The ARM Architecture

Peng-Sheng Chen

Fall, 2017

Outline

- The Acorn RISC Machine
- Architectural inheritance
- The ARM programmer's model
- ARM development tools

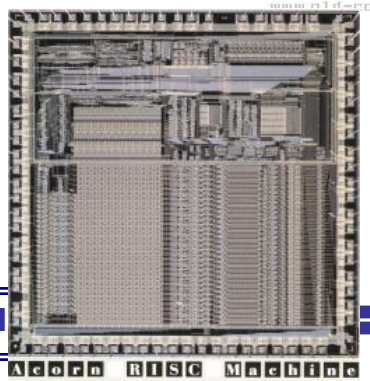
Outline

- The Acorn RISC Machine
- Architectural inheritance
- The ARM programmer's model
- ARM development tools

History (1)

- Acorn Computers
  - Produce desktop PCs
  - Target at the educational markets in the UK
- October, 1983
  - Begin to design microprocessors
- April 26, 1985
  - ARM1
  - Less than 25,000 transistors

Acorn business computer (ABC-310)
- CPU: 6502



History (2)

- ARM2 (1990)
  - No cache and MMU
  - Multiply and multiply-accumulate instructions
  - Coprocessor interface
  - 18 MHz in a 2 micron process
  - Archimedes desktop PC
  - VLSI Technology: VL86C010

Acorn Archimedes PC

History (3)

- In 1989, Acorn was continued to find sales in education, specialist, and hobbyist markets.
- VLSI Technology managed to find other companies willing to use the ARM processor in their designs, especially as an embedded processor.
- Apple was looking to enter the completely new field of personal digital assistants (PDAs).

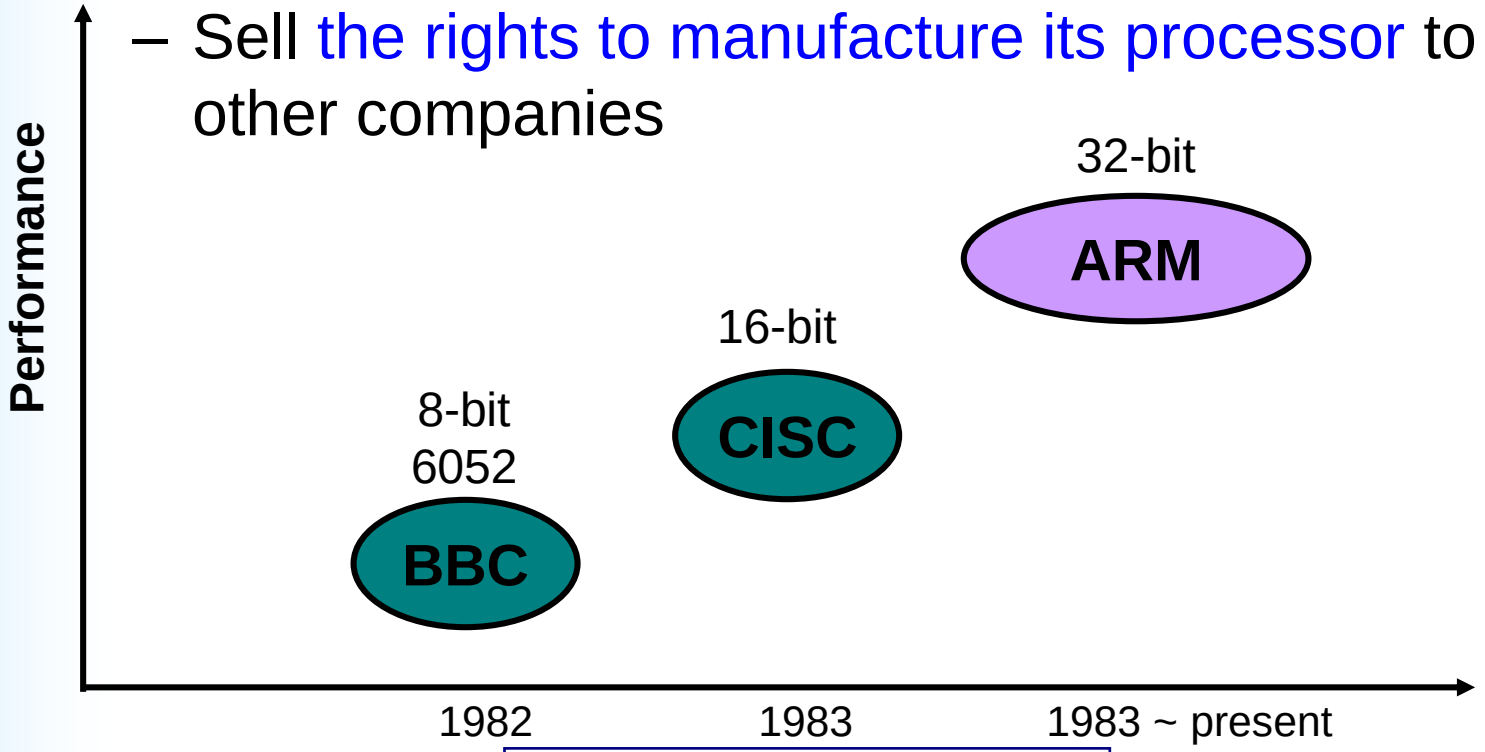
History (4)

- Apple's interest for its new device led to the creation of an entirely separate company to develop it. => Advanced RISC Machine
- The new company consisting of money from Apple, twelve Acorn engineers, and free tools from VLSI Technology, changed the name of the architecture from Acorn RISC Machine to Advanced RISC Machine.

Robin Saxby, managing director

The Creation of ARM Ltd.

- In 1990, ARM stood for **A**corn **R**ISC **M**achine
- Later, ARM stood for **A**dvanced **R**ISC **M**achine
- New business model
  - Sell the rights to manufacture its processor to other companies



ARM Ltd.

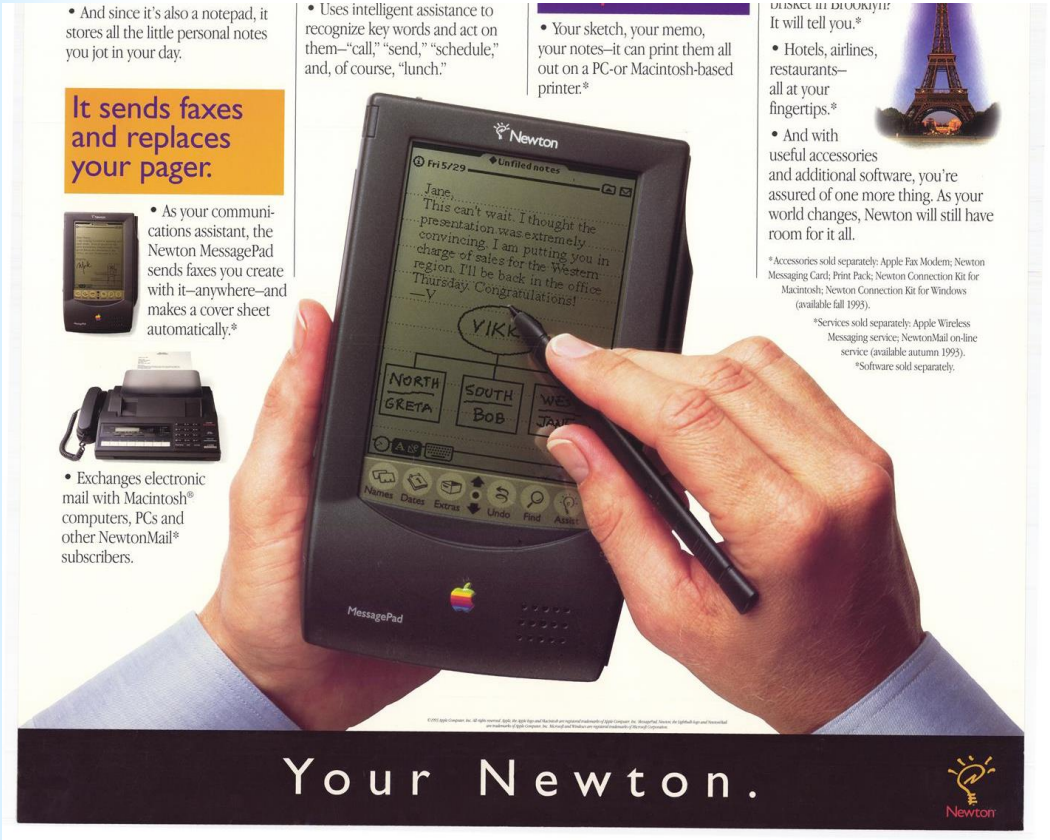
- Founded in November 1990
  - Advanced RISC Machine Limited
  - Spun out of Acorn Computers

- 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
  - Intellectual property (IP) company

- Also develop technologies to assist with the design-in of the ARM architecture
  - Software tools, boards, debug hardware, application software, bus architectures, peripherals etc.

