The ARM8

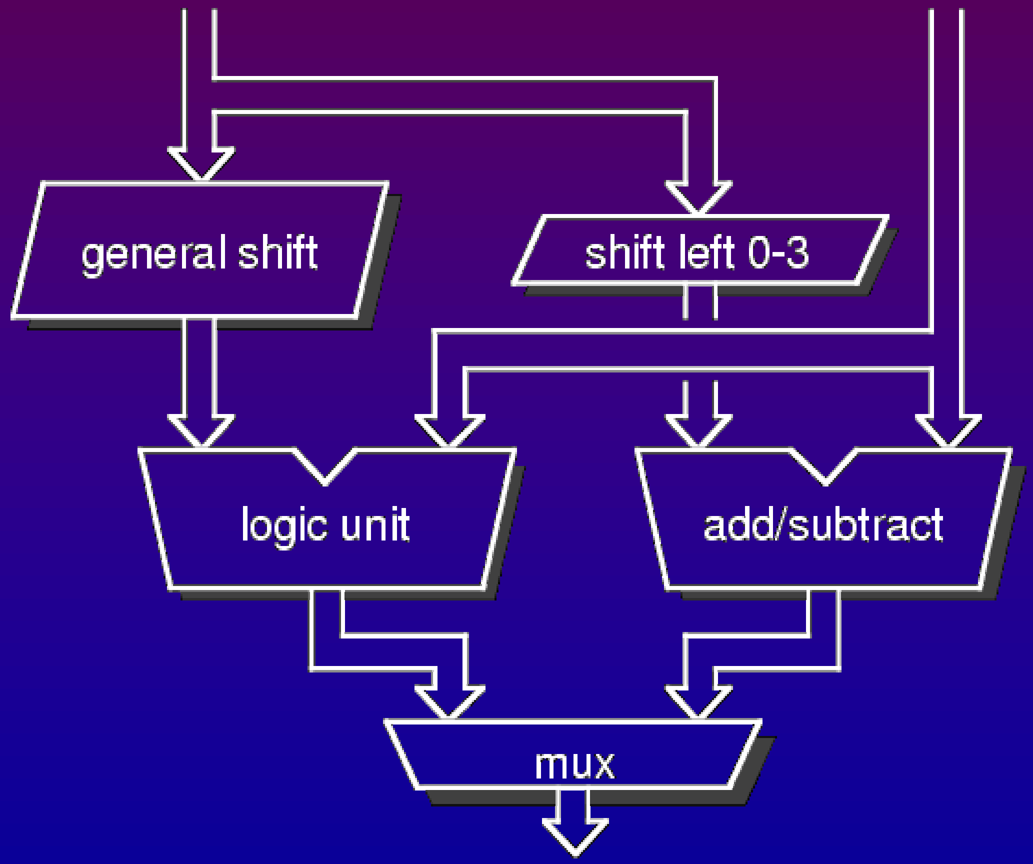
- ARM8 was designed for higher performance than ARM7 through:
  - an increased clock rate
    - simpler logic in each pipeline stage
    - a deeper pipeline
  - a reduced CPI (Clocks Per Instructions)
    - reducing the number of pipeline slots some instructions take
    - removing pipeline stalls

The ARM8 Pipeline

- ARM8 uses a 5-stage pipeline:
  - instruction fetch
    - performed by an autonomous prefetch unit
  - instruction decode and register read
  - execute (shift and ALU)
    - optimized to fit into a single pipeline stage
  - data memory access
  - write-back results

The ARM8 Pipeline (2)

- The optimized ALU/shifter arrangement
  - only carefully chosen shifts in series with adder



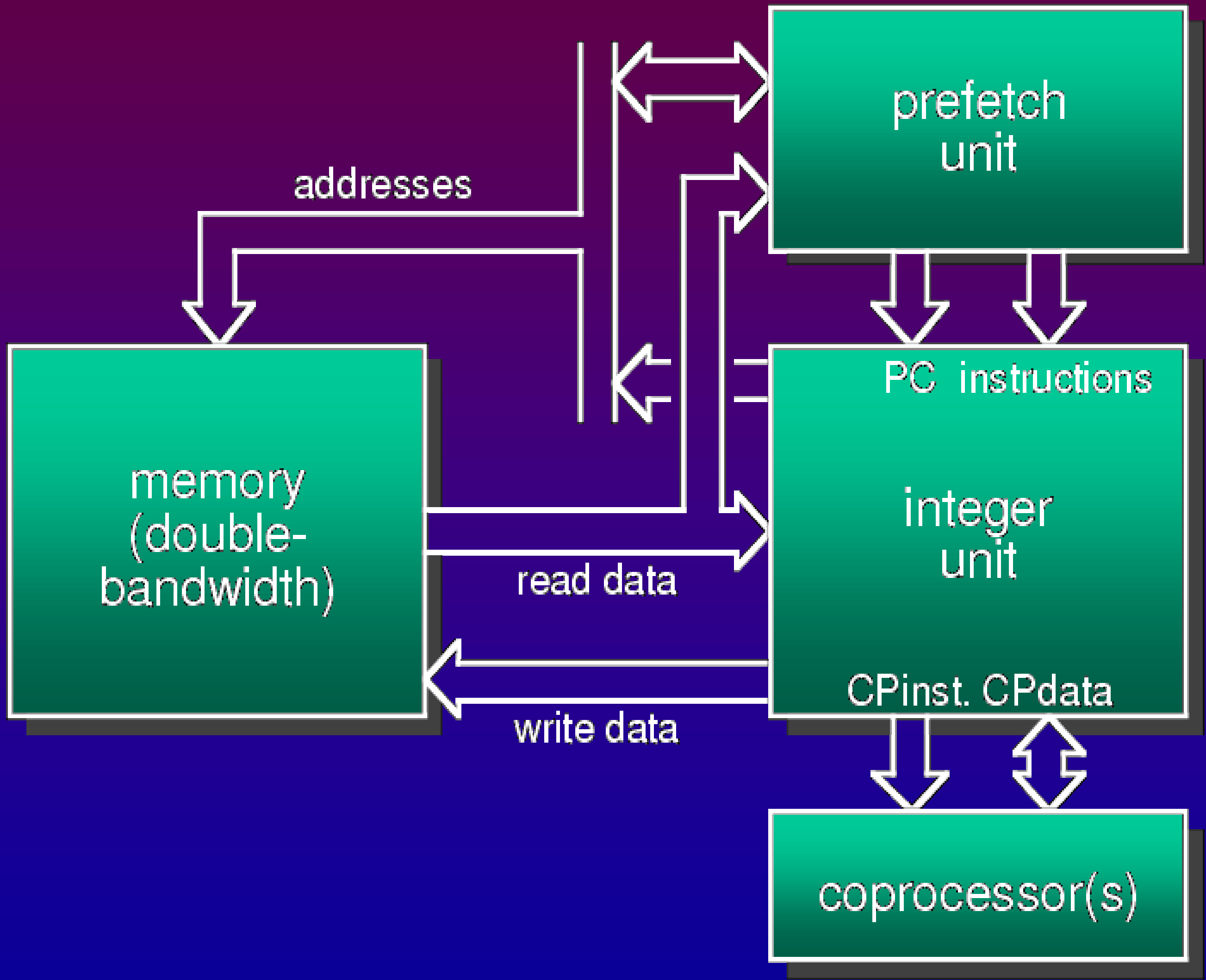
The ARM8 (2)

