[Microprocessor Applications]
ARM Architecture

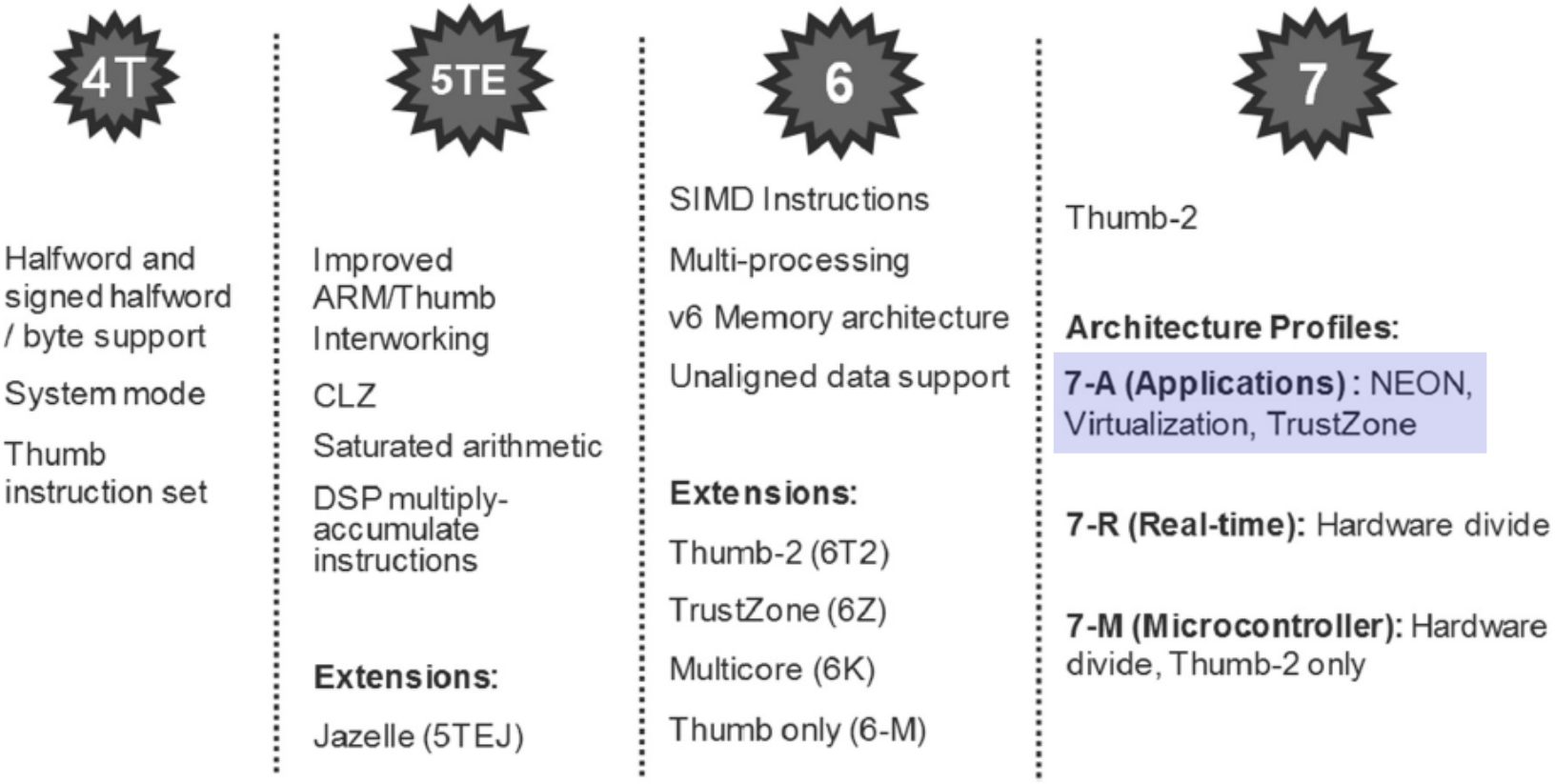
Chester Sungchung Park
SoC Design Lab, Konkuk University
Webpage: http://soclab.konkuk.ac.kr

ARM Architecture

- Architecture describes the user's view
  - **Instruction set**
  - Visible registers
  - Memory management table structures
  - Exception handling model etc.
- Organization describes the user-visible implementation of the architecture
  - Pipeline structure
  - Cache
  - Table-walking hardware
  - Translation look-aside buffer etc.