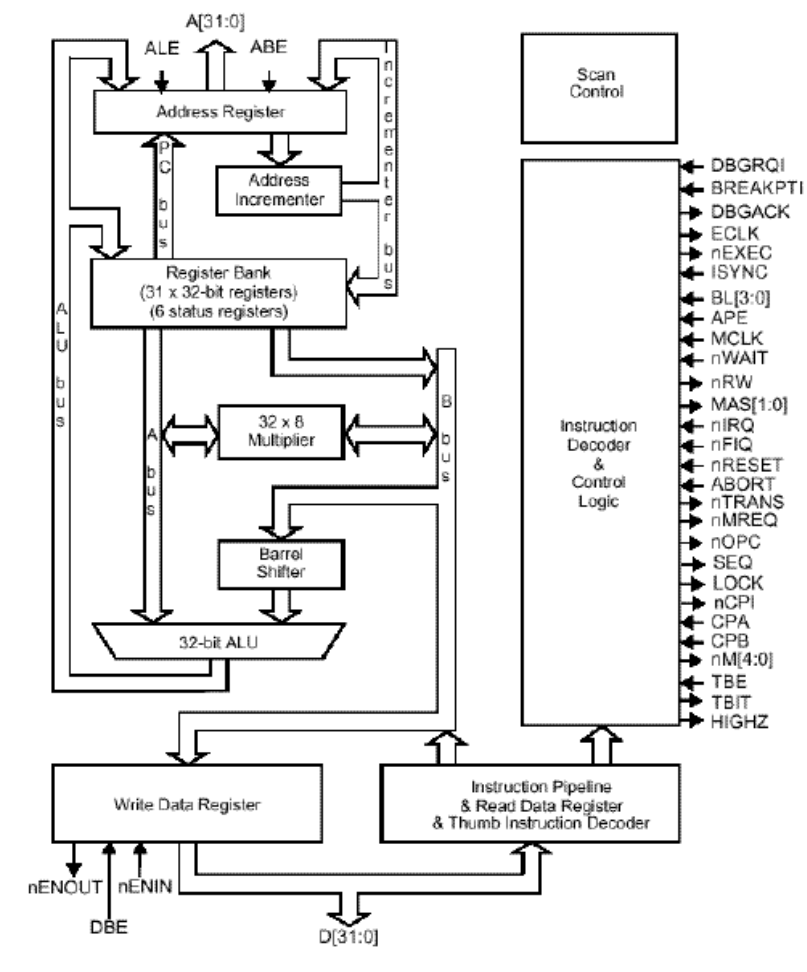
reference: http://www.intel.com/education/highered/modelcurriculum.htm

- ARM610
  - ARM6 core with 4K cache, MMU



Apple Newton PDA (1993)

- ARM7 TDMI



iPod

- processor: PortalPlayer PP5002 (SoC) which contains dual embedded 90 MHz ARM 7TDMI processors.

ARM Partnership Model

reference: http://www.arm.com

ARM的成功之道

- 持續發展新的處理器技術
  - Low power (省電)
  - High performance for embedded system
- 適當的市場切入點 (時機)
  - Embedded system的興起
  - Ex: cell phone, PDA, portable multimedia player, …

Outline

- The Acorn RISC Machine
- Architectural inheritance
- The ARM programmer's model
- ARM development tools

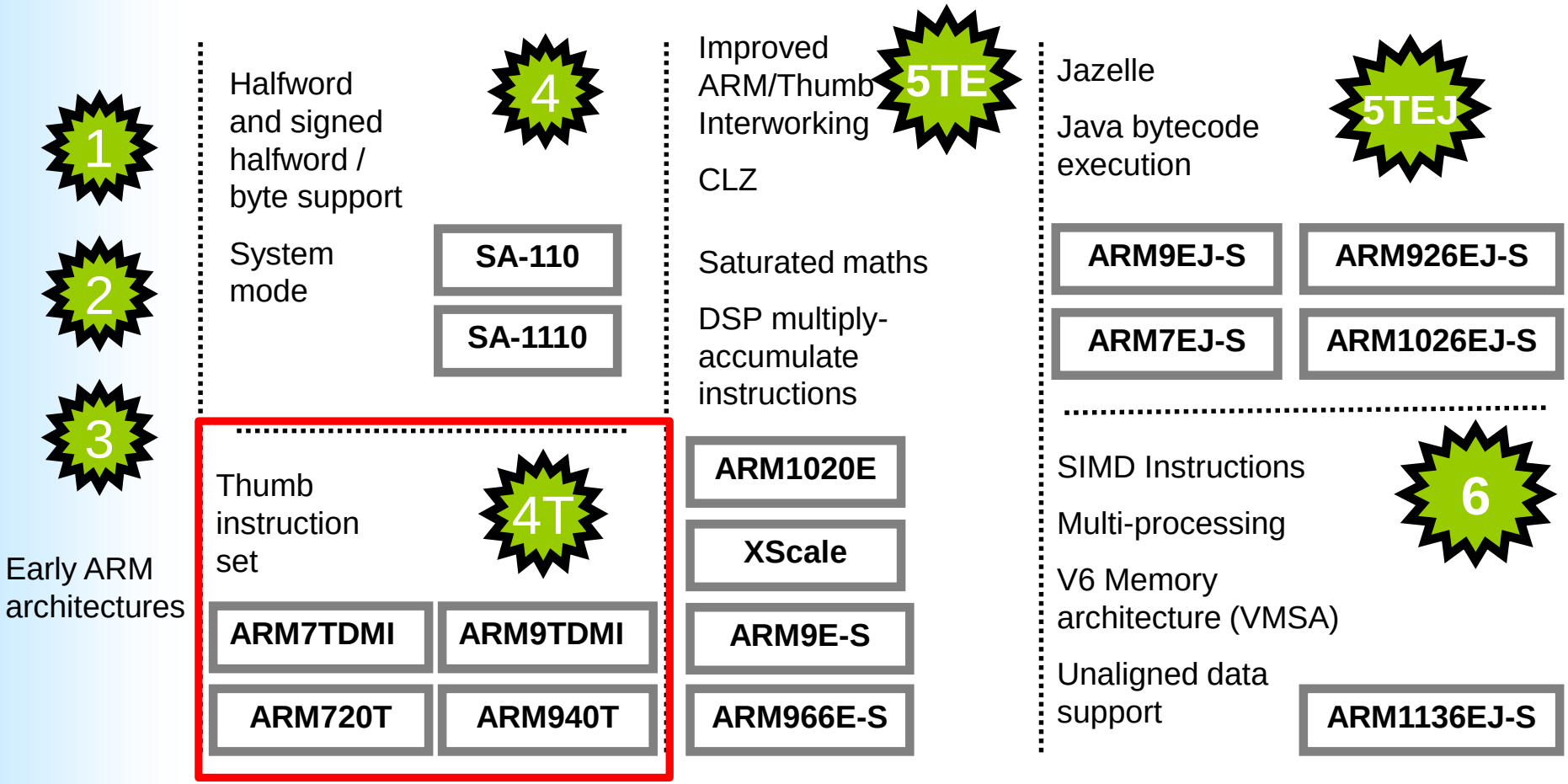
Features Used (1)

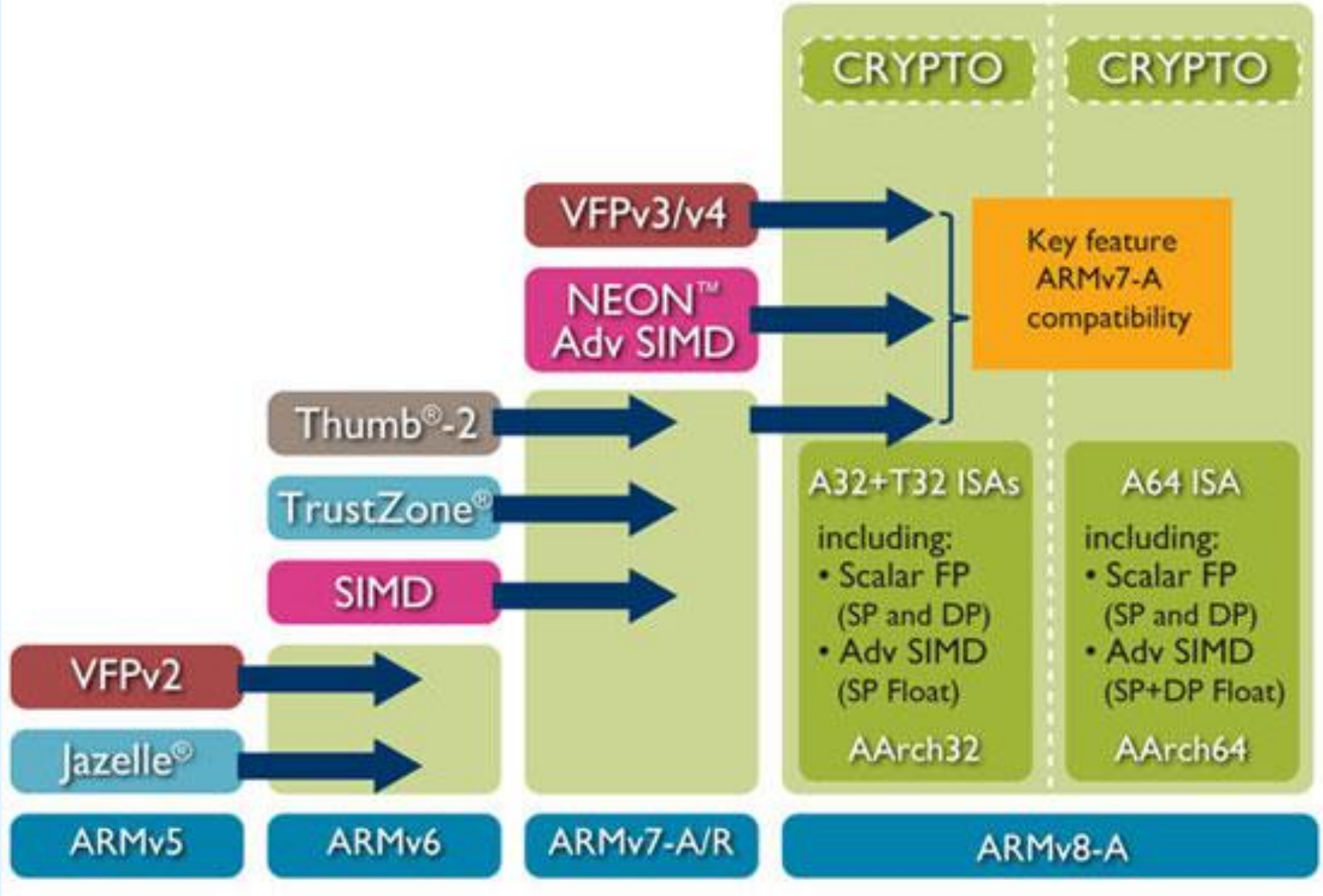
- Incorporate a number of features from the Berkeley **RISC** design
  - An uniform register file load/store architecture, where data processing operates only on register contents, not directly on memory contents.
  - Simple addressing modes, with all load/store addresses determined from register contents and instruction fields only.

