Advanced Micro-controllers - ARM

DESD @ Sunbeam Infotech

Development of the ARM Architecture

ARM modes

① SVC
② IRQ
③ FIQ
④ Undef
⑤ Abt
⑥ User

v1

v2

v3

Early ARM architectures

Apple Newton PDA

v4

Halfword and signed halfword / byte support

⑦ System mode

(Intel) Strong ARM

SA-110

SA-1110

v4T

Thumb instruction set

LPC2148

ARM7TDMI | **ARM9TDMI**

ARM720T | **ARM940T**

+ MMU

v5TE

Enhanced DSP

Improved ARM/Thumb Interworking

✓ CLZ
└ Count leading zero.

✓ Saturated maths - USAT, SSAT

✓ DSP multiply-accumulate instructions MLA $(w = x \times y + z)$

ARM1020E

XScale

Motorola → FreeScale

ARM9E-S

ARM966E-S

v5TEJ

Jazelle

Java bytecode execution (DBX)

ARM9EJ-S | **ARM926EJ-S**

ARM7EJ-S | **ARM1026EJ-S**

v6

Single Instru Multiple Data

SIMD Instructions
e.g. QADD8, ...

Multi-processing → Multi-core

V6 Memory architecture (VMSA) → Virtual memory System Architecture

Unaligned data support

ARM1136EJ-S

Development of the ARM Architecture

v4	v5	v6	v7
✓ Halfword and signed halfword / byte support ✓ System mode ✓ Thumb instruction set (v4T)	✓ Improved interworking ✓ CLZ ✓ Saturated arithmetic DSP MAC instructions Extensions: ✓ Jazelle (5TEJ)	✓ SIMD Instructions ✓ Multi-processing ✓ v6 Memory architecture ✓ Unaligned data support Extensions: Thumb-2 (6T2) TrustZone® (6Z) ✓ Multicore (6K) ✓ Thumb only (6-M)	Thumb-2 Architecture Profiles 7-A - Applications 7-R - Real-time 7-M - Microcontroller

Appln ↑ OS (Linux/ Win).

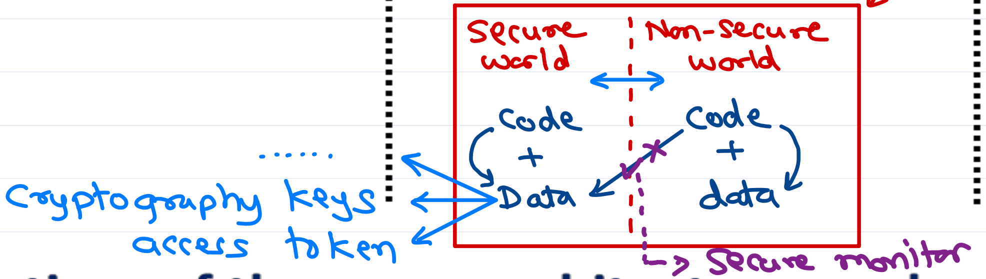
7-A → OS

7-R → RTOS → Tasks

7-M → Bare Metal ↓ Firmware

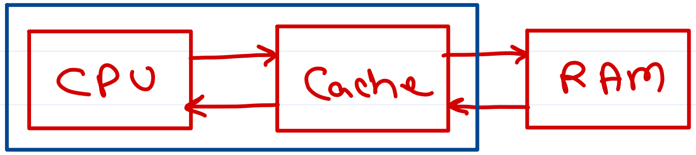
Neon → SIMD engine

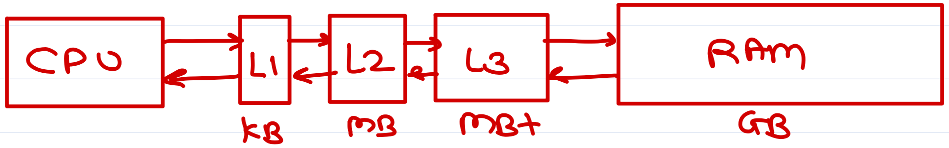
Thumb2



ARM v8 : 64-bit ARM.

- **Note that implementations of the same architecture can be different**
  - Cortex-A8 - architecture v7-A, with a 13-stage pipeline
  - Cortex-A9 - architecture v7-A, with an 8-stage pipeline

