Introduction to ARM

Module 4: Advanced Concepts 8051 Timers/Counters - Modes and Applications. Serial Data Transfer – SFRs of serial port, working, Programming the 8051 to transfer data serially. Introduction to ARM - ARM family, ARM 7 register architecture. ARM programmer's model. System software Assembler, Interpreter, Compiler, Linker, Loader, Debugger.

SAMSUNG
Exynos

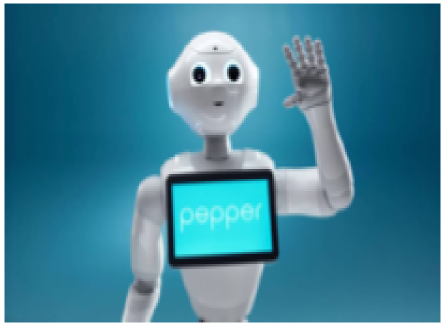
MediaTek
helio G35

A14

ARM(Advanced RISC Machine)

- The ARM processor is a Reduced Instruction Set Computer (RISC).
- The ARM was originally developed at Acorn Computers Limited of Cambridge, England, between 1983 and 1985.
- Established to meet the requirement for low-power and cost-sensitive embedded applications.
- The ARM is supported by a toolkit which includes an instruction set emulator for hardware modeling and software testing and benchmarking, an assembler, C and C++ compilers, a linker and a symbolic debugger.

A Little about ARM – The company



- Originally Acorn RISC Machine (ARM)
- Later Advanced RISC Machine
- Then it became ARM Ltd owned by ARM Holdings (parent company)
- In 2016 SoftBank bought ARM for $31 billion
- ARM:
  - Develops the architecture and licenses it to other companies
  - Other companies design their own products that implement one of those architectures—including systems-on-chips (SoC) and systems-on-modules (SoM) that incorporate memory, interfaces, radios, etc.
  - It also designs cores that implement this instruction set and licenses these designs to a number of companies that incorporate those core designs into their own products.
- ARM Processors
  - RISC based processors
  - In 2010 alone, **6.1 billion** ARM-based processor, representing **95%** of smartphones, **35%** of digital televisions and set-top boxes and **10%** of mobile computers
  - over 100 billion ARM processors produced as of 2017
  - The most widely used instruction set architecture in terms of quantity produced

ARM…

- ARM is a most widely used 32-bit instruction set architecture.
- Relative simplicity makes it suitable for lower power devices
- ARM-7, ARM-9, ARM-11 and CORTEX are the different families of ARM processor
- Used extensively in consumer electronics including, mobile phones, digital media and music players, handheld game consoles, calculators etc

ARM…

- The first ARM chip was developed from the foundation of RISC machines of which some of the features they accepted and some of them rejected.
- Those that were used were
  - **a load-store architecture**
  - **fixed-length 32-bit instructions**
  - **3-address instruction formats**.

ARM…

- The features that were employed on the Berkeley RISC designs which were rejected by the ARM designers were:
  - **Register windows**
  - **Delayed branches**
  - **Single-cycle execution of all instructions**

ARM…

- ARM Limited has incorporated a novel mechanism called the Thumb architecture, into some version of the ARM processor
- The Thumb instruction set is a 16-bit compressed from of the original 32-bit ARM instruction set.
- Employs dynamic decompression hardware in the instruction pipeline.
- Thumb code density is better than that achieved by most CISC processors.

Main features of the ARM architecture

- A large set of registers, all of which can be used for most purposes
- A load-store architecture
- 3-address instructions (that is, the two source operand registers and the result register are all independently specified)
- Conditional execution of every instruction
- The inclusion of very powerful load and store multiple register instructions
- 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 instruction set extension through the coprocessor instruction set, including adding new registers and data types to the programmer's model.
- If the Thumb instruction set is considered part of the ARM architecture, we could also add:
  - a very dense 16-bit compressed representation of the instruction set in the Thumb architecture

Features of ARM…

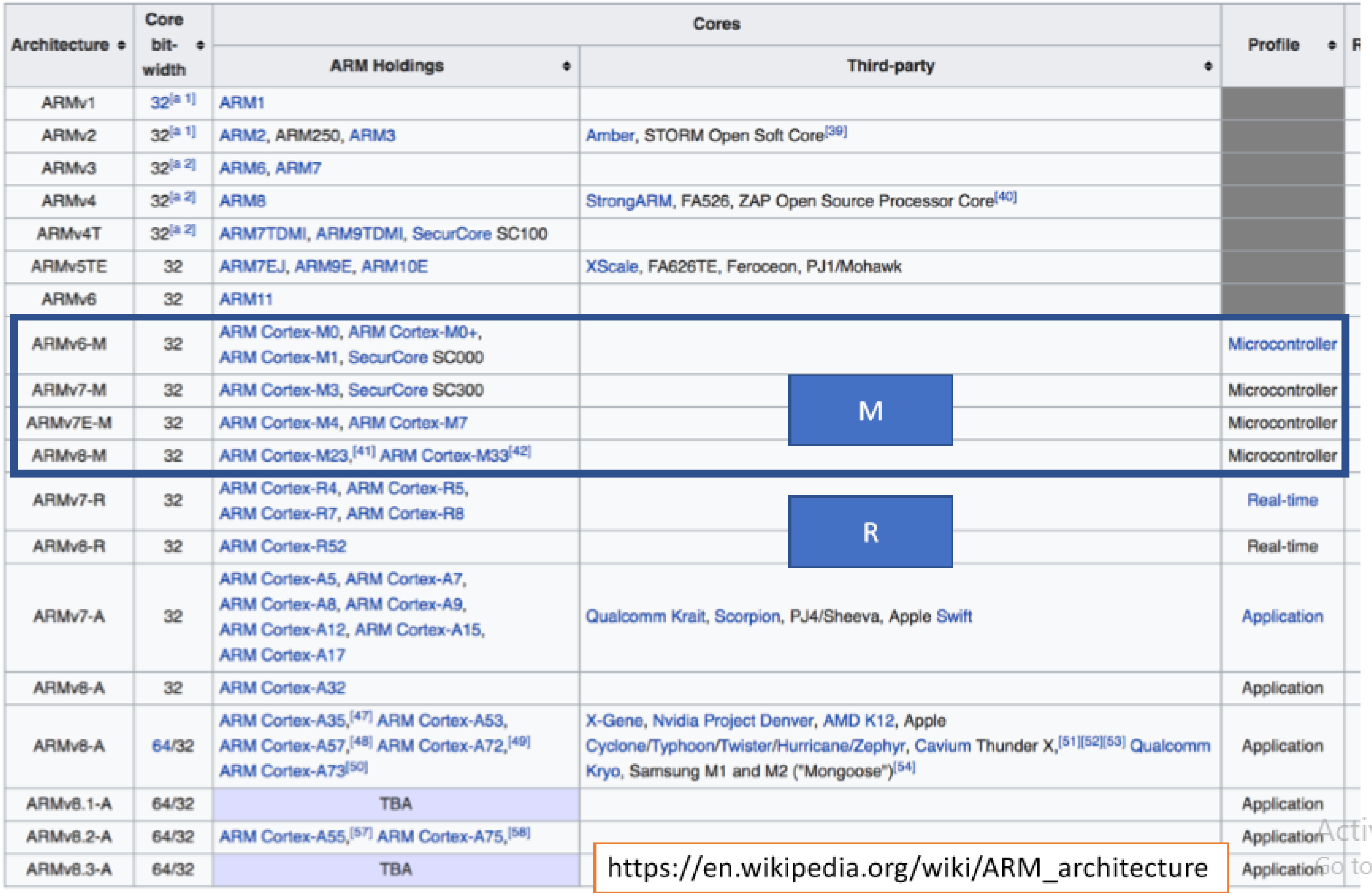
- ARM Processors have a good speed of execution to power consumption ratio.
- They have a wide range of clock frequency ranging from 1MHz to few GHz.
- They support direct execution of Java byte codes using ARM’s Java Jazelle DBX.
- ARM Processors have built in hardware for debugging.
- Supports enhanced instructions for DSP operations.
- High Performance
- Low Power Consumption

