The ARM Architecture

With a focus on v7A and Cortex-A8

Agenda

- Introduction to ARM Ltd

  ARM Processors Overview

  ARM v7A Architecture/Programmers Model

  Cortex-A8 Memory Management

  Cortex-A8 Pipeline

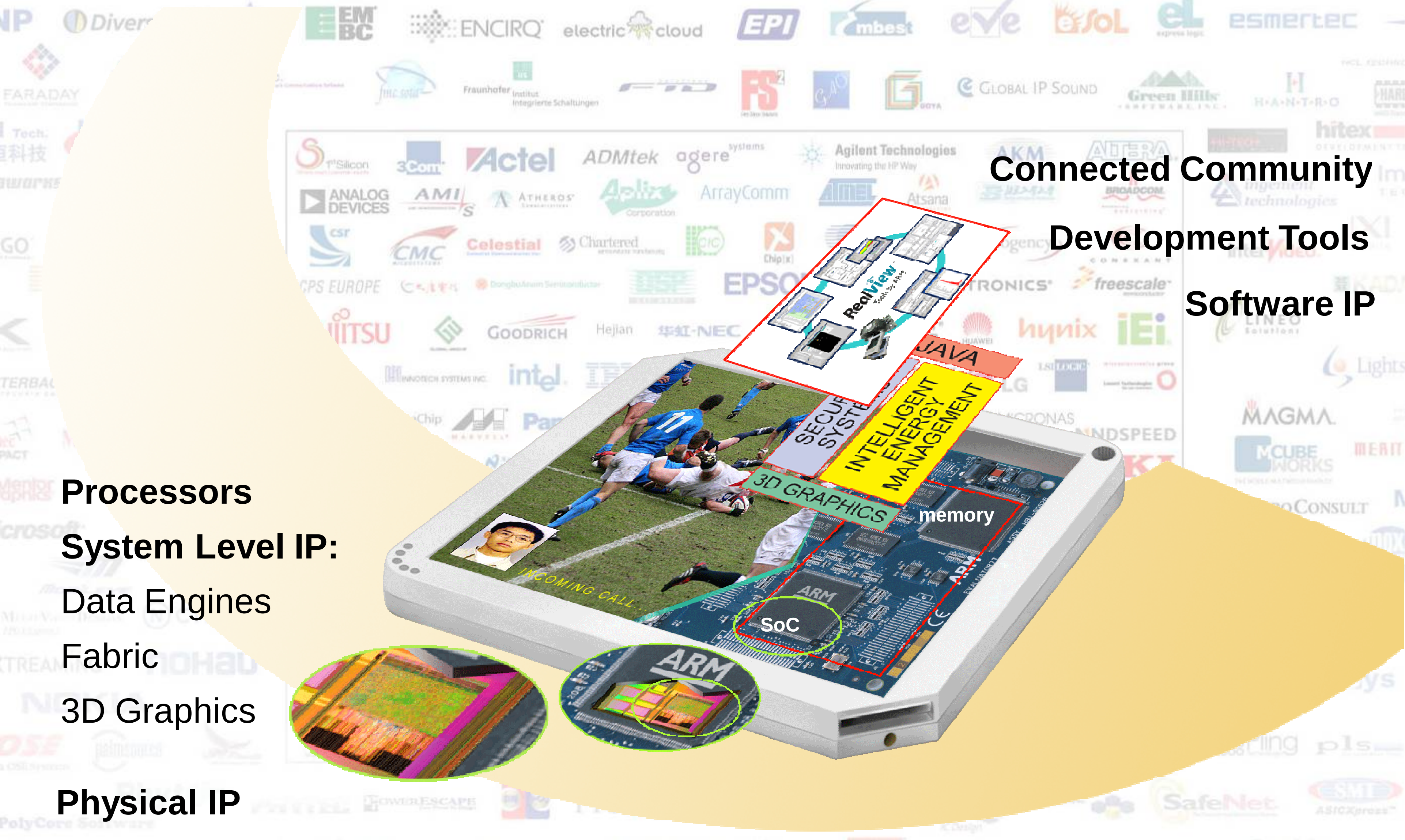
ARM Ltd

- Founded in November 1990
  - Spun out of Acorn Computers
  - Initial funding from Apple, Acorn and VLSI

- Designs the ARM range of RISC processor cores
  - Licenses ARM core designs to semiconductor partners who fabricate and sell to their customers
  - ARM does not fabricate silicon itself

- Also develop technologies to assist with the design-in of the ARM architecture
  - Software tools, boards, debug hardware
  - Application software
  - Bus architectures
  - Peripherals, etc

ARM’s Activities



Connected Community

Development Tools

Software IP

Processors

System Level IP:

Data Engines

Fabric

3D Graphics

Physical IP

Huge Range of Applications

Equipment Adopting 32-bit ARM Microcontrollers

Agenda

Introduction to ARM Ltd

- ARM Processors Overview

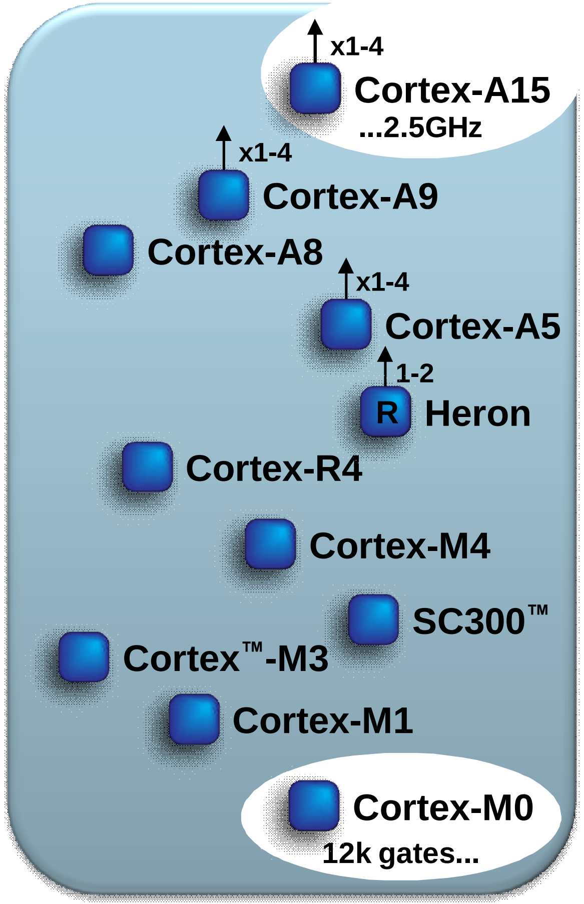
ARM v7A Architecture/Programmers Model

Cortex-A8 Memory Management

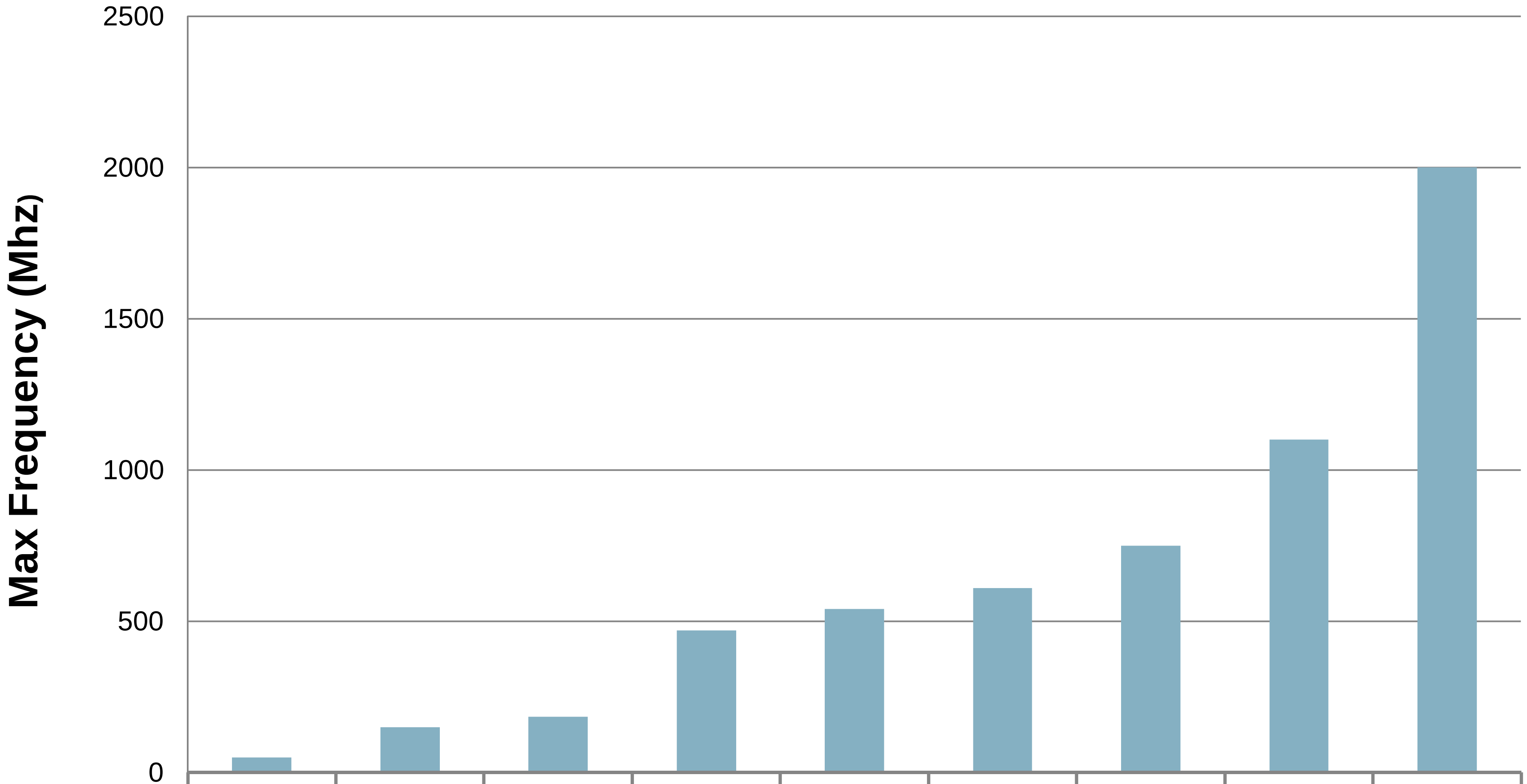
Cortex-A8 Pipeline

ARM Cortex Processors (v7)

- ARM Cortex-**A** family (v7-A):
  - Applications processors for full OS and 3rd party applications
- ARM Cortex-**R** family (v7-R):
  - Embedded processors for real-time signal processing, control applications
- ARM Cortex-**M** family (v7-M):
  - Microcontroller-oriented processors for MCU and SoC applications



Relative Performance*



	Cortex-M0	Cortex-M3	ARM7	ARM926	ARM1026	ARM1136	ARM1176	Cortex-A8	Cortex-A9 Dual-core
Max Freq (MHz)	50	150	184	470	540	610	750	1100	2000
Min Power (mW/MHz)	0.012	0.06	0.35	0.235	0.36	0.335	0.568	0.43	0.5

*Represents attainable speeds in 130, 90, 65, or 45nm processes

Cortex family

Cortex-A8

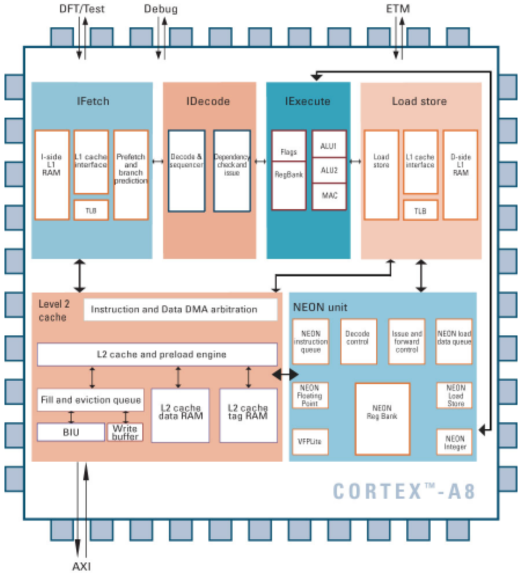
- Architecture v7A
- MMU
- AXI
- VFP & NEON support

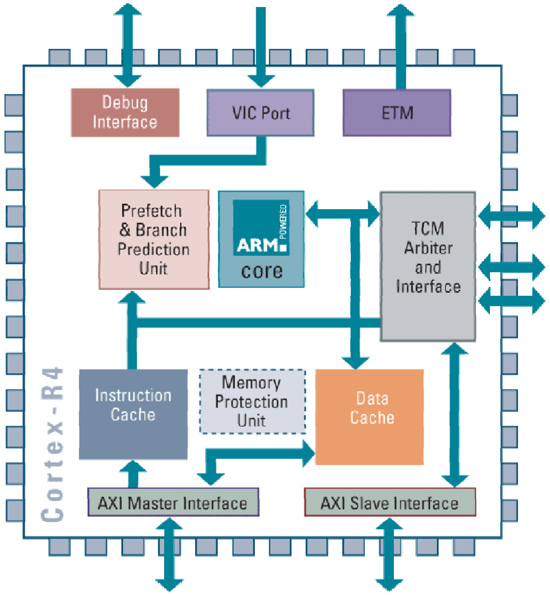
Cortex-R4

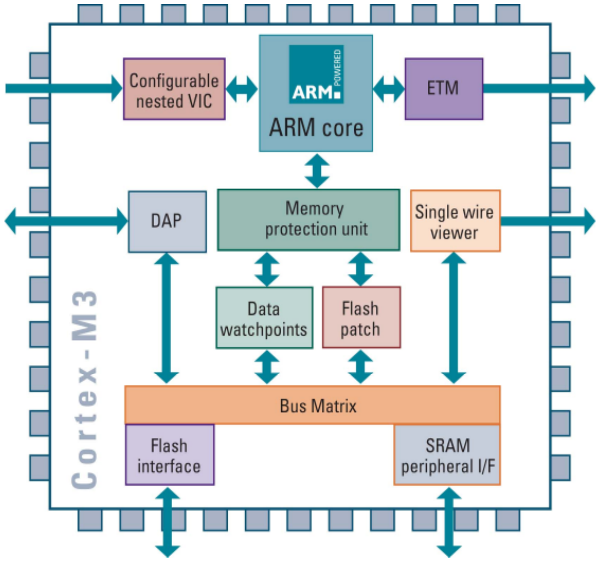
- Architecture v7R
- MPU (optional)
- AXI
- Dual Issue