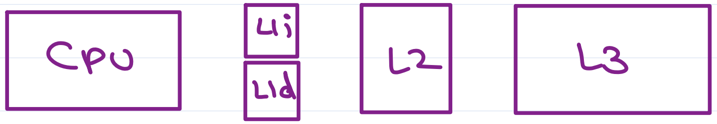




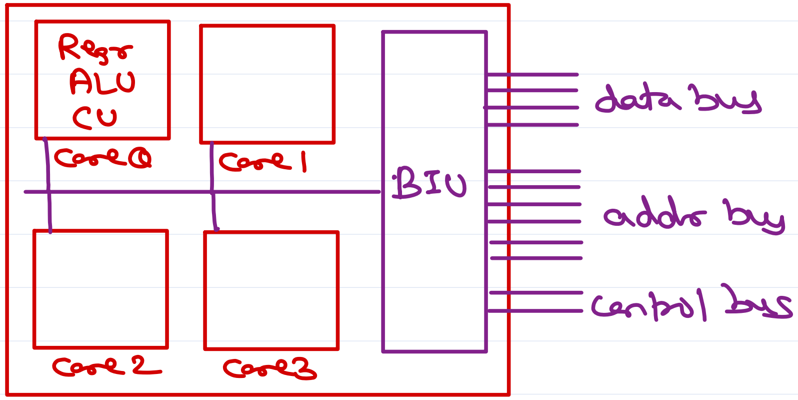
Instruction cache : stores code/instrns.

Data Cache : stores data only.

Unified Cache: stores code + Data.