Features Used (2)

- RISC (Reduced Instruction Set Computers)
  - Easy to design and implementation for hardware designers
  - Compilers dominate the performance
- Enhancements to a basic RISC architecture enable ARM processors to achieve a good balance of high performance, small code size, low power consumption and small silicon area.

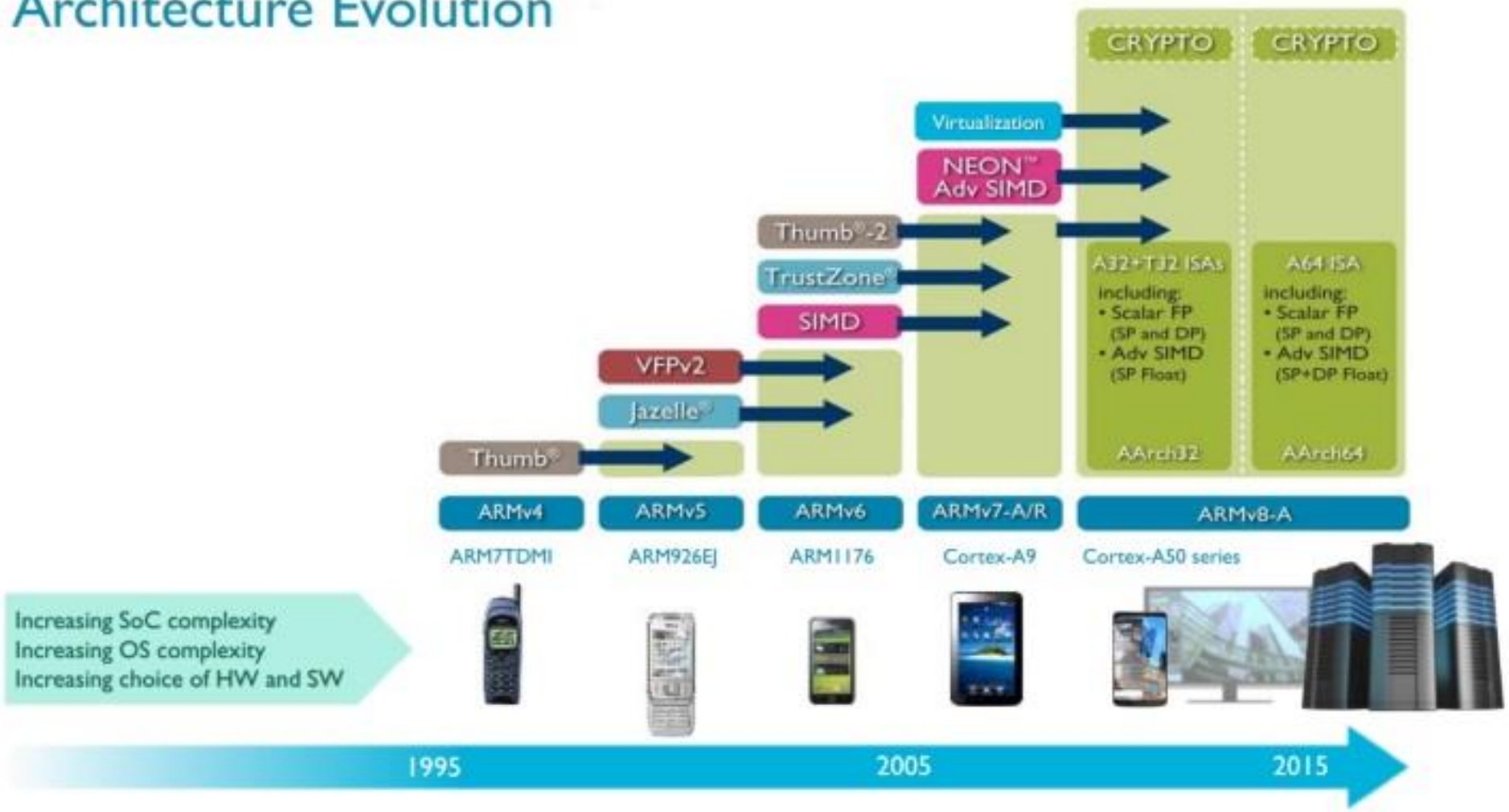
Development of the ARM Architecture (1)





http://www.arm.com/products/processors/instruction-set-architectures/index.php

Architecture Evolution



Reference from: http://www.androidauthority.com/developing-arm-everything-need-know-389402/

Development of the ARM Architecture (2)

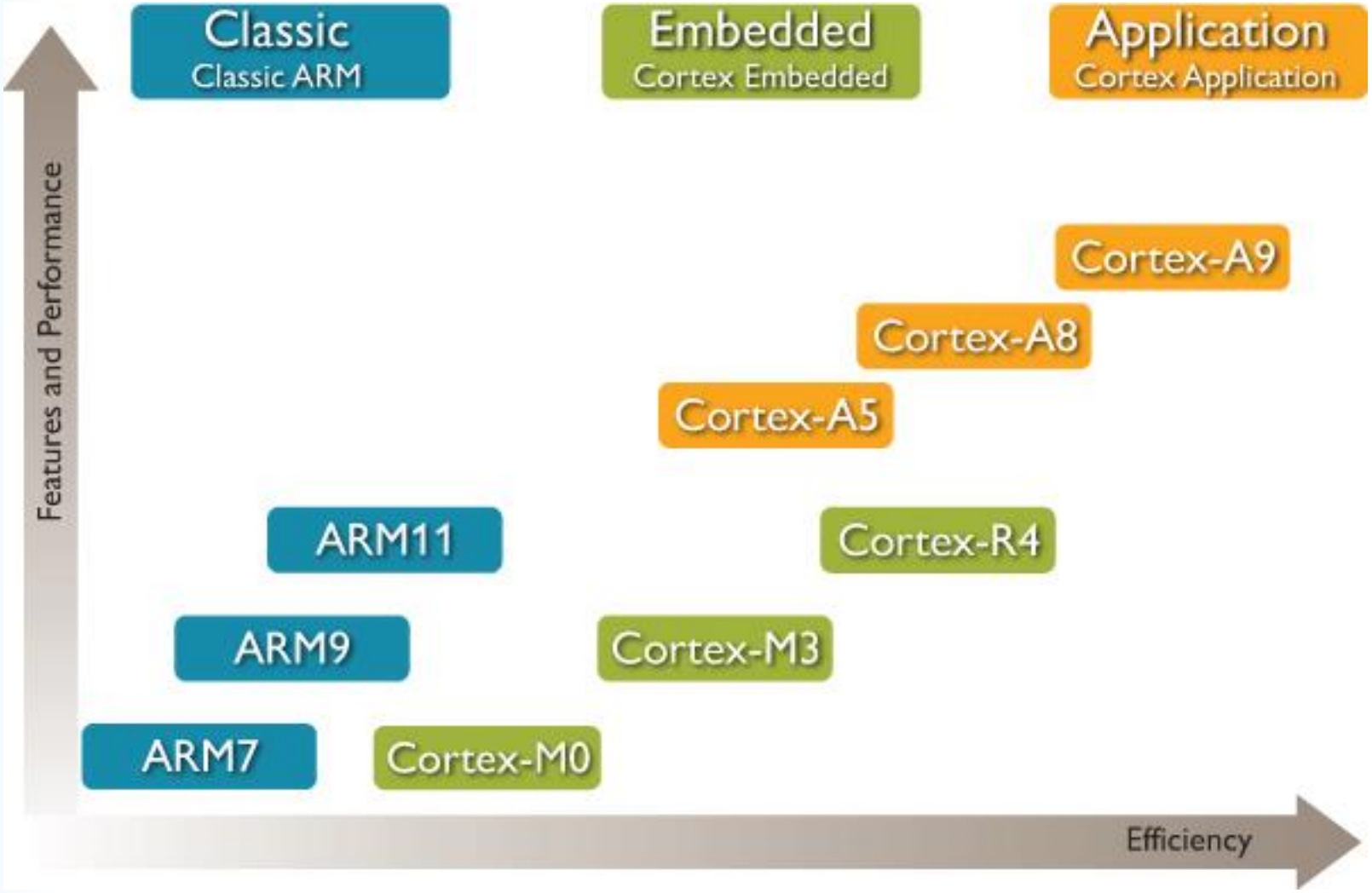
- iPhone 5
  - A6 processor (Dual cores, Cortex A9, ISA: ARM v7)
- iPhone 8
  - A11 processor (ARM v8-A, six-core CPU)
- Galaxy S8
  - Exynos 8895 (Octa-core, four-core custom CPU, four-core Cortex A53)

