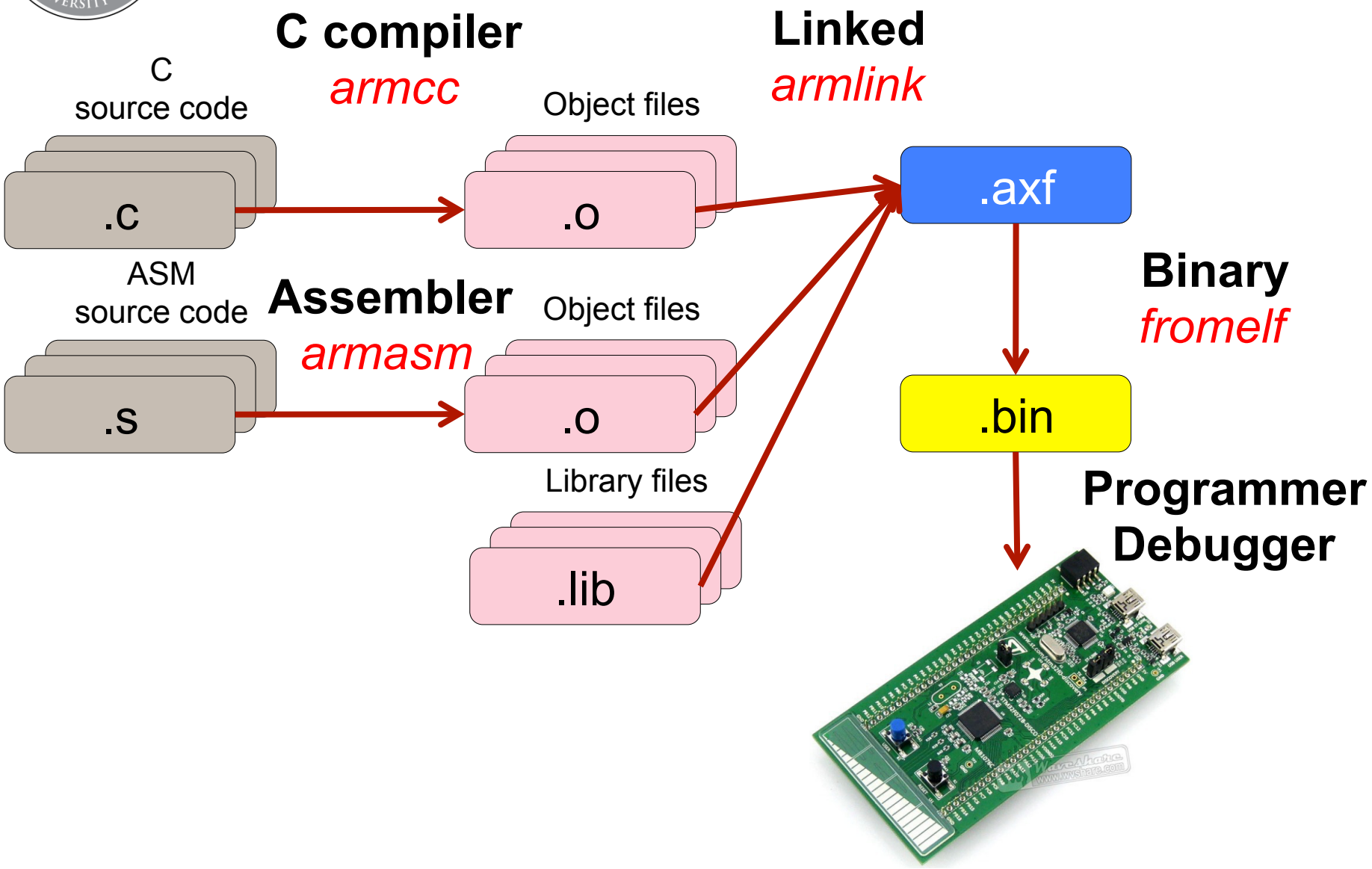
Develop. board or final HW



SOFTWARE PART

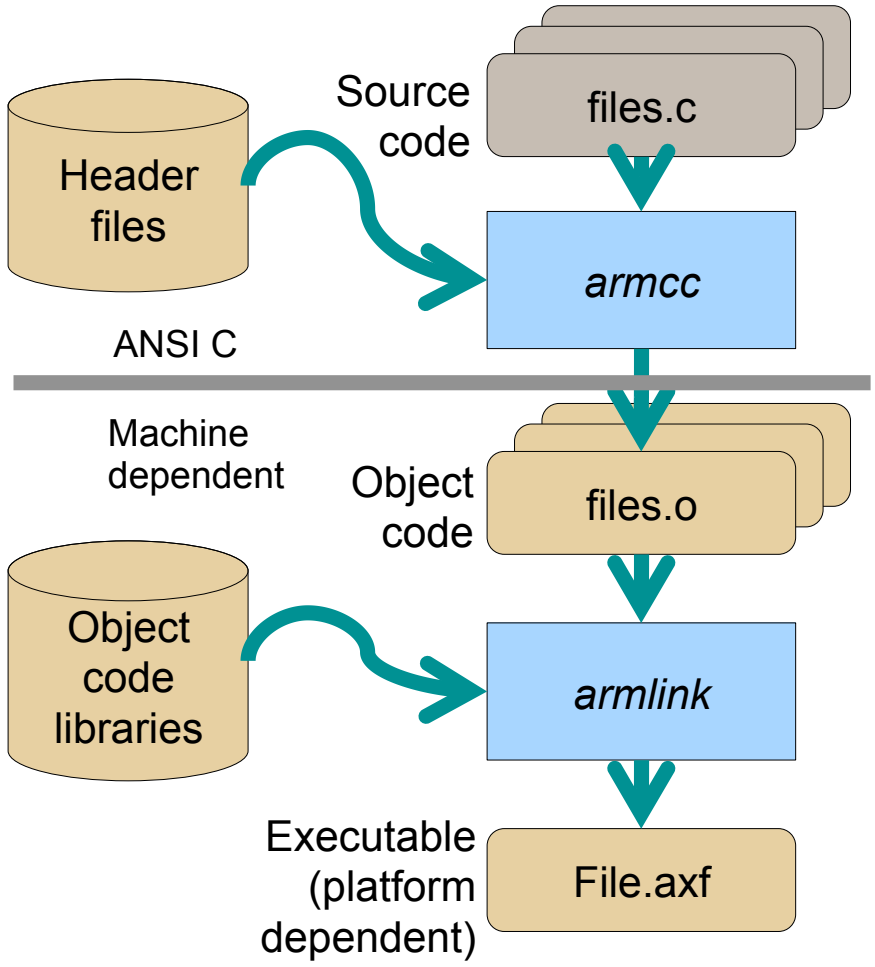
HARDWARE PART

Keil C-Based cross-compilation flow



Seeking compatibility in compilation

- Two separated phases
  1. Source code is easy to migrate
     - Standard source language (e.g., ANSI C)
     - Standard multi-platform libraries (e.g., *stdio*, *stdlib*, etc.)
  2. Object code is not portable, not even among compilers for the same architecture
     - Third-party object code linked in platform-dependent phase



Example: compiling C source code

```
#include <nds.h>
#include <stdio.h>
#define TRUE 1
#define FALSE 0

int main(void)
{
    int i;
    i = 5 * 2;
    printf("5 times 2 is %d.\n", i);
    printf("TRUE is %d.\n", TRUE);
    printf("FALSE is %d.\n", FALSE);
}
```

Handled by the pre-processor

Handled by the compiler

Implemented in C library for the target platform

Memory System

Memory and Processor – The big love

- What does a processor do?

  It manipulates data!

  Where are the data located?

  In memories!

A processor only sees memory!