terminal> lscpu

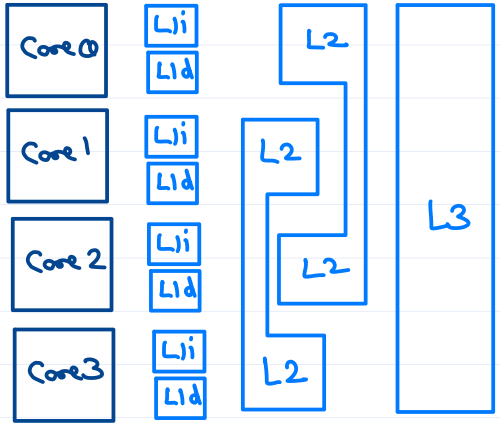


Logical Cores
↑
Core = Hardware Threads (SMT)
↓
(Intel) Hyper Threading

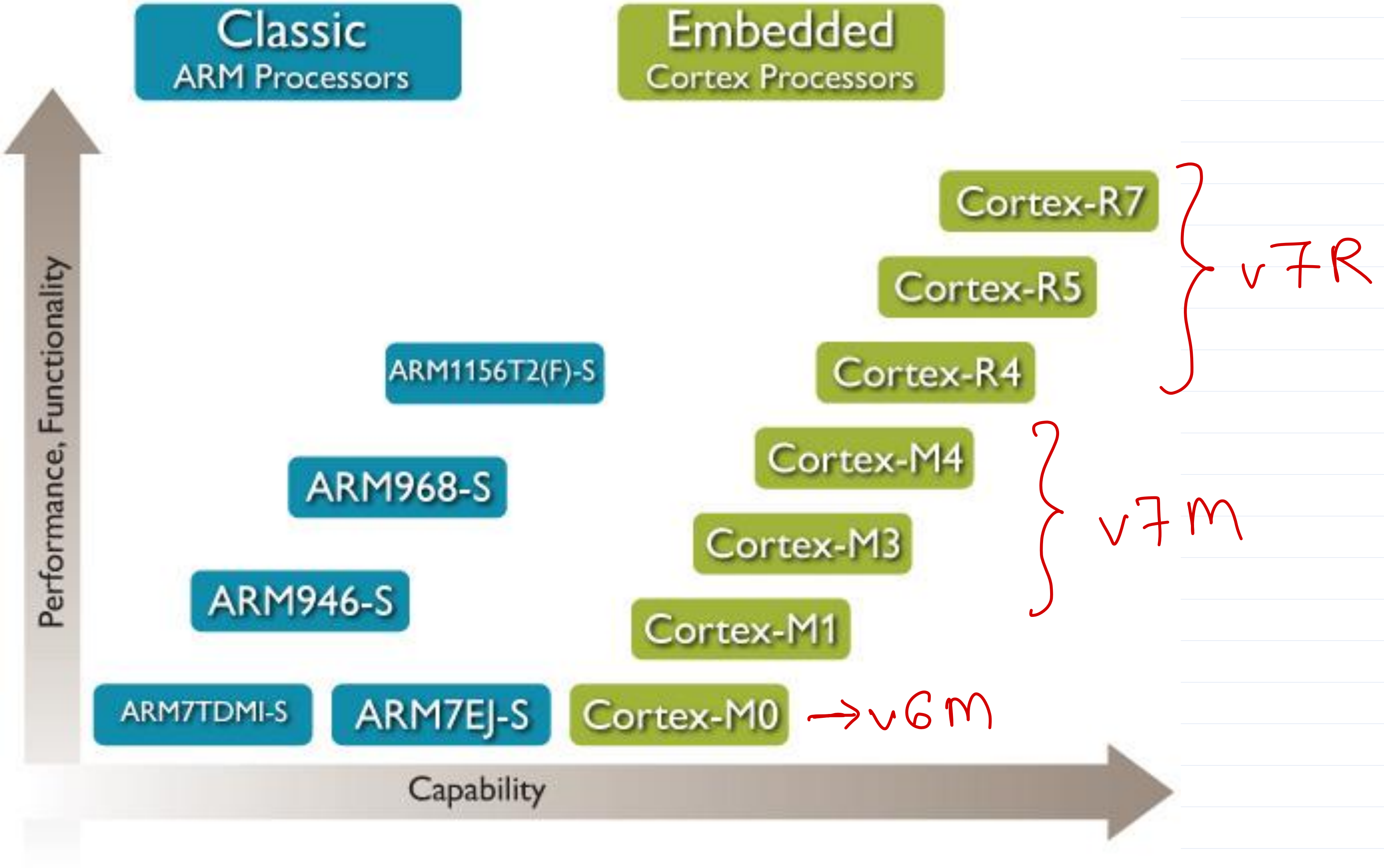
lscpu

num of sockets : 1 (?)
Cores per socket : 4 (?) - physical cores
threads per core : 2 (?) - logical cores (SMT)
CPUs : 8 = sockets x cores x threads

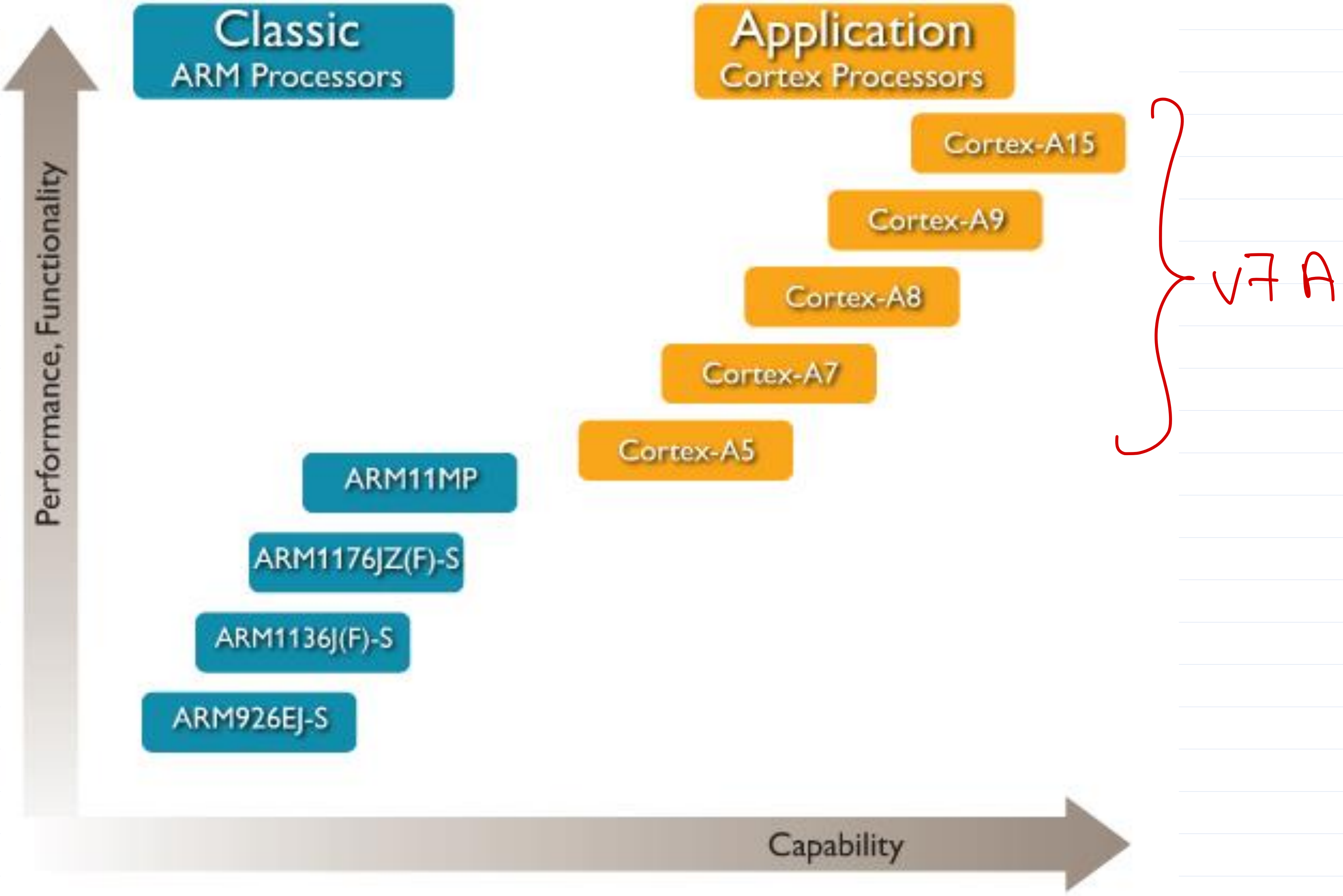
lscpu -p



Embedded Processors



Application Processors



Agenda

- Introduction
- ARM Architecture Overview
- ARMv7-AR Architecture
  - Programmer’s Model
  - Memory Systems
- ARMv7-M Architecture
  - Programmer’s Model
  - Memory Systems
  - Floating Point Extensions
- ARM System Design
- Software Development Tools

