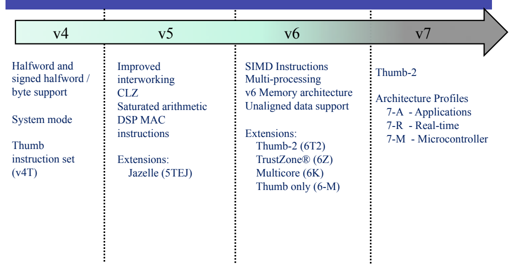
Section 7. Memory System

Memory Organizations and Cache

Development of the ARM Architecture



v4: Halfword and signed halfword / byte support; System mode; Thumb instruction set (v4T)

v5: Improved interworking; CLZ; Saturated arithmetic; DSP MAC instructions; Extensions: Jazelle (5TEJ)

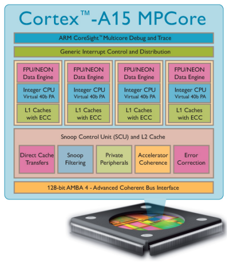
v6: SIMD Instructions; Multi-processing; v6 Memory architecture; Unaligned data support; Extensions: Thumb-2 (6T2), TrustZone® (6Z), Multicore (6K), Thumb only (6-M)

v7: Thumb-2; Architecture Profiles: 7-A - Applications, 7-R - Real-time, 7-M - Microcontroller

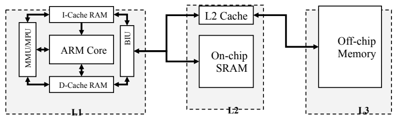
- **Note that implementations of the same architecture can be different**
  - Cortex-A8 - architecture v7-A, with a 13-stage pipeline
  - Cortex-A9 - architecture v7-A, with an 8-stage pipeline

Cortex-A15 MPCore

- 1-4 processors per cluster
- Fixed size L1 caches (32KB)
- Integrated L2 Cache
  - 512KB – 4MB
- System-wide coherency support with AMBA 4 ACE
- Backward-compatible with AXI3 interconnect
- Integrated Interrupt Controller
  - 0-224 external interrupts for entire cluster
- CoreSight debug
- Advanced Power Management
- Large Physical Address Extensions (LPAE) to ARMv7-A Architecture
- Virtualization Extensions to ARMv7-A Architecture

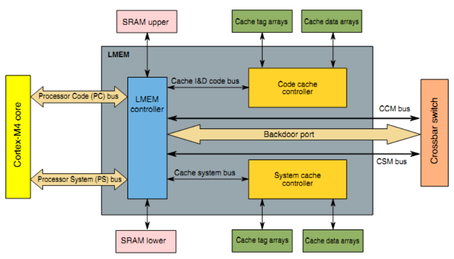

Cortex™-A15 MPCore
ARM CoreSight™ Multicore Debug and Trace
Generic Interrupt Control and Distribution
FPU/NEON Data Engine
Integer CPU Virtual 40b PA
L1 Caches with ECC
Snoop Control Unit (SCU) and L2 Cache
Direct Cache Transfers
Snoop Filtering
Private Peripherals
Accelerator Coherence
Error Correction
128-bit AMBA 4 - Advanced Coherent Bus Interface

L1 and L2 Caches

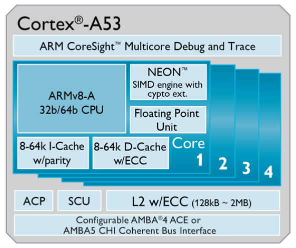


- Typical memory system can have multiple levels of cache
  - Level 1 memory system typically consists of L1-caches,
  - Level 2 memory system (and beyond) depends on the system design
- Memory attributes determine cache behavior at different levels
  - Controlled by the MMU
  - Inner Cacheable attributes define memory access behavior in the L1 memory system
  - Outer Cacheable attributes define memory access behavior in the L2 memory system (if external) and beyond (as signals on the bus)
- Before caches can be used, software setup must be performed

Local Memory Controller

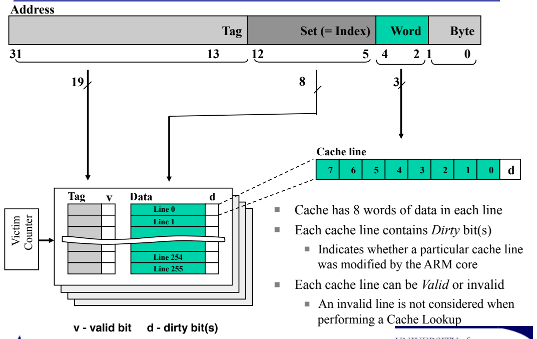


Cortex A53 Architecture


Cortex®-A53
ARM CoreSight™ Multicore Debug and Trace
ARMv8-A 32b/64b CPU
NEON™ SIMD engine with cypto ext.
Floating Point Unit
8-64k I-Cache w/parity
8-64k D-Cache w/ECC
Core 1
2
3
4
ACP
SCU
L2 w/ECC (128kB ~ 2MB)
Configurable AMBA®4 ACE or AMBA5 CHI Coherent Bus Interface