ARM Architecture

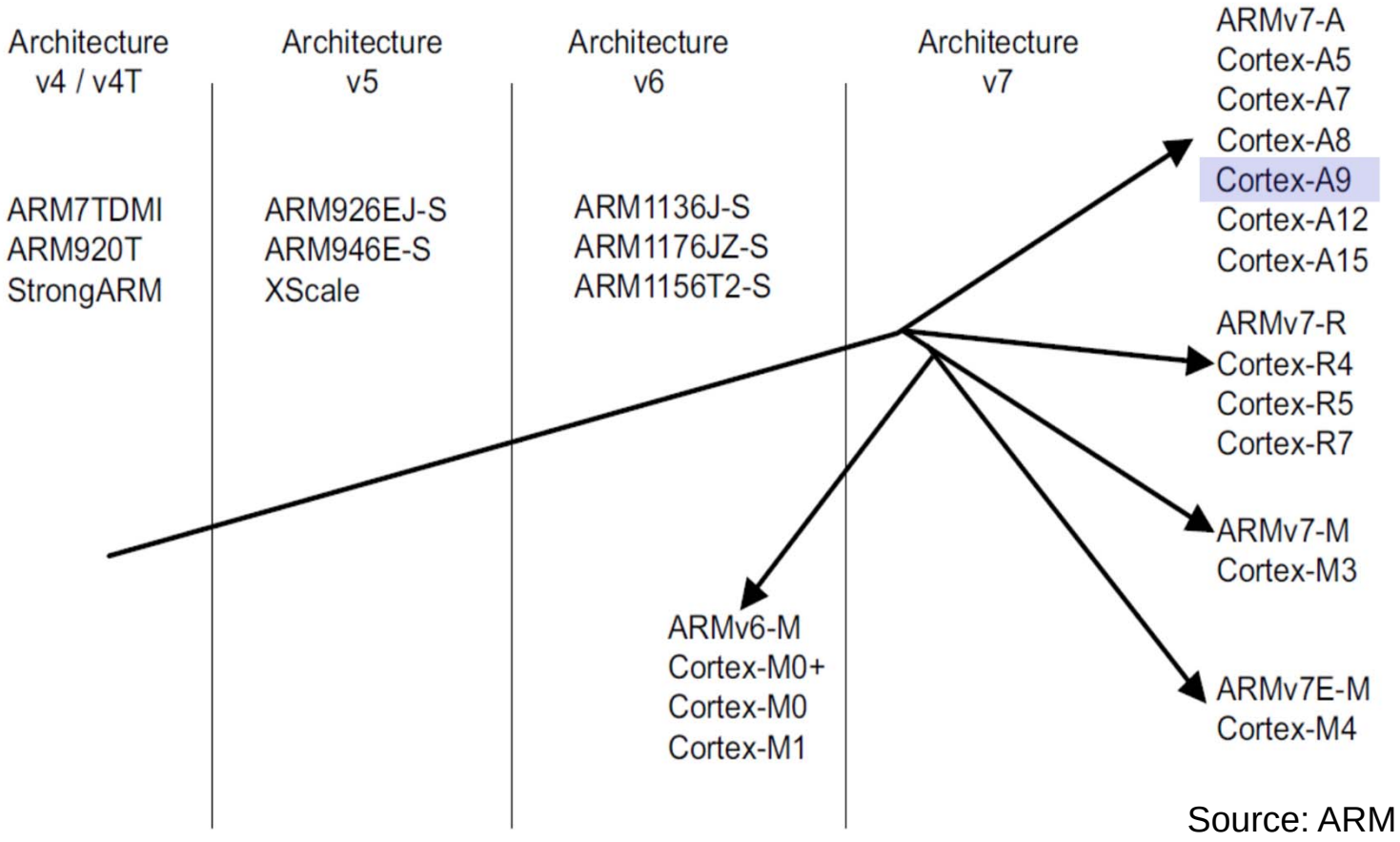
- Development of ARM architecture



Source: ARM

ARM Architecture

- Architecture and processors



Source: ARM

Cortex-A9 Processor

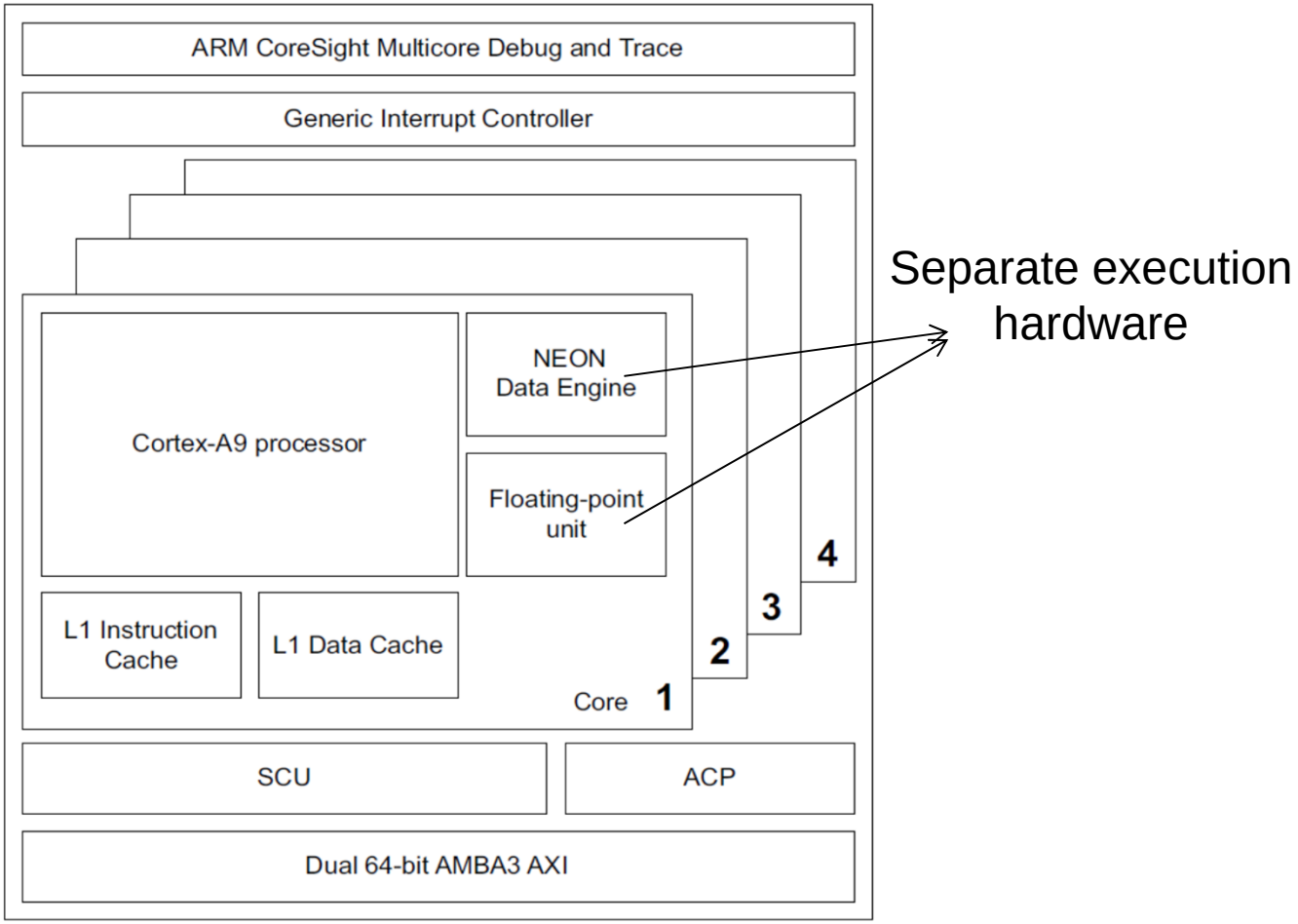
- Implements the ARMv7-A architecture that includes
  - ARM Thumb-2 32-bit instruction set architecture
    - ✓ Code density of Thumb (16-bit) & performance of ARM (32-bit)
  - Advanced SIMD architecture extension, NEON, for multimedia applications (e.g., 3D graphics, image processing)
  - Vector Floating-Point version 3 (VFPv3) architecture extension for floating-point computation (IEEE 754 standard)
  - Security extension for enhanced security
  - ARMv7 Debug architecture for Security Extensions and CoreSight

Cortex-A9 Processor

- Main features
  - Superscalar, variable length, out-of-order pipeline with dynamic branch prediction
  - Full implementation of ARMv7 instruction set
  - Harvard L1 memory with MMU
  - Two 64-bit AXI master interfaces (one for data, another for instruction)
  - ARMv7 Debug architecture
  - Support for trace with Program Trace Macrocell (PTM)
  - Support for advanced power management with up to 3 power domains
  - Optional Preload Engine
  - Optional Data Engine with MPE and VFPv3