Cortex-M3

- Architecture v7M
- MPU (optional)
- AHB Lite & APB







Agenda

Introduction to ARM Ltd

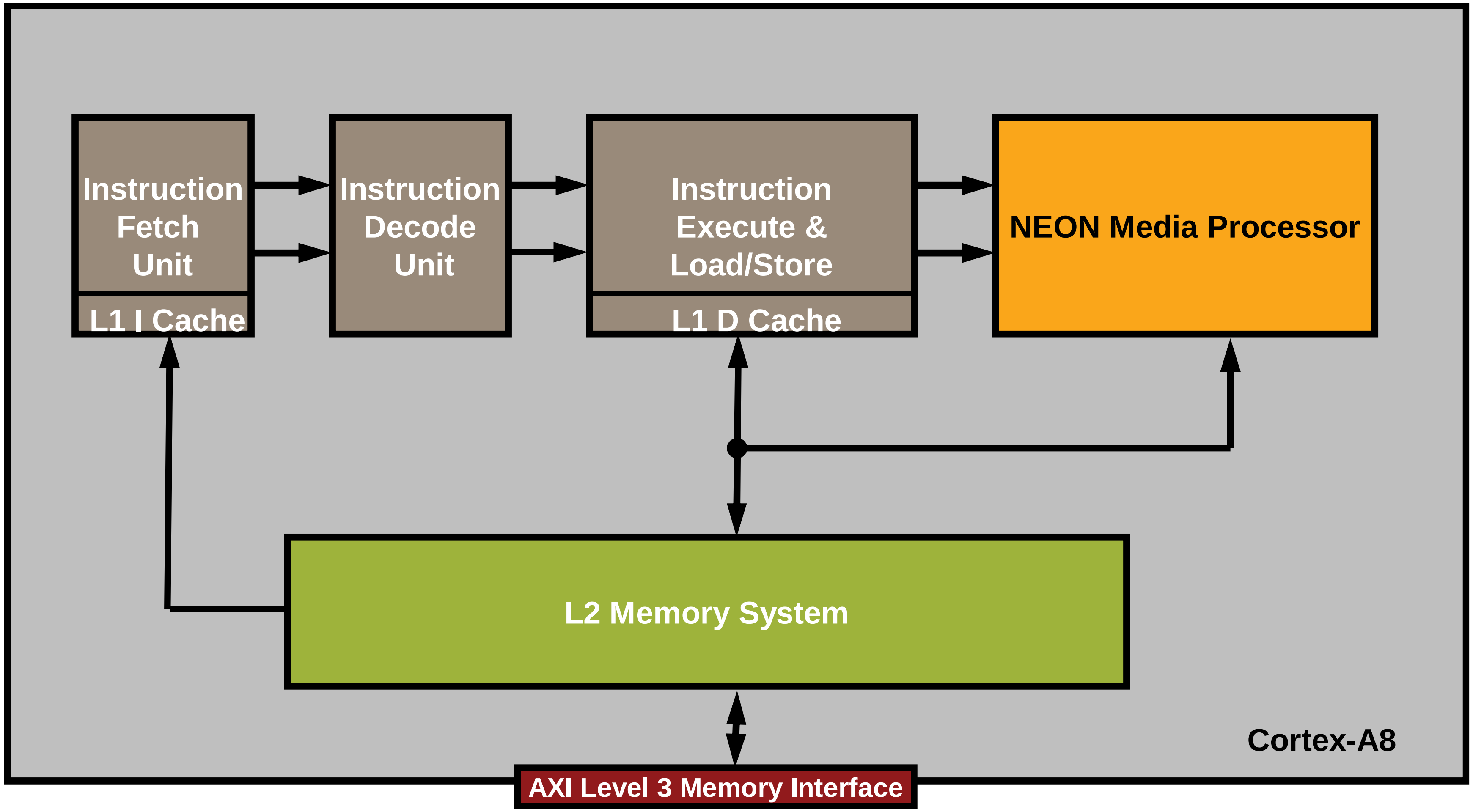
ARM Processors Overview

- ARM v7A Architecture/Programmers Model

Cortex-A8 Memory Management

Cortex-A8 Pipeline

Cortex-A8 Block Diagram



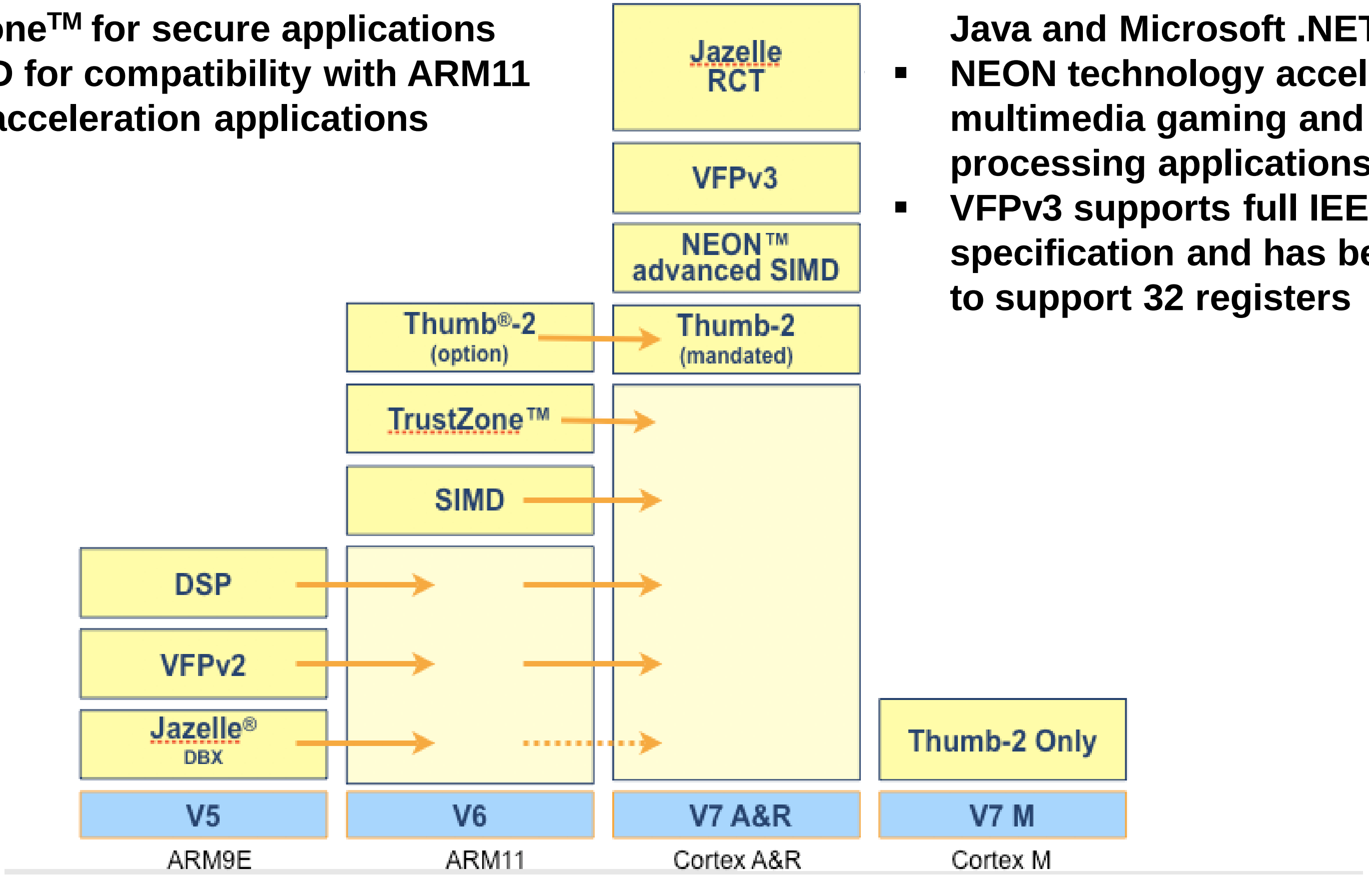
ARM Cortex-A Architecture

Cortex A Base Architecture

- Thumb-2 technology for power efficient execution
- TrustZone™ for secure applications
- v6 SIMD for compatibility with ARM11
- media acceleration applications

Cortex-A8 Extensions

- Jazelle-RCT for efficient acceleration of execution environments such as Java and Microsoft .NET
- NEON technology accelerating multimedia gaming and signal processing applications
- VFPv3 supports full IEEE 754 specification and has been expanded to support 32 registers



