Università
della
Svizzera
italiana

Facoltà
di
scienze
informatiche

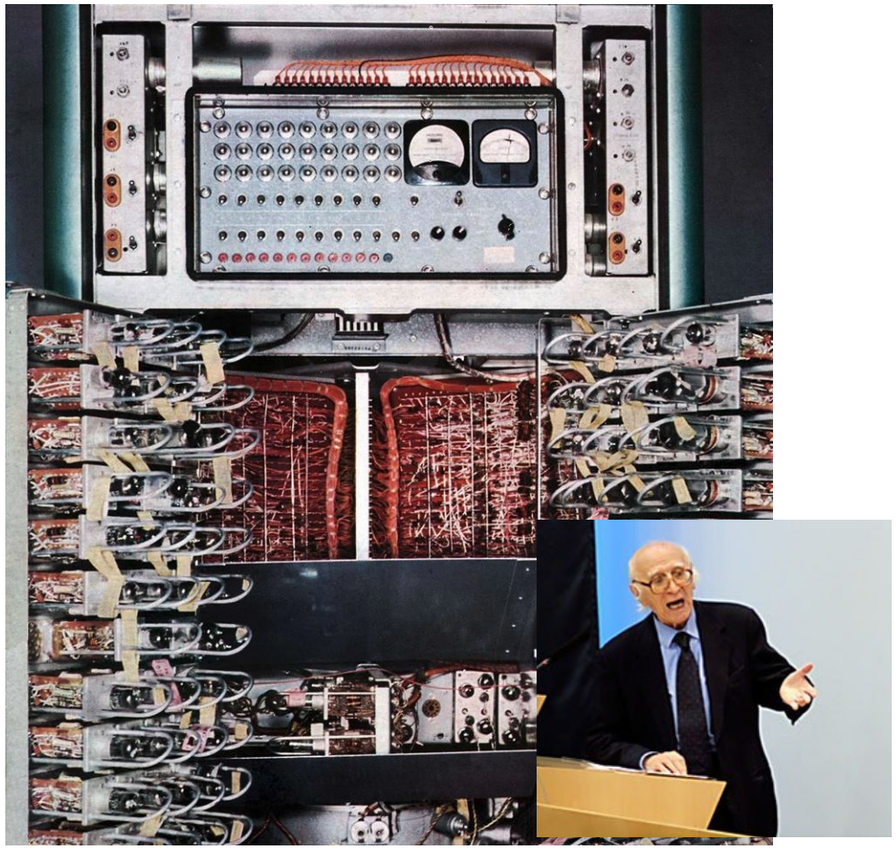
Edge Computing in the IoT

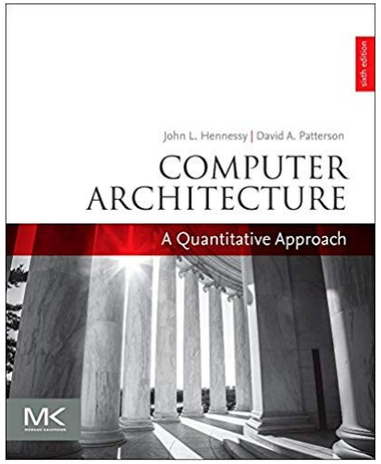
Embedded Computer Architectures

Alberto Ferrante (alberto.ferrante@usi.ch)

Digital systems: sixty years and still evolving…

- The PC you buy today is faster and it has lots more storage than a computer that barely thirty years ago cost 1 M$...
- Seventy years ago: there was just one “class” of computers
- Today: solutions range from microcontrollers (less than 1$ apiece, in quantities) to supercomputers connecting tens of thousands of microprocessors (some M$...)