- Reducing the CPI
  - ARM7 uses the memory on nearly every clock cycle
    - for either instruction fetch or data transfer
  - therefore a reduced CPI requires
    more than one memory access per clock cycle
- Possible solutions are:
  - separate instruction and data memories
  - double-bandwidth memory

The ARM8 (3)

- Double-bandwidth memory
  - exploits the sequential nature of accesses
    - instruction fetches are mainly sequential
    - load and store multiples are sequential
  - each clock cycle allows a random access
  - the next sequential word is available half a cycle later
    - lower cost than, and equivalent performance to, a 64-bit memory

ARM8 Processor Core



ARM9TDMI pipeline

- ARM7TDMI pipeline:

Instruction Fetch	Thumb ARM decompress	ARM decode Reg Select	Reg Read	Shift	ALU	Reg Write
FETCH	DECODE		EXECUTE			

- ARM9TDMI pipeline:

Instruction Fetch	ARM or Thumb Inst Decode		Shift + ALU	Memory Access	Reg Write	
	Reg Decode	Reg Read				
FETCH	DECODE		EXECUTE	MEMORY	WRITE	

  – Thumb instructions are decoded directly

ARM9TDMI

- EmbeddedICE
  - as ARM7TDMI, plus:
    - hardware single-stepping
    - breakpoints on exceptions
- On-chip coprocessor support:
  - for floating-point, DSP, and so on

Process	0.35 µm	Transistors	111.000	MIPS	220
Metal layers	3	Die area	5 mm^2	Power	<800 mW
Vdd	3.3 V	Clock	0 to 200 MHz	MIPS/W	>280

ARM10

- Targets multi-media digital consumer applications
  - high-performance hand-held devices (organizers, smart phones)
  - set top boxes
  - sophisticated UI and 2D-/3D-graphics rendering
  - high performance printers
- Vector floating point Copro (VFP 10) delivering 600 MFLOPS
- Parallel instruction execution

ARM System Design

- History of ARM
- ARM Instruction Set
- Thumb Instruction Set
- ARM Cores
- ARM Cache Modeling
- ARM CPUs
- ARM Coprocessors

Cache Modeling

- Memory hierarchy
- Cache organization
- ARMulator
- Cache modeling using Cheetah

Memory hierarchy

- A typical system has several different memory subsystems:
  - processor registers:~100 bytes, 2ns
    - access is a small part of a clock cycle
  - on-chip cache or RAM: ~10 Kbytes, 10ns
    - accessed at the processor clock rate
  - off-chip ROM and RAM : ~Mbytes, 100ns
    - access costs several processor cycles
  - backup store: ~ GBytes, 10ms

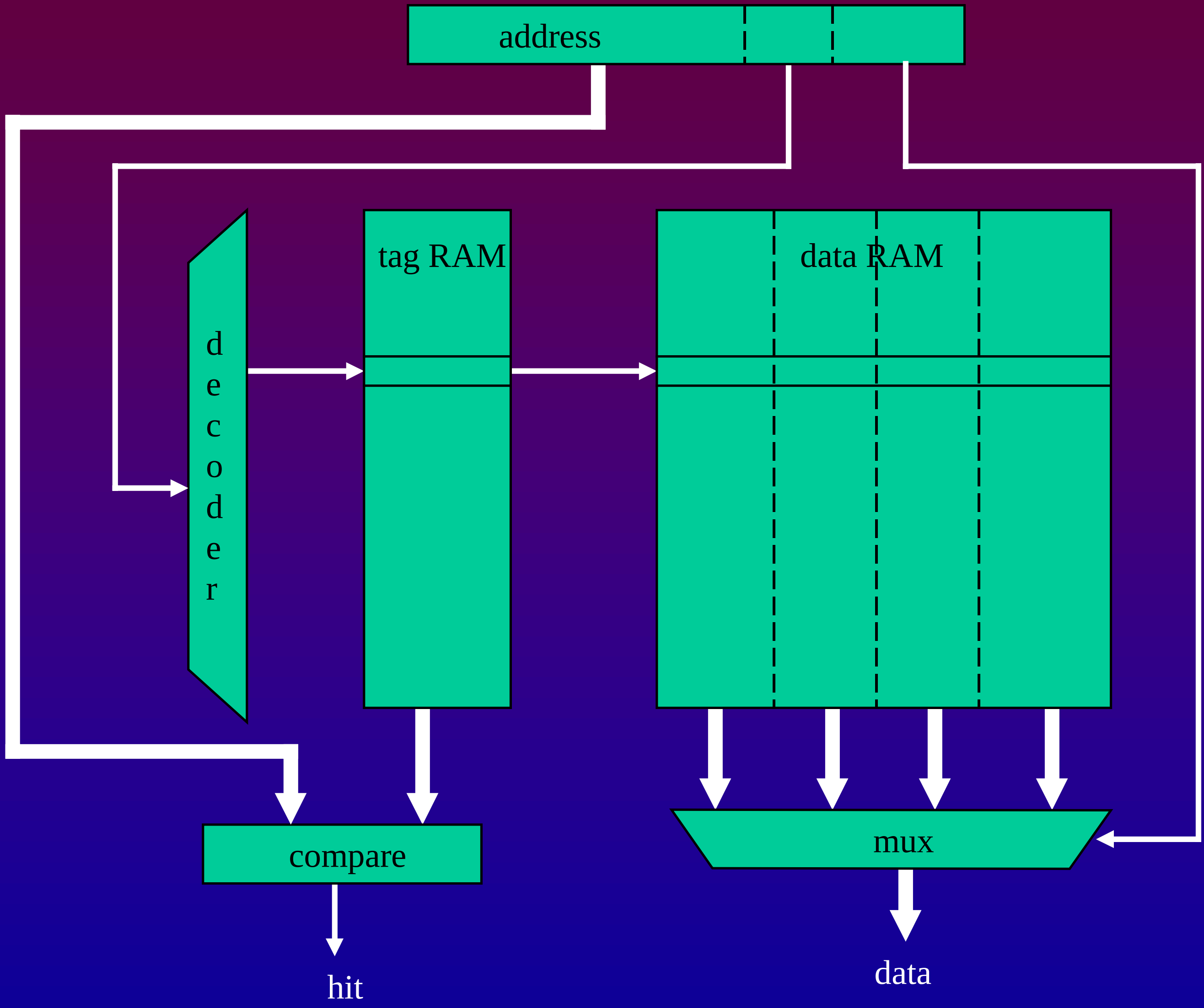
Memory hierarchy

- The objective is to approach:
  - the performance of the fasted memory...
  - ...at the cost/bit of the slowest memory
- Feasible because programs display:
  - *temporal locality*
    - accesses to a location are clustered in time
  - *spatial locality*
    - accesses are clustered in the address space