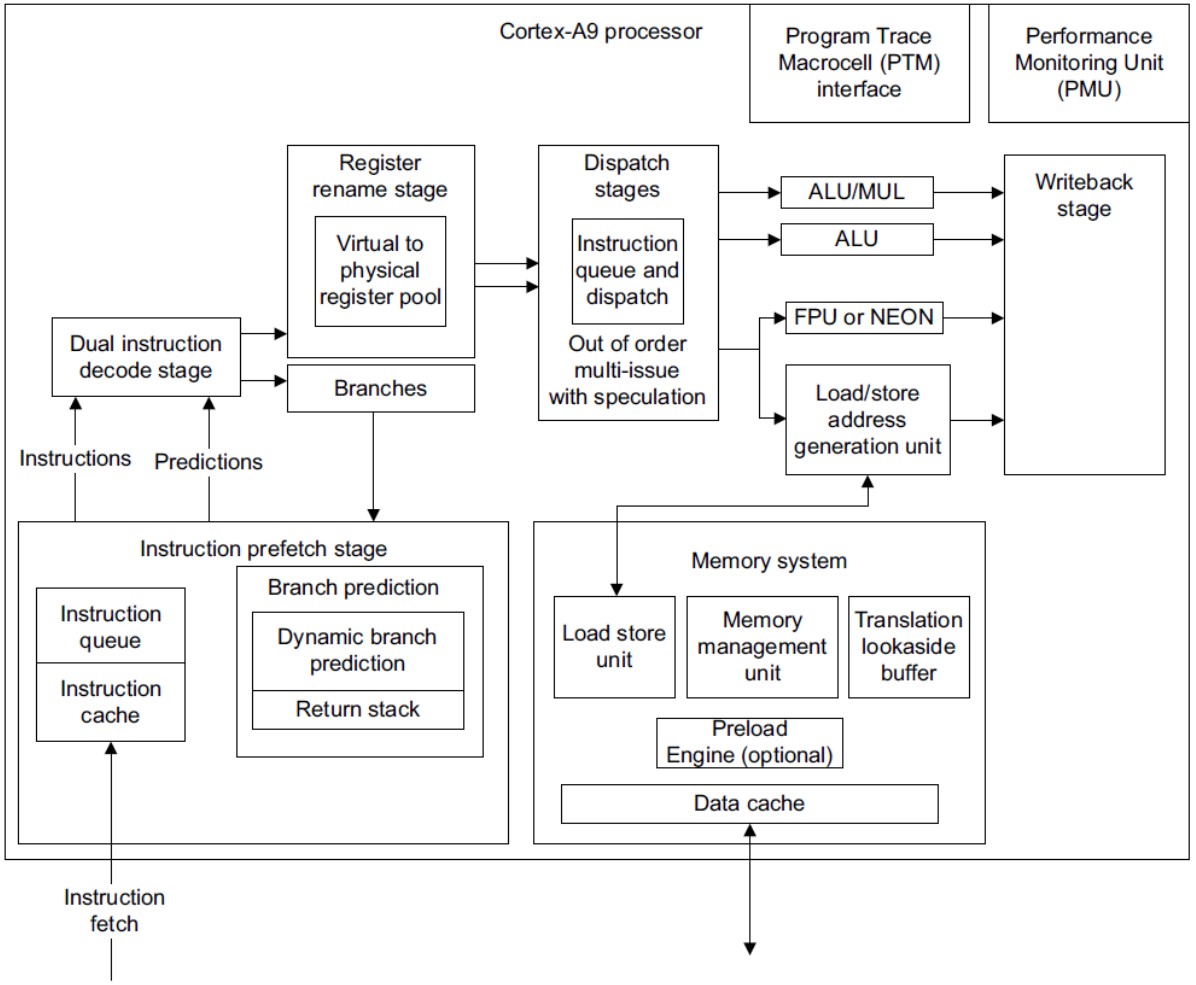
Cortex-A9 Processor

- Block diagram



Cortex-A9 Processor

- Block diagram (more detailed)



Data Types & Arithmetic

- Data types in memory
  - Byte (8 bits)
  - Halfword (16 bits)
  - Word (32 bits)
  - Doubleword (64 bits)
- Data types in registers (32-bit)
  - One 32-bit pointer
  - One unsigned/signed 32-bit integer
  - One unsigned 16-bit/8-bit integer (zero-ext.)
  - One signed 16-bit/8-bit integer (sign-ext.)
  - Two 16-bit integers
  - Four 8-bit integers
  - Half of 64-bit unsigned/signed integer

ARM Processor Modes

- The current mode determines the set of available registers and the privilege of the executing software

Processor mode		Encoding	Privilege level	Implemented	Security state
User	usr	10000	PL0	Always	Both
FIQ	fiq	10001	PL1	Always	Both
IRQ	irq	10010	PL1	Always	Both
Supervisor	svc	10011	PL1	Always	Both
Monitor	mon	10110	PL1	With Security Extensions	Secure only
Abort	abt	10111	PL1	Always	Both
Hyp	hyp	11010	PL2	With Virtualization Extensions	Non-secure only
Undefined	und	11011	PL1	Always	Both
System	sys	11111	PL1	Always	Both

Privileged modes

Exception modes

- Mode changes can be made under software control, or caused by an exception

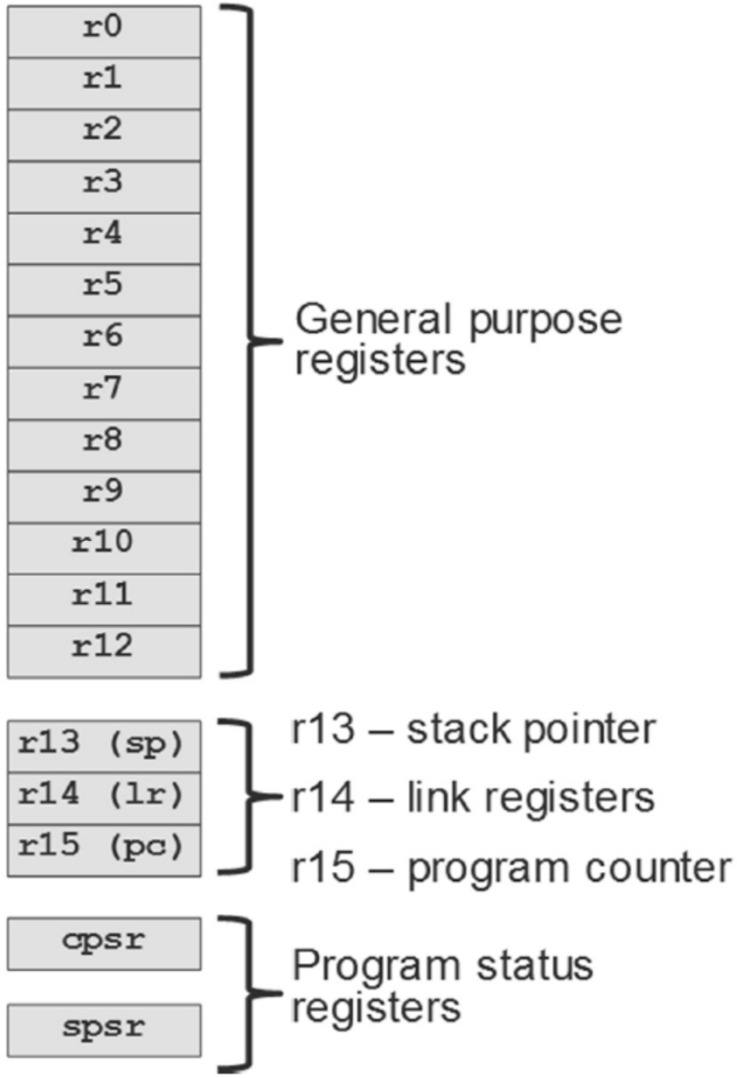
ARM Core Registers

- 16 32-bit core registers seen on the application level
  - Selected from a larger set of registers on the system level (register banking)

