RealView®
Tools by ARM®

ARM Architecture Overview

Development of the ARM Architecture

- **Processor Architecture = Instruction Set + Programmer's model**

ARM7TDMI
ARM922T

Thumb instruction set

ARM926EJ-S
ARM946E-S
ARM966E-S

Improved ARM/Thumb Interworking

DSP instructions

Extensions:

Jazelle (5TEJ)

ARM1136JF-S
ARM1176JZF-S
ARM11 MPCore

SIMD Instructions

Unaligned data support

Extensions:

Thumb-2 (6T2)

TrustZone (6Z)

Multicore (6K)

Cortex-A8/R4/M3/M1

Thumb-2

Extensions:

v7A (applications) – NEON

v7R (real time) – HW Divide

V7M (microcontroller) – HW Divide and Thumb-2 only

- Note: Implementations of the same architecture can be very different
  - ARM7TDMI - architecture v4T. Von Neuman core with 3 stage pipeline
  - ARM920T - architecture v4T. Harvard core with 5 stage pipeline and MMU

ARM Architecture profiles

- Application profile (ARMv7-A → e.g. Cortex-A8)
  - 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
- Real-time profile (ARMv7-R → e.g. Cortex-R4)
  - Protected memory (MPU)
  - Low latency and predictability 'real-time' needs
  - Evolutionary path for traditional embedded business
- Microcontroller profile (ARMv7-M → e.g. Cortex-M3)
  - Lowest gate count entry point
  - Deterministic and predictable behavior a key priority
  - Deeply embedded use

Programmer's Model

RealView® Tools by ARM®

Data Sizes and Instruction Sets

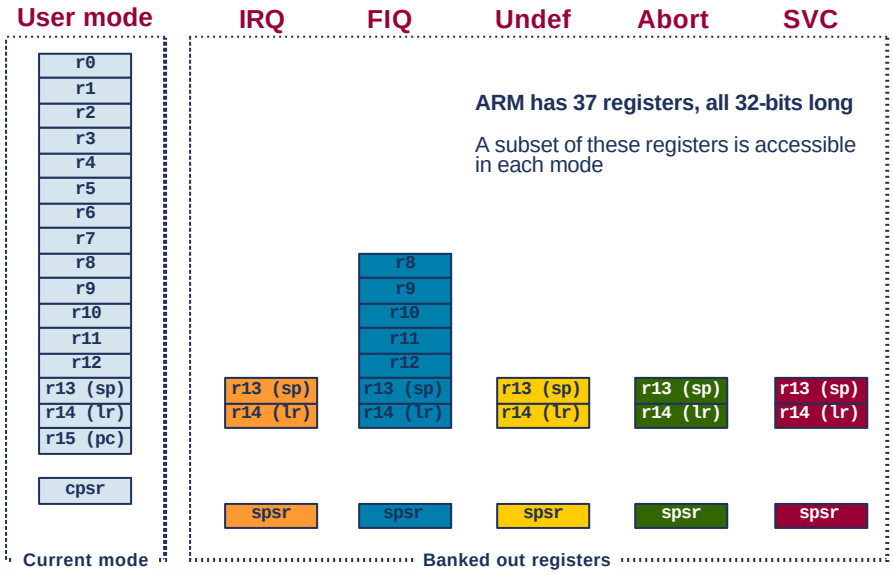
- When used in relation to the ARM:
  - **Halfword** means 16 bits (two bytes)
  - **Word** means 32 bits (four bytes)
  - **Doubleword** means 64 bits (eight bytes)
- Most ARMs implement two instruction sets
  - 32-bit **ARM** Instruction Set
  - 16-bit **Thumb** Instruction Set
- Latest ARM cores introduce a new instruction set Thumb-2
  - Provides a mixture of 32-bit and 16-bit instructions
  - Maintains code density with increased flexibility
- Jazelle-DBX cores can also execute Java bytecode