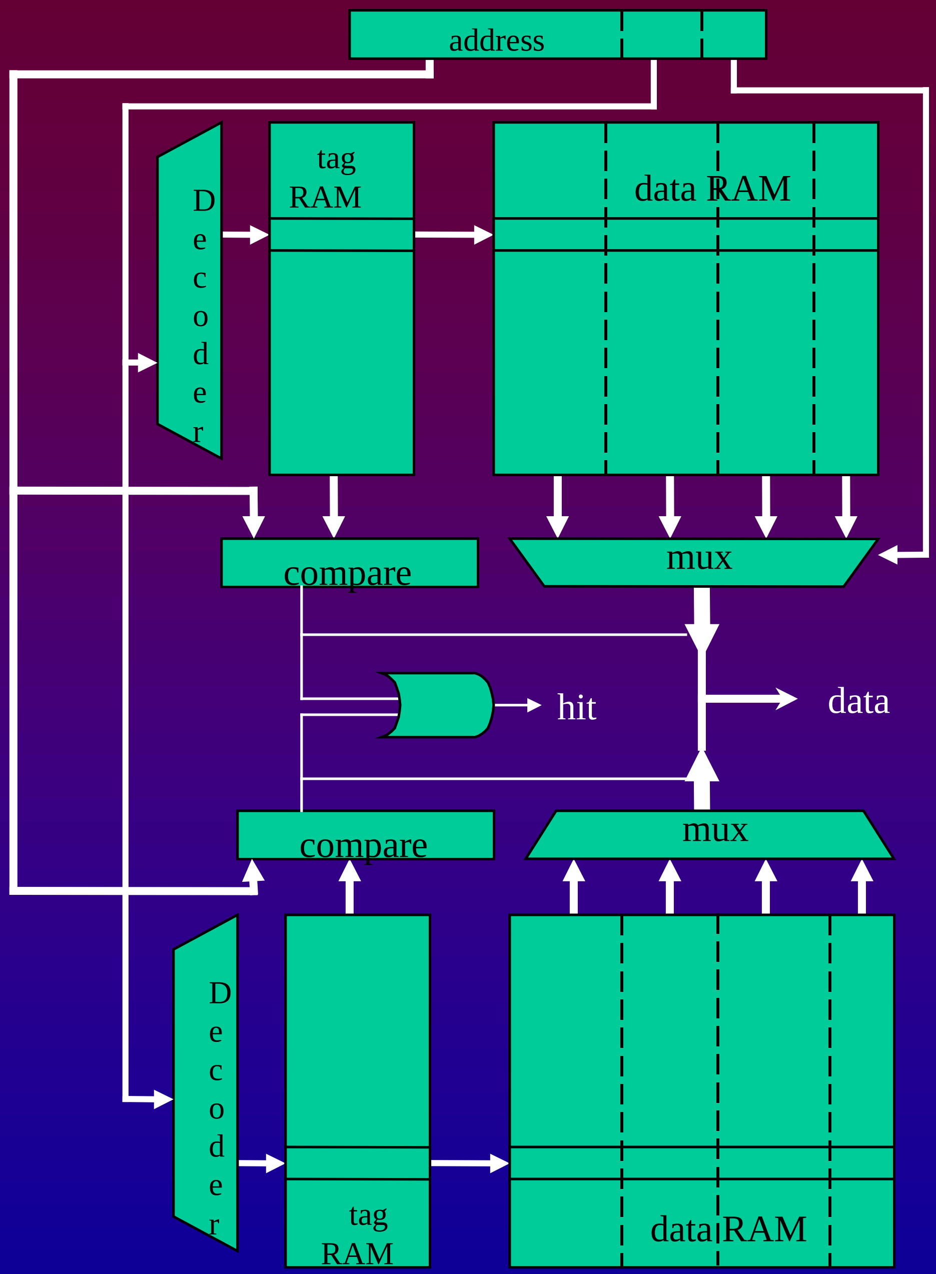
Cache organization

- There are many ways to arrange a cache:
  - separate or mixed instructions and data?
  - How much memory should be loaded on a cache *miss*?
  - How flexible should the allocation of cache space be?
  - How should writes be handled?

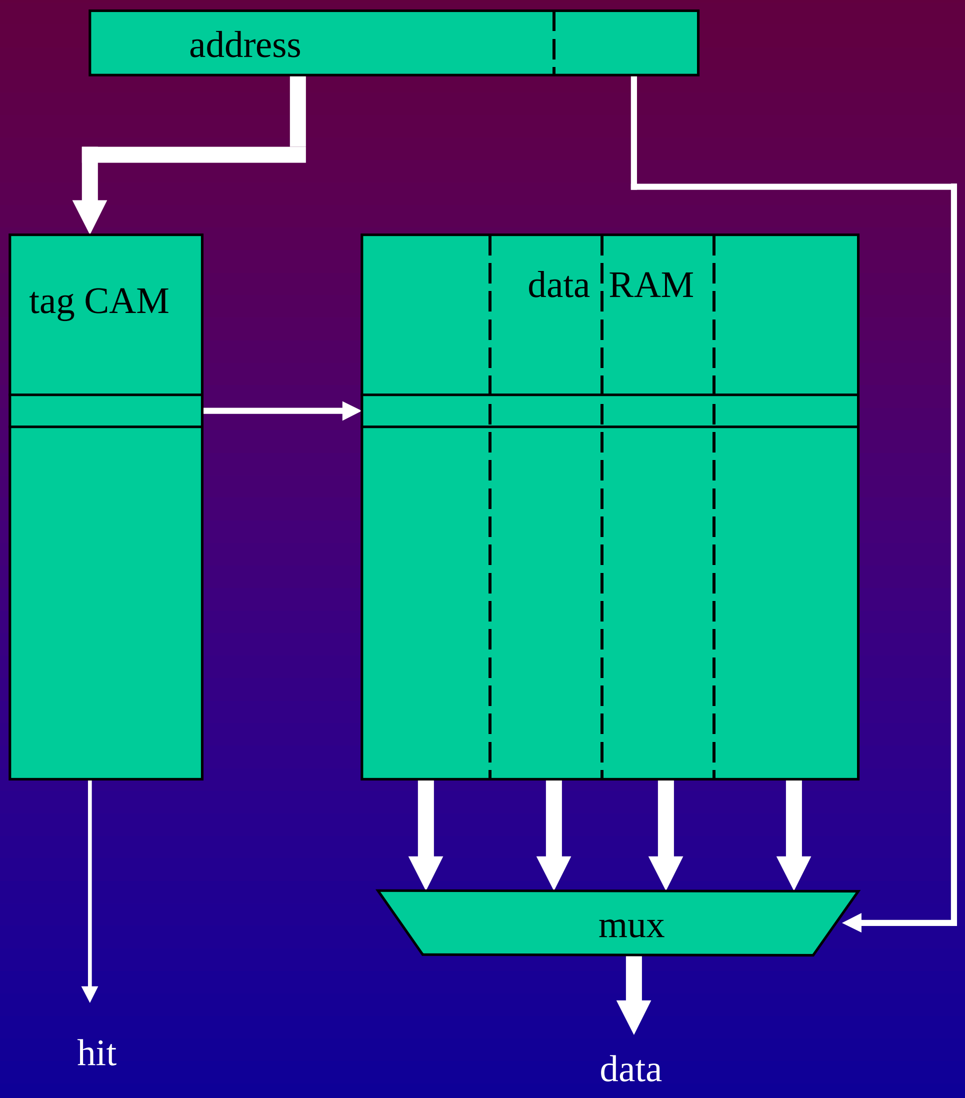
Direct-mapped cache organization



2-way set-associative cache organization



Fully associative cache organization



Cache write strategies

- Write-through
  - all data is written to memory; matching cache locations are updated
- Write-through with write buffer
  - all data is written to memory, but the write is performed through a buffer
- Write-back
  - the processor writes to the cache - main memory is only updated on flushes.

