SPRING 2016
CMPE 364

Microprocessor Based Design

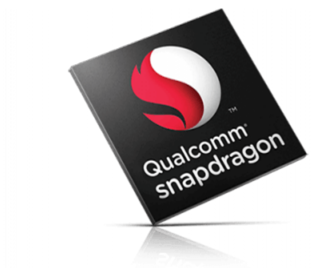
Dr. Ryan Riley
(Slides adapted from Dr. Mohamed Al-Meer)

Apollo Guidance Computer

- 64 Kbyte of memory
- 1.024 MHz
- 16-bit registers

- Took the first men to the moon in 1969

Snapdragon 835



- 8 GB memory (maybe more)
- 2.45 GHz, 8-cores
- 64-bit registers
- Integrated GPU capable of full HD virtual reality
- Supports up to a 4K screen
- Integrated multi-gigabit wireless
- Plays Angry Birds *really* fast

Lecture Objectives

- Objectives
  - An Introduction to Embedded Systems with ARM processor
- Expected to achieve:
  - Short history of ARM
  - ARM in industry
  - Know about ARM registers

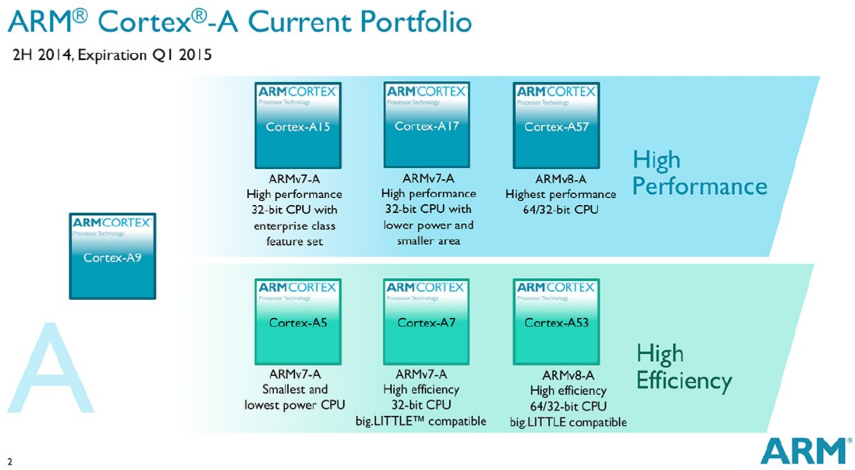
ARM PROCESSOR FUNDAMENTALS

Introduction

ARM PROCESSOR FUNDAMENTALS

- The ARM processor, like all RISC processors, uses a *load-store architecture*.
  - ARM is a RISC processor.
  - It is used for small size and high performance applications.
  - Simple architecture – low power consumption
- has two instruction types for transferring data in and out of the processor:
  - Load: instructions copy data from memory to registers in the core
  - Store: copy data from registers to memory
- no data processing instructions directly manipulate data in memory

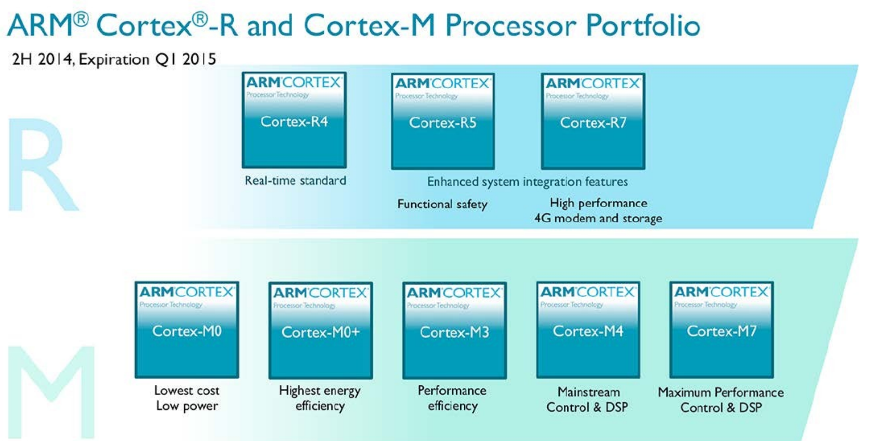
What is ARM CORTEX-A?



Cortex-A Application Processors are defined by the processor's ability to execute complex operating systems, including Linux, Android, Chrome OS, Tizen, Microsoft Windows (CE/Embedded) and others. This class of processors integrates a Memory Management Unit (MMU) to manage the memory requirements of complex OSs and enable the download and execution of third party software.

•Smartphones *Feature Phones *Tablets / eReaders *Adv. Personal Media Players
•Digital Television *Set-top Boxes/Satellite Receivers *High-End Printers
•Personal Navigation Devices *Server/Enterprise *Wearables *Home Networking

What is ARM CORTEX R and M?