- ARM has 3 instruction set states
  - 32 bit ARM instruction set
  - 16 bit Thump instruction set
  - 8 bit Jazelle instruction set
- ARM – 32 bit load-store architecture with every instruction being conditional
- Thump – 16 bit with only branch instruction being conditional and only half of the registers used.
- Jazelle – Allows java byte code to be directly executed in ARM architecture – improves performance by 5x-10x

7 Basic Processor Modes

- **Supervisor** – Entered on RESET and when a software interrupt instruction is executed
- **User** – Unprivileged mode under which most of the task run
- **System** – A privileged user mode for the operating system
- **FIQ** – Entered when a high priority interrupt is raised
- **IRQ** – Entered when a low priority is raised
- **Abort** – Used to handle memory access violations
- **Undef** – Used to handle undefined instructions

- The letters or words after “ARM” are used to indicate the features of a processor
- *T – **T**humb Instruction Set*
- *D – JTAG **D**ebug*
- *M – Fast **M**ultiplier*
- *I – Embedded **I**CE Macrocell*
- E – **E**nhanced Instructions
- J – **J**azelle
- F – Vector **F**loating-point Unit
- S – **S**ynthesizable
- **ARM7TDMI-S architecture based LPC2148 Processor.**



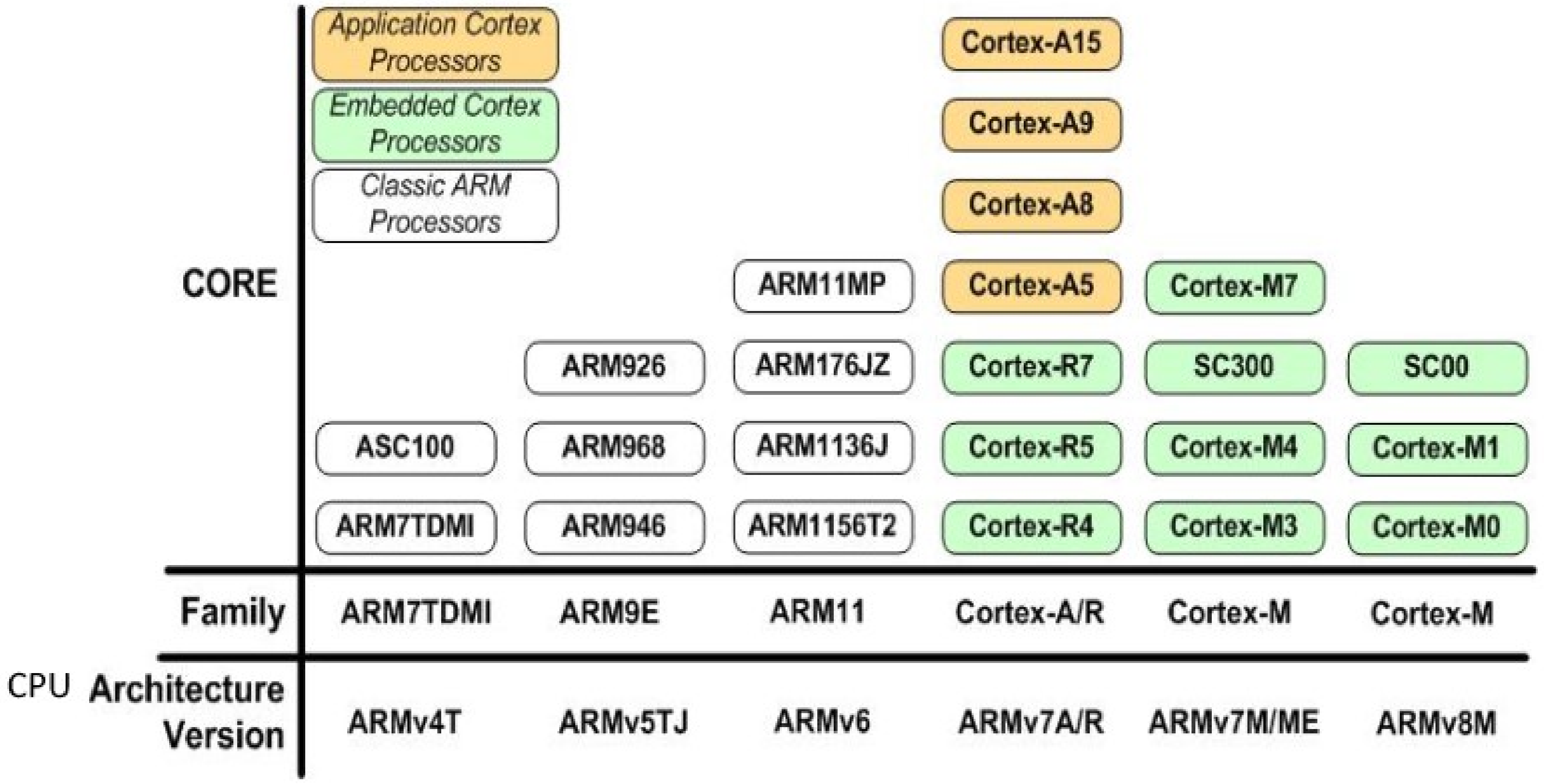
Architecture	Core bit-width	Cores		Profile	R
		ARM Holdings	Third-party		
ARMv1	32[a 1]	ARM1		▨	
ARMv2	32[a 1]	ARM2, ARM250, ARM3	Amber, STORM Open Soft Core[39]	▨	
ARMv3	32[a 2]	ARM6, ARM7		▨	
ARMv4	32[a 2]	ARM8	StrongARM, FA526, ZAP Open Source Processor Core[40]	▨	
ARMv4T	32[a 2]	ARM7TDMI, ARM9TDMI, 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	
ARMv8-M	32	ARM Cortex-M23,[41] ARM Cortex-M33[42]		Microcontroller	
ARMv7-R	32	ARM Cortex-R4, ARM Cortex-R5, ARM Cortex-R7, ARM Cortex-R8		Real-time	
ARMv8-R	32	ARM Cortex-R52		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	Qualcomm Krait, Scorpion, PJ4/Sheeva, Apple Swift	Application	
ARMv8-A	32	ARM Cortex-A32		Application	
ARMv8-A	64/32	ARM Cortex-A35,[47] ARM Cortex-A53, ARM Cortex-A57,[48] ARM Cortex-A72,[49] ARM Cortex-A73[50]	X-Gene, Nvidia Project Denver, AMD K12, Apple Cyclone/Typhoon/Twister/Hurricane/Zephyr, Cavium Thunder X,[51][52][53] Qualcomm Kryo, Samsung M1 and M2 ("Mongoose")[54]	Application	
ARMv8.1-A	64/32	TBA		Application	
ARMv8.2-A	64/32	ARM Cortex-A55,[57] ARM Cortex-A75,[58]		Application	
ARMv8.3-A	64/32	TBA		Application	