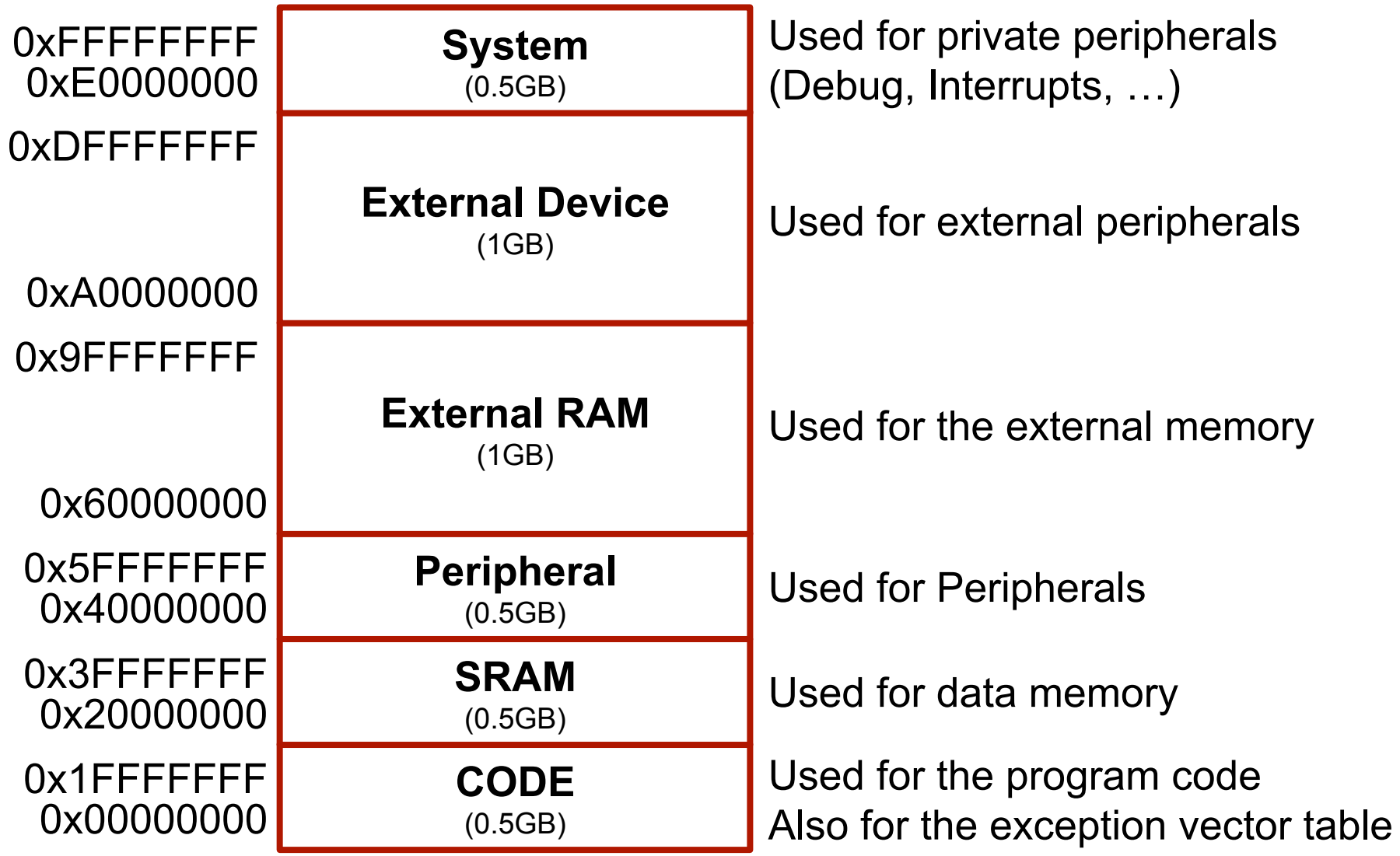
- How does it access memory?

Through a data bus (with access and data)

Overview

- All ARM Cortex-M processors have a 4GB of memory address space
- This space is architecturally defined into a number of regions
- Each region having a recommended usage to help software porting between different devices