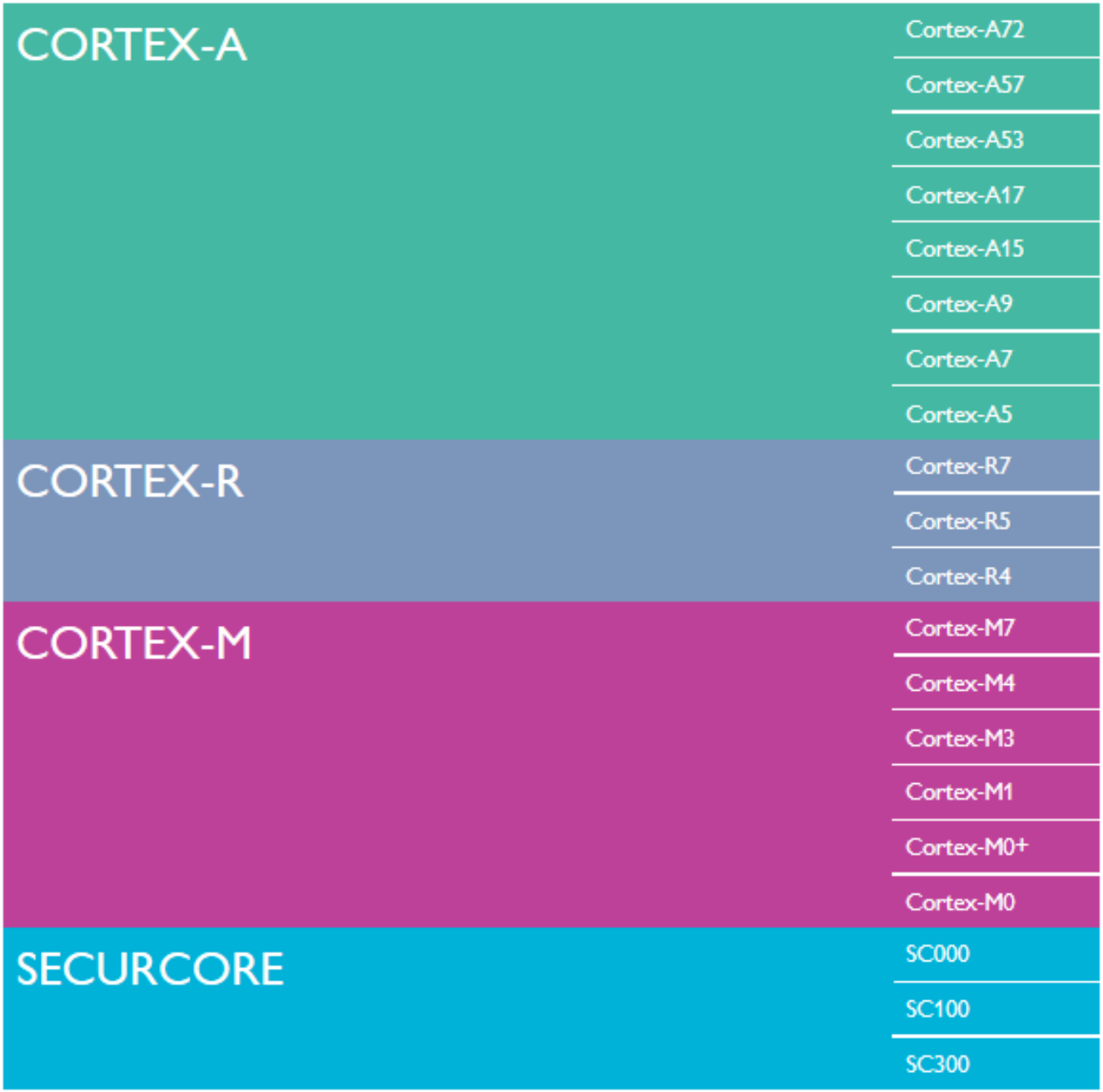
ARM Processors (1)

- Application processor
  - Cortex-A series: high performance processors for feature rich Operating Systems
  - Open platforms running complex operating systems for wireless, consumer and imaging applications.
  - Applications
    - Smartphones
    - Netbooks
    - Digital TV

ARM Processors (2)

- Real-time embedded processor
  - Cortex-R series
  - Embedded real-time systems for mass storage, automotive, industrial and networking applications.
- Embedded processor
  - Cortex-M Series
  - Cost-sensitive solutions for deterministic microcontroller applications
- Specialist processor
  - Secure core
  - Secure applications including smart cards and SIMs.


Classic
Classic ARM
Embedded
Cortex Embedded
Application
Cortex Application
Features and Performance
Cortex-A9
Cortex-A8
Cortex-A5
ARM11
Cortex-R4
ARM9
Cortex-M3
ARM7
Cortex-M0
Efficiency


CORTEX-A
Cortex-A72
Cortex-A57
Cortex-A53
Cortex-A17
Cortex-A15
Cortex-A9
Cortex-A7
Cortex-A5
CORTEX-R
Cortex-R7
Cortex-R5
Cortex-R4
CORTEX-M
Cortex-M7
Cortex-M4
Cortex-M3
Cortex-M1
Cortex-M0+
Cortex-M0
SECURCORE
SC000
SC100
SC300

Outline

- The Acorn RISC Machine
- Architectural inheritance
- The ARM programmer's model
- ARM development tools

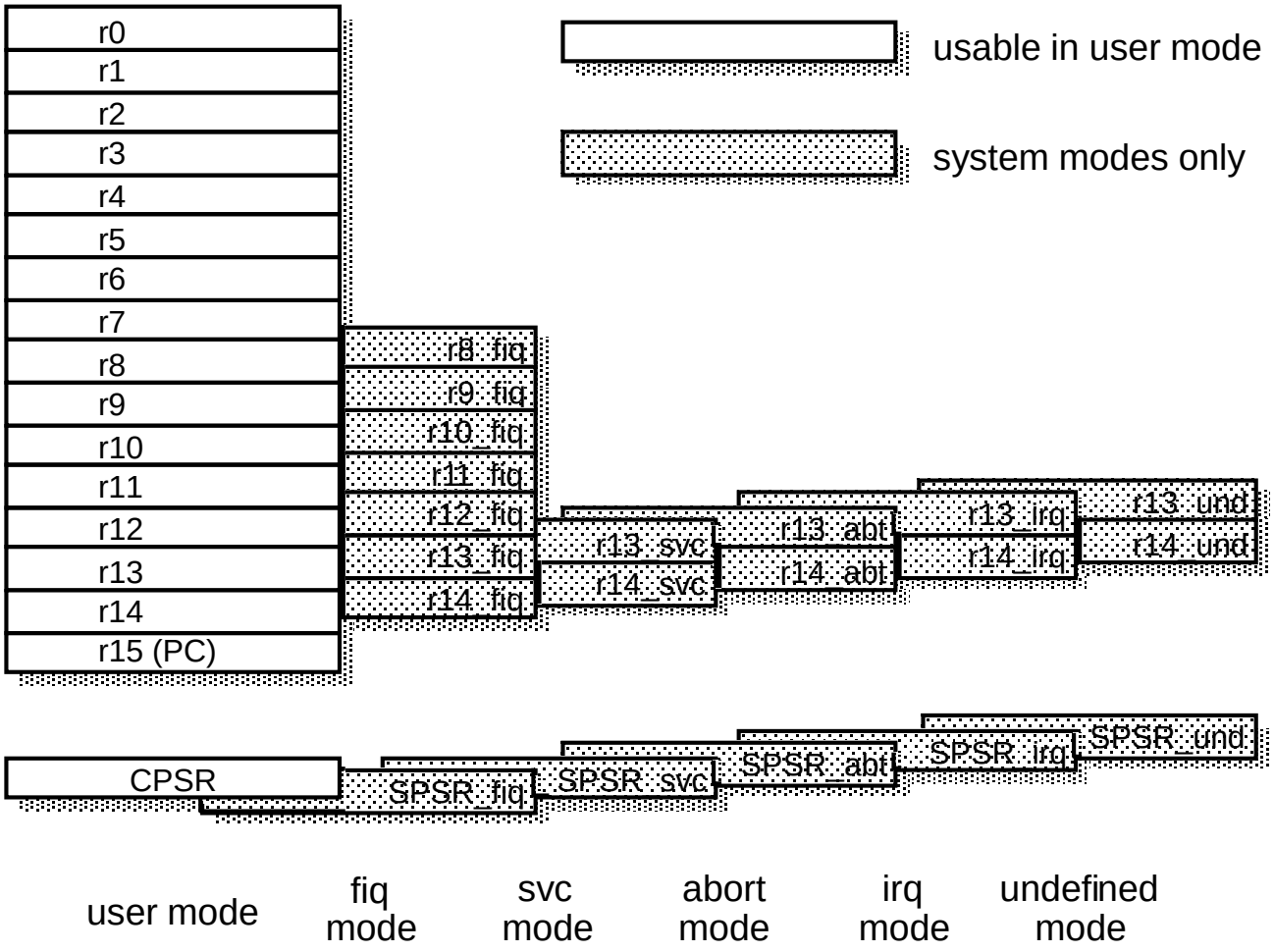
Data Sizes and Instruction Sets

- 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/Thumb2 Instruction Set
- Jazelle cores can also execute Java bytecode