M

R

https://en.wikipedia.org/wiki/ARM_architecture

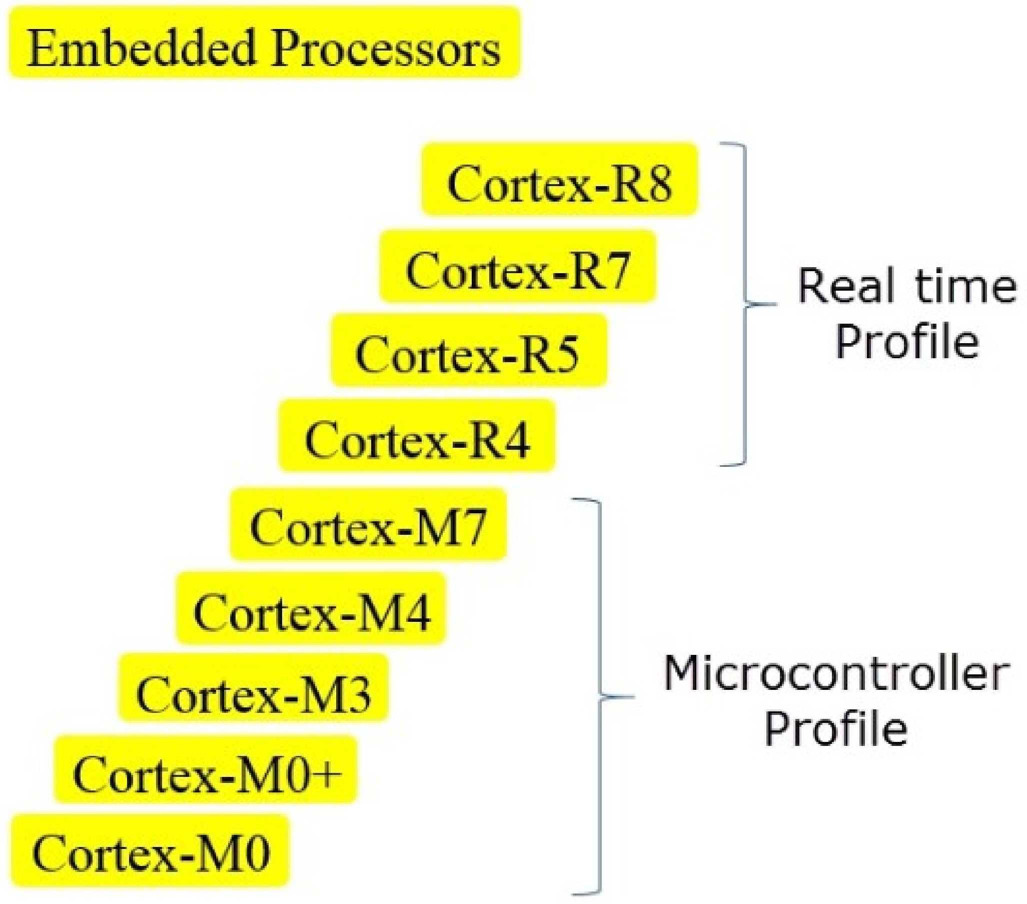
ARM Family and Architecture



- 🟧 *Application Cortex Processors*
- 🟩 *Embedded Cortex Processors*
- ⬜ *Classic ARM Processors*

CORE				Cortex-A15		
				Cortex-A9		
				Cortex-A8		
			ARM11MP	Cortex-A5	Cortex-M7	
		ARM926	ARM176JZ	Cortex-R7	SC300	SC00
	ASC100	ARM968	ARM1136J	Cortex-R5	Cortex-M4	Cortex-M1
	ARM7TDMI	ARM946	ARM1156T2	Cortex-R4	Cortex-M3	Cortex-M0
Family	ARM7TDMI	ARM9E	ARM11	Cortex-A/R	Cortex-M	Cortex-M
CPU **Architecture Version**	ARMv4T	ARMv5TJ	ARMv6	ARMv7A/R	ARMv7M/ME	ARMv8M