CRC 102-A – Prof. Luigi Dadda

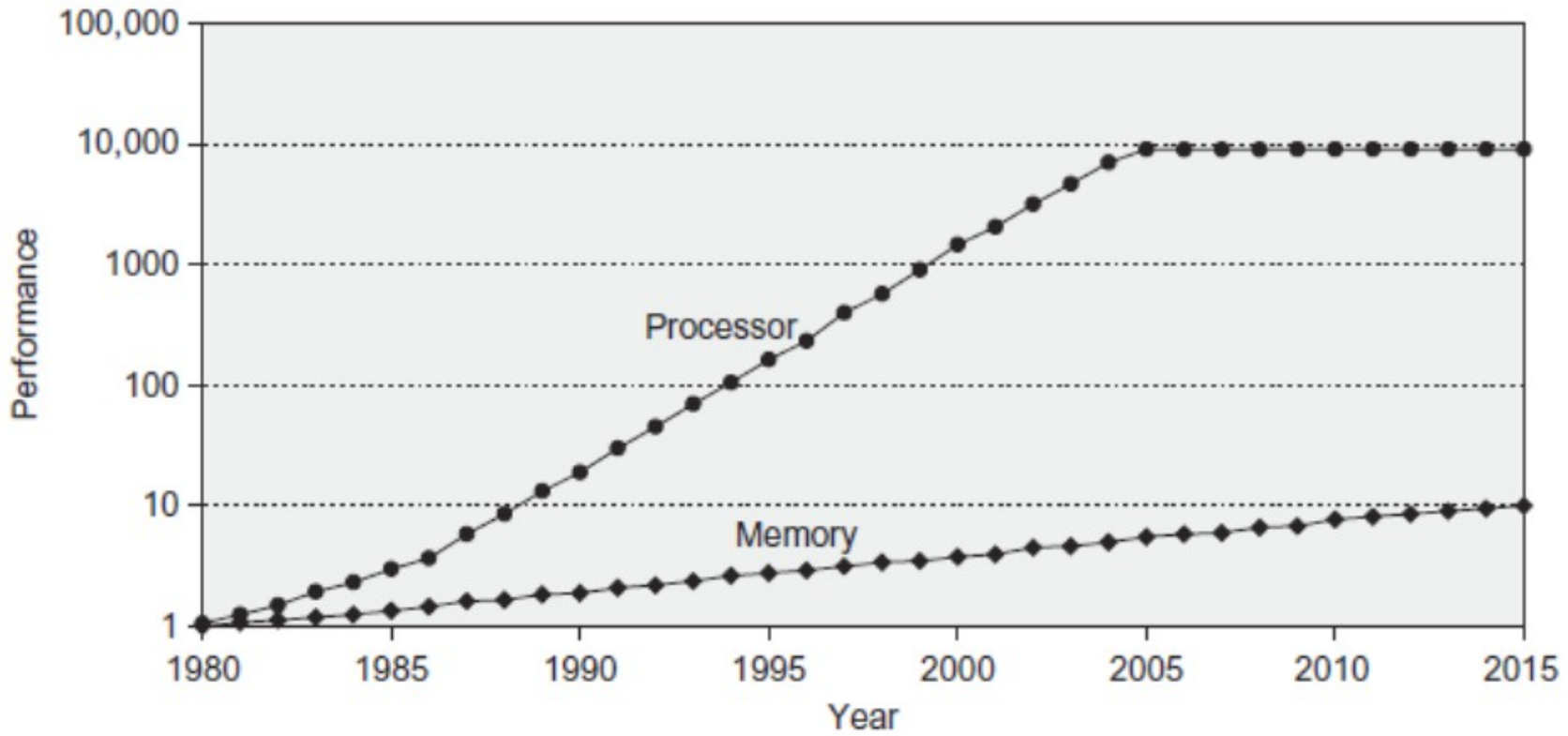
Architecture: which definition?

- **Abstract Architecture**
  - The functional specification
- **Concrete Architecture**
  - An implementation of an Abstract Architecture
  - Often denoted *microarchitecture*

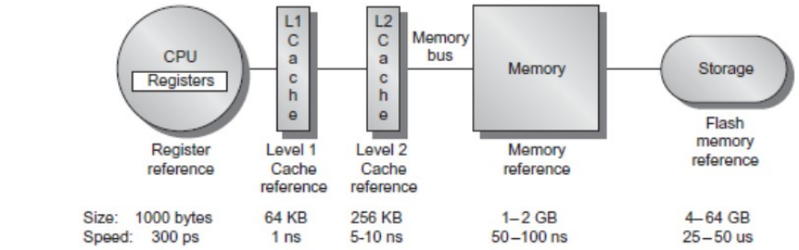
Definition of architecture

- Abstract Architecture: a “black box” specification of a machine – could be seen:
  - From the programmer’s point of view: we deal with a programming model, equivalent to the description of the machine language
  - From the designer’s point of view: we deal with a hardware model (black-box description given to the designer, including additional information, such as interfacing standards)
- **Instruction Set Architecture (ISA)**: abstract architecture defined by the set of instructions that the computer is capable of decoding and executing
  - The same ISA can be implemented with different concrete architectures
    - e.g.: ISA X86, implemented by architectures provided by Intel, AMD, …

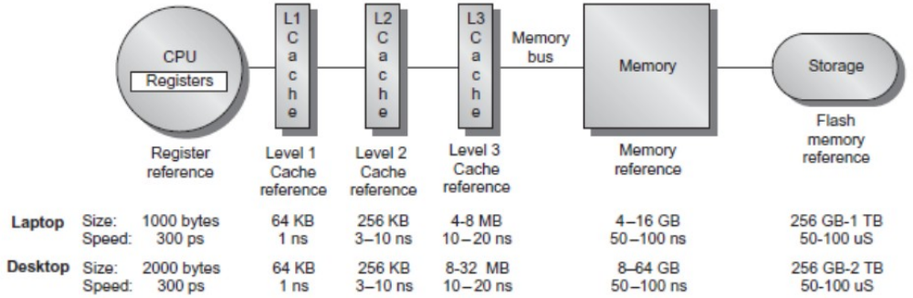
Memory Performance Gap



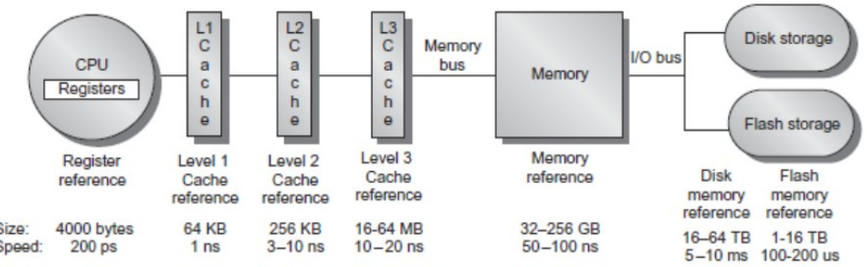
Memory Hierarchy



(A) Memory hierarchy for a personal mobile device



(B) Memory hierarchy for a laptop or a desktop



(C) Memory hierarchy for server

Non-volatile Memory

- Embedded systems need to store data even when the power is turned off
  - ROM: read-only memory, the most basic non-volatile memory, the contents of which is fixed at the chip factory
    - Useful for mass produced products that only need to have a program and constant data stored, and these data never change
      - Such programs are known as **firmware**, suggesting that they are not as “soft” as software
  - EEPROM, electrically-erasable programmable ROM
    - Several forms, but it is possible to write to all of these, often can be programmed in the field
    - The write time is typically much longer than the read time, and the number of writes is limited during the lifetime of the device