Cache organizational options

- There are many design decisions involved in choosing the best cache
  - some of the issues are summarized below:

Organizational feature	Options		
Cache-MMU relationship	Physical cache	Virtual cache	
Cache contents	Unified instruction and data cache	Separate instruction and data caches	
Associativity	Direct-mapped RAM-RAM	Set-associative RAM-RAM	Fully associative CAM-RAM
Replacement strategy	Cyclic	Random	LRU
Write strategy	Write-through	Write-through with write buffer	Write-back

Cache power-efficiency

- What is influence of organization on power-efficiency?
  - a high hit rate minimizes off-chip activity
    - hit rate increases with associativity (up to 4)
  - set-associative caches burn more power
    - due to the increased number of active sense amplifiers
  - CAM (in fully associative caches) is also power-hungry

Cache power-efficiency

- How can cache power-efficiency be improved?
  - use serial tag and data accesses in a set-associative cache
    - enable only the relevant data RAM
  - segment the CAM in a fully associative cache
    - discussed further in ARM600
  - exploit sequential address sequences

Sequential Access

- Accessing memory locations in same line: bypass tag look-up
  - increases access speed
  - saves power
- ARM CPUs generate a signal, when next memory access is sequential
- using current address and sequential signal: deduce that the access will fall in same line

Cache Speed

- High associativity caches: best hit rate
- Sequential CAM-RAM access: limits cycle time
- Lower associativity: parallel tag and data access
- Beyond 4-way associativity: gains in hit rate small
- However: Fast CAM-RAM cache is much simpler than

  4-way associative RAM-RAM Cache

Power Optimization

- Minimize overall system power!
  - good hit rate necessary
- highly associative CAM-RAM or set-associative RAM-RAM:
  - strongly influenced by low-level circuit issues
- 75% of all access are sequential
- sequential access may reduce performance by 25% and reduce cache power requirements by a factor of 2 or 3!

On-chip RAM

- System benefits of on-chip memory:
  - increased performance - no wait states
  - reduced power consumption
  - improved EMC
- On-chip RAM is used in preference to a cache in some embedded systems:
  - it it simpler, cheaper and uses less power
  - its behavior is more deterministic
  - however it requires explicit management

ARMulator

- Emulates ARM processor cores
- Can be extended to model nearly any ARM based system
- Three levels of accuracy:
  - instruction accurate
  - cycle-accurate
  - timing-accurate
- Initial evaluation of design alternatives:
  - instruction accurate model

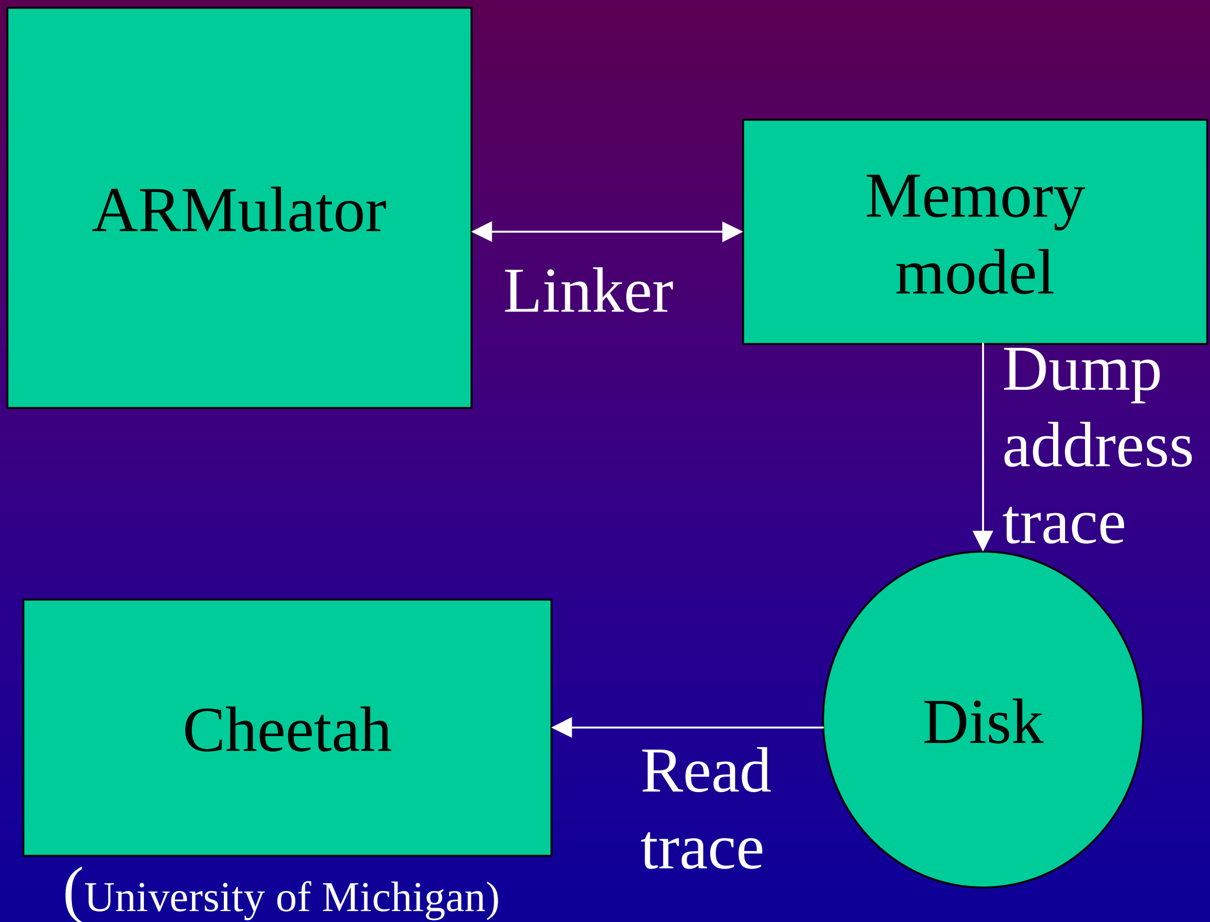
Instruction Accurate Model

- Model of ARM processor:

  not customizable, handles communication with debugger
- Memory interface:

  fully customizable memory model
- Coprocessor interface
- Operating system interface