- ARM Processors family can be divide into
  - ARM Classic Processors
  - ARM Embedded Processors
  - ARM Application Processors
- ARM Classic processors include ARM7, ARM9 and ARM11 families and ARM7TDMI
- ARM7 based processors are still used in many small and simple 32-bit devices


Embedded Processors
Cortex-R8
Cortex-R7
Cortex-R5
Cortex-R4
Real time Profile
Cortex-M7
Cortex-M4
Cortex-M3
Cortex-M0+
Cortex-M0
Microcontroller Profile

Application Processors

High Performance	High Efficiency	Ultra-High Efficiency
Cortex-A73	Cortex-A53	Cortex-A35
Cortex-A72	Cortex-A9	Cortex-A32
Cortex-A57	Cortex-A8	Cortex-A7
Cortex-A17		Cortex-A5

ARM Cortex Processors

- ARM Cortex-A family:
  - **A**pplications processors
  - Support OS and high-performance applications
  - Such as Smartphones, Smart TV
- ARM Cortex-R family:
  - **R**eal-time processors with high performance and high reliability
  - Support real-time processing and mission-critical control
- ARM Cortex-M family:
  - **M**icrocontroller
  - Cost-sensitive, support SoC

Cortex - M

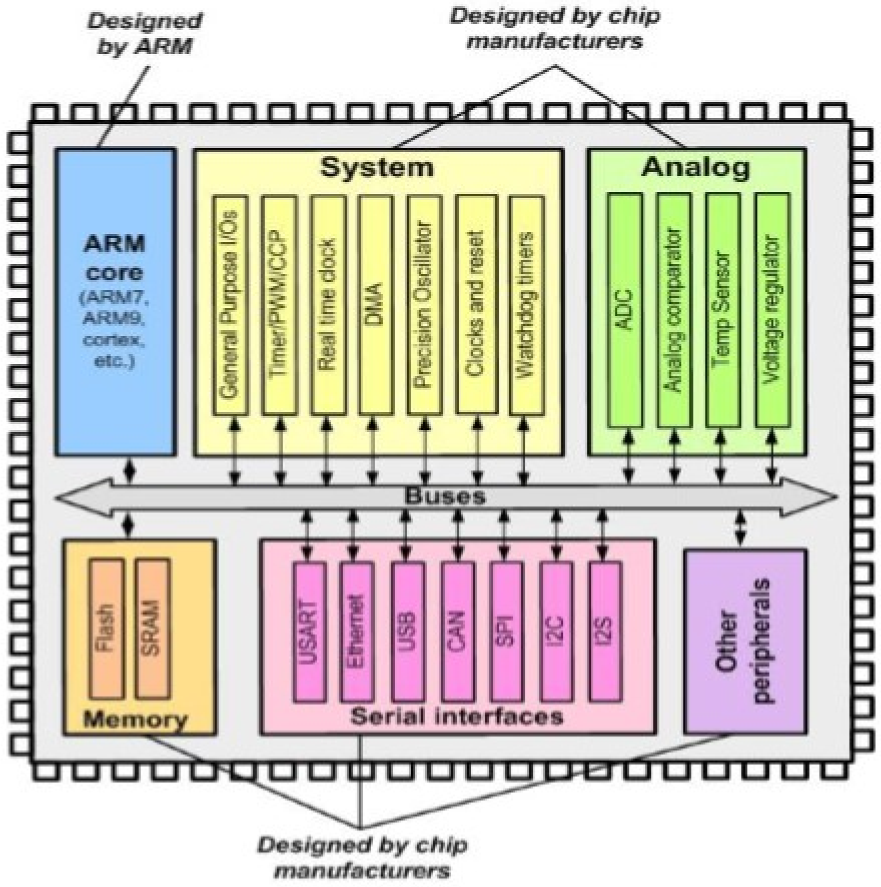
- Cortex-M is a great trade-off between performance, cost, efficiency;
- Used for IoT, various applications.
- Has on-chip peripherals
- Core is licensed by ARM

CORTEX-M: CORE + Peripherals

- Core
  - Memory
    - FLASH: Non-Volatile / Instruction memory
      SRAM/DRAM: Volatile / data memory
  - Processor
    - ALU
    - Processor Control Unit (CPU)
    - Registers
      - Special Purpose Registers
      - General Purpose Registers
  - Buses
    - Data Bus
    - Instruction Bus
    - Bus bridge to connect diff. buses
    - Advanced High-performance Bus (AHB)
    - Advanced Peripheral Bus (APB)
  - GPIO

- Peripherals
  - ADC
  - LCD Controller
  - SPI
  - I2C
  - Etc.

ARM Simplified Block Diagram System on Chip(SoC)



Applications

iPhone 5 Teardown

The A6 processor is the first Apple System-on-Chip (SoC) to use a custom design, based off the **ARMv7** instruction set.

iPhone 6 Teardown

The A8 processor is the first 64-bit ARM based SoC. It supports **ARM A64, A32, and T32** instruction set.

iPhone 7 Tea n

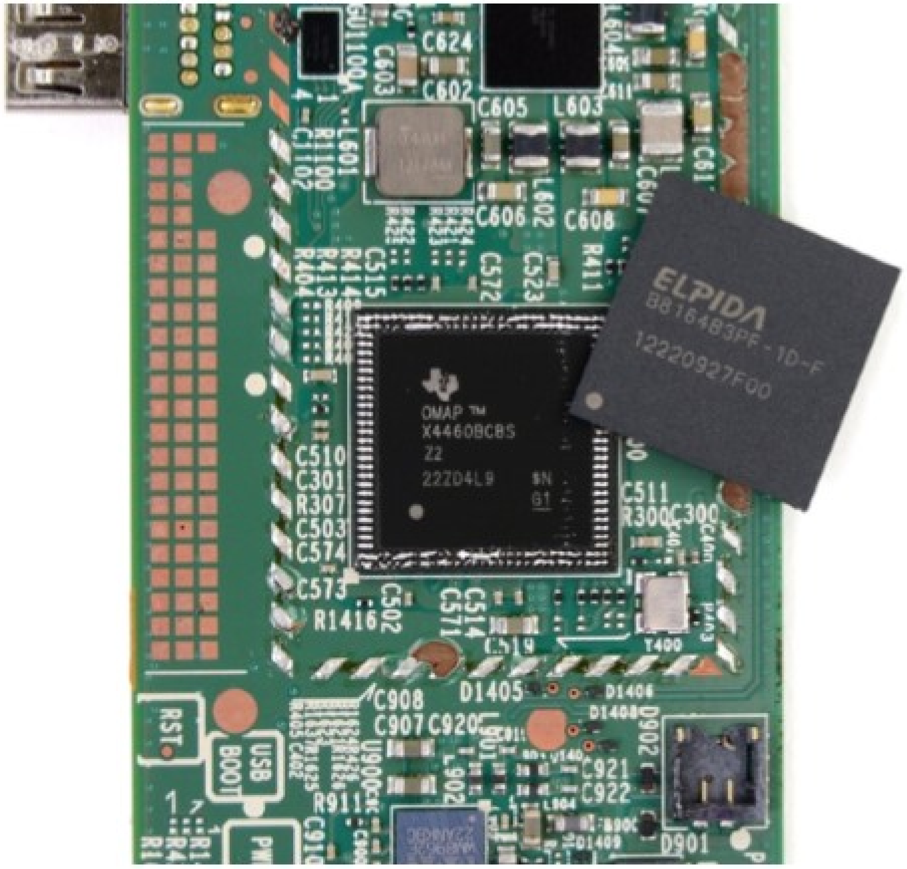
A10 processor:

- 64-bit system on chip (SoC)
- **ARMv**8-A core

Apple Watch

- Apple S1 Processor
  - **32-bit ARMv7-A** compatible
  - # of Cores: **1**
  - CMOS Technology: 28 nm
  - L1 cache 32 KB data
  - L2 cache 256 KB
  - GPU PowerVR SGX543

Kindle HD Fire

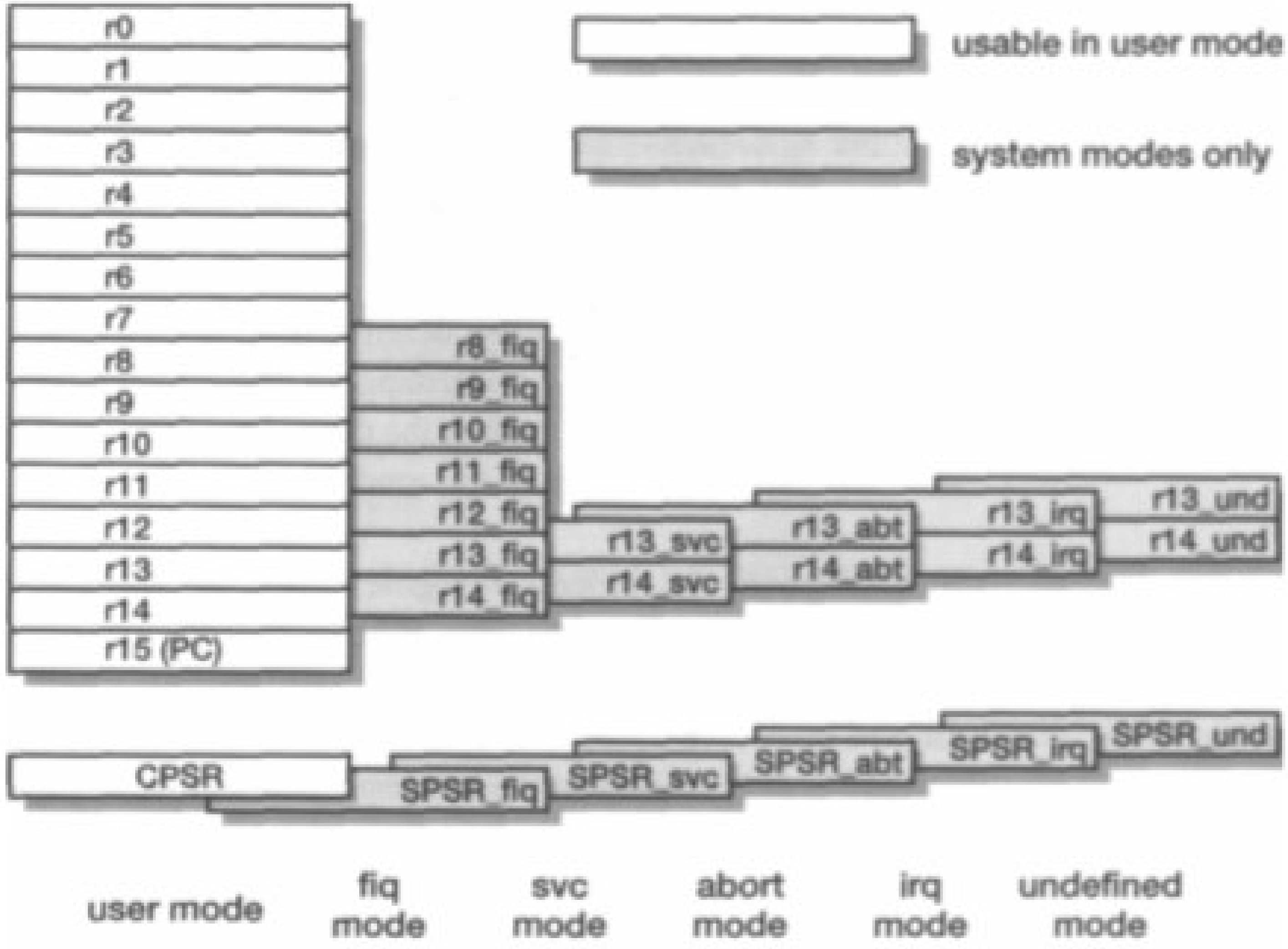


Texas Instruments **OMAP 4460** dual-core processor

The ARM programmer's model

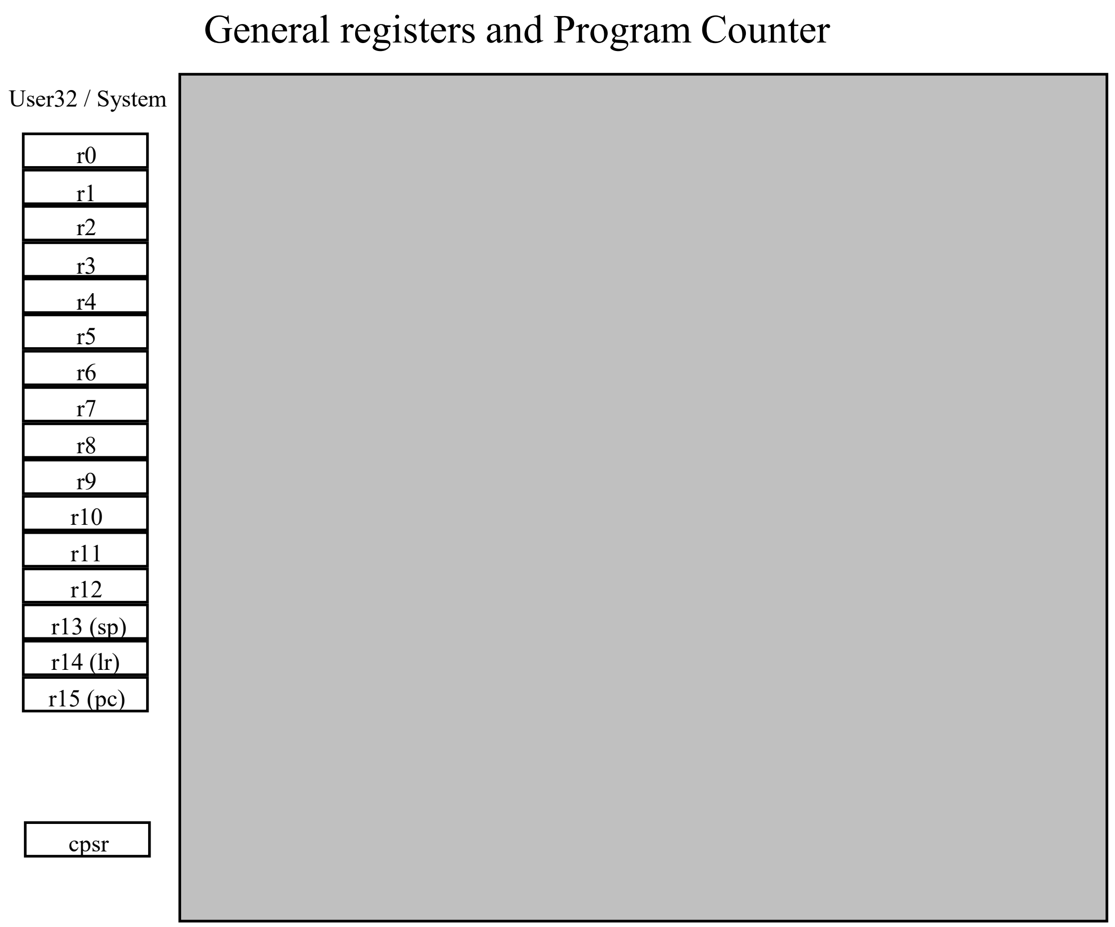
- A processor's instruction set defines the operations that the programmer can use to change the state of the system incorporating the processor.
- Total available registers in ARM processors is divided into visible registers and invisible registers.
- This state usually comprises the values of the data items in the processor's visible registers and the system's memory.
- Each instruction can be viewed as performing a defined transformation from the state before the instruction is executed to the state after it has completed.
- Although a processor will typically have many invisible registers involved in executing an instruction, the values of these registers before and after the instruction is executed are not significant; only the values in the visible registers have any significance

- Programmers model of a processer help the programmer to use the core component of the processor to program it.
- Programmers model of ARM processor includes the details of the main core, registers of the processor, operating modes, and instruction set.
- Understanding programmers model is important for the reusability of the processor and software development tools.



ARM's visible registers.

Register Organisation



- ARM register set includes 37 registers of each 32 bit long.
  - 1 dedicated program counter
  - 1 dedicated current program status register
  - 5 dedicated saved program status registers.
  - 30 general purpose registers.
  - Out of 37, 20 registers are hidden from program at different times. These registers are called banked registers

- The usability of the registers is depend on the mode the processor is currently entered in