Data Sizes and Instruction Sets

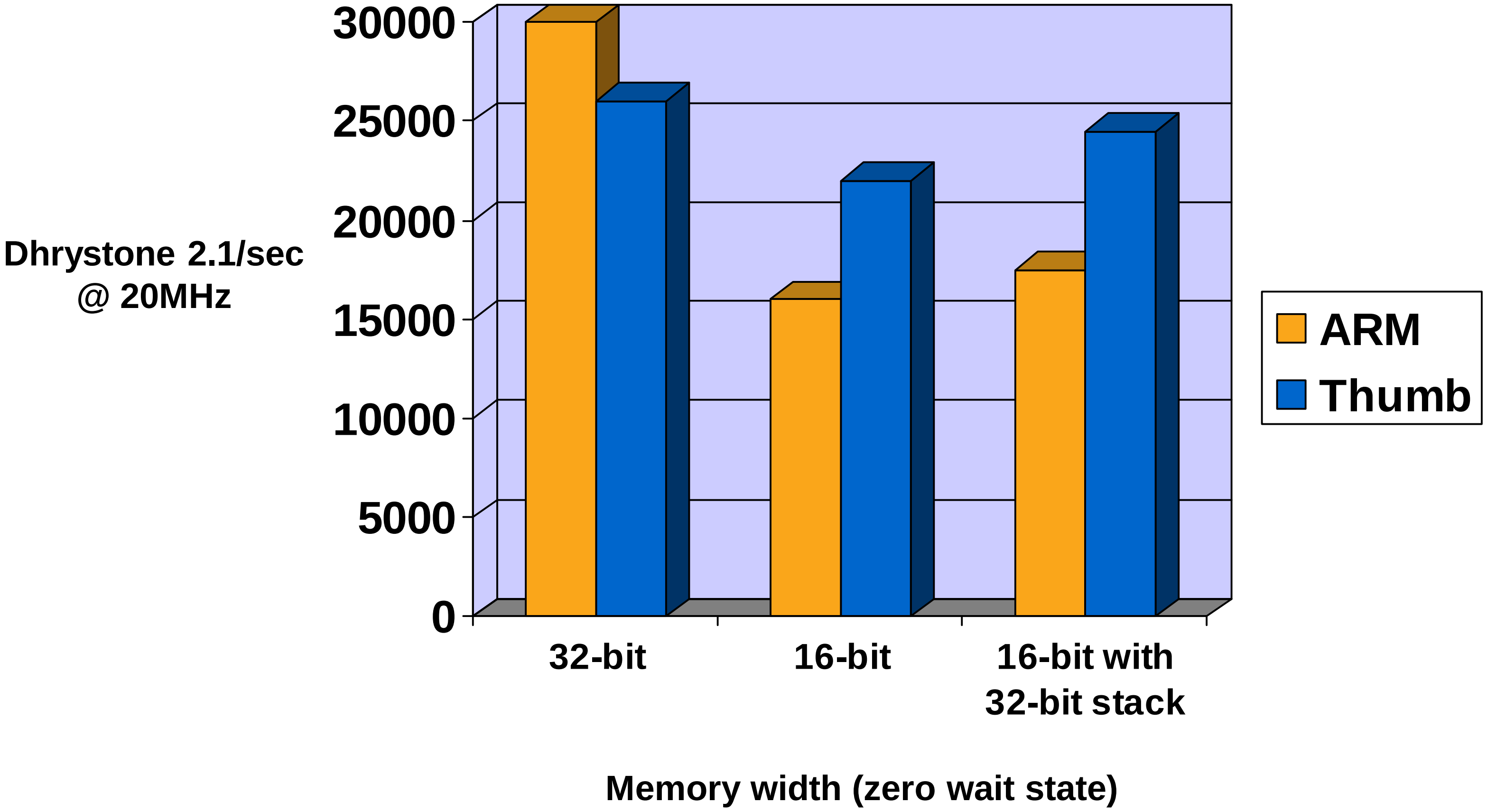
- The ARM is a 32-bit architecture.

- When used in relation to the ARM:
  - **Byte** means 8 bits
  - **Halfword** means 16 bits (two bytes)
  - **Word** means 32 bits (four bytes)

- Most ARM's implement two instruction sets
  - 32-bit ARM Instruction Set
  - 16-bit Thumb Instruction Set

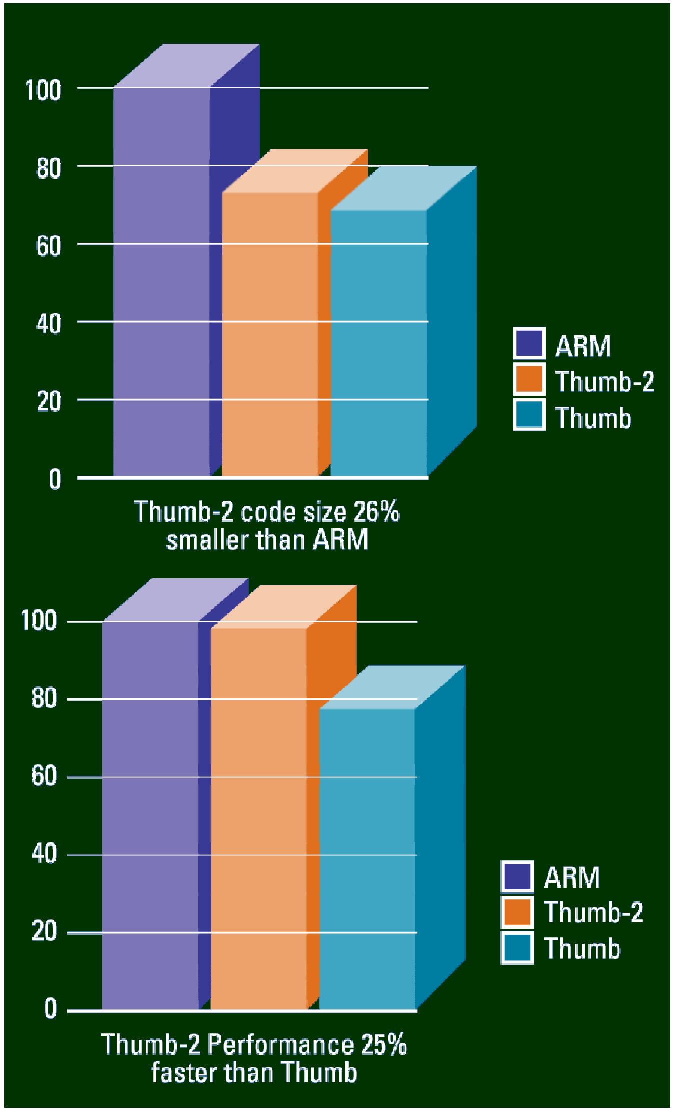
- Jazelle cores can also execute Java bytecode

ARM and Thumb Performance



The Thumb-2 instruction set

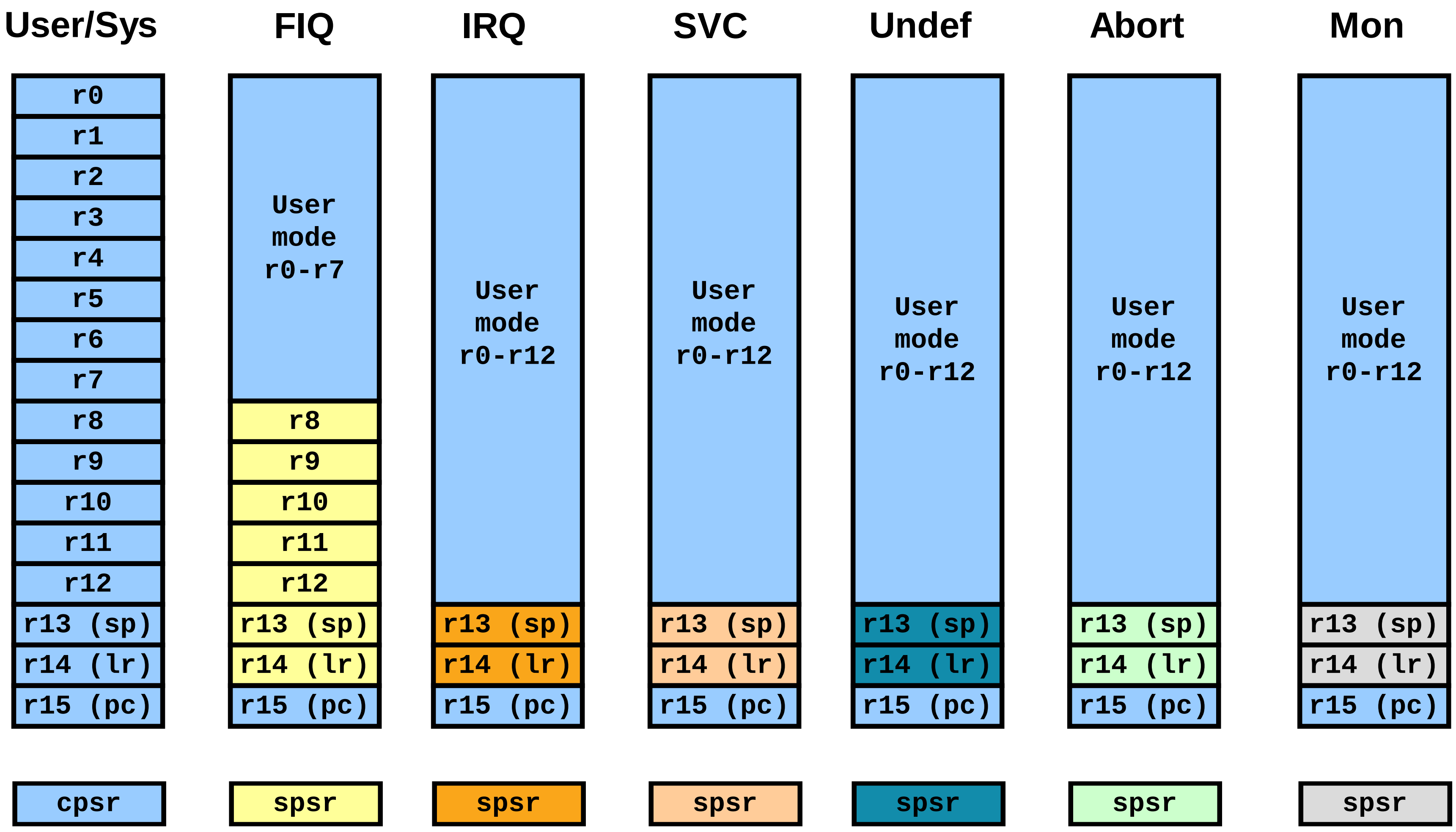
- Variable-length instructions
  - ARM instructions are a fixed length of 32 bits
  - Thumb instructions are a fixed length of 16 bits
  - Thumb-2 instructions can be either 16-bit or 32-bit
- Thumb-2 gives approximately 26% improvement in code density over ARM
- Thumb-2 gives approximately 25% improvement in performance over Thumb