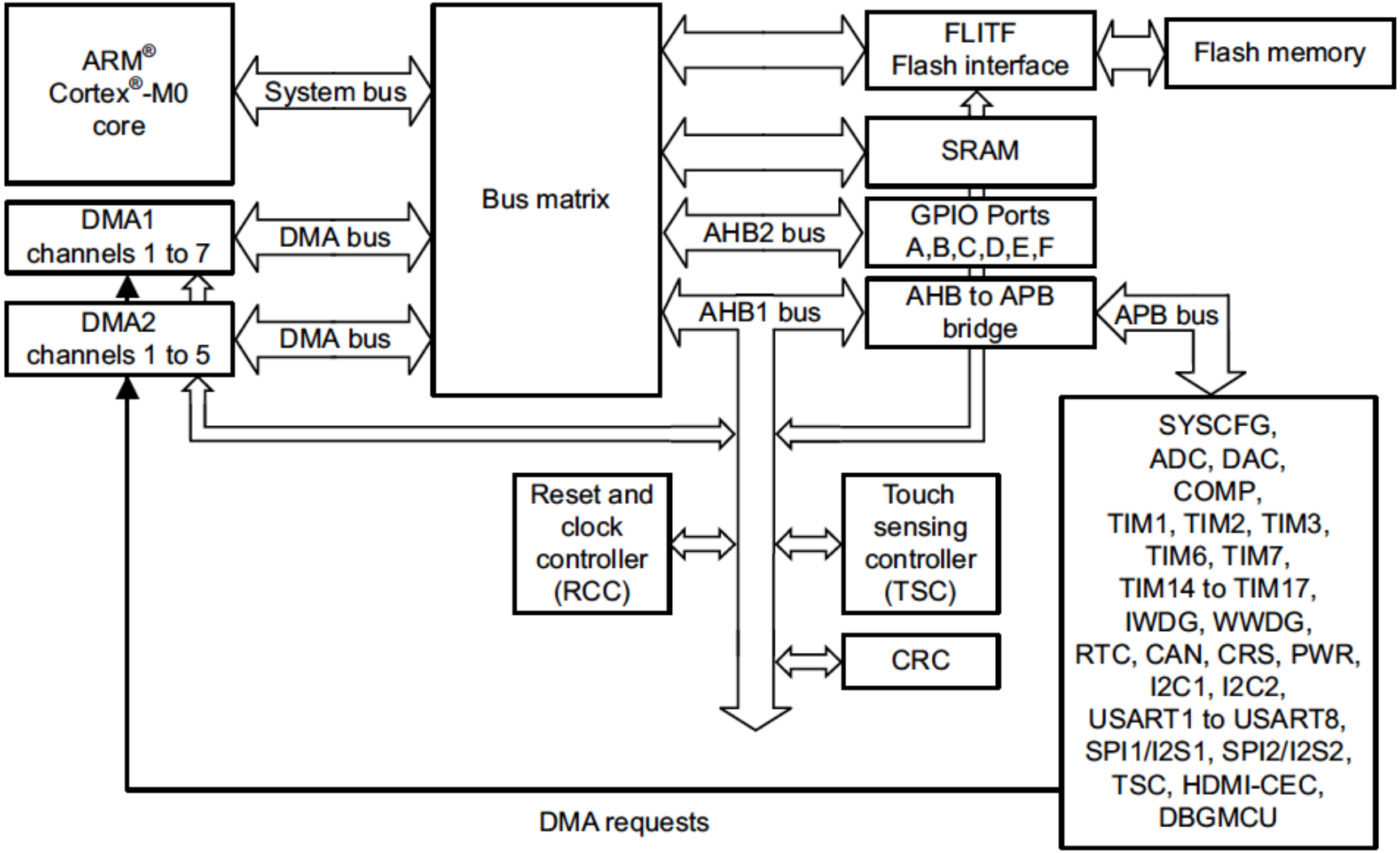
ARM-Cortex M0 Memory Map



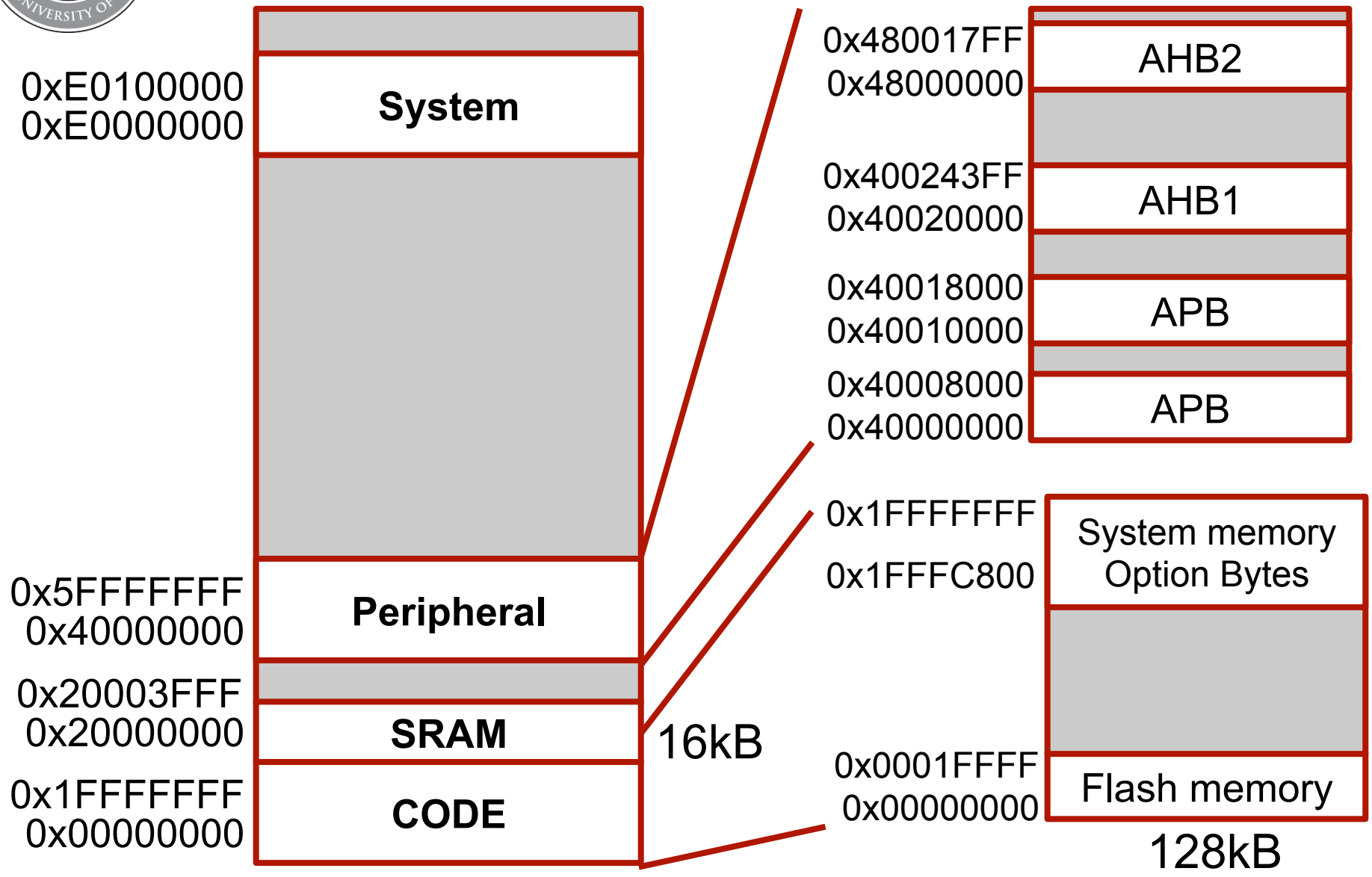
Vendor-specific Memory Design

- Although all Cortex-M0 have this fixed memory map, the usage of the memory is very flexible.
- In an embedded system, there are different types of memory:
  - Flash
  - SRAM
  - ROM
- There are different manipulation strategies:
  - Regular access
  - Stack
  - Heap
- How does the memory map of a real system look like?