- When writing user-level programs, only the 15 general purpose 32-bit registers(r0 to r14), the program counter (r15) and the current program status register(CPSR) need to be considered
- The remaining registers are used only for system-level programming and for handling exceptions (for example interrupts)

- 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)

- The ARM maintains a few separate/duplicate registers that are available only when the processor is operation in a certain mode.
- Modes have their own local/private SP,LR and PSR
- The FIQ mode also has separate r8-r12 registers.
- These duplicate registers (including the duplicate GPRs in FIQ mode) make mode and context switching faster and more efficient.
- Also help decrease the amount of work required to do proper context switching when changing modes

Registers r0 to r7

- Lower order registers
- Can be accessed by all 16bit thumb instructions and all 32bit RISC instructions
- 32 bit registers
- Default value will be unpredictable

Registers r8 to r12

- Higher orders registers
- Can be accessed by all 32 bit RISC instructions
- 32 bit registers
- Default values will be unpredictable

R13-Stack Pointer

- The processor uses SP as a pointer to the active stack
- In the thumb instruction set, most instructions cannot access SP. The only instructions that can access SP are those designed to use SP as stack pointer.
- The ARM instructions set provides more general access to the SP, and it can be used as a general-purpose registers.
- However, ARM deprecates the use of SP for any purpose other than as stack pointer

R14 – Link Register

- **LR, the link register**
  - A link register is a special-purpose register which holds the address to return to when a subroutine or function is called.
  - When software does not require the LR for linking, it can use it for other purposes.
  - If you have multiple call to different subroutine it is mandatory to save the content of LR
  - The common method is to push the LR to stack in the beginning of your subroutine