Application level view	System level view								
	User	System	Hyp †	Supervisor	Abort	Undefined	Monitor ‡	IRQ	FIQ
R0	R0_usr								
R1	R1_usr								
R2	R2_usr								
R3	R3_usr								
R4	R4_usr								
R5	R5_usr								
R6	R6_usr								
R7	R7_usr								
R8	R8_usr								R8_fiq
R9	R9_usr								R9_fiq
R10	R10_usr								R10_fiq
R11	R11_usr								R11_fiq
R12	R12_usr								R12_fiq
SP	SP_usr		SP_hyp	SP_svc	SP_abt	SP_und	SP_mon	SP_irq	SP_fiq
LR	LR_usr			LR_svc	LR_abt	LR_und	LR_mon	LR_irq	LR_fiq
PC	PC								
APSR	CPSR								
			SPSR_hyp	SPSR_svc	SPSR_abt	SPSR_und	SPSR_mon	SPSR_irq	SPSR_fiq
			ELR_hyp						

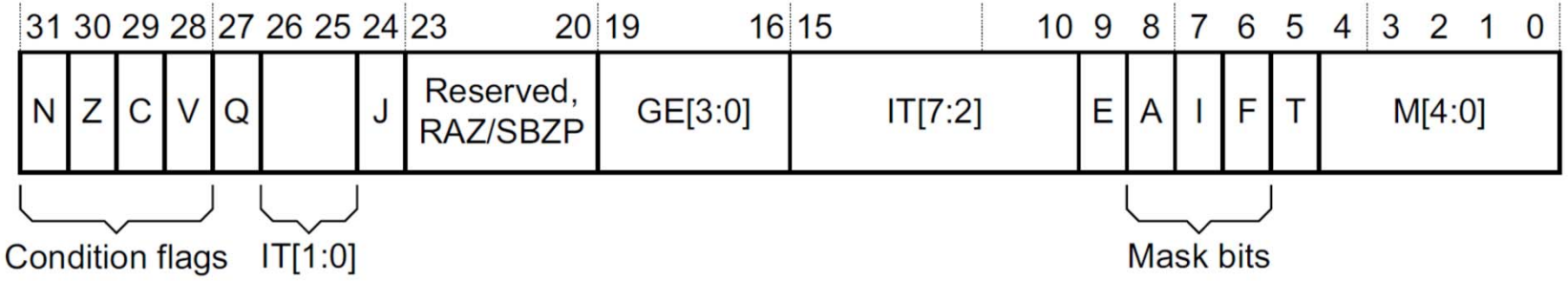
ARM Core Registers

- Application-level registers
  - r13: stack pointer (SP)
  - r14: link register (LR)
  - r15: program counter (PC)
- Program status register
  - Current PSR (CPSR)
  - Saved PSR (SPSR)
    - ✓ Only present in exception modes



ARM Core Registers

- Program status register
  - Condition flags
    - ✓N = Negative **result** from ALU **(1 if negative)**
    - ✓Z = Zero **result** from ALU **(1 if zero)**
    - ✓C = ALU operation carried out **(1 if carried out)**
    - ✓V = ALU operation overflowed **(1 if overflow/underflow)**
  - Mode field
    - ✓Current mode of processor

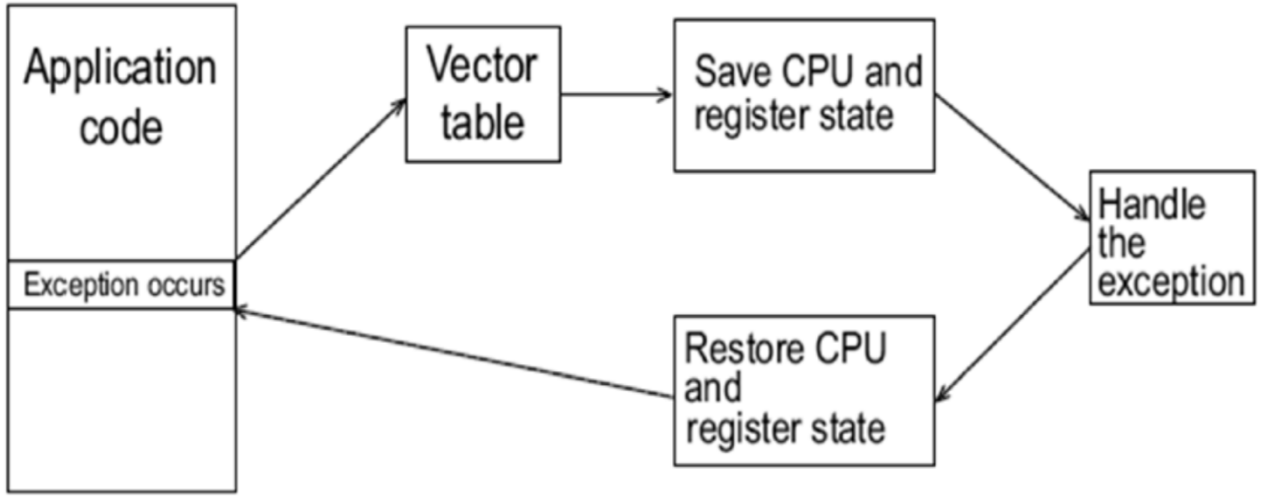


Exceptions

- Exceptions are conditions or system events that requires remedial action or an update of system status by privileged software
- When exception occurs, the system halts normal execution and instead executes a dedicated software routine (exception handler)
- Interrupt enables microprocessors to respond to external asynchronous events (e.g., button being pressed, timeout etc.)

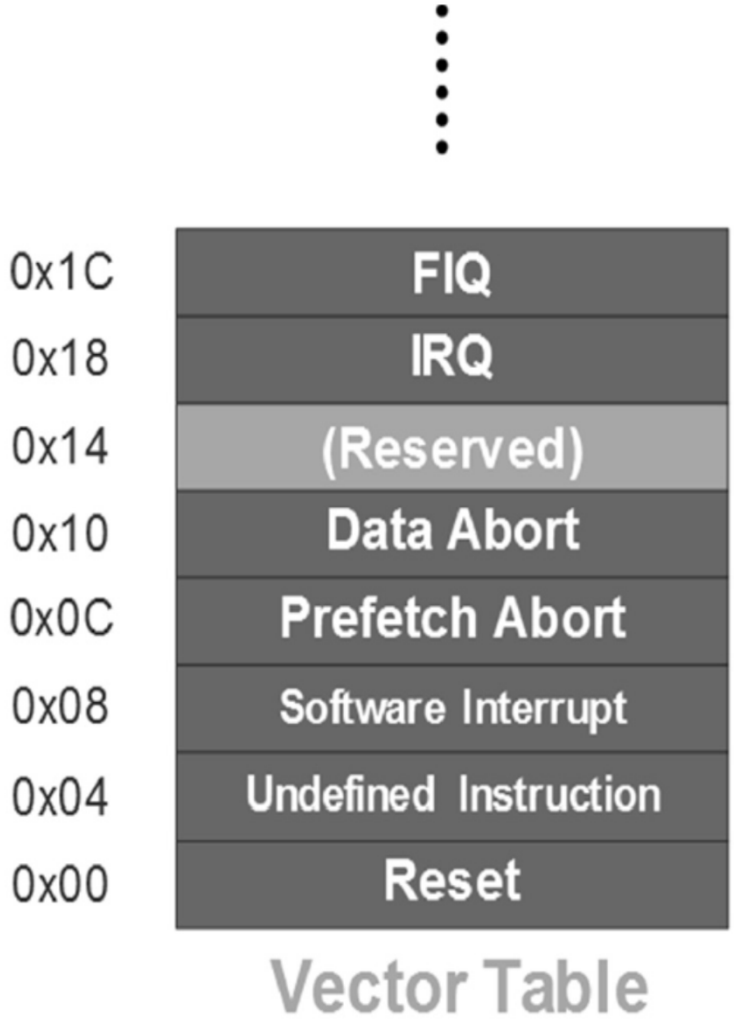
Exceptions

- Exception entry
  - Copies CPSR into SPSR
  - Sets appropriate CPSR bits
    - ✓ State, endianness, mode, interrupt etc.
  - Stores return address in LR
  - Sets PC to vector address
- Exit from handler
  - Restores CPSR from SPSR
  - Restores PC from LR