STM32F072RB System Architecture



STM32F051R8 Memory Map



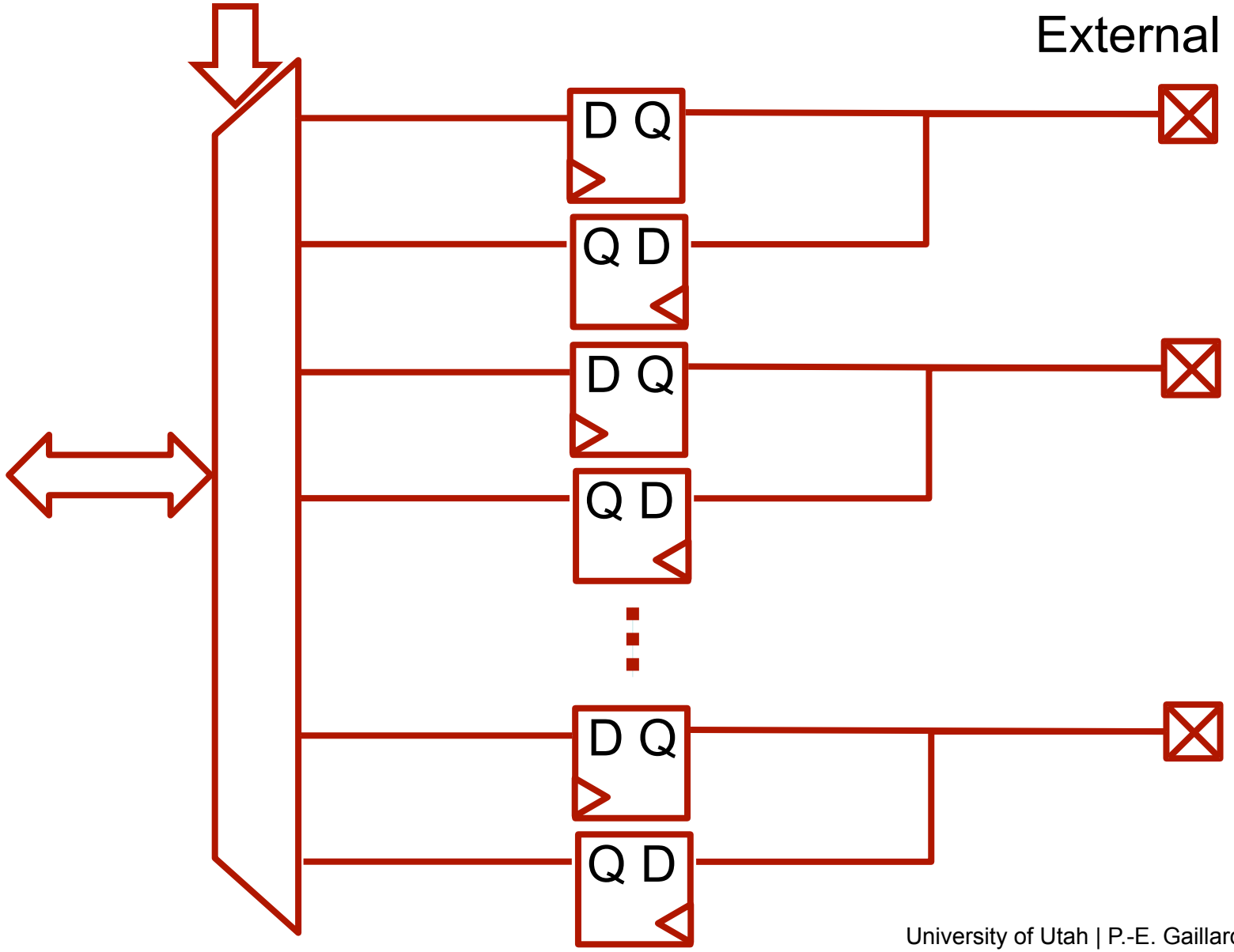
A few Words about AHB and APB busses

- The Cortex-M0 core has a 32-bit system interface based on the protocol AHB-Lite
- AHB (Advanced High-performance Bus) is a protocol defined in the AMBA standard.
- AMBA (Advanced Microcontroller Bus Architecture) is developed by ARM and is an industry standard.
- AHB supports read/write transfers with:
  - 32-, 16- and 8-bit data
  - Pipelined operation
  - Support of wait states
  - Support of slave responses
- For slower device, a secondary bus segment is often used based on the APB (Advanced Peripheral Bus) protocol.
- The APB bus is connected to the AHB-Lite via a bus bridge and may run at a different clock speed.

