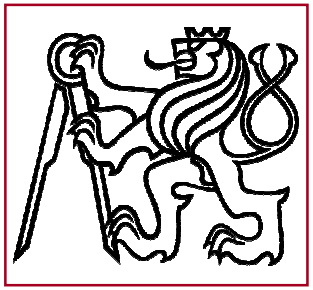
Y35PES

Programming for Embedded Systems

Ondřej Špinka, Pavel Němeček
spinkao@fel.cvut.cz nemecp1@fel.cvut.cz
http://dce.felk.cvut.cz/pes



Why to use ARM

- Low-cost, wide-spread architecture with great development support
- Low power consumption
- Good performance / Watt ratio
- *A lot* of various chips with ARM core available from various manufacturers, equipped with wide range of peripherals
- Good backwards compatibility, chips available from different manufacturers
- Variety of cores optimized for different applications available
- Optimistic future outlook (ARM architecture is definitely going to last for a lot of years, if not decades)

ARM Architecture Overview

- **ARM** stands for **Advanced** (previously **Acorn**) **RISC Machine**
- Designed by the Acorn company in 1983-1986 (Roger Wilson and Steve Furber)
- Probably most widely used embedded processor core nowadays, licensed by most of the MCU manufacturers
- 32-bit load-store RISC architecture, fixed instruction length (32 bits), mostly single-cycle execution
- Every instruction is conditional (!)
- Instruction pipeline (3 – 13 stages, depending on core version)
- Orthogonal register architecture
- Multiple load-store register instructions *(***SIMD - Single Instruction Multiple Data***)*
- 32-bit **barrel shifter** (multiple bit shift can be executed without performance penalty after most arithmetic instructions (!))
- Coprocessor interface *(***VFP – Vector Floating Point***)*
- Optional - **Thumb** instruction architecture (compressed 16-bit instruction set)

ARM x StrongARM x XScale

- A joint project involving Acorn and DEC (Digital Equipment Corporation) in 1995 resulted in the **StrongARM** development.
- StrongARM was a high-performance branch of the ARM family, clocked as high as 206MHz.
- Intel acquired a part of DEC IP (Intellectual Property) as a part of lawsuit settlement between the two companies in 1996, including the StrongARM design. Intel later replaced their ailing i860 and i960 product lines with StrongARM.
- The StrongARM development line was later renamed to **XScale**.

ARM Today

- **ARM7** core
  - 3-stage pipeline
  - MMU (Memory Management Unit) support
  - Used in low-cost cellular phones, mp3 players etc.
- **ARM9** core
  - 5-stage pipeline
  - Separate data and instruction cache
  - Used in higher-end mobile devices
- **ARM11** core
  - 7-stage pipeline
  - Enhanced performance over ARM9 and ARM7 families
  - DSP (Digital Signal Processor) and SIMD (Single Instruction Multiple Data) extensions (sometimes also referred to as NEON)
  - Used in PDAs and high-performance mobile devices, routers…

ARM Naming Conventions

- ARM [x][y][z][T][D][M][I][E][J][F][S]
  - x … family
  - y … MMU type (optional)
  - z … cache (optional)
  - T … Thumb instruction set (optional)
  - D … JTAG support (optional)
  - M … fast multiplier (optional)
  - I … Embedded ICE (In Circuit Emulator) macrocell
  - E … Enhanced instructions (implies TDMI)
  - J … Jazelle – Java hardware accelerator
  - F … Floating point coprocessor
  - S … Synthesizable version
- Therefore, ARM7TDMI implies…
  - ARM7 family
  - No MMU, no cache, no FPU, no Java
  - Thumb instruction set, JTAG and ICE support, fast multiplier