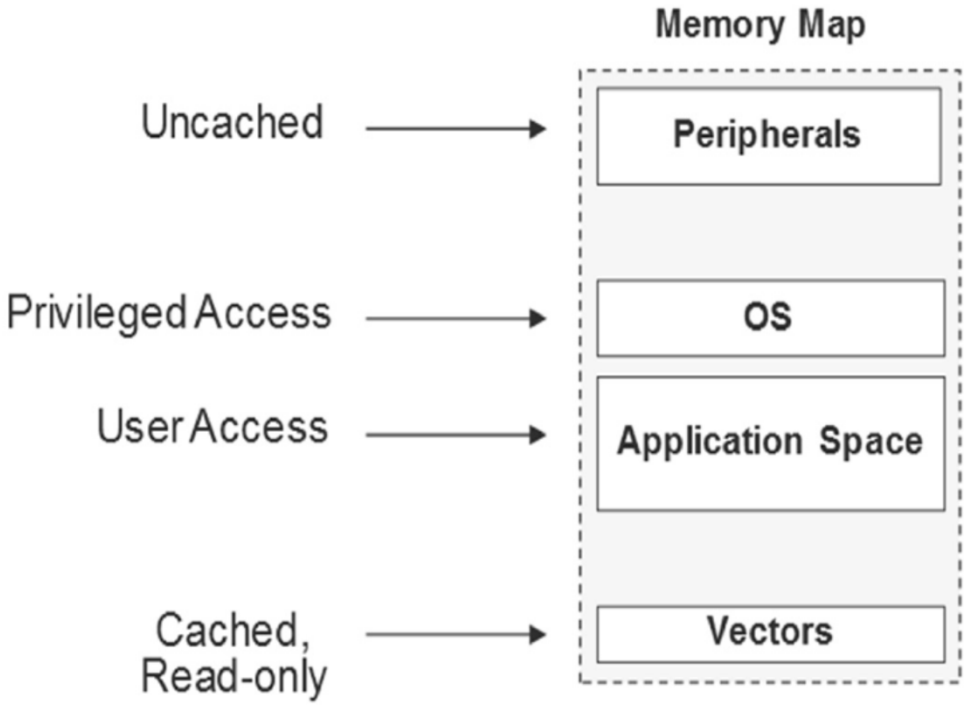
Exceptions

- Exception types
  - Reset
  - Undefined
  - Prefetch/data abort
  - Interrupt (IRQ, FIQ)
- Vector table
  - One entry per exception type
  - One instruction at each entry
  - Branch instruction except for FIQ placing the handler itself directly



Memory Model

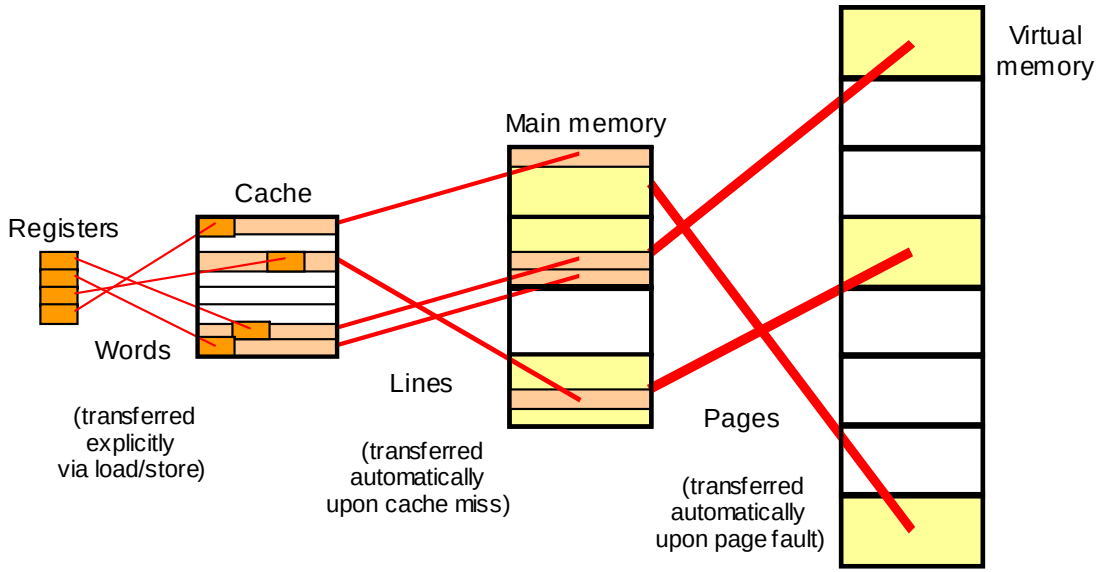
- Different types of memory and peripherals define how to access different devices
- Read/write permissions and memory types defined for each address region



Memory Model

- Memory types
  - Controls
    - ✓Memory access ordering rules
    - ✓Caching/ buffering behavior
  - Mutually exclusive memory types supported
    - ✓Normal: data, instruction
    - ✓Device: device/peripherals, data
    - ✓Strongly-ordered: device/peripherals, data