Processor Modes

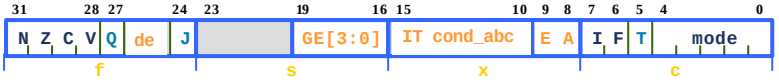
- The ARM has seven basic operating modes:
  - Each mode has access to own stack and a different subset of registers
  - Some operations can only be carried out in a privileged mode

	Mode	Description	
Exception modes	Supervisor (SVC)	**Entered on reset and when a Software Interrupt instruction (SWI) is executed**	Privileged modes
	FIQ	**Entered when a high priority (fast) interrupt is raised**	
	IRQ	**Entered when a low priority (normal) 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

The ARM Register Set



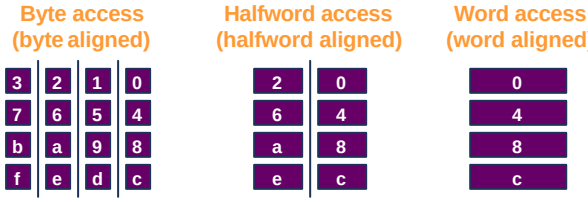
Program Status Registers



- Condition code flags
  - N = Negative result from ALU
  - Z = Zero result from ALU
  - C = ALU operation Carried out
  - V = ALU operation oVerflowed
- Sticky Overflow flag - Q flag
  - Architecture 5TE and later only
  - Indicates if saturation has occurred
- J bit
  - Architecture 5TEJ and later only
  - J = 1: Processor in Jazelle state
- Interrupt Disable bits
  - I = 1: Disables IRQ
  - F = 1: Disables FIQ
- T Bit
  - T = 0: Processor in ARM state
  - T = 1: Processor in Thumb state
  - Introduced in Architecture 4T
- Mode bits
  - Specify the processor mode
- New bits in V6
  - **GE[3:0]** used by some SIMD instructions
  - **E** bit controls load/store endianness
  - **A** bit disables imprecise data aborts
  - **IT [abcde]** IF THEN conditional execution of Thumb2 instruction groups

Data alignment

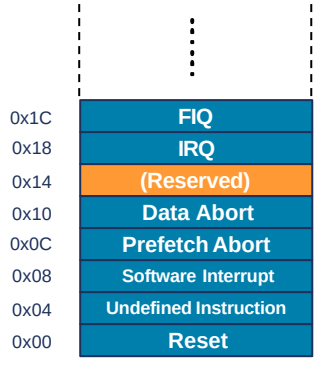
- Prior to architecture v6 data accesses must be appropriately aligned for access size
  - Unaligned addresses will produce unexpected/undefined results



- Unaligned data can be accessed using multiple aligned accesses combined with shift/mask operations

Exception Handling

- When an exception occurs, the core:
  - 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>



Vector Table

Vector table can also be at **0xFFFF0000** on most cores

- Must be done in ARM state in most cores, but...
  ...Thumb-2 capable cores can do this in Thumb state

Introduction to Instruction Sets

RealView® Tools by ARM®

ARM Instruction Set

- All instructions are 32 bits long / many execute in a single cycle
- Instructions are conditionally executed
- A load / store architecture
  - Example data processing instructions

```
SUB     r0,r1,#5
ADD     r2,r3,r3,LSL #2
ADDEQ   r5,r5,r6
```

```
r0 = r1 - 5
r2 = r3 + (r3 * 4)
IF EQ condition true r5 = r5 + r6
```

  - Example branching instruction

```
B       <Label>
```

```
Branch forwards or backwards relative to
current PC (+/- 32MB range)
```

  - Example memory access instructions

```
LDR     r0,[r1]
STRNEB  r2,[r3,r4]
STMFD   sp!,{r4-r8,lr}
```

```
Load word at address r1 into r0
IF NE condition true, store bottom byte
of r2 to address r3+r4
Store registers r4 to r8 and lr on
stack. Then update stack pointer
```