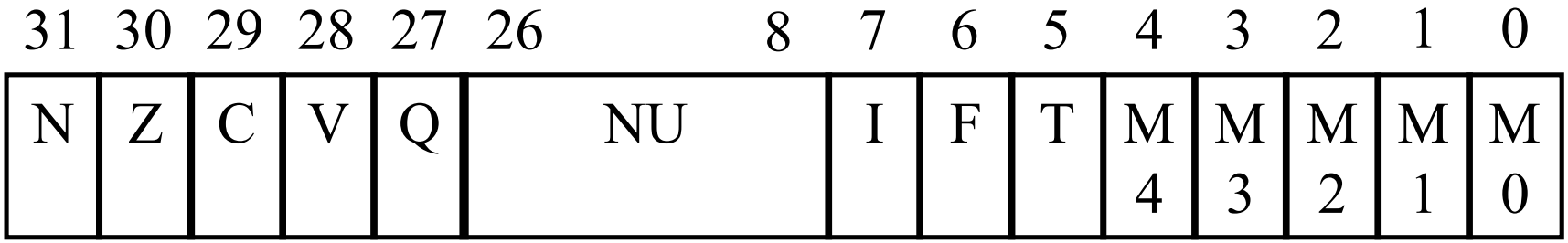
ARM Programming Model (1)

- ARM has a total of 37 registers.
- 31 of those are described as general – purpose 32-bit registers, but only 16 of them (denoted **R0-R15**) are visible at a time.
  - Two of them serve for a given purpose; **R14** is the **Link Register (LR)** and **R15** is the **Program Counter (PC)**. The LR holds the address of the next instruction after a branch and link (BL) instruction, or an exception. As the ARM instructions are 32-bits long and must be word-aligned, the two least significant bits of the PC always reads 0. When PC is written, those bits must be 0, otherwise the result is undefined. The PC is pointing (stage levels x 4) bytes ahead of the instruction currently being executed, to the instruction being fetched.
  - Remaining 14 registers can be used for any purpose, although **R13** is normally used as the **Stack Pointer (SP)**.
  - R13 and R14 registers are **banked** across all processor modes (explained later).
- Remaining 6 registers are identical and serve as the **Program Status Registers**. Only 2 of them are active at a time (given the processor mode). Those two registers are referred to as the **Current Program Status Register (CPSR)** and the **Saved Program Status Register (SPSR)**.
  - CPSR contains **condition code flags**, **interrupt disable bits**, **processor mode** and other status info.
  - SPSR serves as a backup of the CPSR if a exception occurs.

ARM Programming Model (2)

Status Control Register Structure:

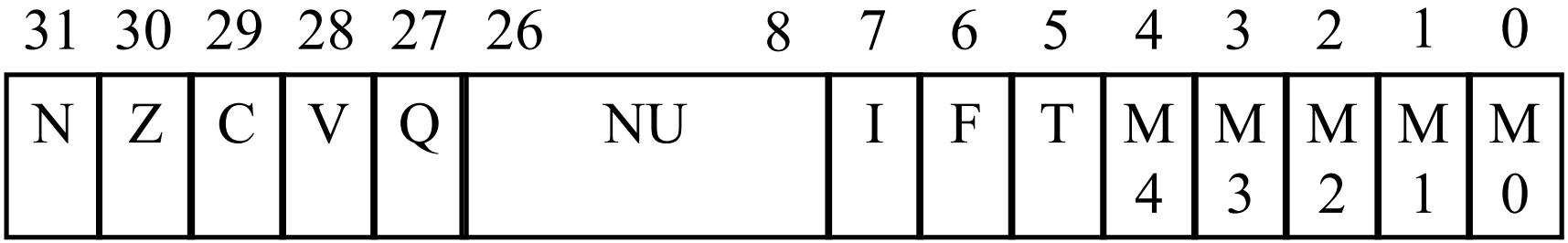
- Bits **M0 – M4 (mode bits)** determine current mode in which the processor operates.
- On "T" architectures, the **T-bit** determines the instruction mode; T = 0 means that the processor is executing ARM instructions, while T = 1 indicates that Thumb instructions are executed. On "non-T" architectures this bit should always be zero, otherwise each instruction triggers the **undefined instruction exception**.
- The **F-bit** disables FIQ when set (see later).
- The **I-bit** disables IRQ when set (see later).
- The **NU (Not Used)** bits are unused and should not be accessed.