Cache Evaluation



(University of Michigan)

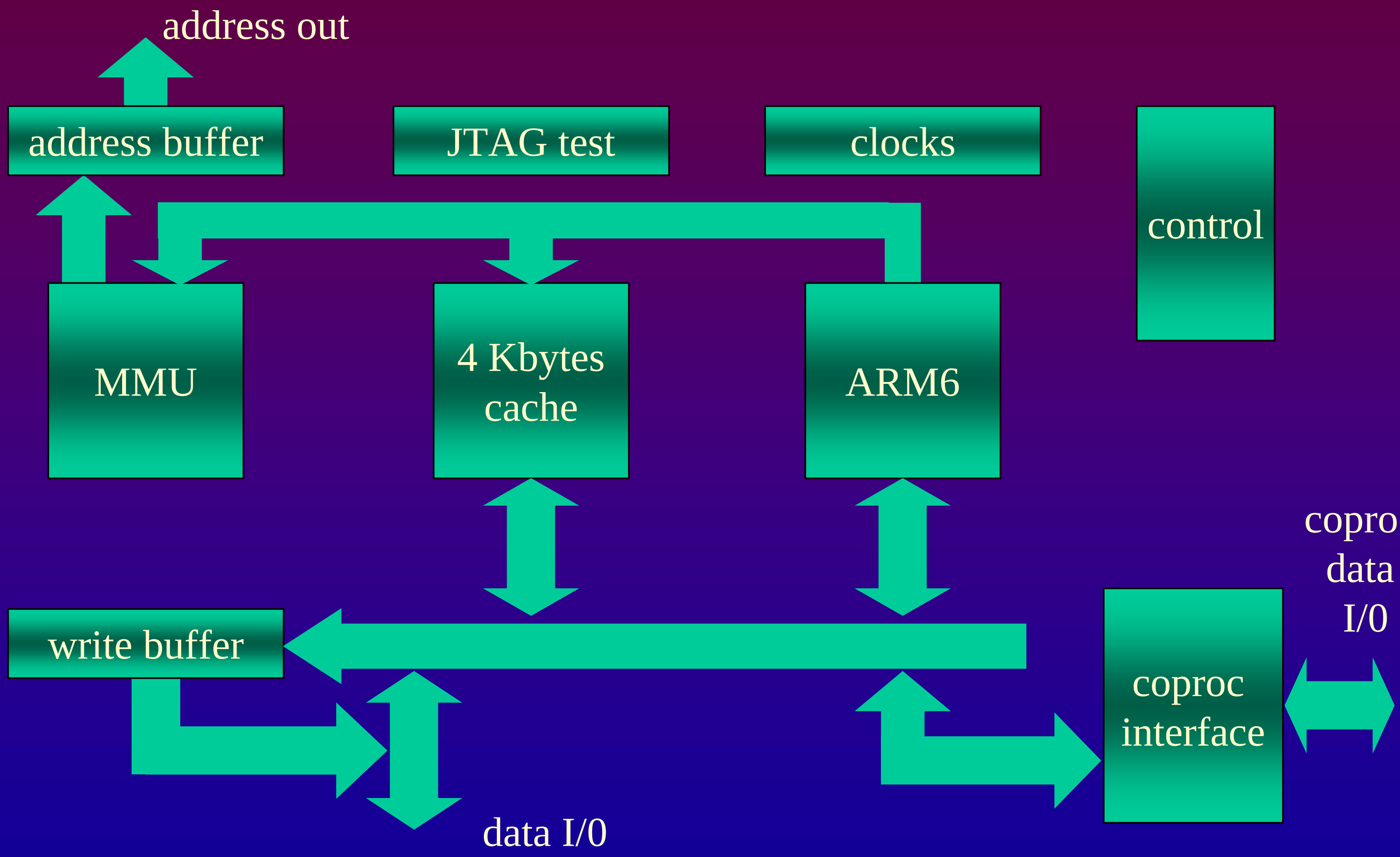
ARM System Design

- History of ARM
- ARM Instruction Set
- Thumb Instruction Set
- ARM Cores
- ARM Cache Modeling
- ARM CPUs
- ARM Coprocessors

ARM600

- General features:
  - ARM6 processor core
  - 4 Kbytes 64-way associative cache
    - mixed instruction and data
    - write-through with buffered write
  - MMU
  - support for on- and off-chip coprocessors
  - JTAG test access port

The ARM600 organization



ARM600 cache design

- The ARM600 cache was based on the ARM3 design, where:
  - extensive simulations were performed to evaluate organizational options
  - models were incrementally refined to approach ´real´ memory timings
  - start from ´perfect´ cache
    - this gives an upper bound on performance

`Perfect`cache performance

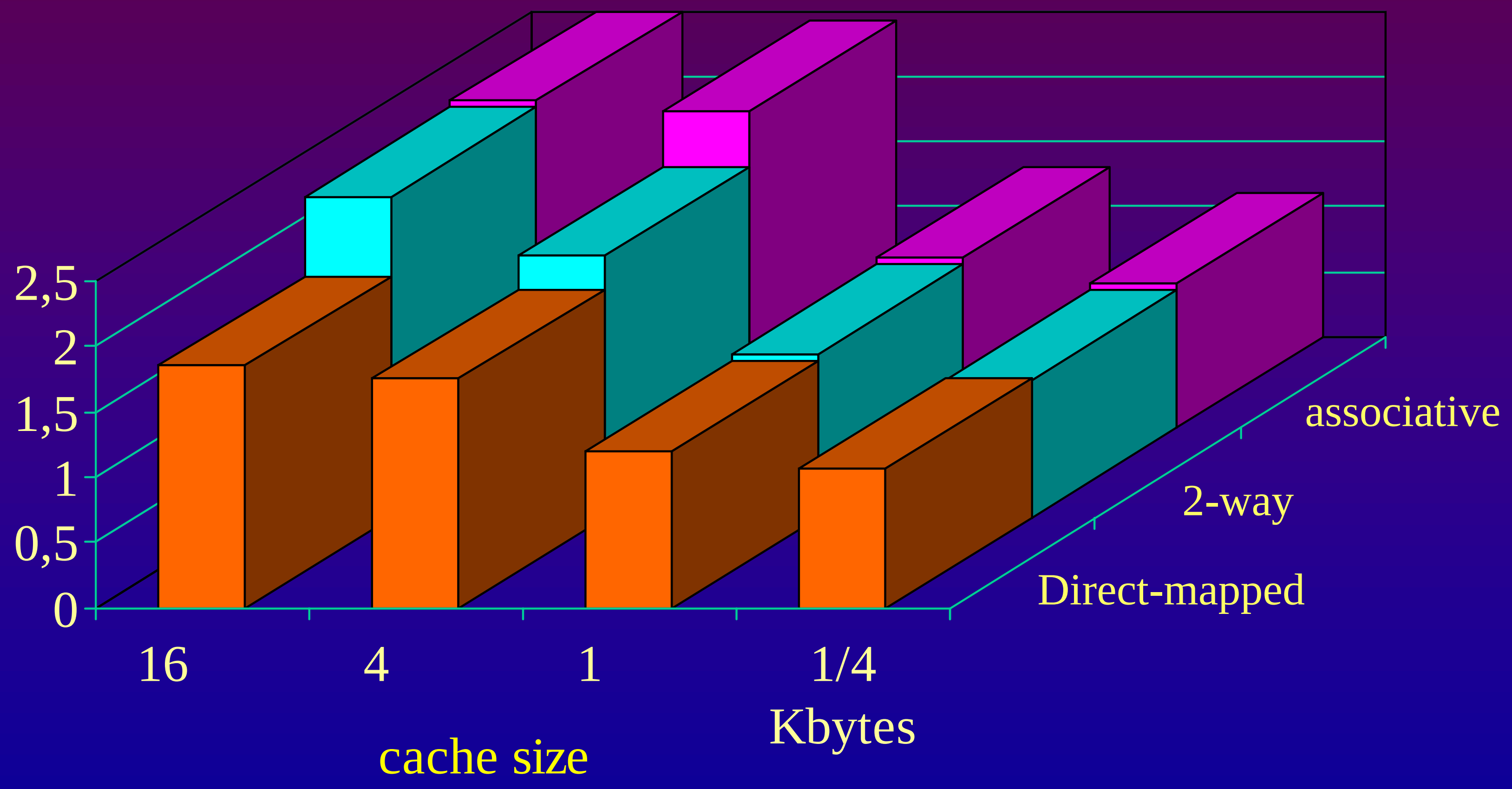
Cache form	Performance
No cache	1
Instruction-only cache	1.95
Instruction and data cache	2.5
Data-only cache	1.13