Non-volatile Memory

- Disk memories can store very large amounts of data, but access times can become quite large
  - Vulnerable to vibration, hence difficult to use in many embedded applications
- Flash memory, a particular category of EEPROM
  - Flash memories have reasonably fast read times, but not as fast as SRAM and DRAM
    - Frequently accessed data will typically have to be moved from the flash to RAM before being used by a program
  - The total number of writes are limited (1-10 MLN), so these memories are not a substitute for working memory
  - Two types of flash memories
    - NOR flash has longer erase and write times, but it can be accessed like a RAM
    - NAND flash is less expensive and has faster erase and write times, but data must be read a block (hundreds to thousands of bits) at a time

Memory Hierarchy Design

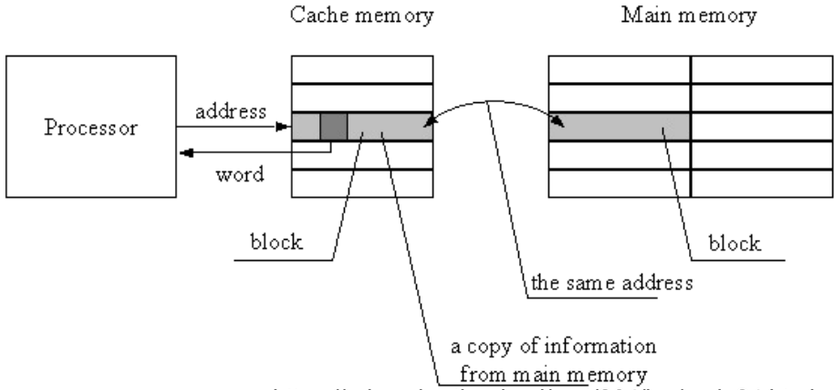
- Memory hierarchy design has become more crucial with multi-core processors:
  - Aggregate peak bandwidth grows with number of cores:
    - Intel Core i7 can generate two memory references per core per clock
    - Four cores and 3.2 GHz clock
      - 25.6 billion 64-bit data references/second +
      - 12.8 billion 128-bit instruction references/second
      - = 409.6 GB/s!
    - DRAM bandwidth is only 8% of this (34.1 GB/s)
    - Requires:
      - Multi-port, pipelined caches
      - Two levels of cache per core
      - Shared third-level cache on chip

Basic principle underlying memory hierarchies: locality

- **Temporal locality**: when a memory entry is referenced, the same entry will be referenced again within a short time with high probability
  - Typical instance: repeated access to instructions and variables within a loop
- **Spatial locality**: whenever a memory entry is referenced, reference will be made shortly to neighboring items with high probability
  - E.g.: instructions within a sequence; data organized as arrays and accessed sequentially
- Locality is strongly application dependent: high (both temporal and spatial) for tight loops operating on stacked data, low in case of repeated procedure calls
  - In some applications, data exhibit locality of one kind only
    - E.g., “streaming” data in video processing (no temporal locality), coefficients used in signal or image processing (the same coefficients are used repeatedly in time – no spatial locality)
    - This may lead to modified use of the memory hierarchy principle

Caches

- Small and fast memories that are used to mask the real speed of RAM, that is much bigger but also much slower
  - Smaller memories are inherently faster
  - For small memories we can adopt costly technologies
- Caches exploit the principles of locality
- A block in cache holds the contents of a block in memory
  - Mapping between cache and memory, as well as block replacement, can be performed by adopting different policies
    - Optimal policies for mapping and replacement depend on the tasks to be executed



https://edux.pjwstk.edu.pl/mat/264/lec/main84.html

- **Caches introduce some uncertainty in memory access times**
  - Cache hit: faster access
  - Cache miss: slower access

Microprocessors in Embedded Devices

- Difficult to define specific characteristics for embedded devices
  - Usually, microprocessors aimed at embedded devices have:
    - Limited energy requirements
    - Limited computational power
    - Limited memory
  - … But there are many exceptions
- Microprocessors are often specifically designed for requirements of embedded devices
  - i.e., general purpose microprocessors are often not suitable: too expensive, too energy hungry, ...

Types of Processors

- Microcontroller
- Microprocessor
- Digital Signal Processor (DSP)
- Graphic Processing Unit (GPUs)

Microcontrollers

- A microcontroller (μC) is a small computer on a single integrated circuit consisting of
  - A relatively simple central processing unit (CPU)
  - Peripheral devices such as memories, I/O devices, and timers
- More than half of all CPUs sold worldwide are microcontrollers
- The simplest microcontrollers operate on 8-bit words and are suitable for applications that require small amounts of memory and simple logical functions
- They may consume extremely small amounts of energy, and often include a sleep mode that reduces the power consumption to nanowatts

DSP Processors

- Processors dedicated to signal processing
  - A signal is a collection of sampled measurements of the physical world, typically taken at a regular rate called the *sample rate*
- Signal processing applications all share certain characteristics:
  - They deal with large amounts of data
  - They typically perform sophisticated mathematical operations on the data
- DPS Processors use architectures that provide the ability of performing the same operation on arrays of data in parallel (SIMD)
  - They are optimized for operating with large amount of data
    - Memory architecture ad access to data in memory
    - Organization of registers
    - Organization of computational units