ARM Programming Model (3)

Status Control Register Structure:

- The **Q-flag** indicates whether overflow and / or saturation occurred in some enhanced DSP instructions (valid for the "E" versions only).
- The **V-flag** indicates the overflow occurrence when executing an arithmetic instruction.
- The **C-flag** is the carry / borrow bit.
- The **Z-flag** indicates zero result when executing an arithmetic instructions (used when determining equality).
- The **N-flag** is set according the 31-bit of the result when executing an arithmetic instruction. If the result is treated as a signed integer, the N-flag indicates the negative sign of the result.



ARM Programming Model (4)

Processor modes:

The ARM core can operate in several **modes** (determined by the M-bits in the CSCR). A processor mode determines the set of registers used and user privileges. A total of 18 registers are active in a given mode. Available modes are as follows:

- **Abort mode** – implements virtual memory and / or memory protection.
- **Fast interrupt mode (FIQ)** – allows high-speed data transfers when handling an interrupt.
- **Interrupt mode (IRQ)** – normal IRQ handling.
- **Supervisor mode** – protected mode for the **Operating System (OS)**.
- **System mode** – used for privileged Operating System tasks.
- **Undefined mode** – supports software emulation of coprocessors.
- **User mode** – normal execution mode.

All modes except the user mode are privileged. Normally, the processor runs in the user mode. In user mode a restricted access to the CSCR is enforced, privileged modes allow full read / write CSCR access.

System Initialization

After startup, an application is needed to set-up the processor and start the application code. This initial set-up includes:

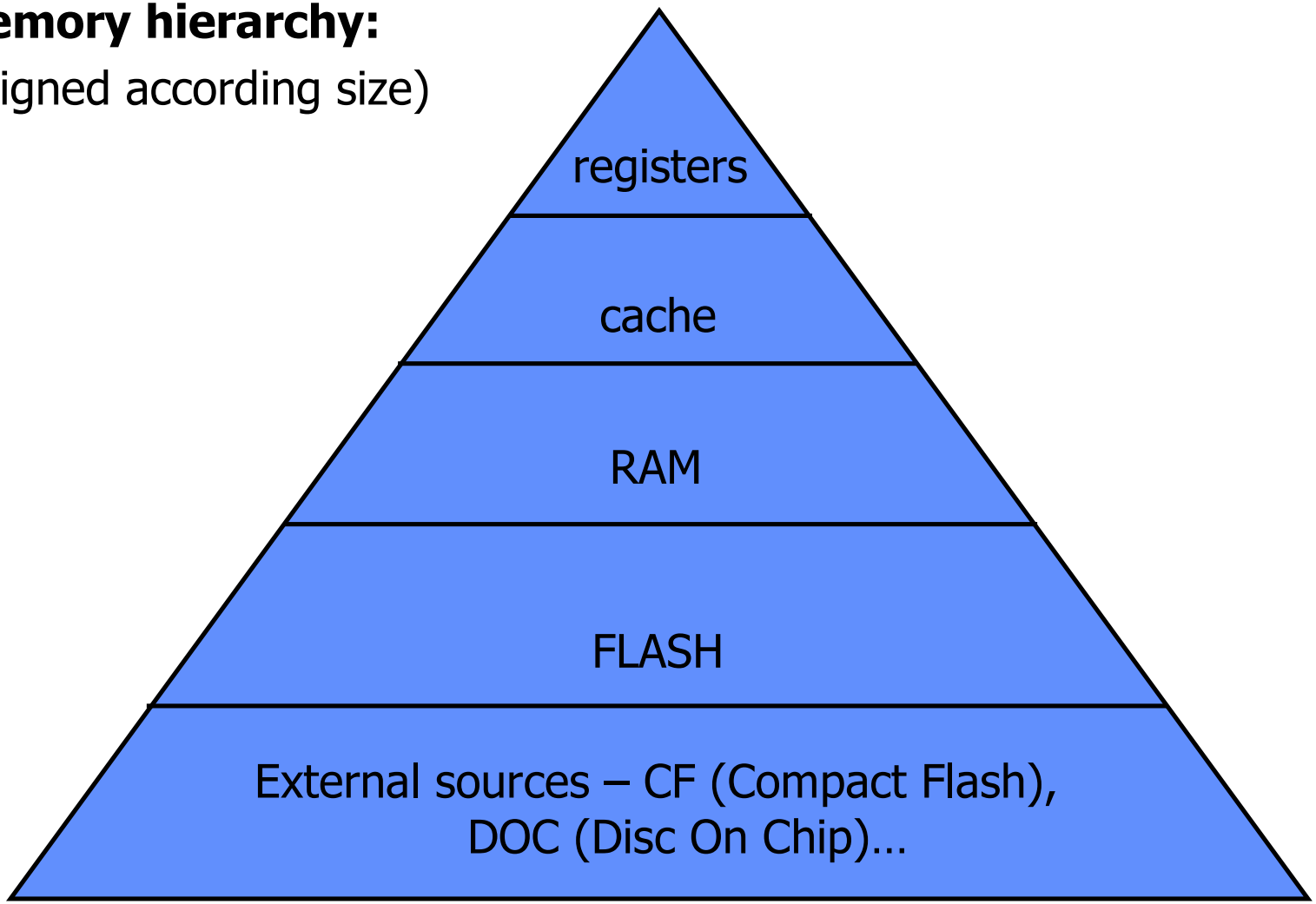
- Initialize stack and heap
- Set the variables in the memory to their initial values
- Load application code into RAM (optional)
- Set the memory protection (optional)
- Set-up interrupt vectors and I/O system (optional)
- Transfer control to the application entry point

The startup code features vary according processor features / application demands. The startup code is usually some kind of library code linked implicitly to the application.

Memory Management (1)

Memory hierarchy:

(Aligned according size)



Memory Management (2)

ARM architecture supports memory protection and virtual memory management (models equipped with MMU only). ARM memory space is divided into regions and programmer can set protection properties for each region individually. Basically, memory can operate in two modes:

- Unprotected mode (no memory protection).
- Protected mode (memory access is hardware protected and violation of memory protection would raise exception.

Memory protection is needed to prevent one application from accidentally rewriting memory portion belonging to another application. It is especially vital when Operating System is used.

Virtual memory is a virtual address space, which can (and normally is) be larger than physical memory address space. VM unifies data access into all memory types. It is divided into **memory pages**. When a page that is currently not loaded in the RAM is being accessed, an exception is triggered, allowing the memory manager to remap memory pages.

Memory Management (3)

ARM MMU:

MMU (Memory Management Unit) is a part of the CPU responsible for memory protection and virtual memory management. It provides following functions:

- Memory pages remapping (Unavailable page access handling)
  - Memory pages can be 4 kB, 64kB or 1MB wide
  - Data abort handler is responsible for fetching pages
- Memory protection (invalid memory access handling)

Memory Management (4)

ARM cache:

Cache is a very fast static associative memory, used to reduce latency when accessing RAM. When a RAM read request is executed, the cache is searched first, and only when requested data are not available in the cache they are actually fetched from the RAM. The cache is updated afterwards. If CPU requests the same data again, they can be obtained from the cache a lot faster then it would have been the case when actually reading RAM.