- assuming realistic clock rates
  - 20 MHz cache operation
  - 8 MHz external memory
- mixed cache gives best performance
  - separate I/D cache not on option with ARM6

Cache organization

- Write-through chosen for simplicity
  - allocate on write miss gave negligible benefits for significant complexity
- Look at size and associativity
  - maximum realistic size was 4 Kbytes
    - 1990 technology!
  - around this size associativity has strong effect on hit rate

Unified cache performance as a function of size and organization



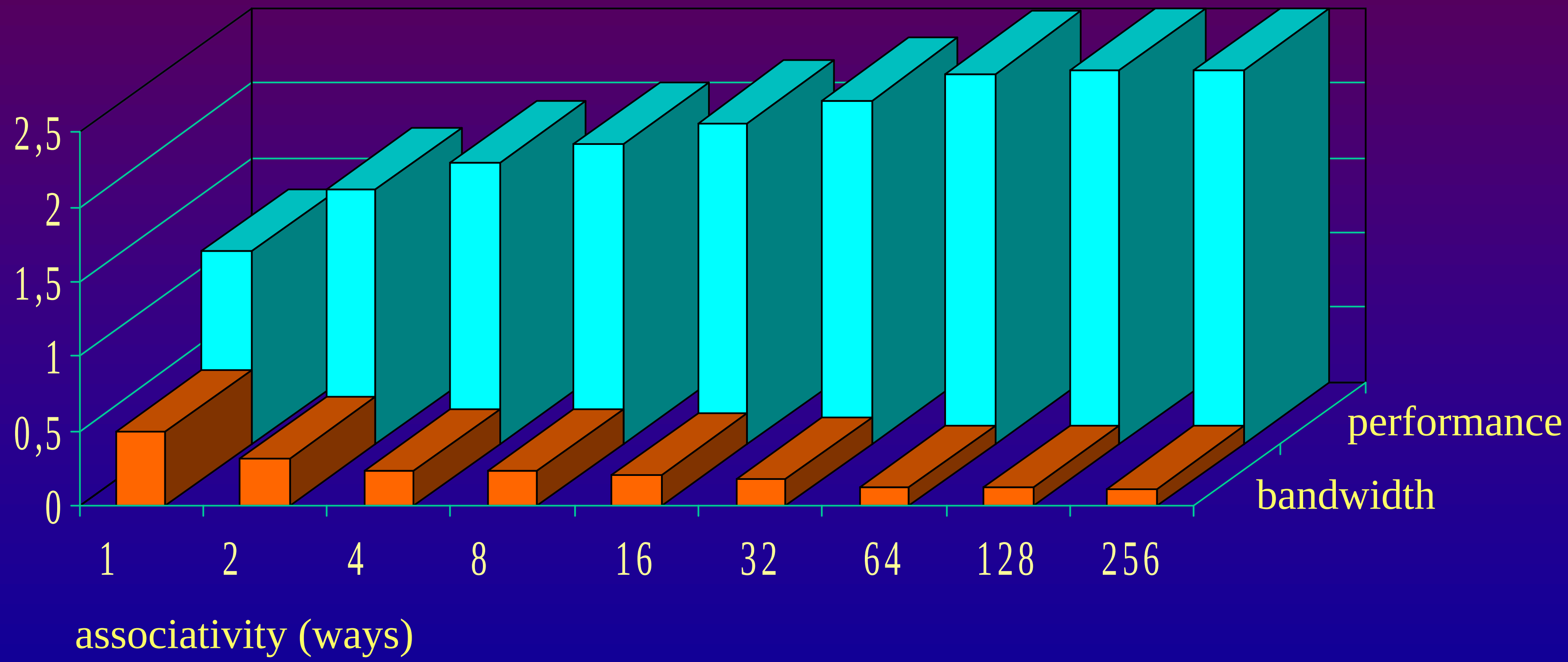
Cache Organization

- So, high associativity is desirable
- Replacement algorithm?
  - LRU (Least Recent Used) is deal
    - but expensive to implement
  - cyclic replacement is simple
    - but has obvious pathological cases
  - random performance as well as LRU
    - and is simple to implement