Architecture ARMv7 profiles

- Application profile (ARMv7-A)
  - Memory management support (MMU)
  - Highest performance at low power
    - Influenced by multi-tasking OS system requirements
  - TrustZone and Jazelle-RCT for a safe, extensible system
  - e.g. Cortex-A5, Cortex-A9

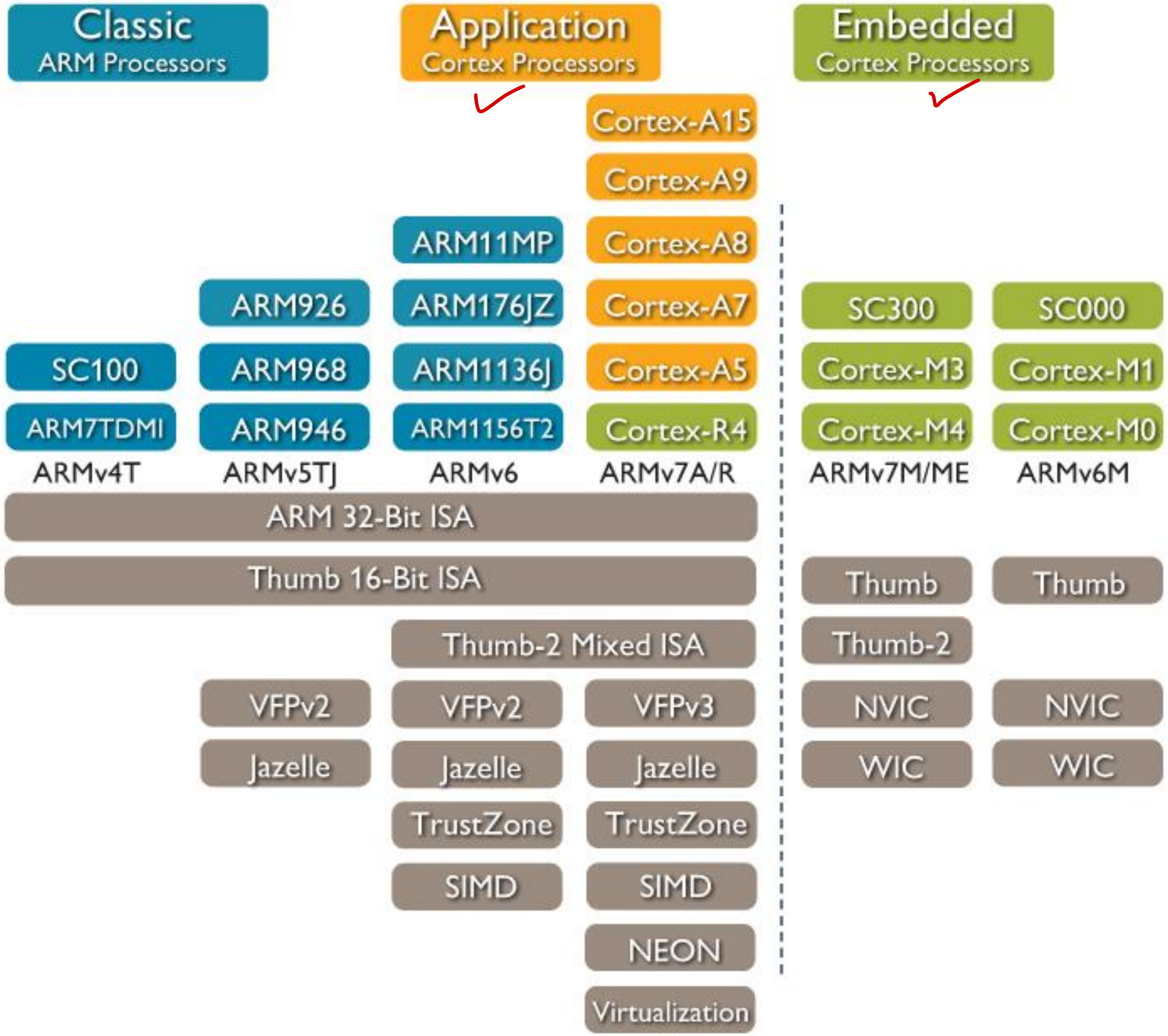
Jazelle → DBX (Direct Bytecode eXecution)
Jazelle → RCT (Runtime Compilation Target)
- Support for JIT (Just In Time) Compiler
- Optimize Java Bytecode interpretation

- Real-time profile (ARMv7-R)
  - Protected memory (MPU)
  - Low latency and predictability 'real-time' needs
  - Evolutionary path for traditional embedded business
  - e.g. Cortex-R4

Interrupt Latency
- Time from arrival of intr till beginning execution of ISR.