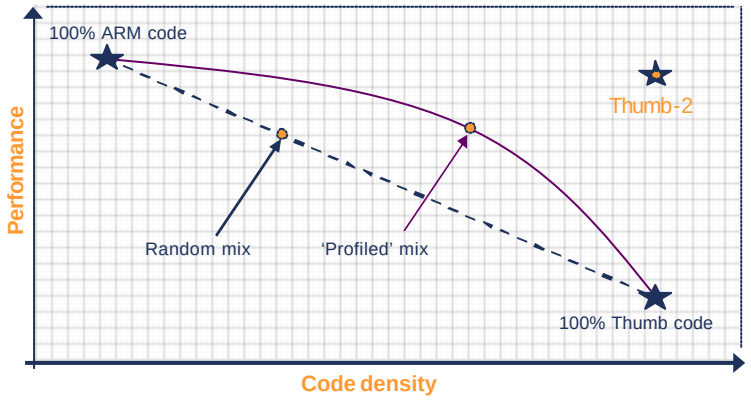
Thumb Instruction Set

- Thumb is a 16-bit instruction set
  - Optimized for code density from C code (~65% of ARM code size)
  - Improved performance from narrow memory
  - Subset of the functionality of the ARM instruction set
- Thumb is not a “regular” instruction set!
  - Constraints are not generally consistent
  - Targeted at compiler generation, not hand coding

Thumb-2 Instruction Set

- Thumb-2 is a major extension to the Thumb ISA
  - Adds 32-bit instructions to implement almost all of the ARM ISA functionality
  - Retains the complete 16-bit Thumb instruction set
- Design objective: ARM performance with Thumb code density
  - No switching between ARM-Thumb states
  - Compiler automatically selects mix of 16 and 32 bit instructions

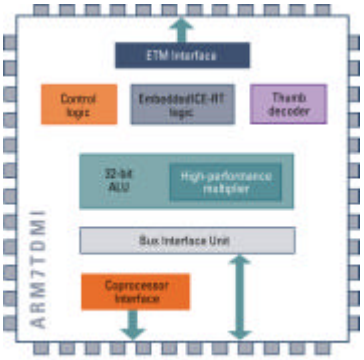
Thumb 2 Performance / Density



Processor Cores

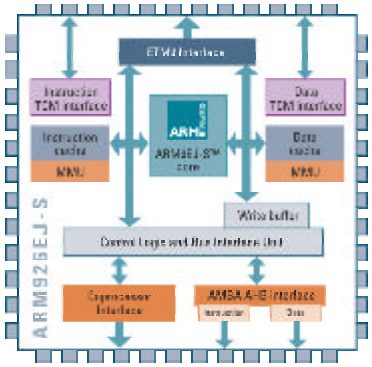
RealView® Tools by ARM®

ARM7TDMI Processor



- Architecture v4T
- 3-stage pipeline
- Single interface to memory

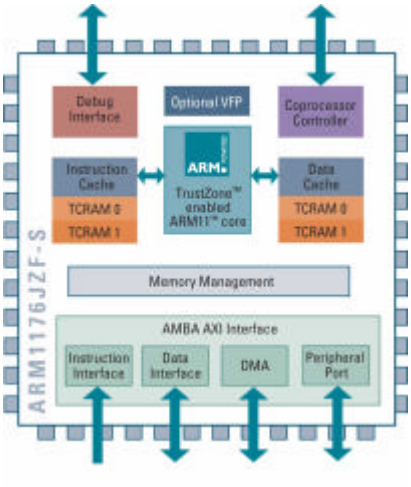
ARM926EJ-S Processor



ARM926EJ-S

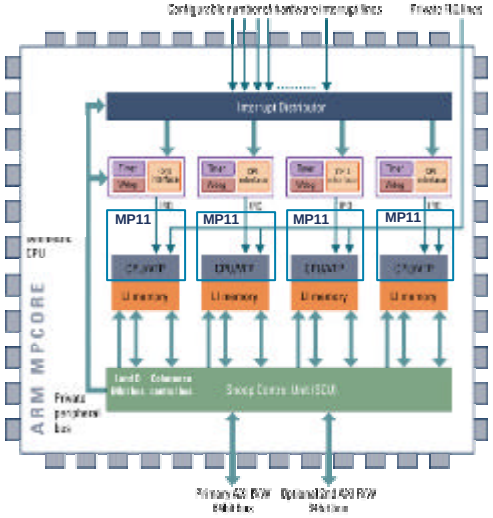
- Architecture v5TE
- 5-stage pipeline
- Single-cycle 32x16 multiplier
- Caches and TCMs
- Memory management unit (MMU)
- 2 AHB memory interfaces
- Jazelle technology