Memory Hierarchy

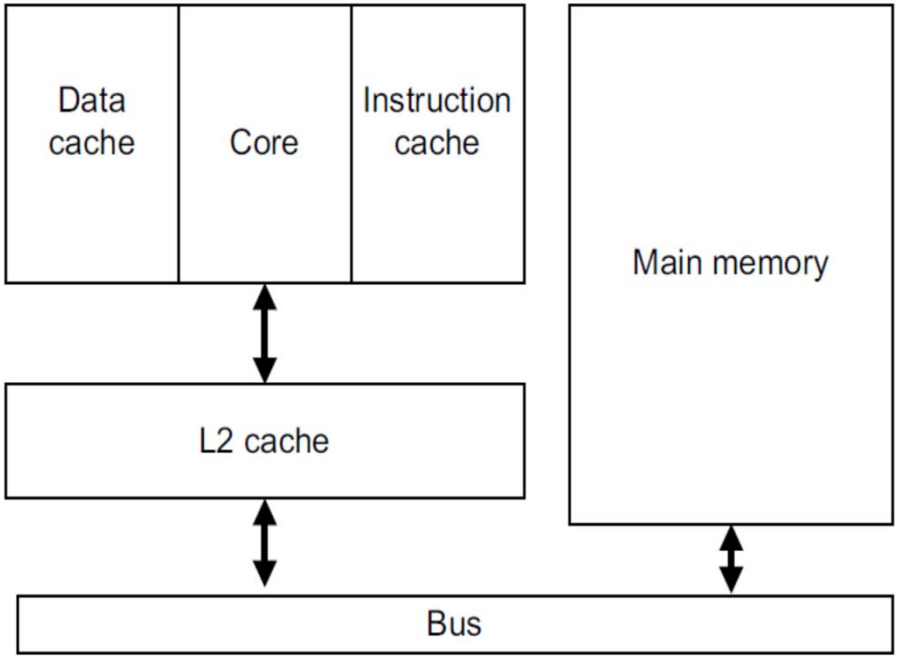


- ✓ Effective **speed-up**
- ✓ **Cache line** (4~64 B)
- ✓ Hit rate: **90~98%**
- ✓ **Set** associative
- ✓ **Pipeline stall** w/ miss

- ✓ Effective **size-up**
- ✓ **Page** (4~64 KB)
- ✓ Hit rate: **99.9%**
- ✓ **Fully** associative
- ✓ **Context switch** w/ miss

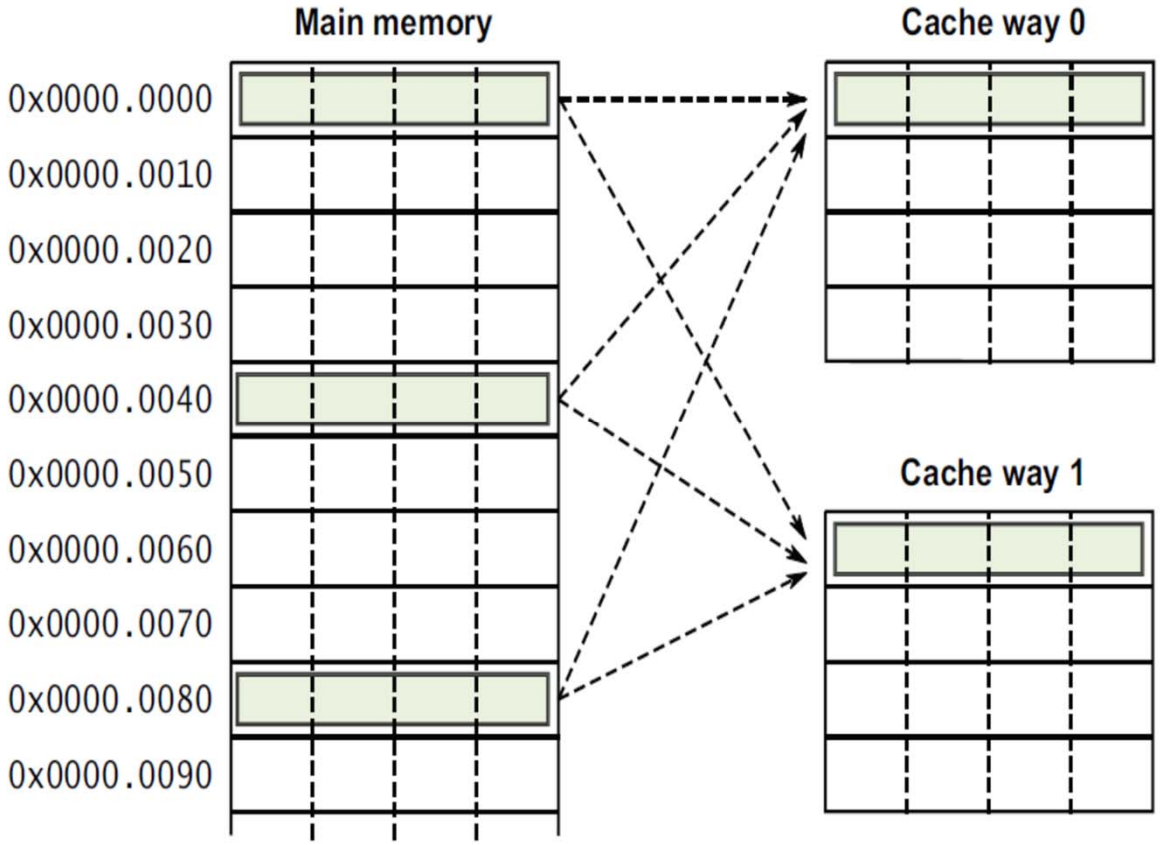
Cache

- L2 cache
  - External & von-Neumann
  - 0KB to 1MB
  - 8 words per line
- L1 cache
  - Integrated & Harvard
  - 16KB/32KB/64KB
  - 4-way set-associative
  - 8 words per line