Processor Modes

- The ARM has **seven** basic operating modes:
  - **User** : unprivileged mode under which most tasks run
  - **FIQ** : entered when a high priority (fast) interrupt is raised
  - **IRQ** : entered when a low priority (normal) interrupt is raised
  - **Supervisor** : entered on reset and when a software interrupt instruction is executed
  - **Abort** : used to handle memory access violations
  - **Undef** : used to handle undefined instructions
  - **System** : privileged mode using the same registers as user mode

ARM's Visible Registers



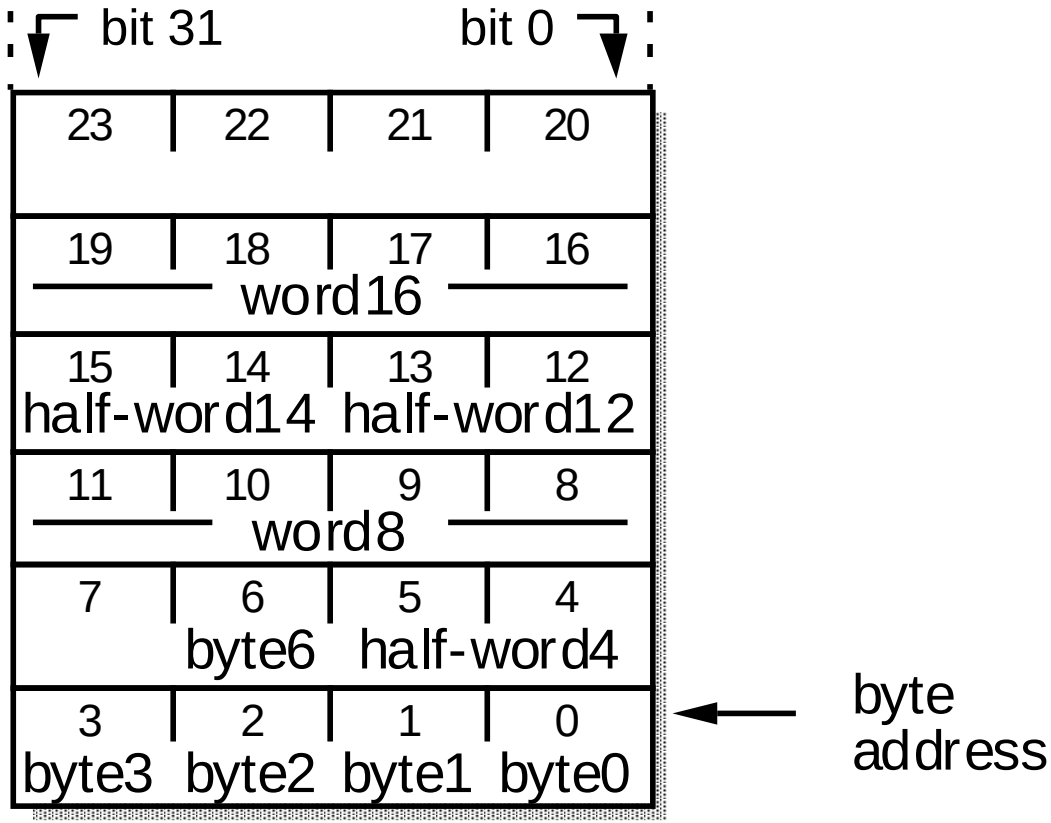
The Registers

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

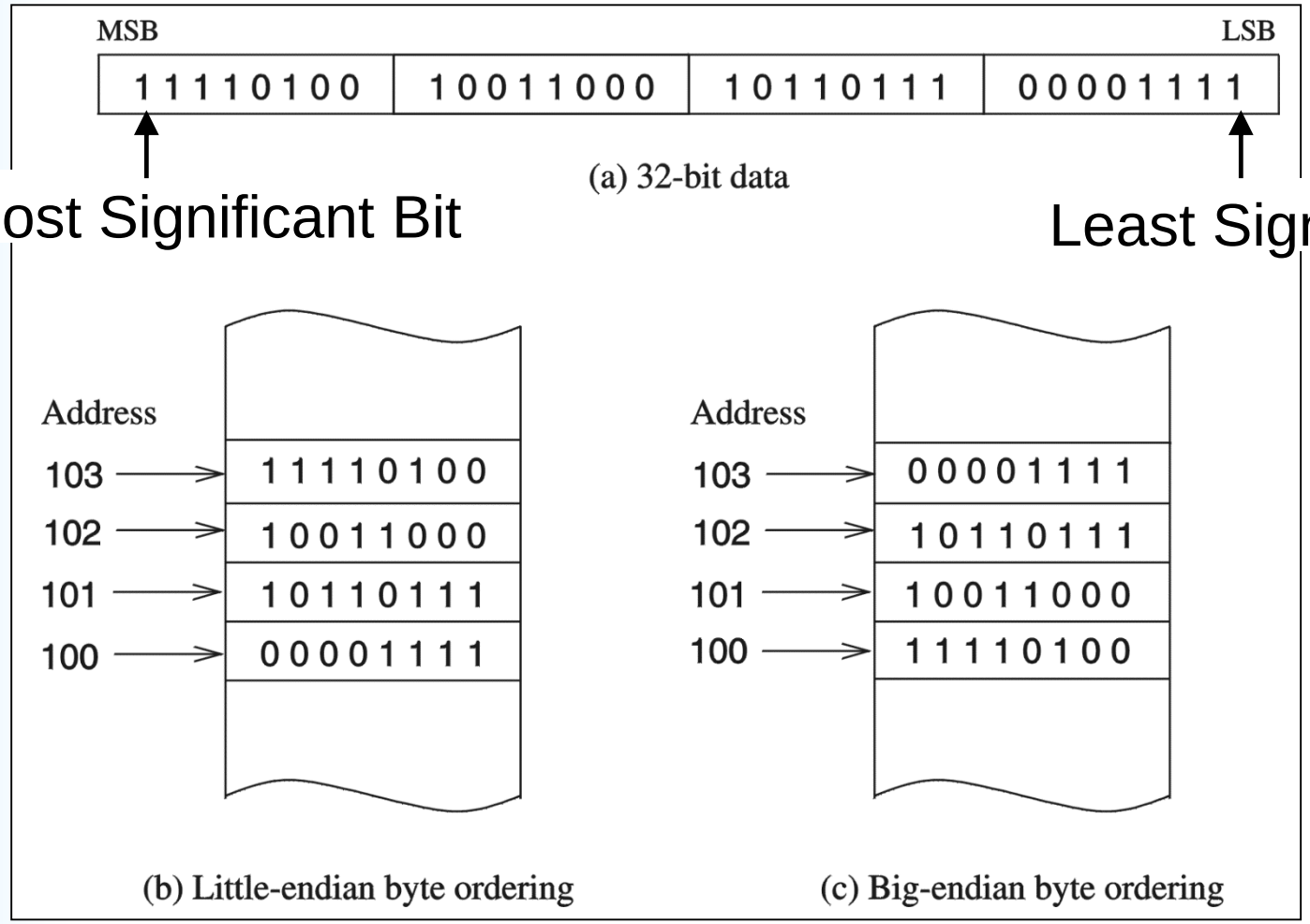
- The current processor mode governs which of several banks is accessible. Each mode can access
  - a particular set of r0-r12 registers
  - a particular r13 (the stack pointer, sp) and r14 (the link register, lr)
  - the program counter, r15 (pc)
  - the current program status register, cpsr

- Privileged modes (except System) can also access
  - a particular spsr (saved program status register)

ARM Memory Organization



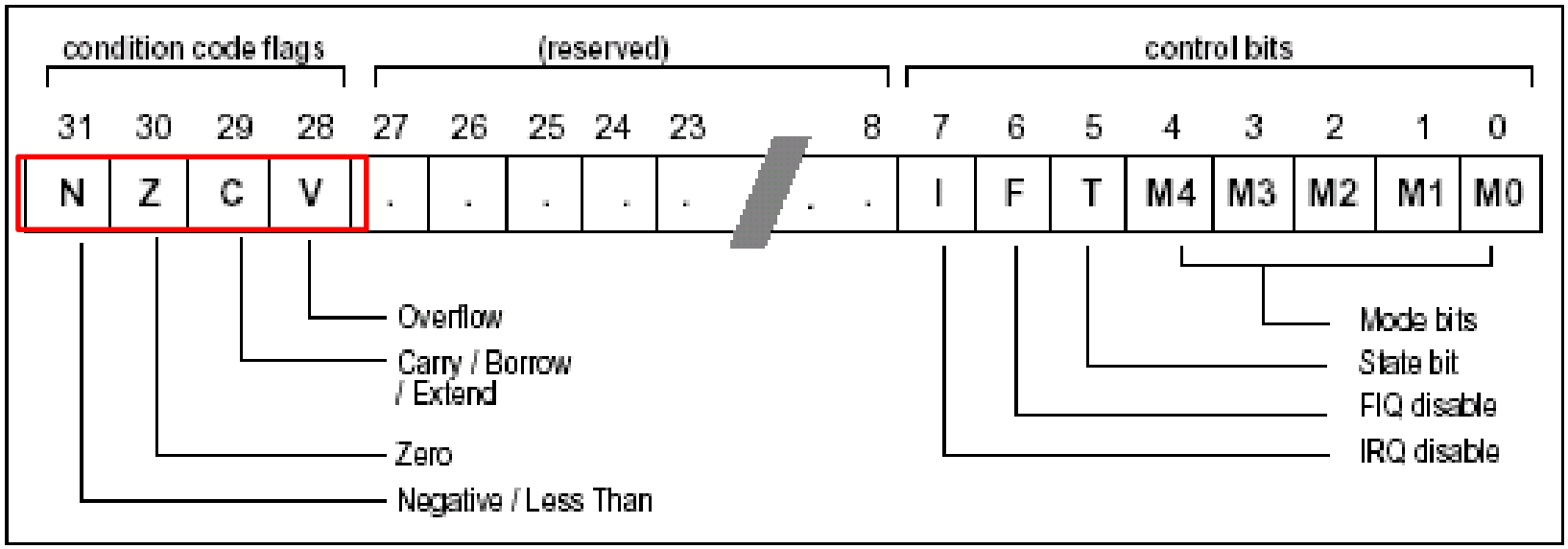
Big Endian and Little Endian (1)