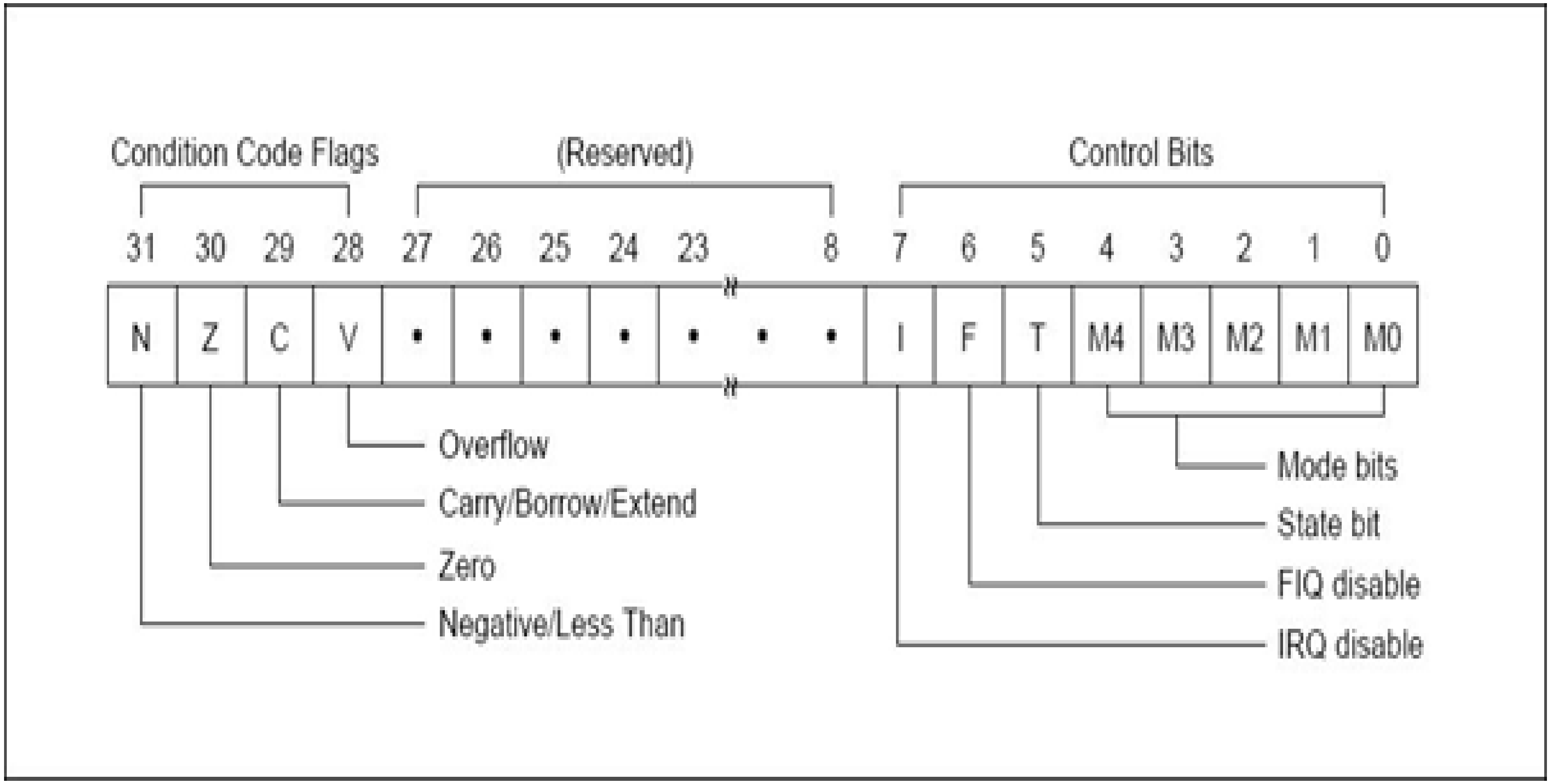
R15 – Program Counter

- The program counter , also referred to as r15
- When executing an ARM instruction, PC reads as the address of the current instruction plus 8
- When executing a Thumb instruction, PC reads as the address of the current instruction plus 4
- Writing an address to PC causes a branch to that address

CPSR

- The CPSR is used in user-level programs to store the condition code bits.
- Used to record the result of a comparison operation and to control whether or not a conditional branch is taken.
- The bits at LSB of the register control the processor mode, instruction set type and interrupt enables and are protected from change by the user-level program

- The condition code flags are in the top four bits of the register and have the following meanings:
- **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.


Condition Code Flags
(Reserved)
Control Bits
31
30
29
28
27
26
25
24
23
8
7
6
5
4
3
2
1
0
N
Z
C
V
I
F
T
M4
M3
M2
M1
M0
Overflow
Carry/Borrow/Extend
Zero
Negative/Less Than
Mode bits
State bit
FIQ disable
IRQ disable

M[4:0]	Mode	Accessible register set	
10000	User	PC, R14..R0	CPSR
10001	FIQ	PC, R14_fiq..R8_fiq, R7..R0	CPSR, SPSR_fiq
10010	IRQ	PC,R14_irq..R13_irq,R12..R0	CPSR, SPSR_irq
10011	Supervisor	PC,R14_svc..R13_svc,R12..R0	CPSR, SPSR_svc
10111	Abort	PC,R14_abt..R13_abt,R12..R0	CPSR, SPSR_abt
11011	Undefined	PC,R14_und..R13_und,R12..R0	CPSR, SPSR_und

Table 2: The Mode Bits

- In the application level view, an ARM processor has:
  - 13 general-purpose 32-bit registers, r0 to r12.
  - Three 32-bit registers with special uses, SP, LR and PC, that can be described as r13 to r15.
- The special registers are:
  - SP, the Stack Pointer
  - LR, the Link Register
  - PC, the Program Counter

The memory system

- The ARM7 is a Von Neumann, load/store architecture
- Only 32 bit data bus for both instruction and data
- Only the load/store instruction (and SWP) access memory.
- In addition to the processor register state, an ARM system has memory state.
- Memory may be viewed as a linear array of bytes numbered from zero up to 2^{32}-l.

- Data items may be 8-bit bytes, 16-bit half-words or 32-bit words.
- Words are always aligned on 4-byte boundaries (that is, the two least significant address bits are zero) and half-words are aligned on even byte boundaries.
- A small area of memory where each byte location has a unique number.
- A byte may occupy any of these locations.
- A word-sized data item must occupy a group of four byte locations starting at a byte address which is a multiple of four.
- Half-words occupy two byte locations starting at an even byte address.