ARM1176JZ(F)-S Processor Core



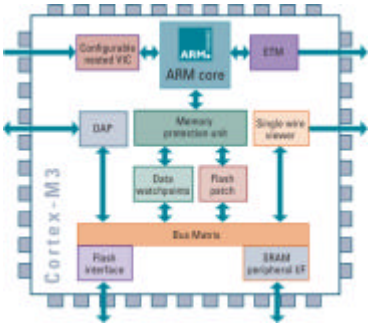
- TrustZone
- 8-stage pipeline
- Branch prediction
- Four AXI memory ports
- IEM (Intelligent Energy Management)
- Integrated VFP coprocessor

ARM11 MPCore Processor



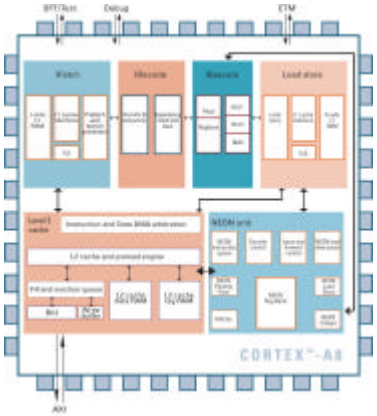
- 1 – 4 MP11 processors
- Cache coherency
- Distributed interrupt controller

ARM Cortex-M3 Processor



- Architecture v7-M (Thumb-2 only) → Very different from previous ARM processors
  - No CPSR register
  - Vector table contains addresses, not instructions
  - Processor automatically saves/restores state in exceptions
  - Only 2 processor modes (Thread/Handler)
  - No Coprocessor 15 3-stage pipeline with static branch prediction
- Atypical Implementation
  - Fixed memory map
  - Integrated interrupt controller
  - Serial-Wire Debug

ARM Cortex-A8 Processor



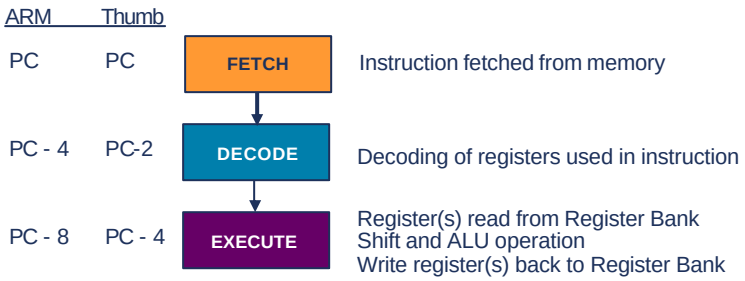
- Architecture v7-A
- 14 stage pipeline
- NEON media processor

The Instruction Pipeline

RealView® Tools by ARM®

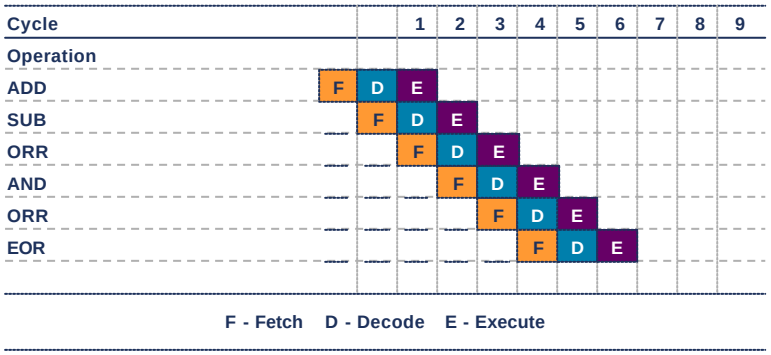
The Instruction Pipeline

- The ARM7TDMI uses a 3-stage pipeline in order to increase the speed of the flow of instructions to the processor
  - Allows several operations to be performed simultaneously, rather than serially