Big Endian and Little Endian (2)

- The standard memory organization used by the ARM
  - **Little-endian (default)**
- ARM can also be configured to work with a "big-endian" memory organization

Current Program Status Registers (CPSR)



- hold information about the most recently performed ALU operation
- control the enabling and disabling of interrupts
- set the processor operating mode

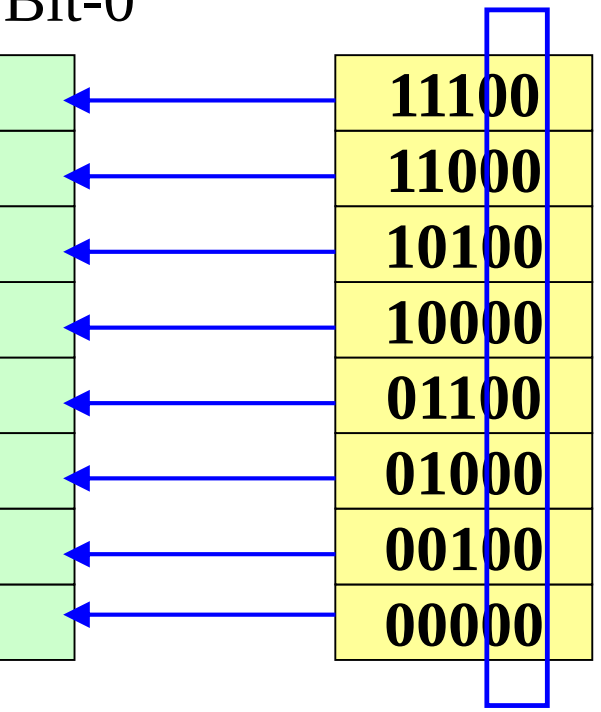
Program Counter (r15)

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

Program Counter (r15)

Bit-31 Bit-0

31	30	29	**28**
27	26	25	**24**
23	22	21	**20**
19	18	17	**16**
15	14	13	**12**
11	10	9	**8**
7	6	5	**4**
3	2	1	**0**



ARM 命名規則

- ARM **{x} {y} {z} {T} {D} {M} {I} {E} {J} {F} {-S}**
  - x: 系列
  - y: 記憶體管理/保護單元
  - z: cache
  - T: Thumb decoder
  - D: JTAG debugger
  - M: 快速乘法器
  - I: 嵌入式追蹤巨集單元
    - 建立在處理器內部用來設置中斷點和觀察點的除錯硬體
  - E: 增強指令 (基於TDMI)
  - J: Jazelle
  - F: 向量浮點單元
  - S: 可合成版本
    - 處理器核心以原始碼形式提供，易用於EDA tools

Outline

- The Acorn RISC Machine
- Architectural inheritance
- The ARM programmer's model
- ARM development tools

Main Components in ADS (1)

- ANSI C compilers – armcc and tcc
- ISO/Embedded C++ compilers – armcpp and tcpp
- ARM/Thumb assembler - armasm
- Linker - armlink
- Project management tool for windows - CodeWarrior
- Instruction set simulator - ARMulator
- Debuggers - AXD, ADW, ADU and armsd
- Format converter - fromelf
- Librarian – armar
- ARM profiler - armprof

ADS: ARM Developer Suite

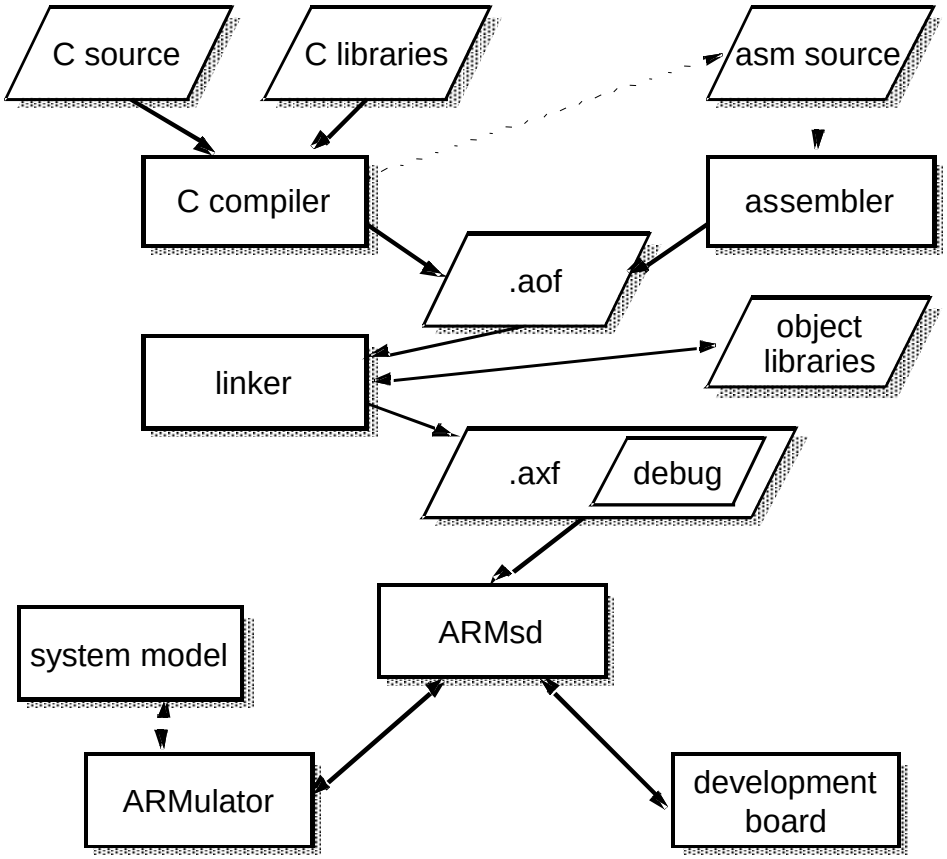
Main Components in ADS (2)

- C and C++ libraries
- Real Time Debug and Trace support
- Support for all ARM cores and processors including ARM9E, ARM10, Jazelle, StrongARM and Intel Xscale