Cortex-A8 Processor Modes

- User - used for executing most application programs
- FIQ - used for handling fast interrupts
- IRQ - used for general-purpose interrupt handling
- Supervisor - a protected mode for the Operating System
- Undefined - entered upon Undefined Instruction exceptions
- Abort - entered after Data or Pre-fetch Aborts
- System - privileged user mode for the Operating System
- Monitor - a secure mode for TrustZone

Cortex-A8 Register File



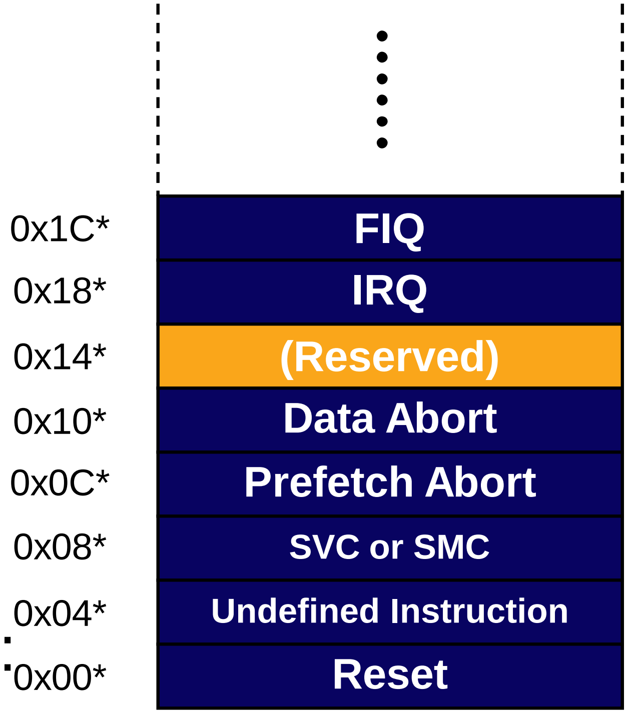
User/Sys	FIQ	IRQ	SVC	Undef	Abort	Mon
r0	User mode r0-r7	User mode r0-r12	User mode r0-r12	User mode r0-r12	User mode r0-r12	User mode r0-r12
r1						
r2						
r3						
r4						
r5						
r6						
r7						
r8	r8					
r9	r9					
r10	r10					
r11	r11					
r12	r12					
r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)
r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)
r15 (pc)	r15 (pc)	r15 (pc)	r15 (pc)	r15 (pc)	r15 (pc)	r15 (pc)
cpsr	spsr	spsr	spsr	spsr	spsr	spsr

Note: System mode uses the User mode register set

Cortex-A8 Exception Handling

- When an exception occurs, the ARM:
  - Copies CPSR into SPSR_<mode>
  - Sets appropriate CPSR bits
    - Change to ARM state
    - Change to exception mode
    - Disable interrupts (if appropriate)
  - Stores the return address in LR_<mode>
  - Sets PC to vector address
- To return, exception handler needs to:
  - Restore CPSR from SPSR_<mode>
  - Restore PC from LR_<mode>

This can only be done in ARM state.



0x1C*	FIQ
0x18*	IRQ
0x14*	(Reserved)
0x10*	Data Abort
0x0C*	Prefetch Abort
0x08*	SVC or SMC
0x04*	Undefined Instruction
0x00*	Reset

Vector Table

* Represents an offset, as vector table can moved to different base addresses

Cortex-A8 Program Status Register

31 27	26 25	24	23 20	19 16	15 10	9 5	4 0
Flags	IT1:0	J	Reserved	GE3:0	IT7:2	E A I F T	M4:0

- New IT field in Program Status Registers
  - Bits 7:5 indicate base condition
  - Bits 4:0 indicate the number of instructions and condition/inverse condition
  - Updated by
    - IT, BX, BLX, BXJ instructions
    - Loads to PC (except in User mode)

- New execution state (CPSR/SPSR)

J bit	T bit	State
0	0	ARM
0	1	Thumb
1	0	Jazelle-DBX
1	1	Thumb2-EE

  - EnterX / LeaveX instructions

Conditional Execution and Flags

- ARM instructions can be made to execute conditionally by postfixing them with the appropriate condition code field.
  - This improves code density *and* performance by reducing the number of forward branch instructions.

```
  CMP    r3,#0
  BEQ    skip
  ADD    r0,r1,r2
skip
```

```
CMP    r3,#0
ADDNE  r0,r1,r2
```

- By default, data processing instructions do not affect the condition code flags but the flags can be optionally set by using “S”. CMP does not need “S”.

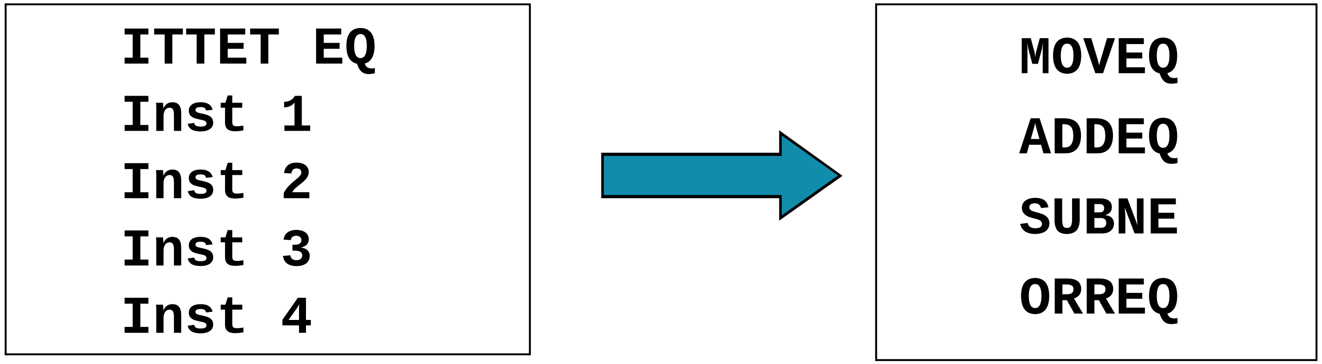
```
loop
  …
  SUBS r1,r1,#1
  BNE  loop
```

decrement r1 and set flags

if Z flag clear then branch

16-bit Conditional Execution

- If – Then (IT) instruction added (16 bit)
  - Up to 3 additional “then” or “else” conditions maybe specified (T or E)
  - Makes up to 4 following instructions conditional