- L1 Cache, 64B line size
  - 2-way set-assoc I-Cache
  - 4-way set-Assoc D-Cache with prefetch engine
- L2 Cache, 64B line size
  - 16-way set-assoc
  - Snoop control unit SCU

Example 32KB ARM cache

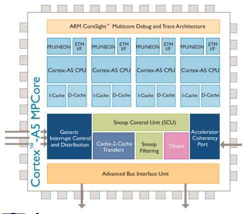


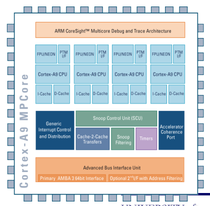
- Cache has 8 words of data in each line
- Each cache line contains *Dirty* bit(s)
  - Indicates whether a particular cache line was modified by the ARM core
- Each cache line can be *Valid* or invalid
  - An invalid line is not considered when performing a Cache Lookup

UNIVERSITY of Rhode Island

Cortex MPCore Processors

- Standard Cortex cores, with additional logic to support MPCore
  - Available as 1-4 CPU variants
- Include integrated
  - Interrupt controller
  - Snoop Control Unit (SCU)
  - Timers and Watchdogs

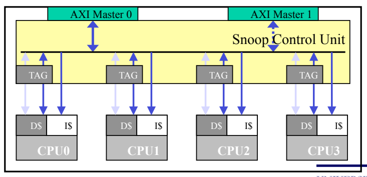

ARM CoreSight™ Multicore Debug and Trace Architecture
FPU/NEON
ETM I/F
Cortex-A5 CPU
I-Cache
D-Cache
Snoop Control Unit (SCU)
Generic Interrupt Control and Distribution
Cache-2-Cache Transfers
Snoop Filtering
Timers
Accelerator Coherency Port
Advanced Bus Interface Unit
Cortex-A5 MPCore


ARM CoreSight™ Multicore Debug and Trace Architecture
FPU/NEON
PTM I/F
Cortex-A9 CPU
I-Cache
D-Cache
Snoop Control Unit (SCU)
Generic Interrupt Control and Distribution
Cache-2-Cache Transfers
Snoop Filtering
Timers
Accelerator Coherence Port
Advanced Bus Interface Unit
Cortex-A9 MPCore

UNIVERSITY of Rhode Island

Snoop Control Unit

- The Snoop Control Unit (SCU) maintains coherency between L1 data caches
  - Duplicated Tag RAMs keep track of what data is allocated in each CPU's cache
    - Separate interfaces into L1 data caches for coherency maintenance
  - Arbitrates accesses to L2 AXI master interface(s), for both instructions and data
- Optionally, can use address filtering
  - Directing accesses to configured memory range to AXI Master port 1



UNIVERSITY of Rhode Island

Memory System Architecture

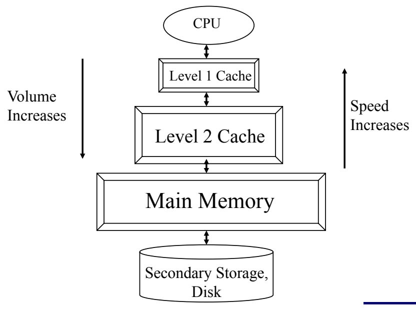
- Concepts of memory hierarchy
  - Quantitative principles of computer design
    - Smaller is faster
    - Amdahl's Law:

      If we make an enhancement on a part of a computer, the overall performance gain is limited by the faction of time when the enhancement part is used.
    - locality properties:

      Spatial locality and temporal locality
  - Speed gap and the principles suggest memory hierarchy
- Design of cache memories
  - Placements or Mapping
    - Direct-mapped, set-associative mapped, and associative
  - Replacement algorithms and cache consistency

UNIVERSITY of Rhode Island 10

Memory System Architecture



UNIVERSITY of Rhode Island 11

Design of Memory Hierarchy

- Memory access time:

$T_{acc} = T_{hit} * hit\_ratio + T_{miss} * (1-hit\_ratio)$

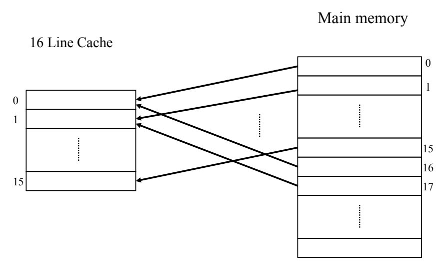
- Increase cache hit ratio
- Minimize miss penalty
- Design of cache memories
  - Placements or Mapping
  - Replacement algorithms
  - Write policy and cache consistency