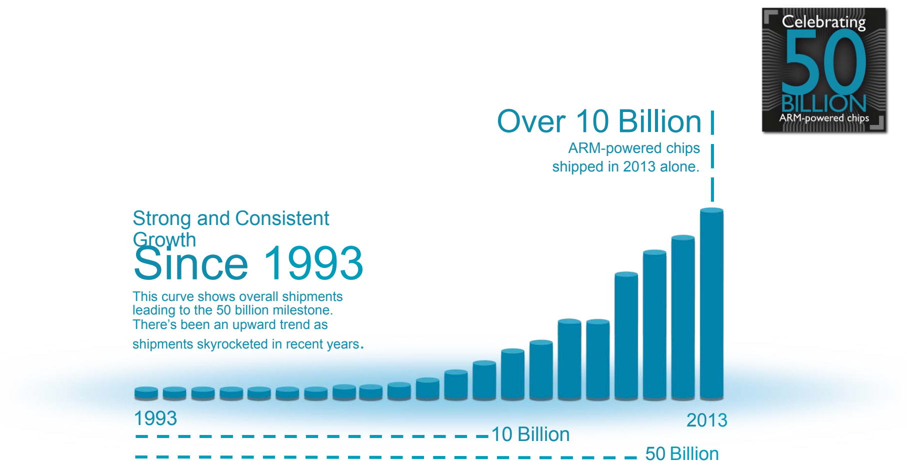
Cortex-M processors are the optimal solution for **low-power embedded computing** applications. The 32-bit Cortex-M processor family is the key to transforming all sorts of embedded systems into smart and connected systems. Often provided as a “black box” with pre-loaded applications, they have limited capability to expand hardware functionality and in most cases no screen.

•Merchant MCUs *Automotive Control Systems *Motor Control Systems

•White Goods controllers *Smart Meters *Sensors
•White Goods controllers *Smart Meters *Sensors *Internet of Things

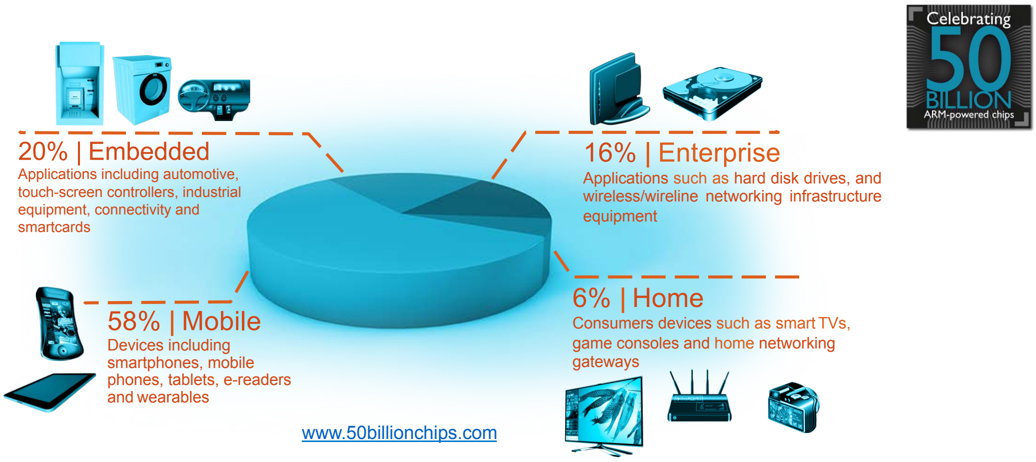
Equipment Adopting ARM Cores

With More Than 50 Billion



www.50billionchips.com

Markets we're POWERING



Architecture	Core bit width	Cores designed by ARM Holdings	Cores designed by third parties	Profile	References
ARMv1	32[a 1]	ARM1			
ARMv2	32[a 1]	ARM2, ARM250, ARM3	Amber, STORM Open Soft Core[37]		
ARMv3	32[a 2]	ARM6, ARM7			
ARMv4	32[a 2]	ARM8	StrongARM, FA526		
ARMv4T	32[a 2]	ARM7TDMI, ARM9TDMI, SecurCore SC100			
ARMv5TE	32	ARM7EJ, ARM9E, ARM10E	XScale, FA626TE, Feroceon, PJ1/Mohawk		
ARMv6	32	ARM11			
ARMv6-M	32	ARM Cortex-M0, ARM Cortex-M0+, ARM Cortex-M1, SecurCore SC000		Microcontroller	
ARMv7-M	32	ARM Cortex-M3, SecurCore SC300		Microcontroller	
ARMv7E-M	32	ARM Cortex-M4, ARM Cortex-M7		Microcontroller	
ARMv7-R	32	ARM Cortex-R4, ARM Cortex-R5, ARM Cortex-R7		Real-time	
ARMv7-A	32	ARM Cortex-A5, ARM Cortex-A7, ARM Cortex-A8, ARM Cortex-A9, ARM Cortex-A12, ARM Cortex-A15, ARM Cortex-A17	Krait, Scorpion, PJ4/Sheeva, Apple A6/A6X	Application	
ARMv8-A	64	ARM Cortex-A35,[38] ARM Cortex-A53, ARM Cortex-A57,[39] ARM Cortex-A72[40]	X-Gene, Nvidia Project Denver, AMD K12, Apple A7/A8/A8X/A9/A9X, Cavium Thunder X,[41][42][43] Qualcomm Kryo	Application	[44][45]
ARMv8.1-A	64	TBA		Application	
ARMv8-R	32	TBA		Real-time	[46][47]
ARMv8-M	32	TBA		Microcontroller	[48]