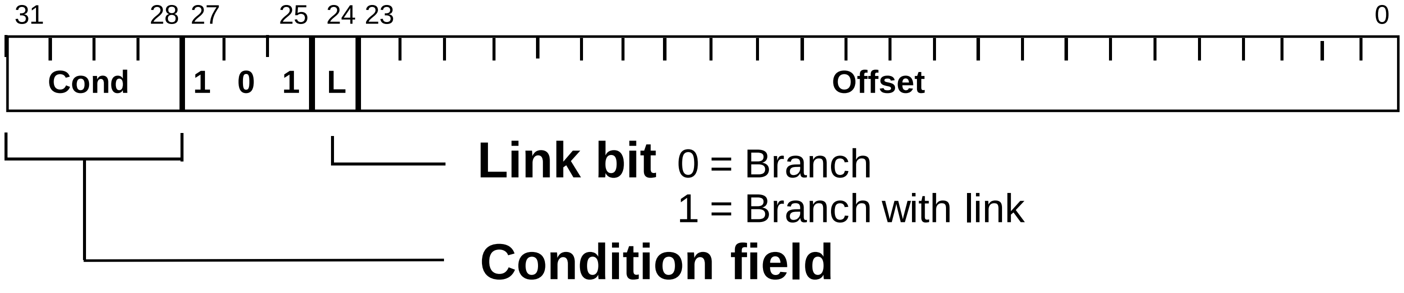
```
ITTET EQ
Inst 1
Inst 2
Inst 3
Inst 4
```

```
MOVEQ
ADDEQ
SUBNE
ORREQ
```

- Any normal ARM condition code can be used
- 16-bit instructions in block do not affect condition code flags
  - Apart from comparison instruction
  - 32 bit instructions may affect flags (normal rules apply)
- Current “if-then status” stored in CPSR
  - Conditional block maybe safely interrupted and returned to
  - Must NOT branch into or out of ‘if-then’ block

Branch instructions

- Branch : B{<cond>} label
- Branch with Link : BL{<cond>} subroutine_label



- The processor core shifts the offset field left by 2 positions, sign-extends it and adds it to the PC
  - ± 32 Mbyte range
  - How to perform longer branches?

Data processing Instructions

- Consist of :
  - Arithmetic: `ADD ADC SUB SBC RSB RSC`
  - Logical: `AND ORR EOR BIC`
  - Comparisons: `CMP CMN TST TEQ`
  - Data movement: `MOV MVN`

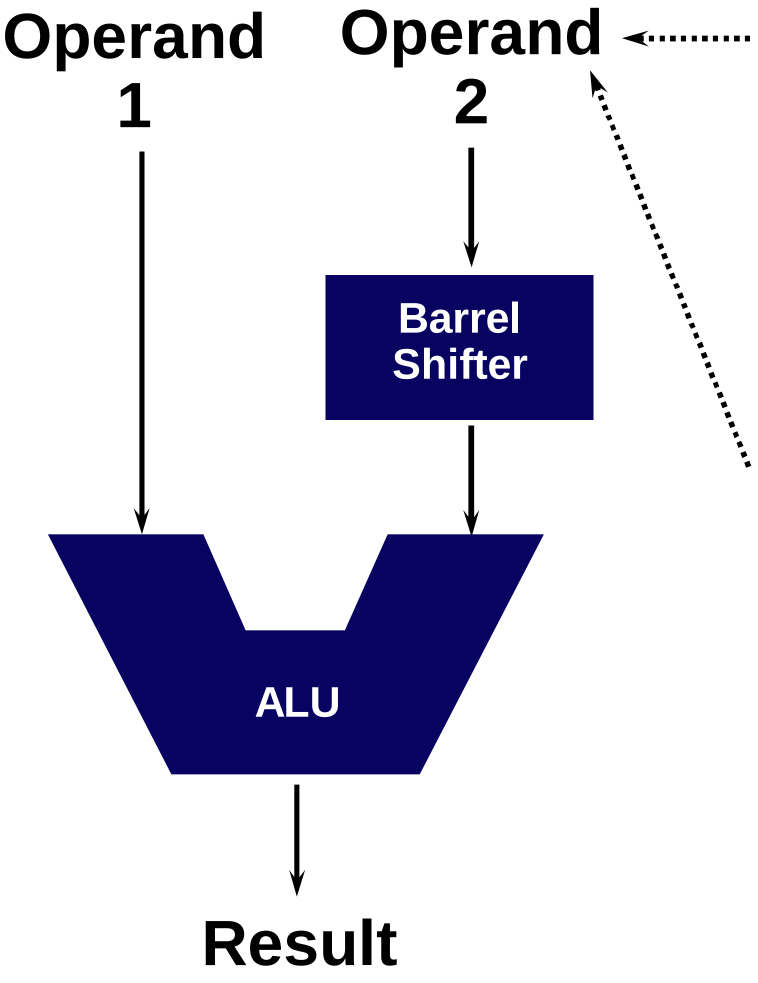
- These instructions only work on registers, NOT memory.

- Syntax:

```
<Operation>{<cond>}{S} Rd, Rn, Operand2
```

  - Comparisons set flags only - they do not specify Rd
  - Data movement does not specify Rn
  - Second operand is sent to the ALU via barrel shifter.

Using a Barrel Shifter:The 2nd Operand



Register, optionally with shift operation

- Shift value can be either be:
  - 5 bit unsigned integer
  - Specified in bottom byte of another register.
- Used for multiplication by constant

Immediate value

- 8 bit number, with a range of 0-255.
  - Rotated right through even number of positions
- Allows increased range of 32-bit constants to be loaded directly into registers

Single register data transfer

LDR	**STR**	Word
LDRB	**STRB**	Byte
LDRH	**STRH**	Halfword
LDRSB		Signed byte load
LDRSH		Signed halfword load

- Memory system must support all access sizes

- Syntax:
  - **LDR**{<cond>}{<size>} Rd, <address>
  - **STR**{<cond>}{<size>} Rd, <address>