- Microcontroller profile (ARMv7-M, ARMv7E-M, ARMv6-M)
  - Lowest gate count entry point
  - Deterministic and predictable behavior a key priority
  - Deeply embedded use (bare-metal programming)
  - e.g. Cortex-M3

Which architecture is my processor?



Agenda

Introduction

ARM Architecture Overview

- ARMv7-AR Architecture
    - Programmer’s Model
    - Memory Systems

ARMv7-M Architecture

- Programmer’s Model
- Memory Systems
- Floating Point Extensions

ARM System Design

Software Development Tools

Architecture ARMv7-AR profiles

- Application profile (ARMv7-A)
  - Memory management support (MMU)
  - Highest performance at low power
  - Influenced by multi-tasking OS system requirements
  - e.g. Cortex-A5, Cortex-A8, Cortex-A9, Cortex-A15

- Real-time profile (ARMv7-R)
  - Protected memory (MPU)
  - Low latency and predictability ‘real-time’ needs
  - Evolutionary path for traditional embedded business
  - e.g. Cortex-R4, Cortex-R5

Cortex-A8

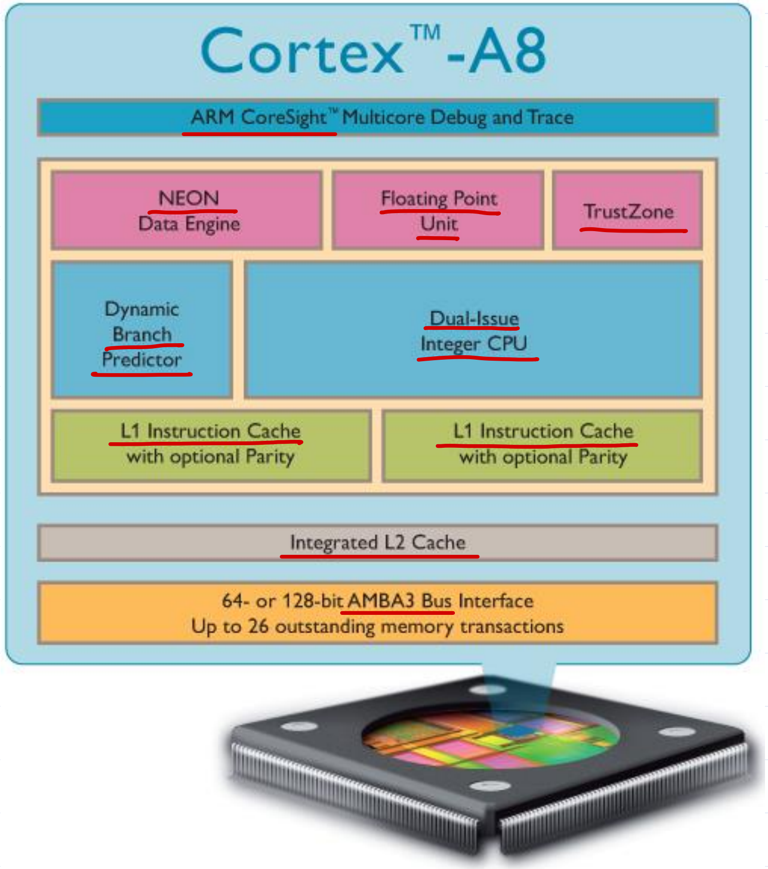
- ARMv7-A Architecture
  - Thumb-2 → Execution Environment
  - Thumb-2EE (Jazelle-RCT)
  - TrustZone extensions
- Custom or synthesized design
- MMU

  → Advance Microcontroller Bus Architecture

  → AMBA Bus → Advanced eXtended Interface
- 64-bit or 128-bit AXI Interface
- L1 caches
  - 16 or 32KB each
- Unified L2 cache
  - 0-2MB in size
  - 8-way set-associative

- **Dual-issue, super-scalar 13-stage pipeline**
  - Branch Prediction & Return Stack
  - NEON and VFP implemented at end of pipeline