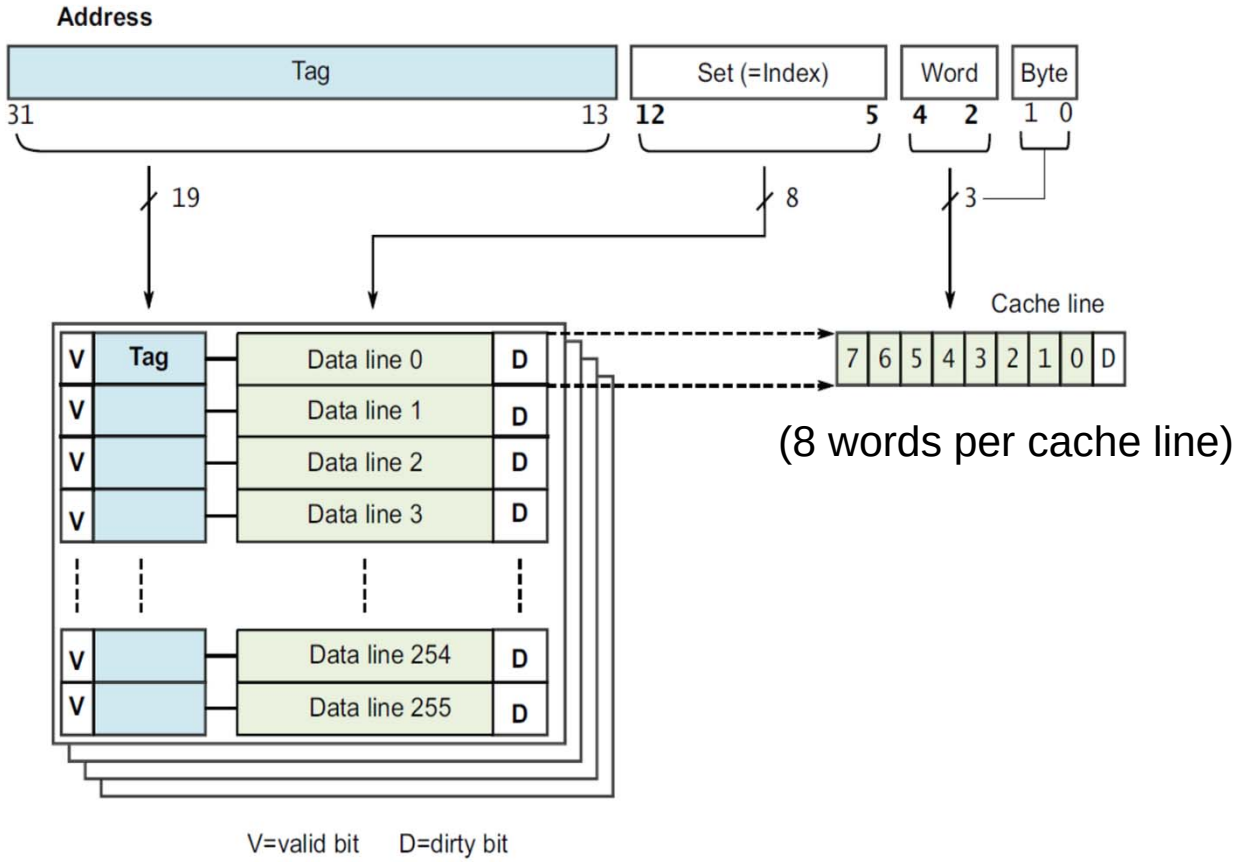
Cache

- Set associative cache



Cache

- Set-associative cache (cont'd)
  - Example: 32KB 4-way set-associative

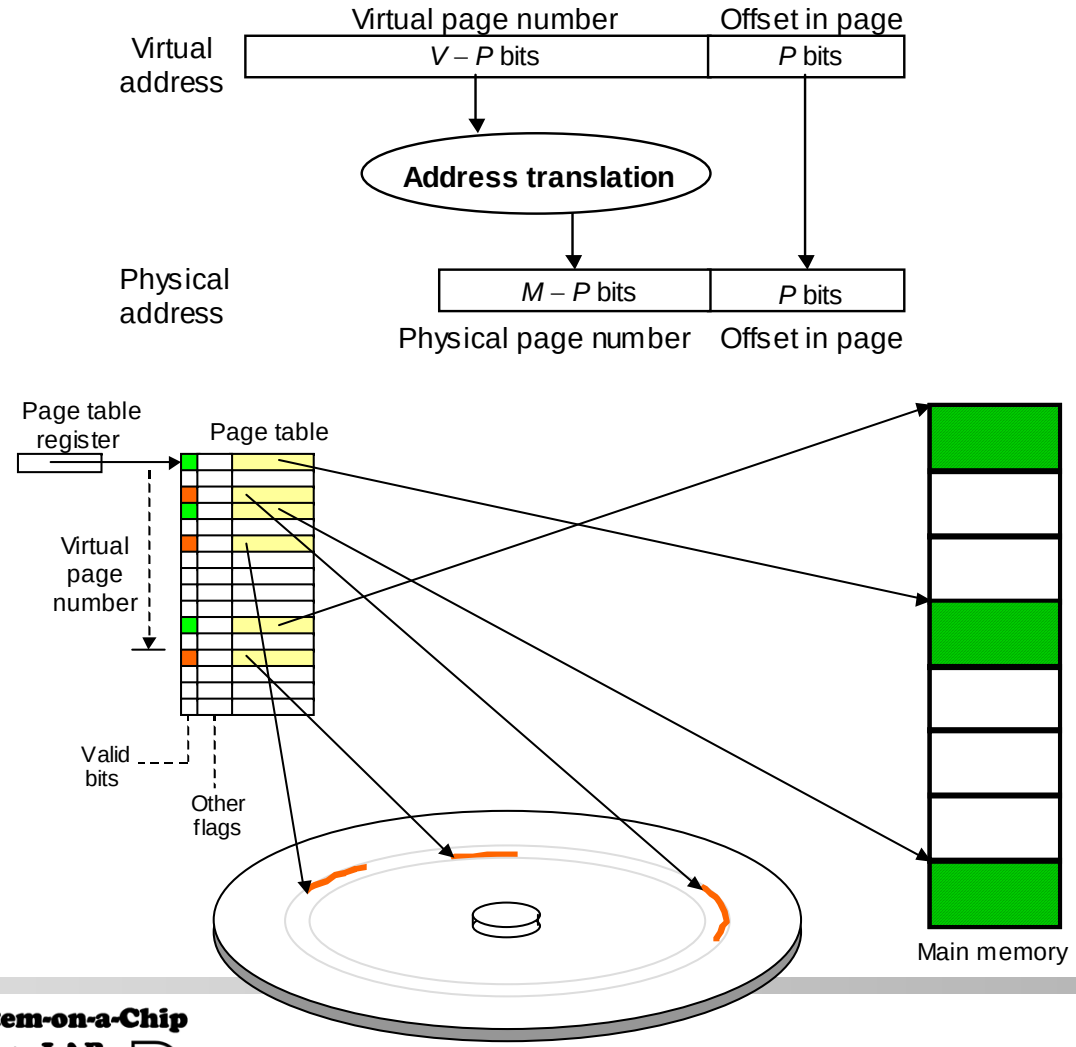


Cache

- **Allocation**: *what causes line-fill*
  - **Read Allocate**: line-fill only on read misses
  - **Write Allocate**: line-fill on both read & write misses
- **Replacement**: *which line to replace (victim)*
  - **Round Robin**: victim counter
  - **Pseudo Random**: random selection
  - **LRU**: least recently used line
- **Write**: *what happens on write hits*
  - **Write Through**: writes to both cache and upper level
  - **Write Back**: writes to cache only