Graphic Processing Units (GPUs)

- A graphics processing unit (GPU) is a specialized processor designed specifically to perform the calculations required in graphics rendering
- Some non-graphic applications can benefit form GPUs
- GPUs are very efficient in executing the same sequence of operations on multiple data, but they are very inefficient when control paths diverge
  - Instructions that operate on a limited subset of the set of data, are very inefficient
- GPUs require a host CPU
- GPUs vs. DSPs:
  - All of its calculations in GPUs use floating-point algorithms, and there are currently no bit or integer arithmetic instructions
  - The storage system in GPUs is a two-dimensional segmented storage space
  - No indirect write instructions:tThe output write address is determined by the raster processor and cannot be changed by the program

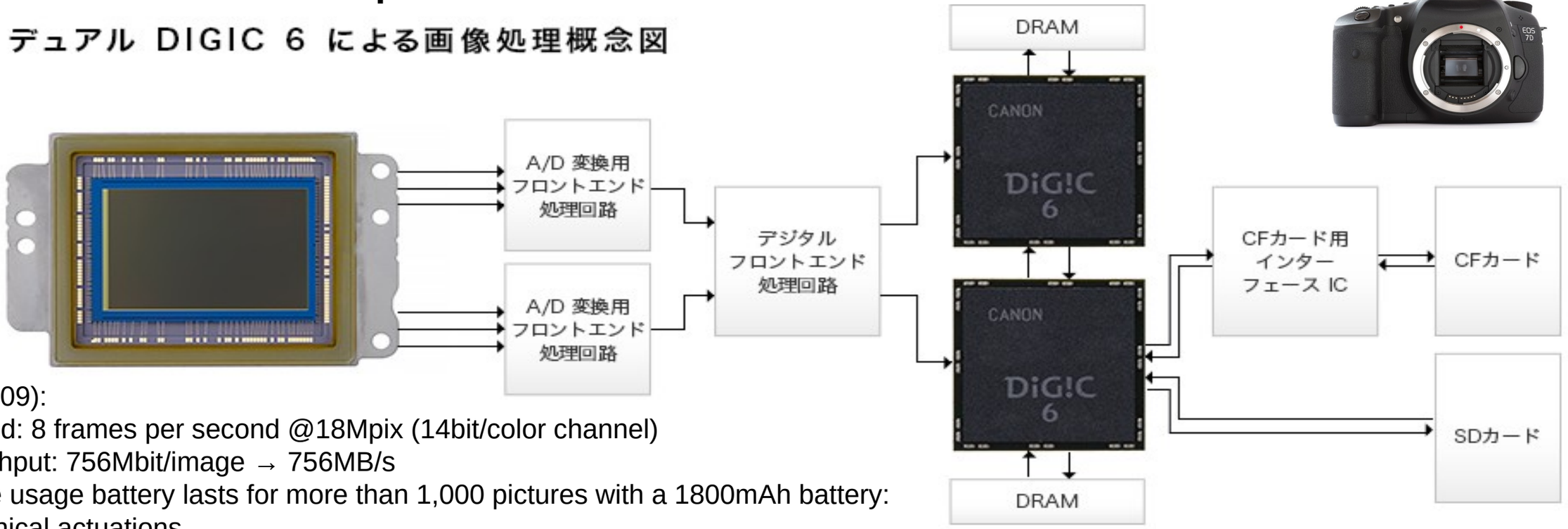
Microprocessors in Embedded Devices - Trends

- In the last years, microprocessors dedicated to embedded devices have become more and more capable from the computational stand point
  - 8-bit architectures are now less frequent; 32-bit architectures have become more usual
  - Multi-core processors are common in the mid and high-performance classes of devices
  - Memory has increased in size, even though still quite limited
- Use of some forms of accelerations, often on-chip, have become frequent:
  - Hardware accelerators (e.g., for cryptography)
  - GPUs
- Pure DSP (Digital Signal Processor) processors are less frequent, but DSP instructions (SIMD) are often included in microprocessors

Parallelism

- Functional parallelism may be seen at different abstraction levels, corresponding to different architectural solutions:
  - Instruction-level (ILP) – fine-grained;
  - Loop-level – middle-grained
  - Procedure-level – middle-grained
  - Program-level – coarse-grained
- Microprocessors have hit the “power wall”: Impossible to keep doubling performance every three years as per the Moore’s law
  - Use of multiple cores to increase performance: only for concurrent applications, that can exploit parallelism!

An Example: Canon EOS 7D / 5DS / 5DS R



EOS 7D (2009):

- Max speed: 8 frames per second @18Mpix (14bit/color channel)
  - Throughput: 756Mbit/image → 756MB/s
- In real-life usage battery lasts for more than 1,000 pictures with a 1800mAh battery:
  - Mechanical actuations
  - Write to solid state memory
  - Image processing
  - Autofocus
  - ...