UNIVERSITY of Rhode Island 12

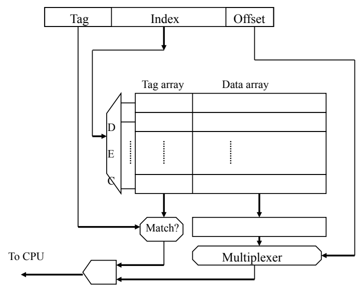
Data Placement: Cache Mapping

- Direct mapped cache:
  - Data with address A is mapped to exactly one cache location
    - A modulo C, where C is number of lines in the cache
  - Simple logic, quick access: each address has 3 fields: Tag---Index---Offset
    - Offset gives a byte in a cache line; Index identifies the cache line corresponding to the data address, and tag compares with high order bits of the address to see if they match
  - Potential line interferences since many memory blocks map to a same cache line

Direct-Mapped Cache



Hardware Organization of Direct-Mapped

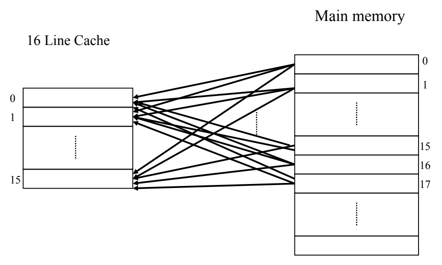


Fully Associative Mapped Cache

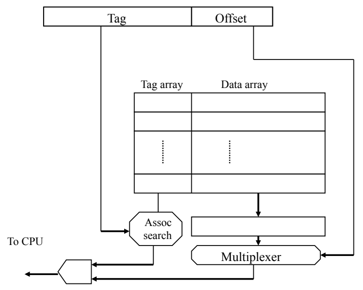
- Direct-mapped cache has high conflict misses
- Why not place a block at any free location?

→ Fully associative cache
  - No restriction as to where to place a cache line
  - each address has 2 fields: Tag---Offset
    - Cache is accessed by associative search, matching tags: content addressable memory
  - No line interferences, only capacity misses

Data Mapping of Fully Associative Cache



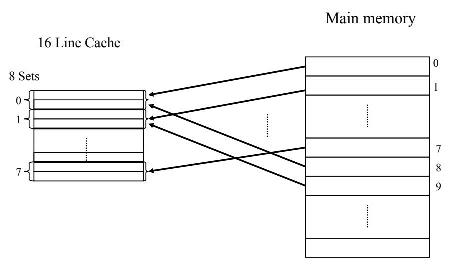
Hardware Organization of Fully Associative Cache



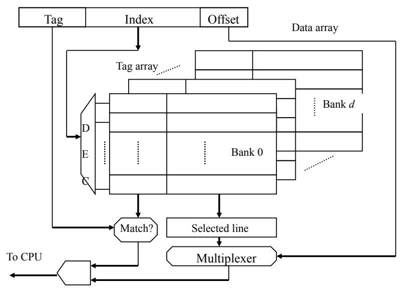
Set-Associative Mapped Cache

- A compromise/hybrid of the above two: associative map within a set and direct-map among sets
  - Flexibility with a set to place data
  - Simple indexing logic to identify a set
  - d-way associative means d lines in a set
  - Cache lines are divided into groups → sets
- Each address has 3 fields: Tag---Index---Offset
  - Index here identify which set a line mapped to
- Reduce conflict misses and less complicated/faster than fully associative

Data Mapping of Set-Associative Cache



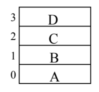
Hardware Organization of Set-Associative Cache



Replacement Algorithms

- LFU, LRU, Random, MRU, etc.
- LRU: Least Recently Used
  - An LRU counter is associated with each line
  - The LRU counters in a set form a logic stack
  - Bottom line is replaced

Memory Reference Sequence:



A B C D B

A is the LR item to be replace when CPU accesses E

LRU Counter Implementation

- Cache hit
  - The LRU counter of the referenced data is set to maximum
  - All other counters that are greater than the original counter of the referenced data are decremented by 1
- Cache miss
  - The line with counter 0 is replaced if the set is full
  - All the counters of lines in the set are decremented by 1
  - The counter of the new line is set to maximum

Write Policy

- Write-through
  - Every write updates both cached copy and memory copy
  - Write-through guarantee consistency but suffer from slow writes
- Write back
  - Write operations performed in cache only
  - Main memory updated only when changed line is replaced
  - Write-back: good performance but potential inconsistency