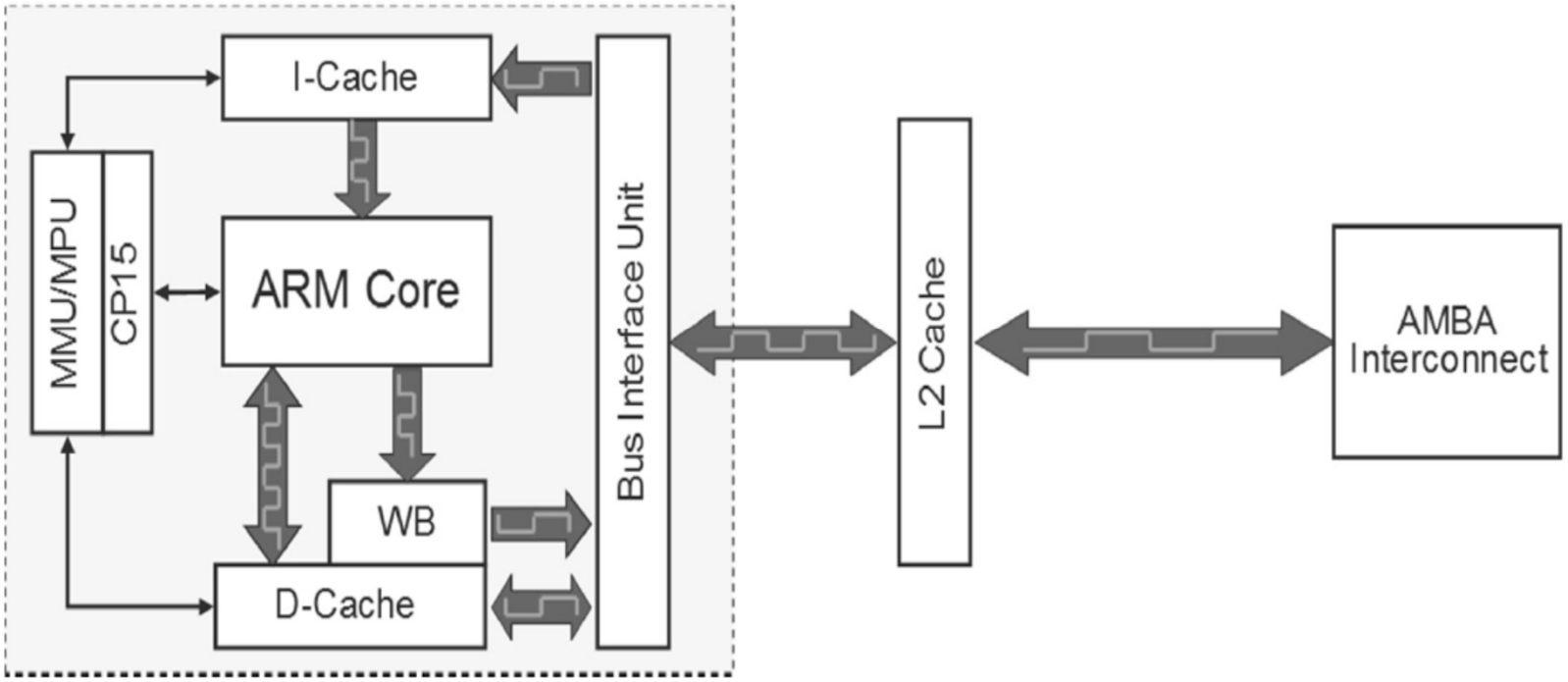
Memory Management

- Mapping virtual to physical addresses



Memory Management

- Memory management unit (MMU)
  - D$: physically-indexed, physically tagged (PIPT)
  - I$: virtually-indexed, physically tagged (VIPT)

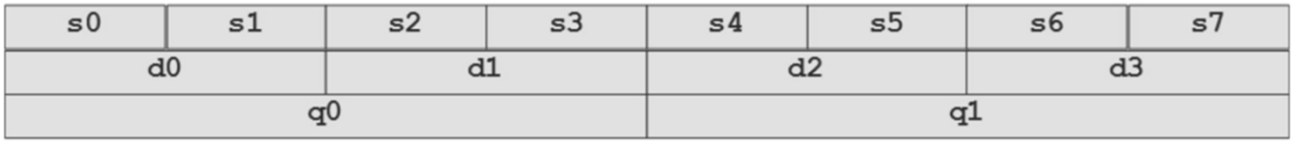


Pipelines

- 8 pipeline stages for Cortex-A9
- Several effects from longer pipeline
  - Branch penalty (due to pipeline flush) increased
  - More complex instruction scheduling
- Branch prediction to avoid pipeline flushes
  - Static & dynamic branch prediction
  - Dynamic branch prediction updates state machine according to branch history
  - Branch target address cache (BTAC) contains recent branches and results
  - Return stack for subroutine return addresses

Floating Point & NEON

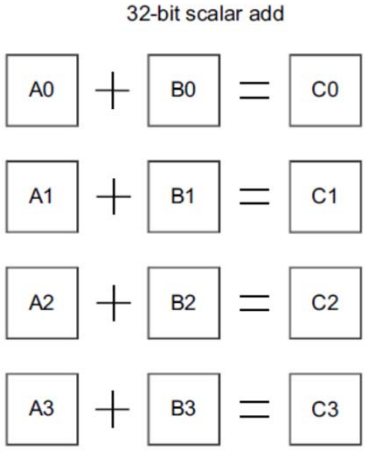
- Optional floating point support (VFPv3)
- Optional NEON
  - Single Instruction Multiple Data (SIMD) extension
- VFP and NEON instructions use a separate set of registers to standard data processing instructions
  - One bank of registers, three ways of accessing them
    - ✓ s<n> 32-bit (single precision)
    - ✓ d<n> 64-bit (double precision)
    - ✓ q<n> 128-bit, NEON only

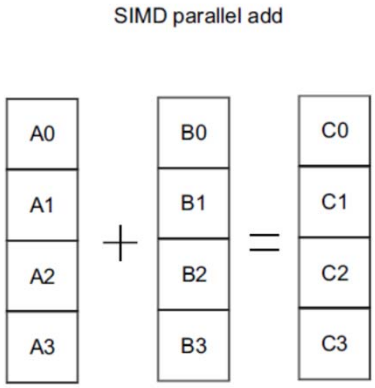


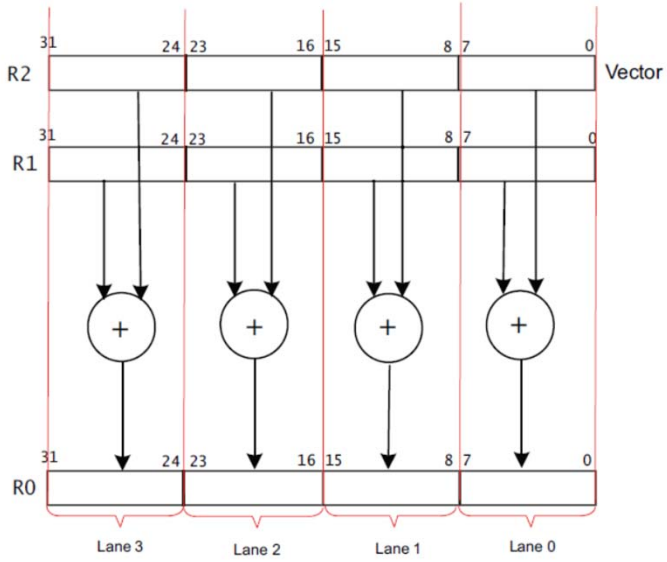
  - Number of registers is implementation-defined

NEON

- SIMD enables a single instruction treats a register value as multiple data elements







Coprocessor

- For earlier ARM processors, additional coprocessors possibly added to expand the instruction set
- For newer ARM processors, **no** user-defined coprocessors allowed
  - Usually better for system designers to use memory mapped peripherals (easier to implement, being free from pipeline concern) → **accelerator**
- ARM coprocessors for internal functions
  - System control (cp15): system ID, caches, MMU, TCMs etc.
  - Debug (cp14): access to debug control registers
  - VFP and NEON (cp10 and cp11)

Instruction Cycle Timing

- Cycle timings are generally complicated to hand calculate
  - Dependent on surrounding instructions in pipeline
  - Memory system activity (e.g., cache miss leading to pipeline stall for tens of cycles) also affects cycle timings
- Execution unit cycles
  - 1~3 cycles for data processing
  - 1~5 cycles for (single) load/store
  - 1~5 cycles for multiplications

References

- Cortex-A9 technical reference manual, ARM DDI0388F
- ARM architecture reference manual, ARM DDI0100I
- ARM architecture reference manual (ARMv7A and ARMv7-R edition), ARM DDI0406C
- ARM Cortex-A series, programmer's guide, ARM DEN0013D