https://cweb.canon.jp/eos/lineup/7dmk2/feature-highquality.html#cmos

ILP

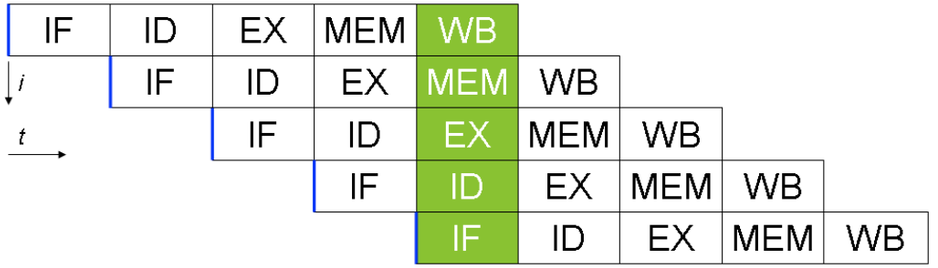
- Multiple instructions from the same object program are executed in parallel
  - ILP does not require ad-hoc language constructs, it is detected by an optimizing compiler
- Two basic techniques of parallel execution:
  - **Pipelining**: a number of (different) functional units are employed in sequence to perform a single “computation” (in our case, machine instruction), forming a pipeline.
  - **Replication of functional units**: replicated units execute the same operations in parallel on different data elements
  - The two techniques may (and do) coexist in the same architecture
    - Two orthogonal concepts
- Applicable in the case of regular ISA
  - Hardly applicable when instructions of different lengths coexist;
- Throughput enhancement for the complete instruction set

Pipelining

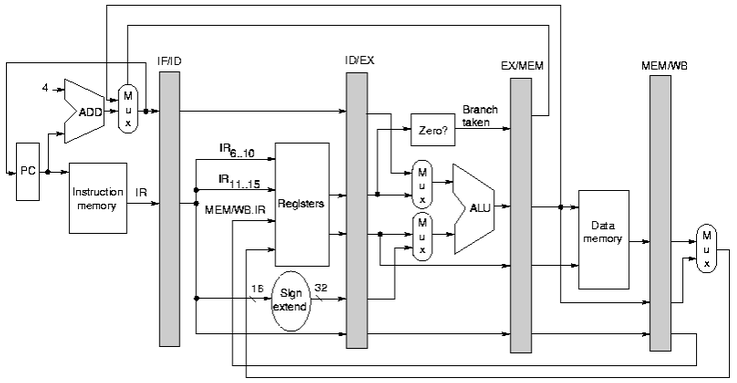
- For each instruction executed by the microprocessor, the following operations need to be performed (load/store architecture):
  1) The instruction is fetched from memory
  2) The instruction is decoded
  3) The operands are loaded into registers
  4) The instruction is executed:
     - For memory instructions:
       - Memory address is computed
       - Memory is written with a value from a register or memory is read and a value is written to a register
     - For branch instructions:
       - The condition is verified
       - The target address is computed
     - For other instructions:
       - The operation is executed
       - Results are written in destination registers
  5) The program counter is updated

Pipelining

- The operations executed for each instruction are logically sequential
- They may be parallelized when multiple instructions are considered (e.g., while executing instruction *n*, the processor can fetch instruction *n+1*)



Control Hazards




```
                addi    r3,r0,0
loop:           lw      r4, a(r3)
                lw      r5, b(r3)
                sub     r5,r5,r4
                addi    r3,r3,4
                bnez    r5,loop
exit:           addi    r5, r5, 10
```

- A branch is not evaluated until after the execution stage
- Until the branch is not evaluated we cannot know if it is taken or not
  - Which are the next instructions to be executed?
- Any instruction following the branch should wait
  - The pipeline is not fully exploited
- Control hazards are difficult to remove and/or optimize

Branch Prediction

- Branch prediction: by far the most used approach, has many alternative solutions: simplest approach:
  - Branch is always predicted not taken: instructions following branch are fetched and sent to the pipeline
  - If branch has to be taken, pipeline is flushed and instructions in it are nullified.
- Prediction implies a speculation on comparison outcome → **speculative execution**
  - With the simple prediction (always taken/not taken):
    - Probability of sequential execution high → increase of performances w.r.t. stall insertion
      - True, e.g., for loops (if properly compiled)
    - Probability of sequential execution near 50% → speculation does not lead to substantial performance increase.

Other Branch Prediction Schemes

- Branch prediction buffer
- BPB with Two-bit prediction
- Correlating predictors
- …

- Effectiveness of any prediction scheme is strictly related to the task to the specific code being executed

Data-level Parallelism

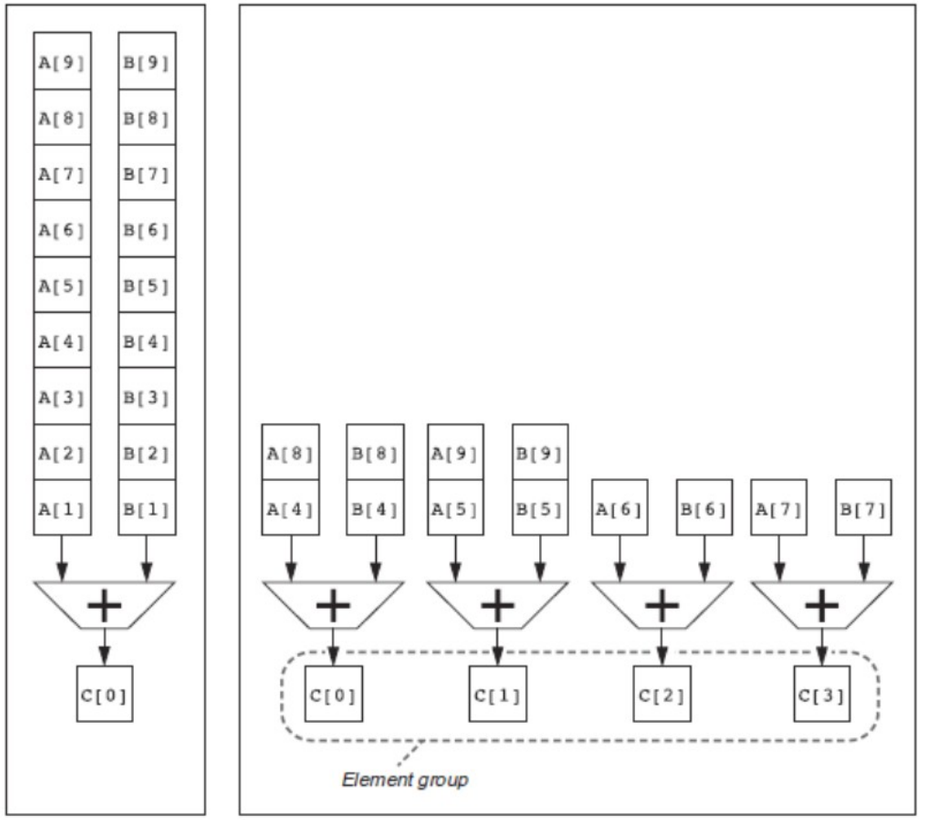
- Single instruction stream, single data stream (SISD): the traditional single processor;
- Single instruction stream, multiple data streams (SIMD):
  - One instruction flow executed simultaneously by multiple processing units operating on different data flows
  - Each processing unit has its own data memory
  - There is just one program memory and one control processor
  - A solution suitable for well-specified application classes