Pros:

- Cache speeds up RAM data access significantly

Cons:

- RAM access behavior is **non-deterministic** when a cache is involved (one cannot determine how fast a RAM read request would be)

Memory Management (5)

Cache properties:

- Cache is a lot faster then RAM (recall **static** x **dynamic** memory)
- Cache is a lot smaller then RAM
- Cache is a **fully associative memory**
- **Write thru** or **write back** – determines the cache behavior when used as a RAM write buffer
- **Data replacement policy** – determines the rules for updating data stored in the cache

An Example of a small ARM-Based MCU

Atmel AT91SAM7XC256

- ARM7TDMI core
- 256kB FLASH, 64kB SRAM
- Up to 55MHz clock speed
- DMA (Direct Memory Access)
- Power Management Controller
- Reset management with brown-out
- Watchdog
- 16-bit TC (Timer Counter), PIT (Periodic Interval Timer), RTT (Real-Time Timer), 4-channel 16-bit PWM (Pulse Width Modulator)
- 2x 32-bit PIO (Parallel I/O)
- 2x USART, 2x SPI, 1x CAN 2.0b, 1x USB 2.0, 1x Ethernet 10/100
- 8-channel 10-bit A/D converter
- AES and 3DES encryption unit
- JTAG interface

Basic Peripherals Overview (1)

- Timers
  - TC (Timer Counter)
    - 3-channel 16-bit timer
    - Frequency measurement, event counting, interval measurement, pulse generation
  - PIT (Periodic Interval Timer)
    - Its purpose is to provide accurate periodic interrupt
    - Useful mainly for Operating Systems
  - RTT (Real-Time Timer)
    - Provides Real-Time clock (counts elapsed seconds)
    - Useful to generate a periodic interrupt at lower time base
    - Can trigger alarm at a pre-set time
    - Can be used as free-running low-speed timer

Basic Peripherals Overview (2)

- PWM (Pulse Width Modulator)
  - Four 16-bit channels
  - Is used to generate periodic signal with variable duty cycle
  - Used to control DC motors, servomechanisms, etc…
- ADC (Analog to Digital Converter)
  - Eight 10-bit channels (multiplexed)
  - Sample & Hold function
  - Internal triggers from Timer Counter

Basic Peripherals Overview (3)

- PIO (Parallel Input / Output)
  - 2 32-bit ports
  - Each pin can be configured separately
  - 5V tolerant
- WDT (Watchdog Timer)
  - Is a free-running timer, which would trigger RESET if not cleared periodically
  - Is used to prevent lock-ups

Basic Peripherals Overview (4)

- USART (Universal Synchronous / Asynchronous Transceiver / Receiver)
  - Synchronous or asynchronous serial interface
  - Full-duplex
  - DMA connection
  - IrDA and RS-485 support
- CAN (Controller Area Network)
  - Well-known industrial serial bus
  - Reliable, fault tolerant
  - Non-destructive data packet arbitration
  - Up to 1Mb/s bit rate

Basic Peripherals Overview (5)

- JTAG (Joint Test Action Group)
  - Test and debug interface standardized by *IEEE 1149.1* standard
  - Designed in 1985, originally used to test complex multi-layer **PCBs (Printed Circuit Boards)**
  - Today the most widely used in-circuit debug standard
  - Allows to gain a **total control** over the tested **IC (Integrated Circuit)**
  - Serves as a "back door" into Embedded Systems

Lecture Summary – Essential Things to Remember

- **ARM** is one of the **most widely used** MCU cores nowadays. It utilizes **32-bit RISC Load-store architecture** with **instruction pipeline**. Every instruction is **conditional**. ARM has **orthogonal register architecture**.
- There are **multiple ARM cores** available, varying in performance and features.
- ARM can operate in **seven different modes**.
- Before running application, the CPU must be correctly **initialized**.
- **ARM MMU** provides several useful functions, such as **memory protection** and **virtualization**.
- **Cache** is used to reduce **latency** when accessing RAM.
- Typical ARM-based embedded MCU has a **wide range of peripherals**.