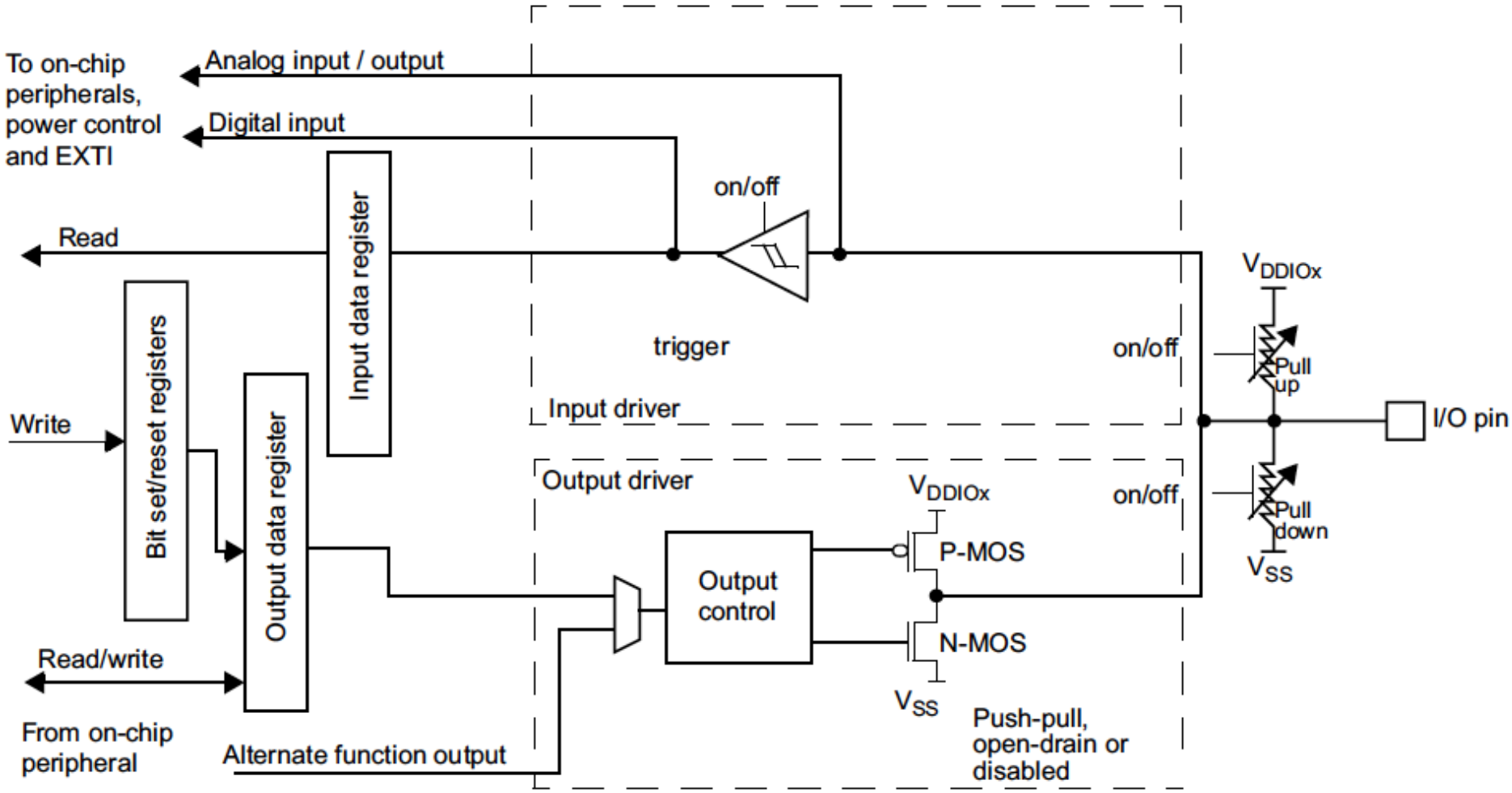
How does a Peripheral Work?

- It is a device interface on a bus
- What components do you need?
  - Bus interface
  - Controls
  - REGISTERS!
- A Peripheral is accessed and controlled as a memory
- You read and write Memory Locations.
- This is called a memory-mapped peripheral

A General View of an I/O Bank



Things are of course a bit more complicated!



Thank you for your attention

Questions?

Laboratory for NanoIntegrated Systems
Department of Electrical and Computer Engineering
MEB building – University of Utah – Salt Lake City – UT – USA