Data-level Parallelism

- DSP processors make use of SIMD
- GPUs make use of SIMD
- Most modern CPUs include some SIMD instructions that provide the ability to exploit SIMD benefits “in the small”

Data-level Parallelism

- SIMD architectures can exploit significant data-level parallelism for:
  - Matrix-oriented scientific computing
  - Media-oriented image and sound processors
  - Example: a single SIMD instruction adds 128 numbers in one clock cycle



ARM v8

- Instruction set architecture:
  - There exist different implementations with different microarchitectures
- RISC ISA
  - Load-store architecture
  - Fixed-length instructions
- Three profiles

Profile	Name	Description
Application	ARMv8-A	Optimized for a large class of general applications for mobile, tablets, and servers.
Real-Time	ARMv8-R	Optimized for safety-critical environments.
Microcontroller	ARMv8-M	Optimized for embedded systems with a highly deterministic operation.

https://developer.arm.com/architectures/cpu-architecture

ARM v8-A

- Two execution states
  - AArch64
    - A64 instruction set
    - Holds addresses in 64-bit registers
    - Allows instructions in the base instruction set to use 64-bit registers
  - AArch32
    - 32-bit Execution state
    - Preserves backwards compatibility with the Armv7-A architecture
      - It can support some features included in the AArch64 state
    - It supports the T32 and A32 instruction sets
- Cannot switch between AAarch64 and AAarch32 in the same application

https://developer.arm.com/architectures/cpu-architecture

ARM v8-A

- Hardware-accelerated cryptography
- NEON double-precision floating-point advanced SIMD
- Architectural extensions:
  - Security Extensions
  - Multiprocessing Extensions
  - Large Physical Address Extension
  - Virtualization Extensions
  - ...

https://developer.arm.com/architectures/cpu-architecture

ARM v8-M

- 32 bit instruction set
  - The registers in the register bank, most data operations, and addresses are 32-bit
    - It also supports data types of various sizes such as 8-bit and 16-bit
    - Supports a limited set of 64-bit operations
  - The 32-bit linear address provides 4GB of address space
    - architecturally pre-defined into several regions with different memory attributes
- Various architectural extensions
  - The Security Extension (Arm TrustZone)
  - The Floating-point Extension
  - The Debug Extension
  - The Digital Signal Processing (DSP) Extension

https://developer.arm.com/architectures/cpu-architecture

ARM v8-R

- It supports the A32 and T32 instruction sets
- A number of features for safety-critical environments
  - No overlapping memory regions
  - Exception model that is compatible with the Armv8-A model
  - Virtualization with support for guest operating systems.
  - Optionally, support for double-precision floating-point and Advanced SIMD
- Several extensions:
  - Multiprocessing Extensions: optional set of extensions that provide a set of features that enhance multiprocessing functionality
  - Generic Timer Extension: optional extension that provides a system timer and a low-latency register interface to it
  - Performance Monitors Extension: defines a recommended performance monitors implementation and reserves register space for performance monitors

https://developer.arm.com/architectures/cpu-architecture

ARM Thumb instruction set

- Thumb instructions provide the ability to optimize for memory and energy
- Is a subset of the most commonly used 32-bit ARM instructions
- Thumb instructions are each 16 bits long
  - On execution, 16-bit Thumb instructions are transparently decompressed to full 32-bit ARM instructions in real time
    - Performed at the decode stage, without performance loss
  - Thumb instructions operate with the standard ARM register configuration, allowing interoperability between ARM and Thumb states.

http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0210c/CACBCAAE.html
http://www.keil.com/support/man/docs/armasm/armasm_dom1359731126163.htm

ARM Thumb instruction set

- Thumb code
  - Is typically 65% the size of standard ARM code
  - Provides a 160% performance increase when running on a 16-bit memory system
- Thumb is ideally suited to embedded applications with restricted memory bandwidth, where code density and footprint is important
- The availability of both 16-bit Thumb and 32-bit ARM instruction sets gives designers the flexibility to choose on a subroutine level, according to the requirements of their applications
  - Mode can be changed through directives and instructions in the assembly code
  - gcc can handle the generation of thumb code at compile time

http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0210c/CACBCAAE.html
http://www.keil.com/support/man/docs/armasm/armasm_dom1359731126163.htm