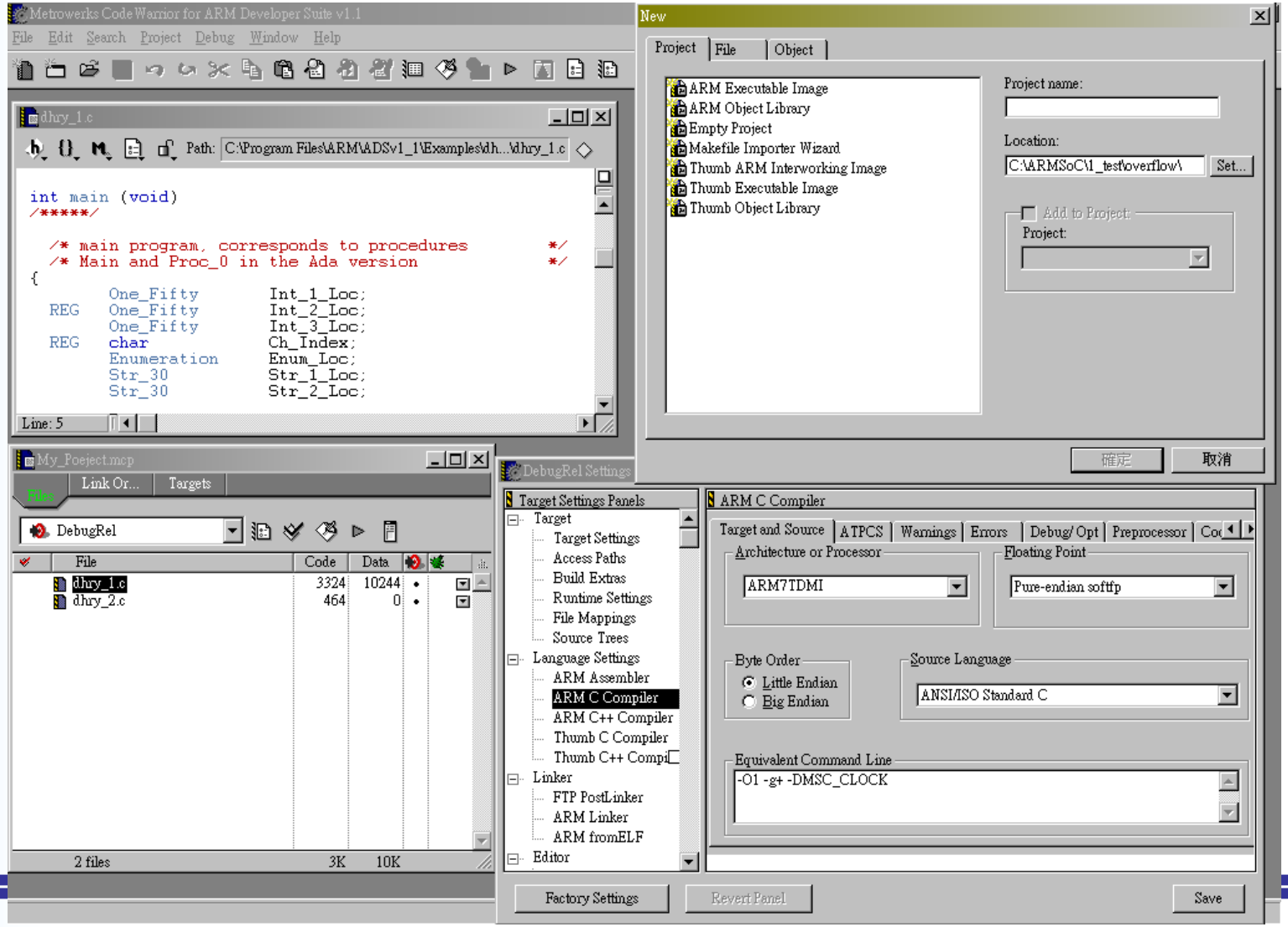
ARMulator

- It models the behaviour of various ARM processor cores in software on host system.
- Various levels of modeling accuracy:
  - Instruction-accuracy
    - give the exact behaviour of the system state
  - Cycle-accuracy
    - Give the exact behaviour of the processor on a cycle-by-cycle basis
  - Timing-accuracy
    - Present signals at the correct time within a cycle, allowing logic delays to be accounted for

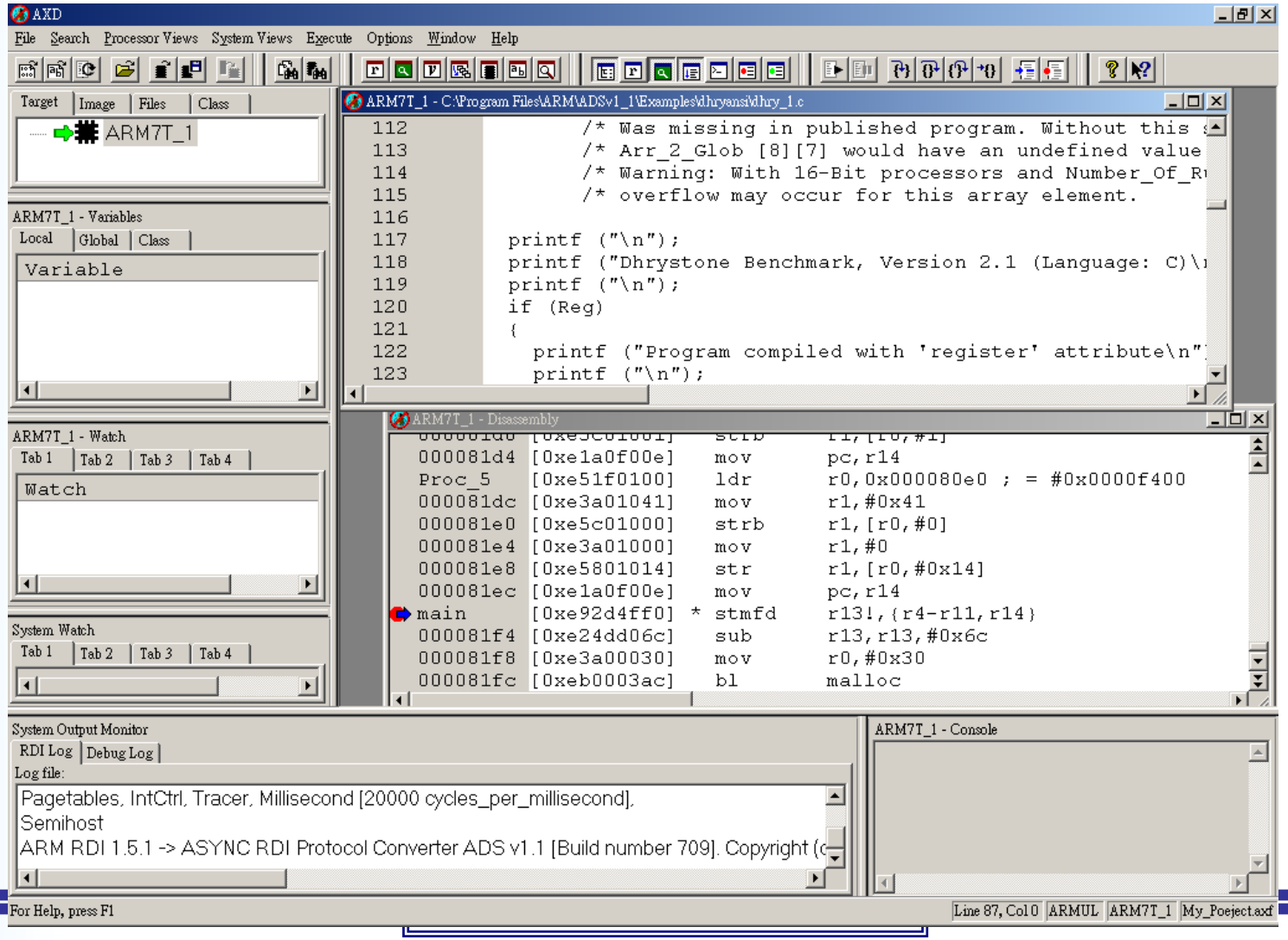
The Structure of ARM Cross-Development Toolkit