A First Look at the ARM Processor: Main Features

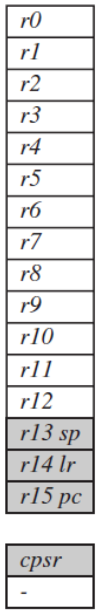
- Load-Store architecture
- Fixed-length (32-bit) instructions
- 3-operand instruction format (2 source operand reg's, 1 result operand reg.)
- Conditional execution of ALL instructions
- Load-Store multiple registers in one instruction
- A single-cycle *n*-bit shift with ALU operation
- Coprocessor instruction interfacing
- Thumb architecture (dense 16-bit compressed instruction set)
- "combines the best of RISC with the best of CISC"

ARM Registers

- Data items are placed in the *register file*
- 16 32-bit registers
  - The sign extend hardware converts signed 8-bit and 16-bit numbers to 32-bit values.
- ARM instructions typically have two source registers, Rn and Rm, and a single result or destination register, Rd.
  - Source operands are read from the register file using the internal buses A and B, respectively.

ARM Registers

- They are identified with the letter *r* prefixed to the register number
  - For example, register 4 is given the label *r4.*
- Next Figure shows the active registers available in *user* mode
- The processor can operate in seven different modes, which we will introduce shortly.
- All the registers shown are 32 bits in size.
- There are up to 18 active registers: 16 data registers and 2 processor status registers.
  - The data registers are visible to the programmer as *r0* to *r15.*



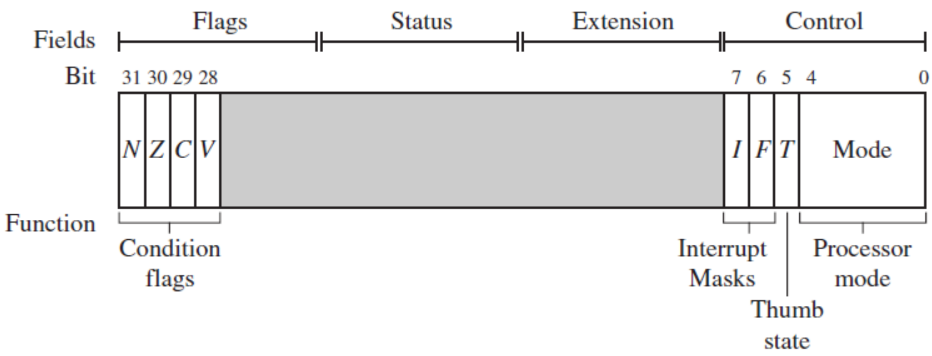
ARM Registers

- The ARM processor has three registers assigned to a particular task or special function: *r13*, *r14*, and *r15*.
  - Given different labels.
- Register *r13* is traditionally used as the stack pointer (***sp***) and stores the head of the stack in the current processor mode.
- Register *r14* is called the link register (***lr***) and is where the core puts the return address whenever it calls a subroutine.
- Register *r15* is the program counter (**pc**)

Current Program Status Register

- The ARM core uses the *cpsr* to monitor and control internal operations.
- The *cpsr* is a dedicated 32-bit register and resides in the register file
- The *cpsr* is divided into four fields, each 8 bits wide: flags, status, extension, and control.
  - The control field contains the processor mode, state, and interrupt mask bits
  - The flags field contains the condition flags

CPSR Register



CPSR Register

- N: Negative; the last ALU operation which changed the flags produced a negative result (the top bit of the 32-bit result was a one).
- Z: Zero; the last ALU operation which changed the flags produced a zero result (every bit of the 32-bit result was zero).
- C: Carry; the last ALU operation which changed the flags generated a carry-out, either as a result of an arithmetic operation in the ALU or from the shifter.
- V: oVerflow; the last arithmetic ALU operation which changed the flags generated an overflow into the sign bit.