ARM Cortex M4

- Architecture: Armv7E-M
- Bus Interface: 3x AHB Lite interface (Harvard bus architecture)
  - ICode, DCode, and System bus interfaces
- Pipeline: 3-stage + branch speculation
- No built-in cache
- Interrupts
  - Implementation-defined number from 1 to 240
  - 8 to 256 priority levels
  - An internal Non Maskable Interrupt
- DSP Extension
  - Single cycle 16/32-bit MAC
  - Single cycle dual 16-bit MAC
  - 8/16-bit SIMD arithmetic
  - Hardware Divide (2-12 Cycles)
- Optional single precision floating point unit, IEEE 754 compliant (**Cortex-M4F**)
- Thumb/Thumb-2
- Memory Protection
  - Optional 8 region MPU with sub regions and background region

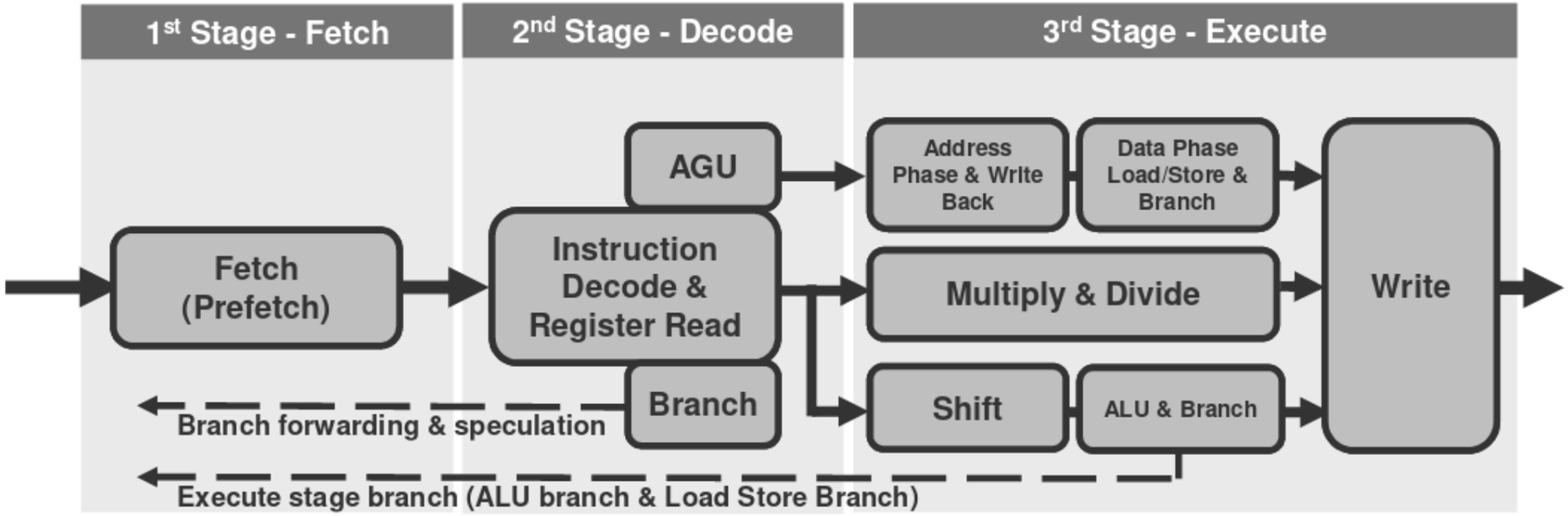
http://infocenter.arm.com/help/topic/com.arm.doc.100166_0001_00_en/arm_cortexm4_processor_trm_100166_0001_00_en.pdf

ARM Cortex M Power Modes

- ARM Cortex-M processors have at least three power modes: Run, Sleep and Deep Sleep
  - Depending on the implementation, more modes can be available
- There are two instructions that can be used to put the system into a low power state:
  - Wait-For-Event (WFE)
    - The processor will enter standby, and wake on the next interrupt or event.
      - e.g., a wake-up event from another CPU
  - Wait-For-Interrupt (WFI)
    - The processor will enter standby, and wake on the next interrupts
- **Sleep-On-Exit** allows the processor to immediately go back to sleep once the interrupt that is being serviced completes

https://www.embedded.com/arm-cortex-m-low-power-mode-fundamentals/

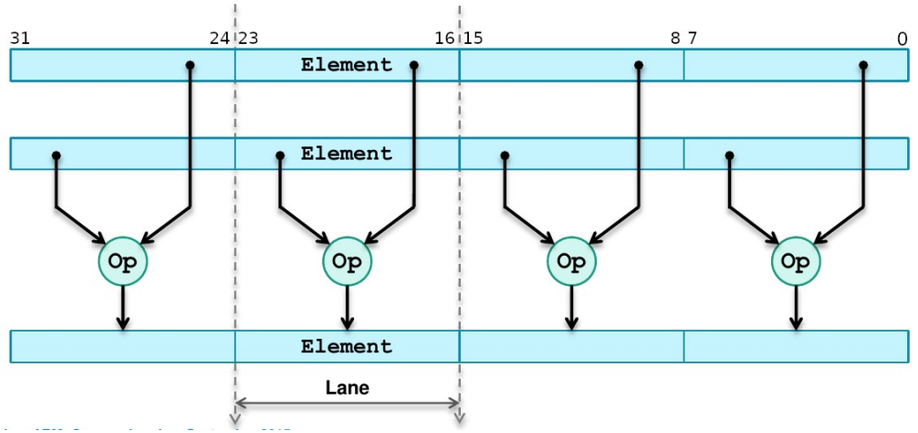
ARM Cortex M4 Pipeline



http://infocenter.arm.com/help/topic/com.arm.doc.100166_0001_00_en/arm_cortexm4_processor_trm_100166_0001_00_en.pdf