    SIMD Floating point



Cortex™-A8

ARM CoreSight™ Multicore Debug and Trace

NEON Data Engine

Floating Point Unit

TrustZone

Dynamic Branch Predictor

Dual-Issue Integer CPU

L1 Instruction Cache with optional Parity

L1 Instruction Cache with optional Parity

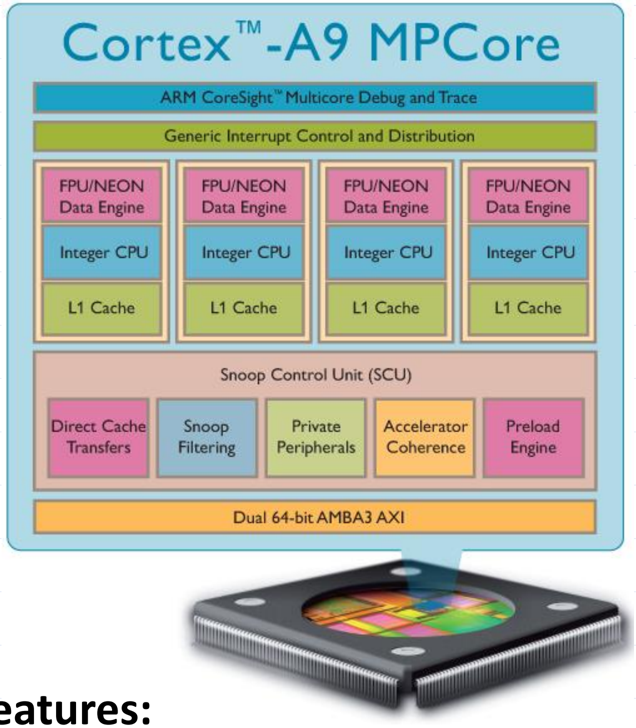
Integrated L2 Cache

64- or 128-bit AMBA3 Bus Interface
Up to 26 outstanding memory transactions

- **Optional features**
  - VFPv3 Vector Floating-Point
  - NEON media processing engine

Cortex-A9

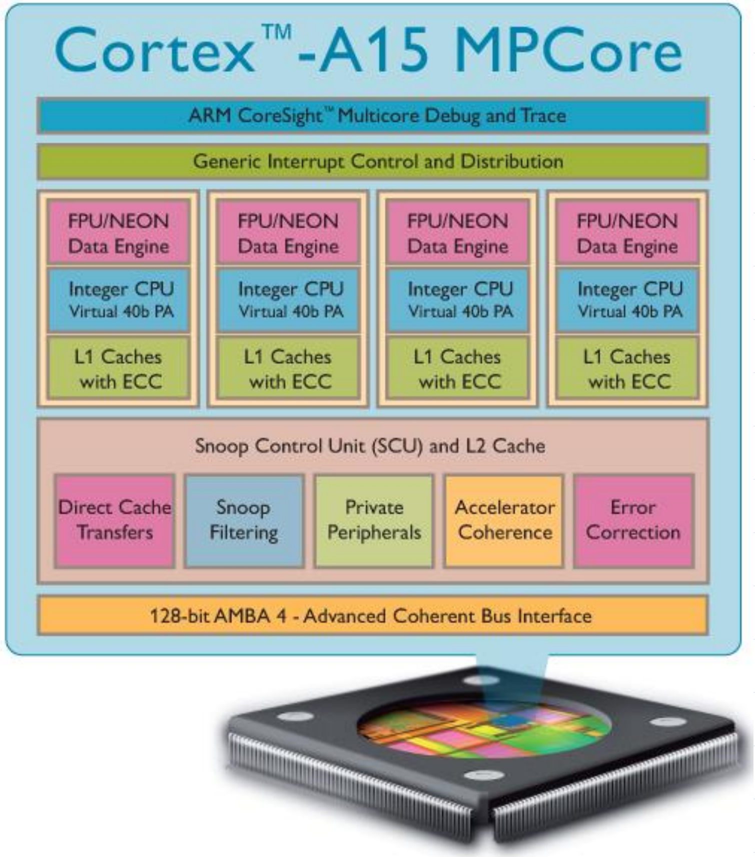
- ARMv7-A Architecture
  - Thumb-2, Thumb-2EE
  - TrustZone support
- Variable-length Multi-issue pipeline
  - Register renaming
  - Speculative data prefetching
  - Branch Prediction & Return Stack
- 64-bit AXI instruction and data interfaces
- TrustZone extensions
- L1 Data and Instruction caches
  - 16-64KB each
  - 4-way set-associative



- **Optional features:**
  - PTM instruction trace interface
  - IEM power saving support
  - Full Jazelle DBX support
  - VFPv3-D16 Floating-Point Unit (FPU) or NEON™ media processing engine

Cortex-A15 MPCore

- 1-4 processors per cluster
- Fixed size L1 caches (32KB)
- Integrated L2 Cache
  - 512KB – 4MB
- System-wide coherency support with AMBA 4 ACE
- Backward-compatible with AXI3 interconnect
- Integrated Interrupt Controller
  - 0-224 external interrupts for entire cluster
- CoreSight debug
- Advanced Power Management