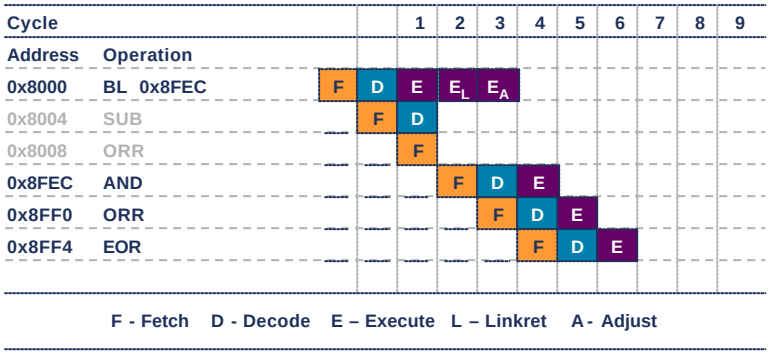
- The PC points to the instruction being fetched, not executed
  - Debug tools will hide this from you
  - This is now part of the ARM Architecture and applies to all processors

Optimal Pipelining



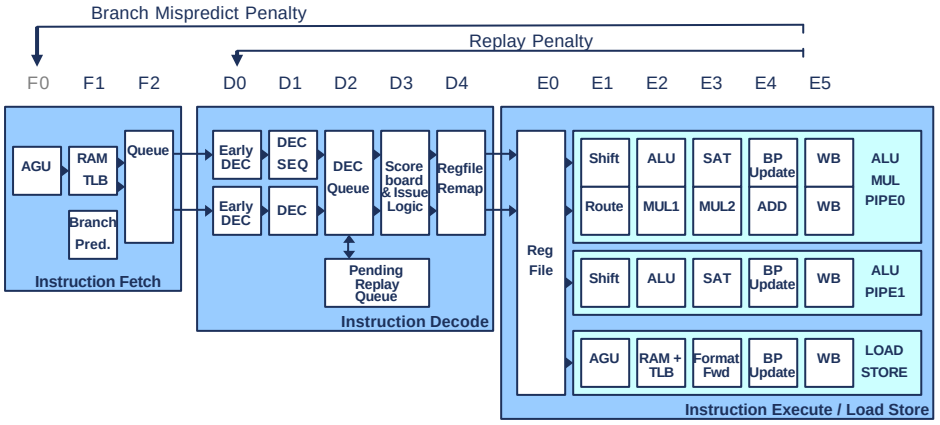
- All operations here are on registers (single cycle execution)
- In this example it takes 6 clock cycles to execute 6 instructions
- Clock cycles per Instruction (CPI) = 1

Branch Pipeline Example



- Breaking the pipeline
- Note that the core is executing in ARM state

Cortex-A8 Integer Pipeline



- Optimising code to make use of the processor pipeline is very difficult
- Leave it to the compiler!!

Reference Slides

RealView® Tools by ARM®

Reference Material

- ARM ARM ("Architecture Reference Manual")
  - ARM DDI 0100E covers v5TE DSP extensions
    - Can be purchased from booksellers - ISBN 0-201-737191 (Addison-Wesley)
    - Available for download from ARM's website
  - ARM v7-M ARM available for download from ARM's website
  - Contact ARM if you need a different version (v6, v7-AR, etc.)
- Steve Furber "ARM system-on-chip architecture" - 2nd edition
  - ISBN 0-201-67519-6 (Addison-Wesley)
- Sloss, Symes & Wright – "ARM System Developer's Guide"
  - ISBN: 1-55860-874-5 (Morgan Kaufman)
- RVCT Assembler Guide
  - Available for download from ARM's website
- Technical Reference Manuals for processor core being used
  - Available for download from ARM's website