ARM Cortex M4 DSP Extension – SIMD Instructions

- SIMD: Singe Instruction Multi Data
  - The same instruction is executed on multiple flows of data
  - Useful for streams of data (e.g., multimedia)
- Instructions that work on 8- or 16-bit data types
- The SIMD instructions allow 2x16-bit or 4x8-bit operations to be performed in parallel
- In gcc use the *-ffast-math* option to generate SIMD code
  - Usually, with SIMD operations we parallelize successive iterations of the same loop
  - Not all code can be converted to SIMD, beware of data dependencies

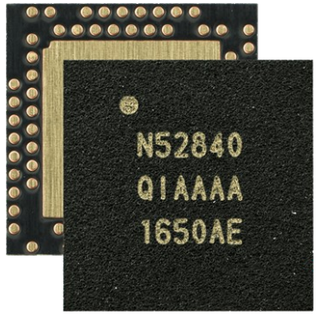


ARM Cortex M7

- Architecture: Armv7E-M
- Highest performance member of the Cortex-M processor family
- Thumb or Thumb2, featuring:
  - Hardware Divide (2-12 Cycles)
  - Single-Cycle Multiply
  - DSP extension
    - Single cycle 16/32-bit MAC
    - Single cycle dual 16-bit MAC
    - 8/16-bit SIMD arithmetic
- Optional single and double precision floating point unit
- Integrated Bit-Field Processing Instructions (bit manipulation)
- Architecturally defined Sleep and Deep Sleep Modes. Integrated WFI and WFE Instructions and Sleep-On-Exit capability.

https://developer.arm.com/Processors/Cortex-M7

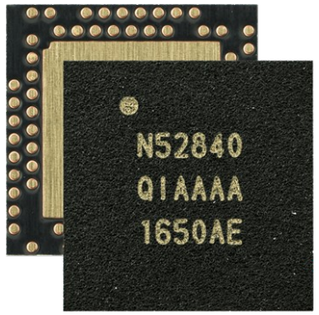
Nordic Semiconductor nRF 52840



- System on Chip containing
  - A microcontroller
  - RAM + Flash
  - Crypto accelerator
  - Bluetooth 5
  - IEEE 802.15.4 radio support
  - USB 2.0
  - ADC
- Implementation of the **ARM Cortex-M4** with **floating point unit** clocked at **64MHz**
  - Digital signal processing (DSP) instructions
  - Single-cycle multiply and accumulate (MAC) instructions
  - Hardware divide
  - 8- and 16-bit single instruction multiple data (SIMD) instructions
  - Single-precision floating-point unit (FPU)

https://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.1.pdf
https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52840

Nordic Semiconductor nRF 52840



- Memory:
  - 1 MB of flash
    - An instruction cache (I-Cache) can be enabled for the ICODE bus in the flash
      - It is possible to enable cache profiling to analyze the performance of the cache using the ICACHECNF register.
  - 256 kB of RAM
- The same physical memory is mapped to both the Data RAM region and the Code RAM region
  - It is up to the application to partition the RAM within these regions so that one does not corrupt the other

https://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.1.pdf
https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52840

ST Microelectronics STM32H747

- Cortex-M7 + Cortex-M4
  - 480 MHz on the Cortex-M7
  - 240 MHz on the Cortex-M4
- Memory:
  - From 1 to 2 Mbytes of Flash memory
  - 1 Mbyte of SRAM
- ADCs: different configurations, e.g., 3× ADCs with 16-bit max. resolution (up to 36 channels, up to 3.6 MSPS)
- DACs: different configurations, e.g., 2× 12-bit DAC (1 MHz)
- GPU: Chrom-ART Accelerator™
- JPEG hardware accelerator for fast JPEG encoding and decoding

https://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.1.pdf
https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52840

ST Microelectronics STM32H747

- Crypto/hash hardware acceleration
  - Secure Firmware Install (SFI) embedded security services to authenticate and protect your software IPs while performing initial programming
  - Secure Boot Secure Firmware Update (SBSFU)
- Multi-power domain architecture enables different power domains to be set in low-power mode to optimize the power efficiency
- Multiple 16- and 32-bit timers running at up to 480 MHz

https://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.1.pdf
https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52840

ARM Cortex-A53

- 8-stage pipelined processor
  - 2-way superscalar
  - In-order execution
    - Out-of-order too expensive in therms of area and power
- 64-byte cache lines
- 10-entry L1 TLB, and 512-entry L2 TLB
- 4 KiB conditional branch predictor, 256-entry indirect branch predictor
- ACE or CHI AMBA bus

Power and Embedded CPUs

- Dynamic power depends on
  - The switching frequency (CPU clock)
  - The Activity Factor: average number of switching events undergone by the transistors in the chip
- The CPU is only a part of the whole system, other parts might impact heavily on power and energy
  - Memory access is expensive
  - Networking is expensive
  - ...

In Summary...

- Definition of architecture
- Memory architectures for embedded systems
- Definition of Microcontrollers, CPUs, DPSs, GPUs
- ARM v8, DSP and Thumb instructions; ARM-based chips