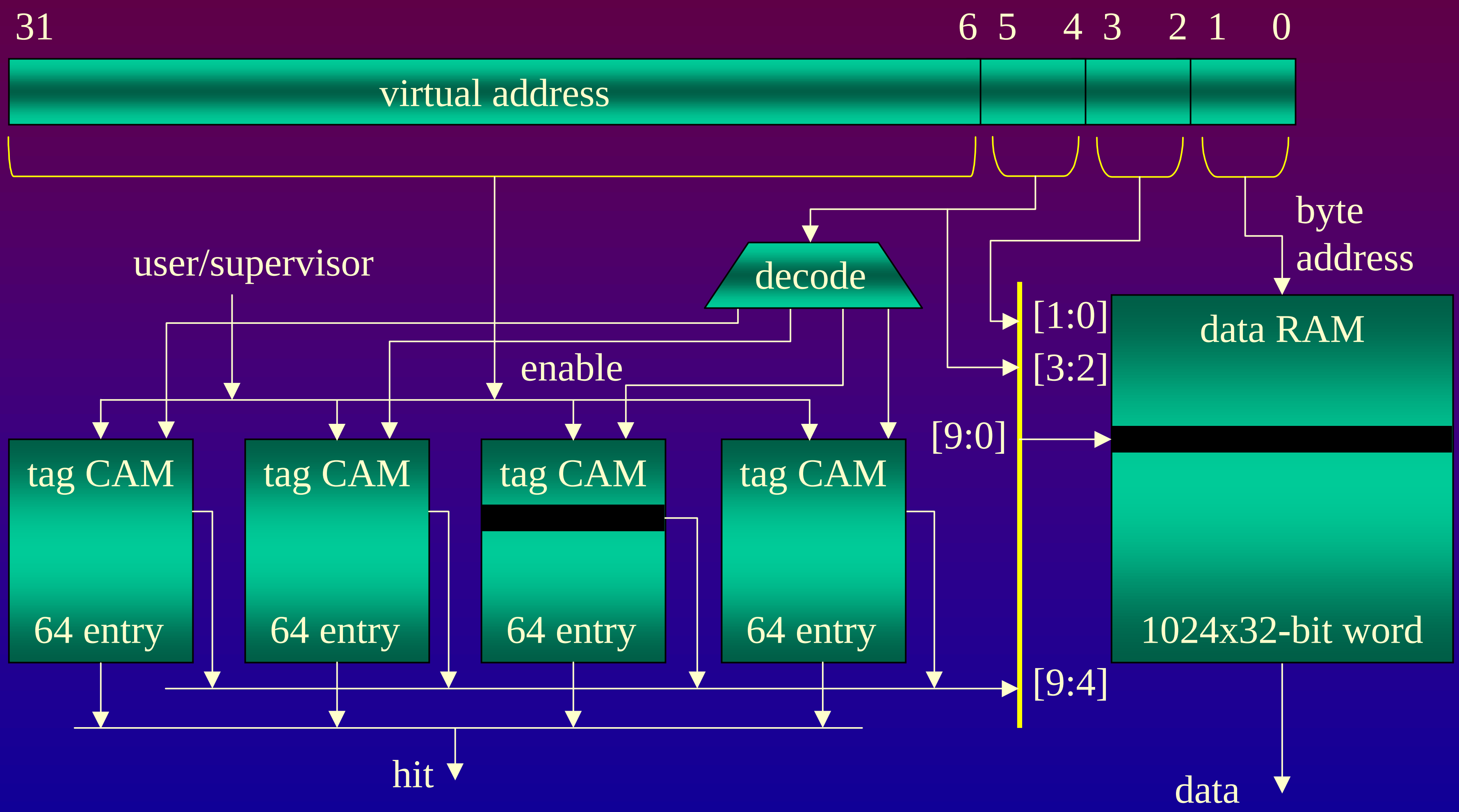
Cache organization

- Full associativity?
  - Very large CAMs consume too much power
  - line size
    - a quad-word line size:
      - reduces the size of the tag store (by 4x)
      - has negligible impact on performance
    - slightly reduced associativity
      - has little effect on performance
      - allows the CAM to be segmented
        - » only active segment uses power

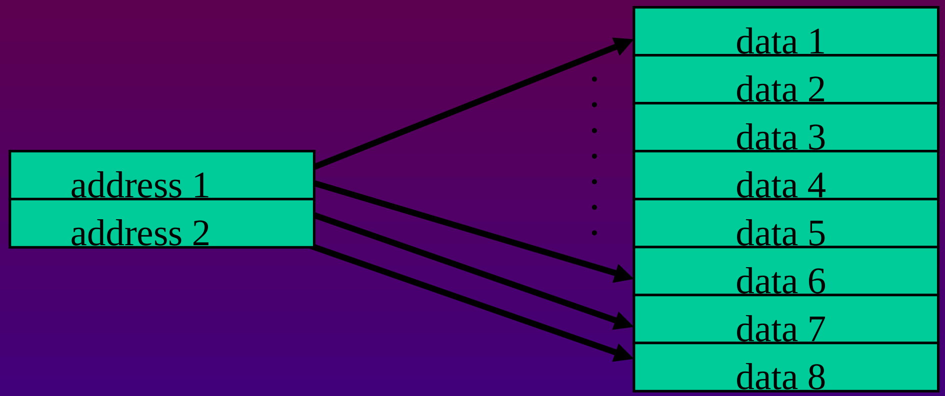
The effect of associativity on performance and bandwidth



ARM600 cache organization



ARM600 write buffer



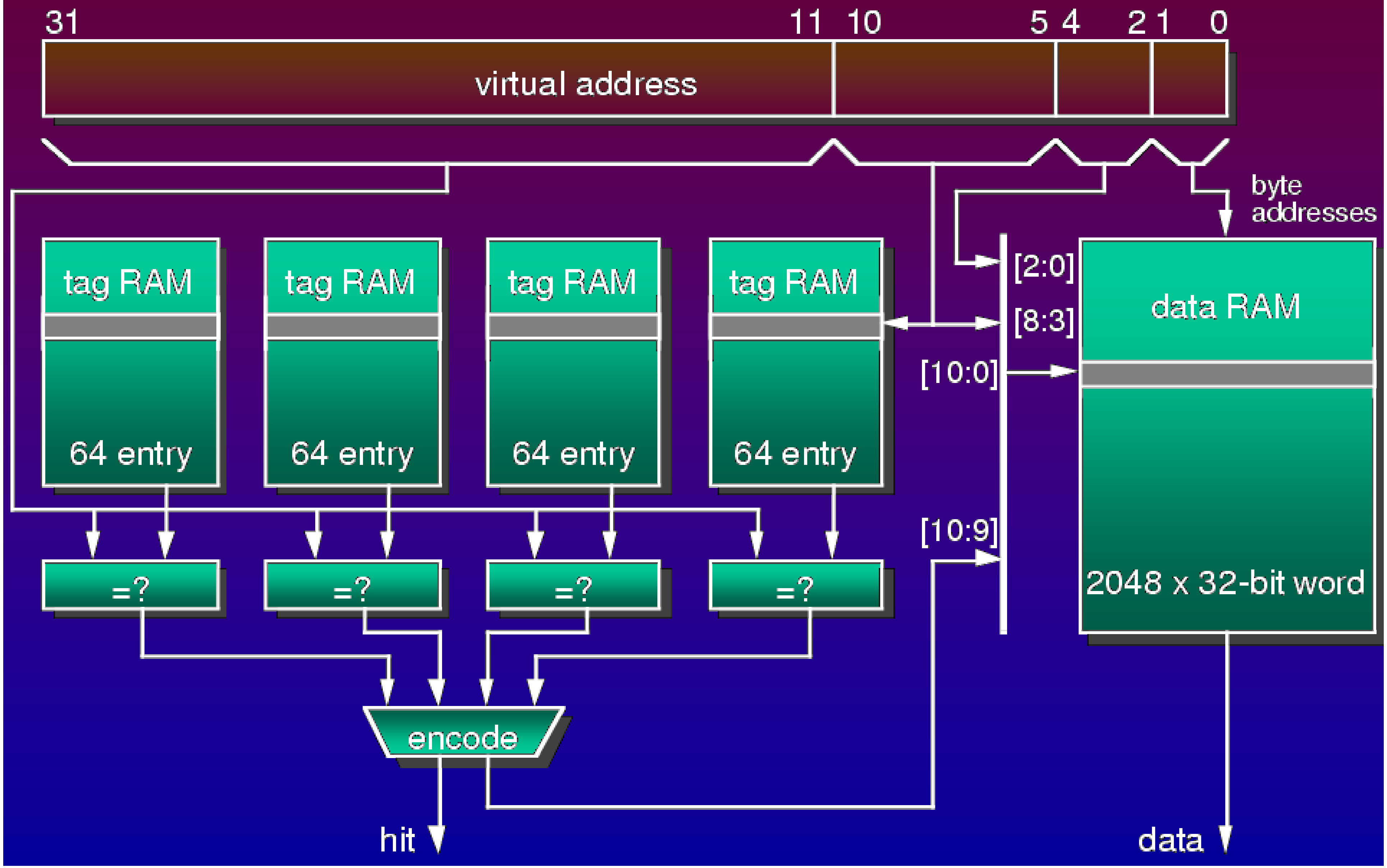
- 2 address, 8 data locations
  - flexible association from address to data
  - only STM can generate multiple data items for one address