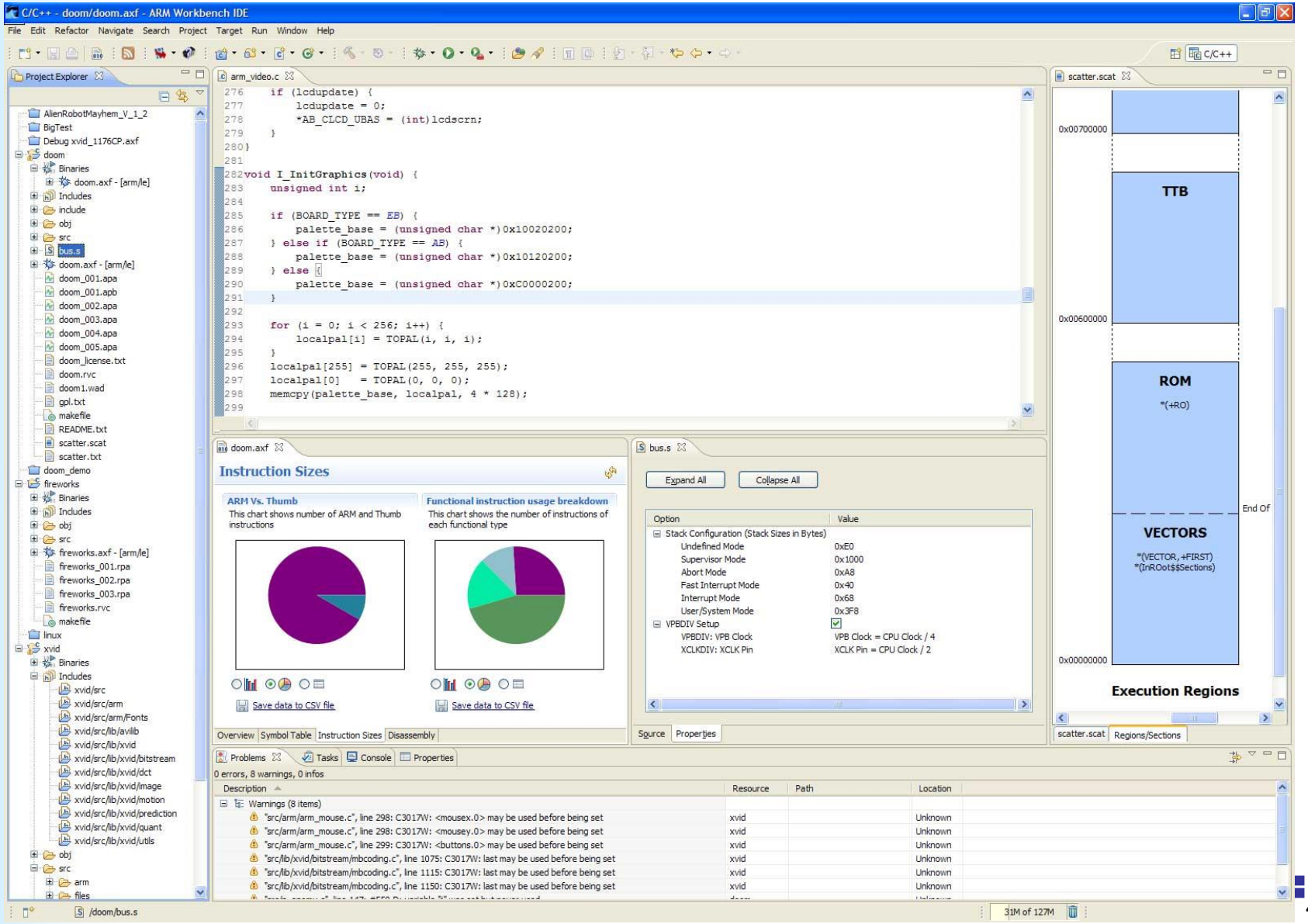
CodeWarrior IDE



AXD


AXD
File Search Processor Views System Views Execute Options Window Help
Target Image Files Class
ARM7T_1
ARM7T_1 - Variables
Local Global Class
Variable
ARM7T_1 - Watch
Tab 1 Tab 2 Tab 3 Tab 4
Watch
System Watch
Tab 1 Tab 2 Tab 3 Tab 4
ARM7T_1 - C:\Program Files\ARM\ADSv1_1\Examples\dhryansi\dhry_1.c
112 /* Was missing in published program. Without this
113 /* Arr_2_Glob [8][7] would have an undefined value
114 /* Warning: With 16-Bit processors and Number_Of_R
115 /* overflow may occur for this array element.
116
117 printf ("\n");
118 printf ("Dhrystone Benchmark, Version 2.1 (Language: C)\
119 printf ("\n");
120 if (Reg)
121 {
122 printf ("Program compiled with 'register' attribute\n"
123 printf ("\n");
ARM7T_1 - Disassembly
000081d4 [0xe1a0f00e] mov pc,r14
Proc_5 [0xe51f0100] ldr r0,0x000080e0 ; = #0x0000f400
000081dc [0xe3a01041] mov r1,#0x41
000081e0 [0xe5c01000] strb r1,[r0,#0]
000081e4 [0xe3a01000] mov r1,#0
000081e8 [0xe5801014] str r1,[r0,#0x14]
000081ec [0xe1a0f00e] mov pc,r14
main [0xe92d4ff0] * stmfd r13!,{r4-r11,r14}
000081f4 [0xe24dd06c] sub r13,r13,#0x6c
000081f8 [0xe3a00030] mov r0,#0x30
000081fc [0xeb0003ac] bl malloc
System Output Monitor
RDI Log Debug Log
Log file:
Pagetables, IntCtrl, Tracer, Millisecond [20000 cycles_per_millisecond],
Semihost
ARM RDI 1.5.1 -> ASYNC RDI Protocol Converter ADS v1.1 [Build number 709]. Copyright (
ARM7T_1 - Console
For Help, press F1
Line 87, Col 0 ARMUL ARM7T_1 My_Poeject.axf

ARM RealView



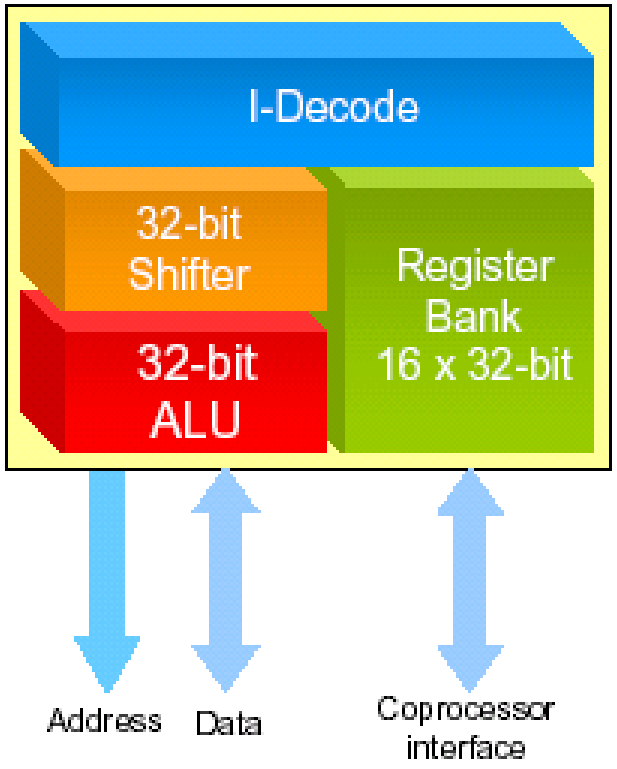
Next

- **Study 32-bit ARM instruction set**
- **Target: ARM v4T**

Backup

Architectural Inheritance

World's best-selling embedded RISC*



- 32-bit Load/Store RISC Architecture
  - 3-operand instructions (ARM)
  - 2-operand instructions (Thumb)
- 16-word register bank
- 32-bit barrel shifter & ALU
- Thumb enabled
- Co-processor Interface
  - implement user-defined instructions
- Fast Interrupt Response
- Extensive exception handling
  - interrupts
  - memory aborts
  - undefined instruction trap
  - software interrupt
- User & Supervisor Modes

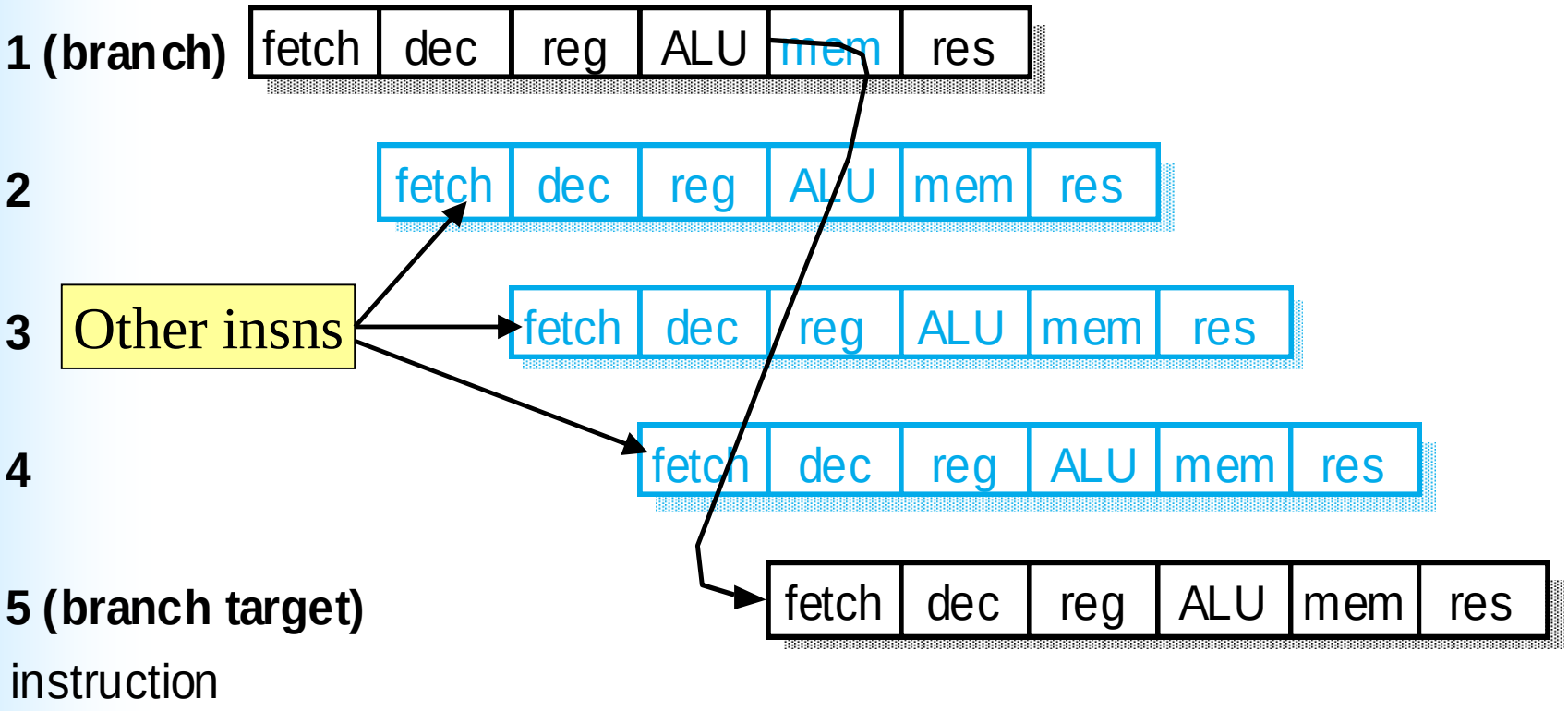
*Based on 1999 sales.
Source: Inside the New Computer Industry.

Features Rejected (1)

- **Register windows**
  - Berkeley RISC processors incorporated a large number of register, 32 of which were visible at any time
  - Ex: Sun SPARC architecture
  - Drawback
    - Large chip area occupied by the large number of registers

Features Rejected (2)

- **Delayed branches**



Features Rejected (3)

- Drawback on delayed branches
  - Remove the atomicity of individual instructions
  - Work well on single issue pipelined processors, but not scale well to super-scalar implementations
  - Interact badly with branch prediction mechanism
  - Make exception handling more complex

Features Rejected (4)

- Single-cycle execution of all instructions
  - ARM executes most data processing instructions in a single clock cycle
  - Many other instructions take multiple clock cycles
  - ARM was designed to use the minimum of cycles required for memory accesses
    - If possible, the extra cycles were used to do something useful
    - ex: auto-indexing addressing mode

Notable Features of ARM Instruction Set (1)

- The load-store architecture
- 3-address data processing instructions
  - Two source operands and the result operand are all independently specified
- Conditional execution of every instruction
- The inclusion of every powerful load and store multiple register instructions

Notable Features of ARM Instruction Set (2)

- The ability to perform a general shift operation and a general ALU operation in a single instruction that executes in a single clock cycle
- Open coprocessor instruction set extension
- A very dense 16-bit compressed representation of the instruction set in the Thumb architecture