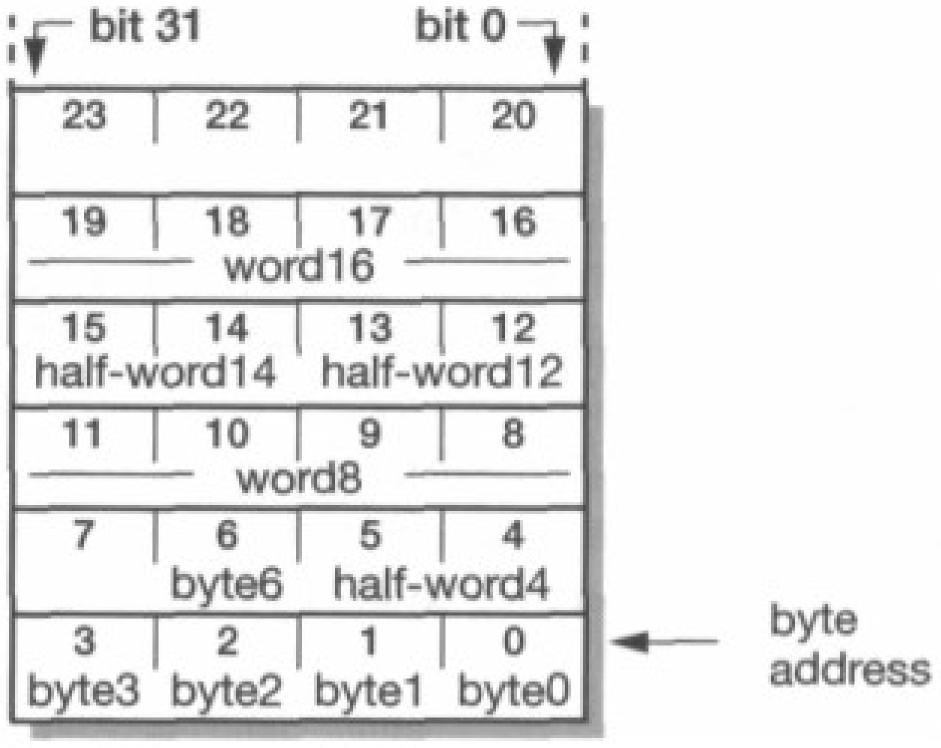
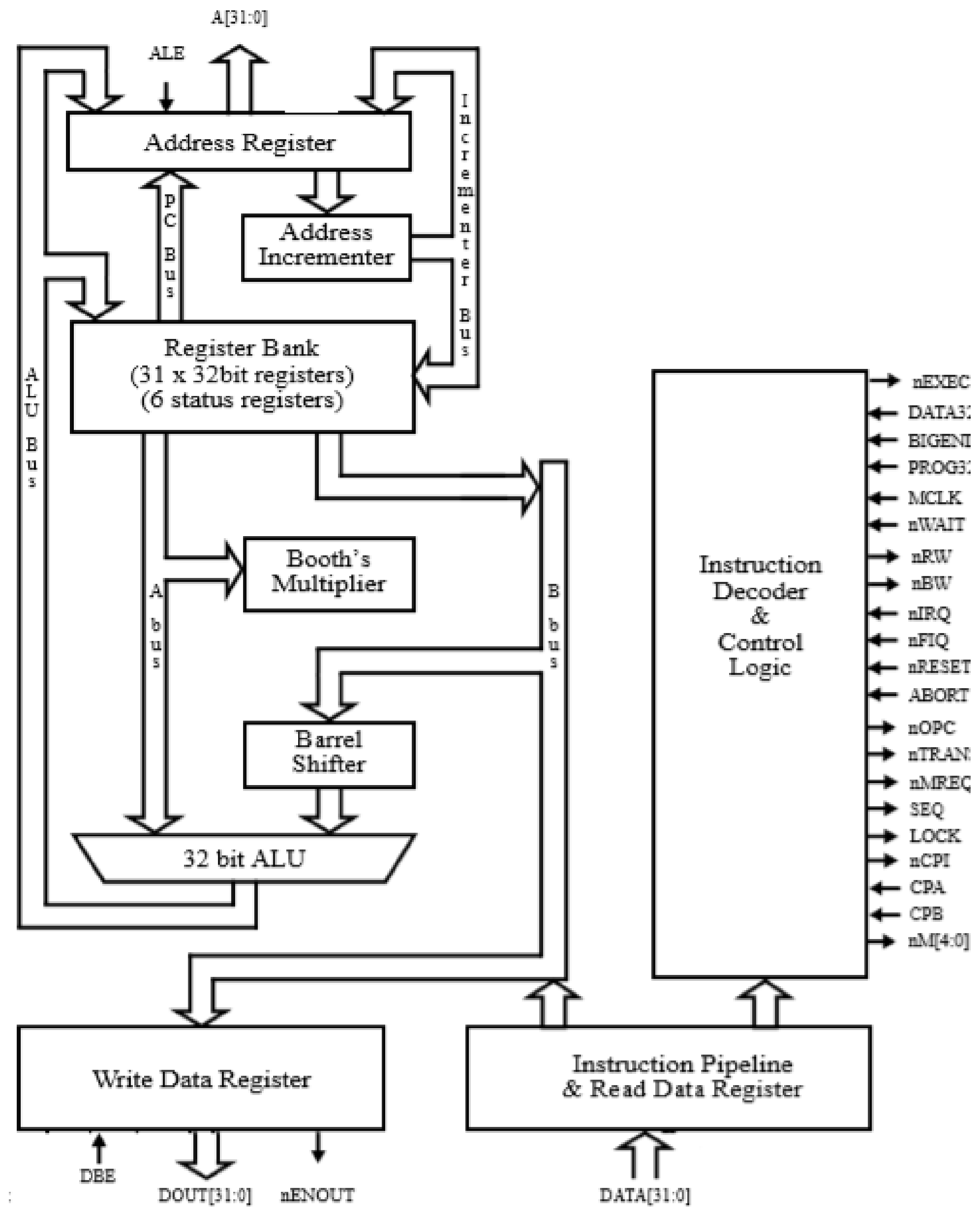


Figure 2.3 ARM memory organization.

ARM7 Block diagram



Load-store architecture

- In common with most RISC processors, ARM employs a load-store architecture.
- This means that the instruction set will only process (add, subtract, and so on) values which are in registers (or specified directly within the instruction itself), and will always place the results of such processing into a register.
- The only operations which apply to memory state are ones which copy memory values into registers (load instructions) or copy register values into memory (store instructions).
- CISC processors typically allow a value from memory to be added to a value in a register, and sometimes allow a value in a register to be added to a value in memory.

- ARM does not support such 'memory-to-memory' operations. Therefore all ARM instructions fall into one of the following three categories:

1. Data processing instructions

   These use and change only register values. For example, an instruction can add two registers and place the result in a register.

2. Data transfer instructions

   These copy memory values into registers (load instructions) or copy register values into memory (store instructions). An additional form, useful only in systems code, exchanges a memory value with a register value.

3. Control flow instructions. Normal instruction execution uses instructions stored at consecutive memory addresses. Control flow instructions cause execution to switch to a different address, either permanently (branch instructions) or saving a return address to resume the original sequence (branch and link instructions) or trapping into system code (supervisor calls).