The ARM610

- The ARM610 is an ARM600
  - without external coprocessor support
  - in a 144-pin TQFP `thin quad flat pack`
  - as used in the original Apple Newton
  - original on 1 µm CMOS, now on 0.6 µm

Process	0.6 µm	Transistors	358.931	MIPS	30
Metal layers	2	Die area	26 mm²	Power	500 mW
Vdd	5 V	Clock	0 to 33 MHz	MIPS/W	60

The ARM700 Cache Organization



The ARM700 and 710

- Differences from the 600 and 610
  - ARM7 core
  - operates at 3v3 as well as 5v
- cache is 8 Kbytes, 4-way, 8-word line
  - later ARM710s have reverted to a 4-word line
- TLB has 64 entries instead of 32
- write buffer has 4 address, 8 data slots
- clock rate increased
  - from 33 MHz to 40 MHz at 5v

ARM System Design

- History of ARM
- ARM Instruction Set
- Thumb Instruction Set
- ARM Cores
- ARM Cache Modeling
- ARM CPUs
- ARM Coprocessors

Design and Application of Cores

- Microprocessor Cores (ARM)
- On-chip buses

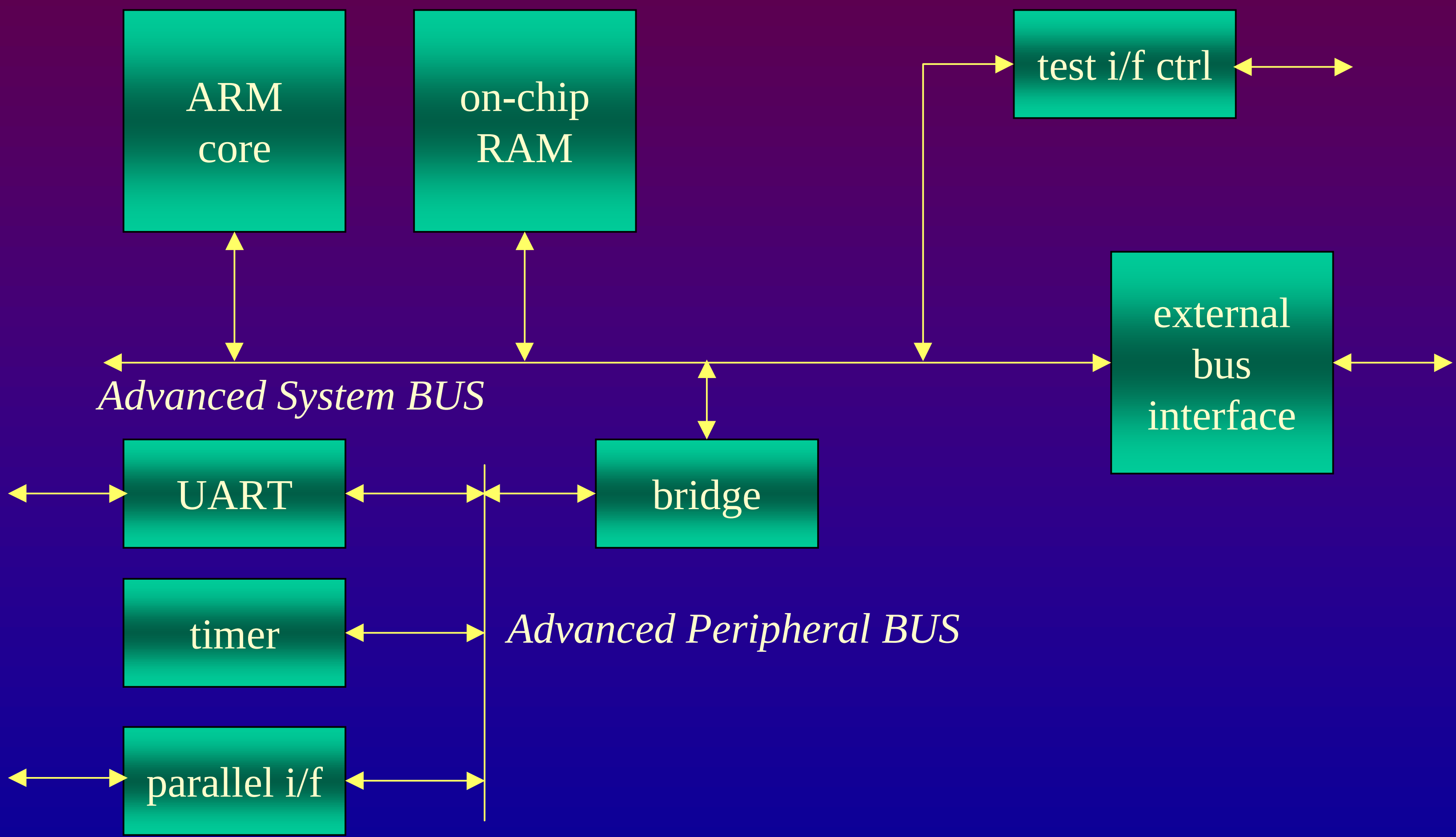
ARM Coprocessors

- Details see ARM6 organization
- No tremendous changes

AMBA

- Advanced Microprocessor Bus Architecture
  - a systematic solution to assembling macrocell-based systems
- AMBA structure:
- Advanced System Bus (ASB)
  - high-performance, multi-master
- Advanced Peripheral Bus (APB)
- interface for low performance peripherals

A typical AMBA-based system



AMBA test interface

- VSLI production test is an economically important issue
  - macrocell based designs present problems
  - how can each macrocell be systematically tested?
- AMBA offers a standardized solution
  - based on 32-bit parallel access, via the bus, to test registers