e.g. **LDREQB**

Agenda

Introduction to ARM Ltd

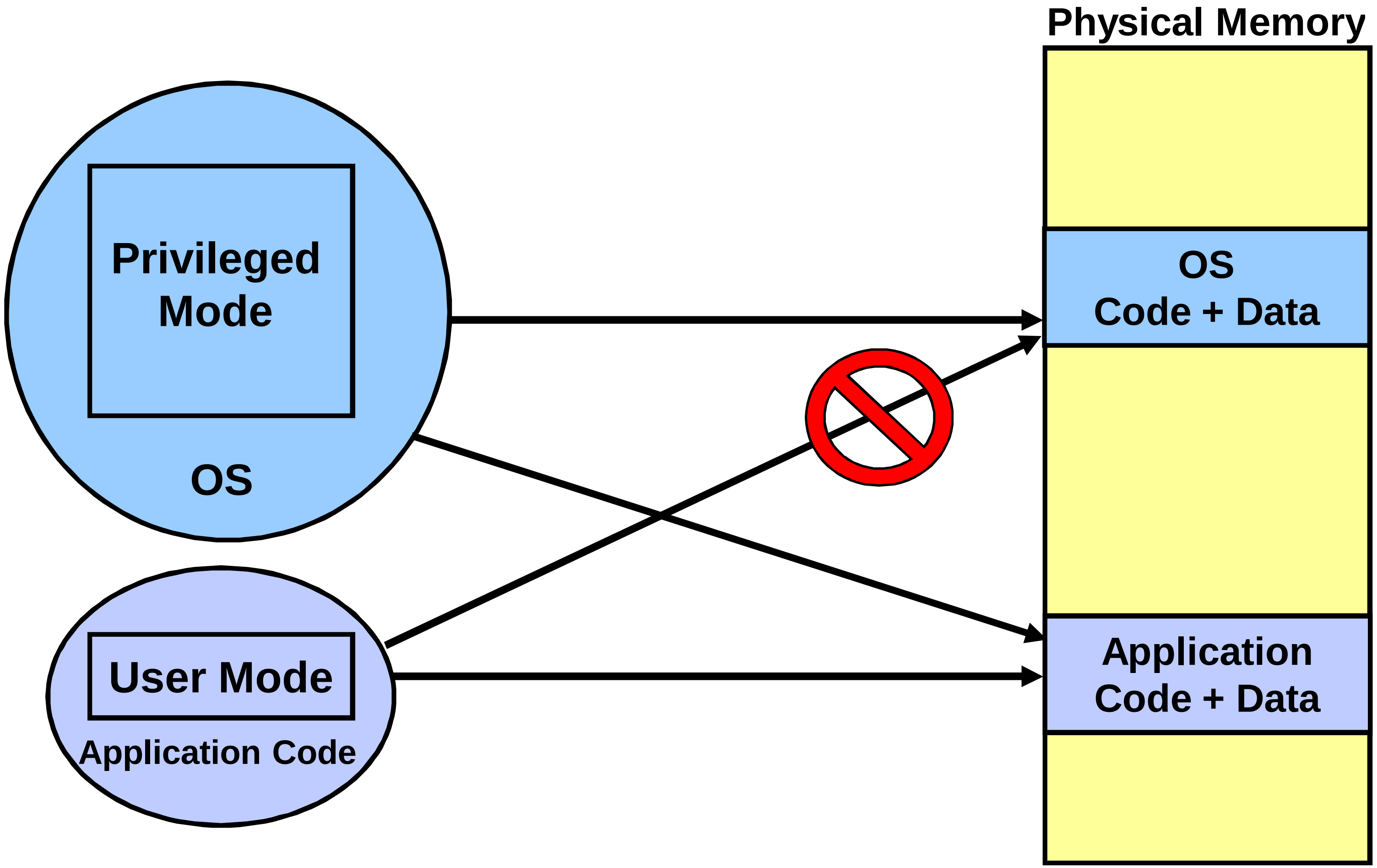
ARM Processors Overview

ARM v7A Architecture/Programmers Model

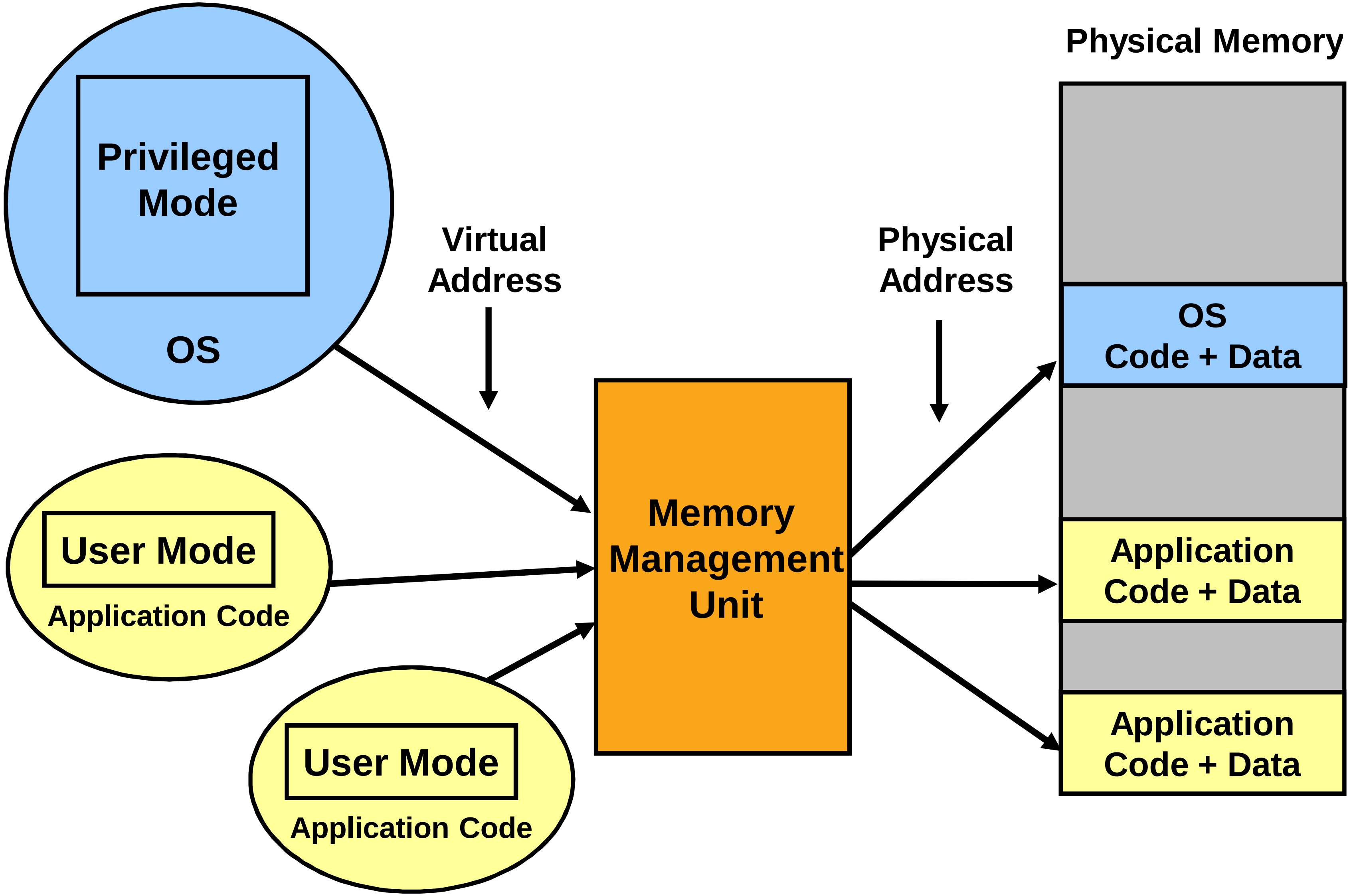
- Cortex-A8 Memory Management

Cortex-A8 Pipeline

Memory Protection



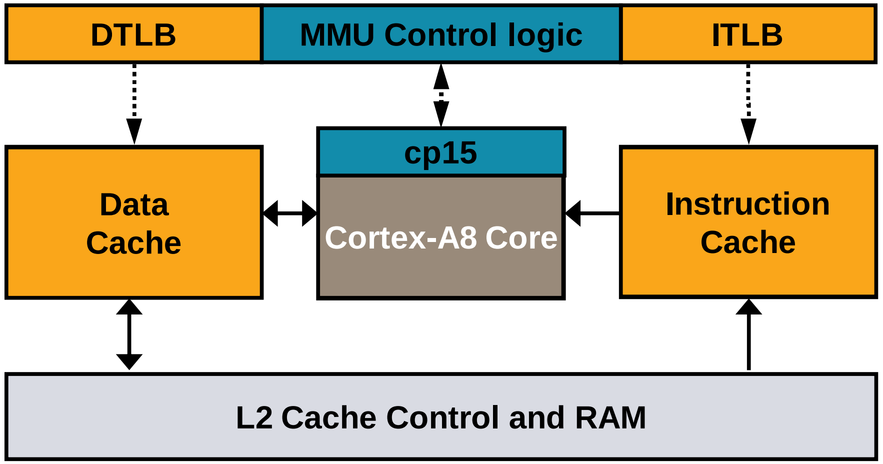
Memory Allocation



Memory Management

- Memory Management Unit (MMU)
  - Controls accesses to and from external memory
  - Assigns access permissions to memory regions
  - Performs virtual to physical address translation

- Instruction and Data Translation Look-Aside Buffers (TLB)
  - Contains recent virtual to physical address translations
  - Associates an ASID with each entry
    - ASID identifies which process is currently active