Cortex™-A15 MPCore
ARM CoreSight™ Multicore Debug and Trace
Generic Interrupt Control and Distribution
FPU/NEON Data Engine
Integer CPU Virtual 40b PA
L1 Caches with ECC
FPU/NEON Data Engine
Integer CPU Virtual 40b PA
L1 Caches with ECC
FPU/NEON Data Engine
Integer CPU Virtual 40b PA
L1 Caches with ECC
FPU/NEON Data Engine
Integer CPU Virtual 40b PA
L1 Caches with ECC
Snoop Control Unit (SCU) and L2 Cache
Direct Cache Transfers
Snoop Filtering
Private Peripherals
Accelerator Coherence
Error Correction
128-bit AMBA 4 - Advanced Coherent Bus Interface

- **Large Physical Address Extensions (LPAE) to ARMv7-A Architecture**
- **Virtualization Extensions to ARMv7-A Architecture**

Data Sizes and Instruction Sets

- ARM is a 32-bit load / store RISC architecture
  - The only memory accesses allowed are loads and stores
  - Most internal registers are 32 bits wide
  - Most instructions execute in a single cycle

- When used in relation to ARM cores
  - ✓ **Halfword** means 16 bits (two bytes)
  - ✓ **Word** means 32 bits (four bytes)
  - ✓ **Doubleword** means 64 bits (eight bytes)

- ARM cores implement two basic instruction sets
  - **ARM** instruction set – instructions are all 32 bits long
  - **Thumb** instruction set – instructions are a mix of 16 and 32 bits
    - Thumb-2 technology added many extra 32- and 16-bit instructions to the original 16-bit Thumb instruction set

- Depending on the core, may also implement other instruction sets
  - ✓ **VFP** instruction set – 32 bit (vector) floating point instructions
  - ✓ **NEON** instruction set – 32 bit SIMD instructions
  - ✓ **Jazelle-DBX** - provides acceleration for Java VMs (with additional software support)
  - ✓ **Jazelle-RCT** - provides support for interpreted languages

Processor Modes

Instruction
= OP code + operands

10111	01101	01100	1000

↓
Decoder → Control

- ARM has seven basic operating modes
  - Each mode has access to its own stack space and a different subset of registers
  - Some operations can only be carried out in a privileged mode

Software Interrupt is used for OS system calls.

exception → exception mode

① reset
② software interrupt
③ fiq interrupt
④ irq interrupt
⑤ prefetch abort
⑥ data abort
⑦ undef

Exception modes

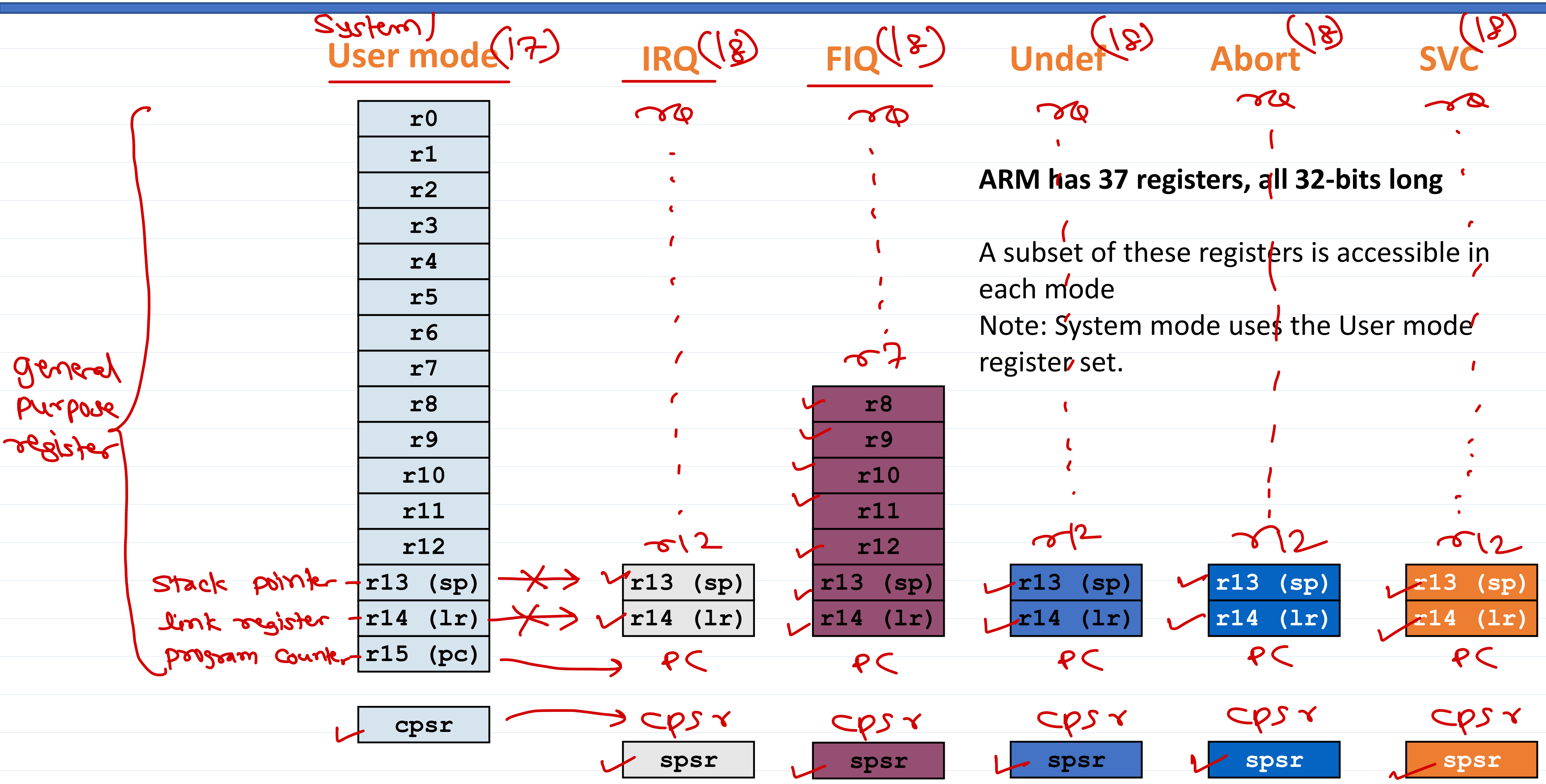
Mode	Description	
Supervisor (SVC)	Entered on reset and when a Supervisor call (earlier SWI) instruction (SVC) is executed	Privileged modes
FIQ	Entered when a high priority (fast) interrupt is raised	
IRQ	Entered when a normal priority interrupt is raised	
Abort	Used to handle memory access violations	
Undef	Used to handle undefined instructions	
System	Privileged mode using the same registers as User mode	
User	Mode under which most Applications / OS tasks run	Unprivileged mode