Naming Conventions

- ARMx1z (e.g. ARM710T) indicates cache & full MMU
- ARMx2z (e.g. ARM720T) indicates cache, MMU & Process ID support
- ARMx3z (e.g. ARM1136J-S) indicates physically mapped caches and MMU
- ARMx4z (e.g. ARM740T) indicates cache and MPU
- ARMx5z (e.g. ARM1156T2-S) indicates cache, MPU and error correcting memory
- ARMx6z (e.g. ARM966E-S) indicates write buffer but no caches
- ARMx7z (e.g. ARM1176JZ-S) indicates AXI bus, & physically mapped caches and MMU
- ARMxy6 (e.g. ARM946E-S) indicates TCMs

Which architecture is my processor?

Processor core	Architecture
ARM7TDMI family	v4T
▪ ARM720T, ARM740T	
ARM9TDMI family	v4T
▪ ARM920T,ARM922T,ARM940T	
ARM9E family	v5TE, v5TEJ
▪ ARM946E-S, ARM966E-S, ARM926EJ-S	
ARM10E family	v5TE, v5TEJ
▪ ARM1020E, ARM1022E, ARM1026EJ-S	
ARM11 family	v6
▪ ARM1136J(F)-S	v6
▪ ARM1156T2(F)-S	v6T2
▪ ARM1176JZ(F)-S	v6Z
▪ ARM11 MPCore	v6
Cortex family	
▪ ARM Cortex-A8	v7-A
▪ ARM Cortex-R4(F)	v7-R
▪ ARM Cortex-M3	v7-M
▪ ARM Cortex-M1	v6-M

- For ARM processor naming conventions and features, please see the Appendix

ARMv4T Cores:

	7TDMI	720T	740T	920T	940T	SA1100
Architecture	von Neumann	von Neumann	von Neumann	Harvard	Harvard	Harvard
Cache	None	8K Unified 4 words/line	8K Unified 4 words/line	16K Instr + 16K Data 8 words/line	4K Instr + 4K Data 4 words/line	16K Instr + 16K Data 4 words/line
Associativity	N/A	4-way	4-way	64-way	64-way	32-way
TCM	No	No	No	No	No	No
Replacement	N/A	Random	Random	Random Round Robin	Random	Round Robin
Write Strategy	N/A	Write Through	Write Through	Write Through Write Back	Write Through Write Back	Write Back
Write Buffer	None	8 Words 4 Addresses	8 Words 4 Addresses	16 Words 4 Addresses	8 Words 4 Addresses	8 Words 4 Addresses
MMU/MPU	None	MMU	MPU	MMU	MPU	MMU
Hi Vectors	No	Yes	No	Yes	Yes	Yes
Streaming	N/A	Yes	Yes	Yes	Yes	Yes
Standby Mode	No	No	No	Yes	Yes	Yes

ARMv5 Cores:

	926EJ-S	946E-S	966E-S	968E-S	1026EJ-S	XScale
Architecture	Harvard	Harvard	Harvard	Harvard	Harvard	Harvard
Cache	4-128K Instr 4-128K Data 8 words/line	0-1024K Instr 0-1024K Data 8 words/line	None	None	0-128K Instr 0-128K Data 8 words/line	32K Instr 32K Data 8 words/line
Associativity	4-way	4-way	N/A	N/A	4-way	32- way
TCM	0-1024K Instr 0-1024K Data	0-1024K Instr 0-1024K Data	0-64M Instr 0-64M Data	0-64M Instr 0-64M Data	0-1024K Instr 0-1024K Data	No
Replacement	Random Round Robin	Random Round Robin	N/A	N/A	Random Round Robin	Random Round Robin
Write Strategy	Write Through Write Back	Write Through Write Back	N/A	Write Through Write Back	Write Through Write Back	Write Through Write Back
Write Buffer	16 Words 4 Addresses	16 Words Data or Address	12 Words Data or Address	12 Words Data or Address	8 Words Data or Address	8 x 16 Bytes Coalescing
MMU/MPU	MMU	MPU	None	None	MMU or MPU	MMU With extensions
Hi Vectors	Yes	Yes	Yes	Yes	Yes	Yes
Streaming	Yes	Yes	N/A	N/A	Yes	Yes
Standby Mode	Yes	Yes	Yes	Yes	Yes	Yes