Agenda

Introduction to ARM Ltd

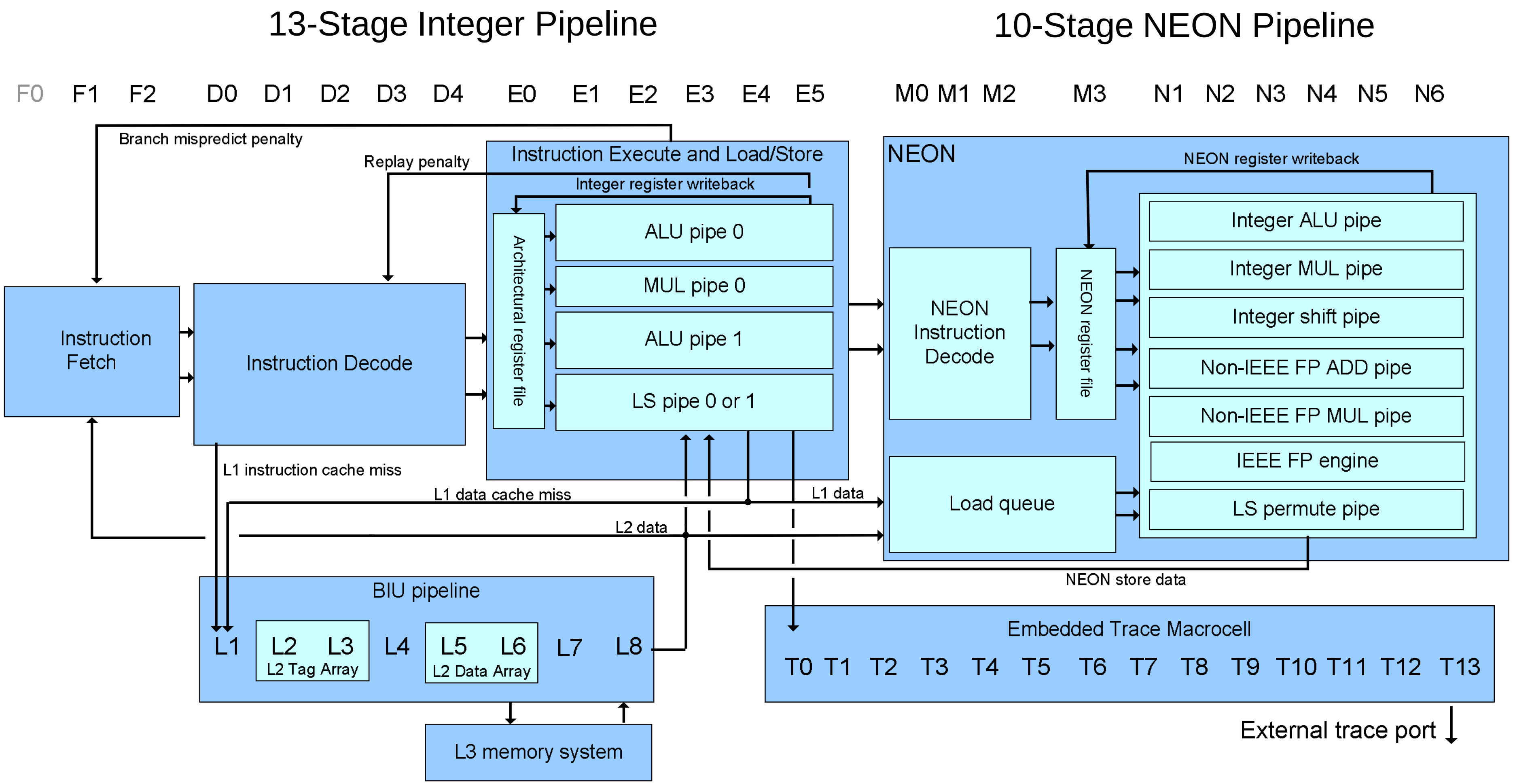
ARM Processors Overview

ARM v7A Architecture/Programmers Model

Cortex-A8 Memory Management

- Cortex-A8 Pipeline

Full Cortex-A8 Pipeline Diagram



Security - TrustZone

- Security – Property of the System which ensures resources of value cannot be copied, damaged or made un-available to genuine users
- Security cannot be foolproof so focus should be on
  - Assets to protect
  - Attacks against which it has to be protected
  - Goal: Attack A on Asset B will take Y days at Z dollars cost
- Need for Security
  - Embedded devices are handling data of increasing value such as Banking data
  - Different market sectors have need different needs. Ex Mobile Sector, Consumer electronics

Cellular Handset SoC Design

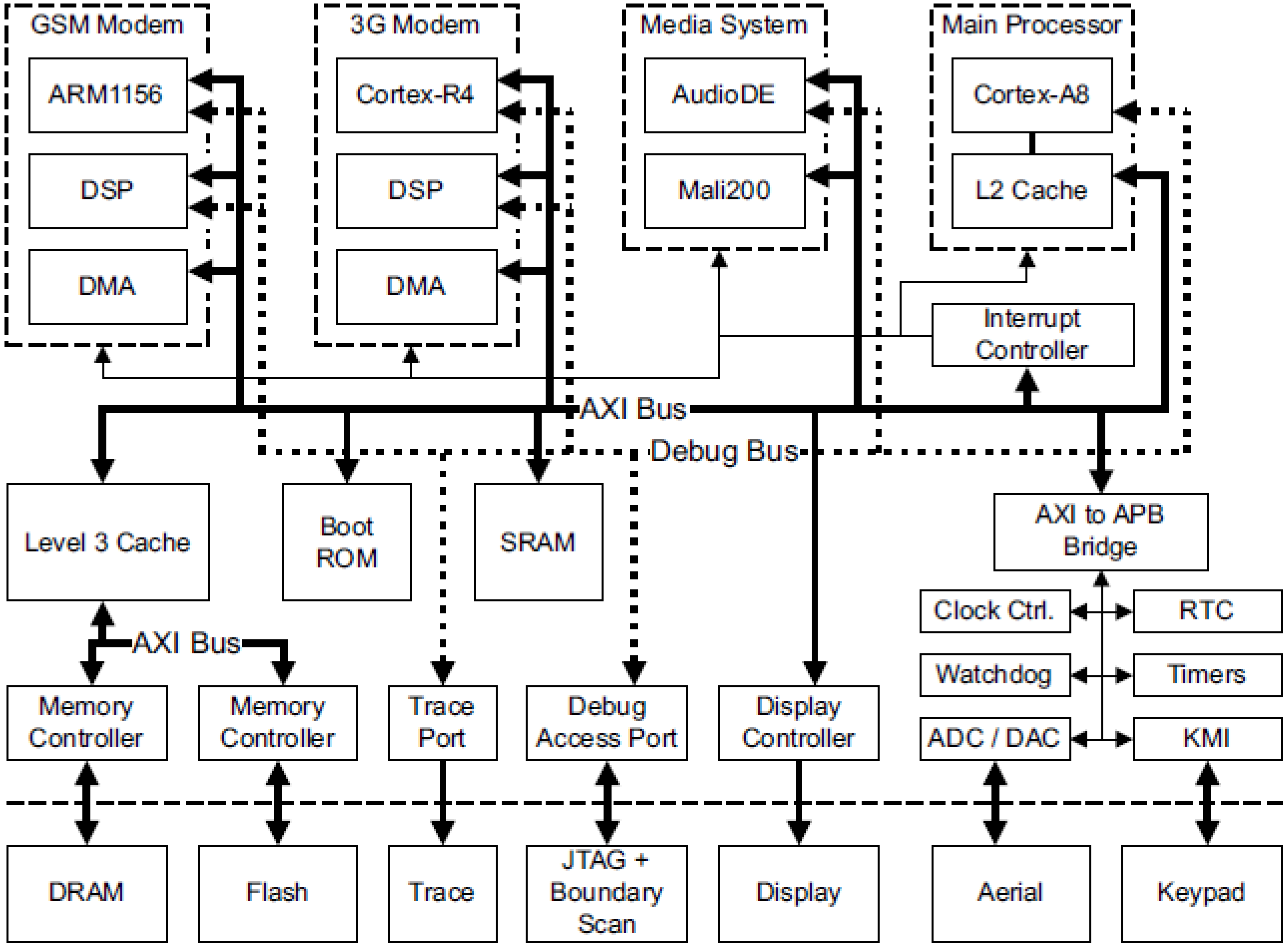


Figure 2-1 : A simplified schematic of a typical cellular handset SoC design

TrustZone

Cortex-A8 References

- Cortex-A8 Technical Reference Manual
- ARM Architecture Reference Manual v7-AR
- RealView Compilation Tools Compiler Reference Guide
- RealView Compilation Tools Compiler User Guide

http://infocenter.arm.com

ARM University Program Resources

- http://www.arm.com/support/university/
- University@arm.com

Fin