SysCalls are fns exposed by the kernel so that user programs can access kernel functionality.

Privileged modes → Data processing, Control/Jump, Status regr manip, ...

Unprivileged mode → Data processing, Control/Jump, ..

The ARM Register Set



ARM has 37 registers, all 32-bits long

A subset of these registers is accessible in each mode

Note: System mode uses the User mode register set.

Current mode

Banked out registers

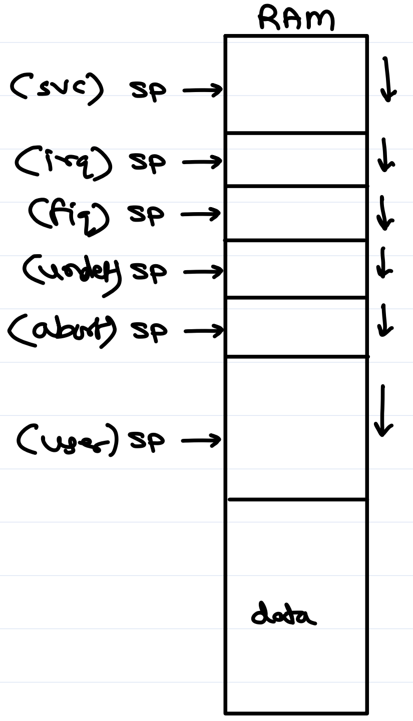
The Registers

- ARM has 37 registers all of which are 32-bits long.
  - 1 dedicated program counter
  - 1 dedicated current program status register (cpsr)
  - 5 dedicated saved program status registers (spsr)
  - 30 general purpose registers

- The current processor mode governs which of several banks is accessible. Each mode can access
  - a particular set of r0-r12 registers
  - a particular r13 (the stack pointer, sp) and r14 (the link register, lr)
  - the program counter, r15 (pc)
  - the current program status register, cpsr

Privileged modes (except System) can also access

- a particular spsr (saved program status register)



Program Counter (r15)

- When the processor is executing in ARM state:
  - All instructions are 32 bits wide
  - All instructions must be word aligned (address multiple of 4)
  - Therefore the **pc** value is stored in bits [31:2] with bits [1:0] undefined (as instruction cannot be halfword or byte aligned).

    ↓ ↓
    0 0

- When the processor is executing in Thumb state:
  - All instructions are 16 bits wide
  - All instructions must be halfword aligned
  - Therefore the **pc** value is stored in bits [31:1] with bit [0] undefined (as instruction cannot be byte aligned).

    ↓
    0

- When the processor is executing in Jazelle state:
  - All instructions are 8 bits wide
  - Processor performs a word access to read 4 instructions at once

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>