ARMv6 Cores:

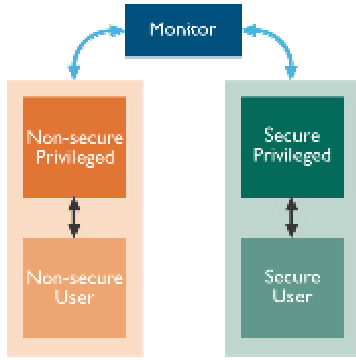
	1136EJ(F)-S	1156T2(F)-S	1176JZ(F)-S	MPCore11
Architecture	Harvard	Harvard	Harvard	Harvard
Cache	4-64K Instr 4-64K Data 8 words/line	0-64K Instr 0-64K Data 8 words/line	4-64K Instr 4-64K Data 8 words/line	16-64K Instr 16-64K Data 8 words/line
Associativity	4-way	4-way	4-way	4-way
TCM	0-64K Instr 0-64K Data	0-256K Instr 0-256K Data	0-64K Instr 0-64K Data	None
Replacement	Random Round Robin	Random Round Robin	Random Round Robin	Random Round Robin
Write Strategy	Write Through Write Back	Write Through Write Back	Write Through Write Back	Write Through Write Back
MMU/MPU	MMU	MPU	MMU	MMU
Hi Vectors	Yes	Yes	Yes	Yes
Streaming	Yes	Yes	N/A	Yes
Standby Mode	Yes	Yes	Yes	Yes
Bus	AHB/APB	AXI	AXI	AXI
VFP Support	Yes	Yes	Yes	Yes

Cortex Cores:

	Cortex-M3	Cortex-M1	Cortex-R4	Cortex-A8
Architecture	Harvard	Harvard	Harvard	Harvard
Cache	None	None	4-64K Instr 4-64K Data 8 words/line	16 or 32 Instr 16 or 32 Data 16 words/line
Associativity	N/A	N/A	4-way	4-way
TCM	None	0-1M Instr 0-1M Data	0-8M Instr 0-8M Data	None
Replacement	N/A	N/A	Random	Random
Write Strategy	N/A	N/A	Write Through Write Back	Write Through Write Back
MMU/MPU	MPU	None	MPU (optional)	MMU
Hi Vectors	No	No	Yes	Yes
Streaming	N/A	N/A	Yes	Yes
Standby Mode	Yes	Yes	Yes	Yes
Bus	AHB Lite/APB	AHB Lite/APB	AXI	AXI
VFP Support	No	No	Yes	Yes

TrustZone Computing

- TrustZone adds a “parallel world” to allow trusted programs and data to be safely separated from the OS and applications
- Introduced for ARM1176, standard for ARMv7-A Cores
- Features:
  - **New Secure Monitor Mode: gate-keeper for secure state**
  - **New S-bit in CP15 to indicate when the processor is running in a secured state**
  - **Security state exposed on external bus accesses to permit security-aware memory and peripherals**
  - **Ability to restrict debug to non-secure state**



NEON Media Processor Features

- Single Instruction Multiple Data (SIMD) Media Processor
- Targets audio and video codecs, image and speech processing, graphics, baseband processing, and general signal processing
- 3 Processing pipelines: Integer/fixed point, single precision floating point, IEEE vector floating point
- Efficient data handling
  - Best use of available memory bandwidth
  - Eliminates data arrangement overhead
  - Operates on separate register file
  - SIMD Framework excellent target for compilers

RealView® Tools by ARM®

End