Information Technology Fundamentals

More Memory

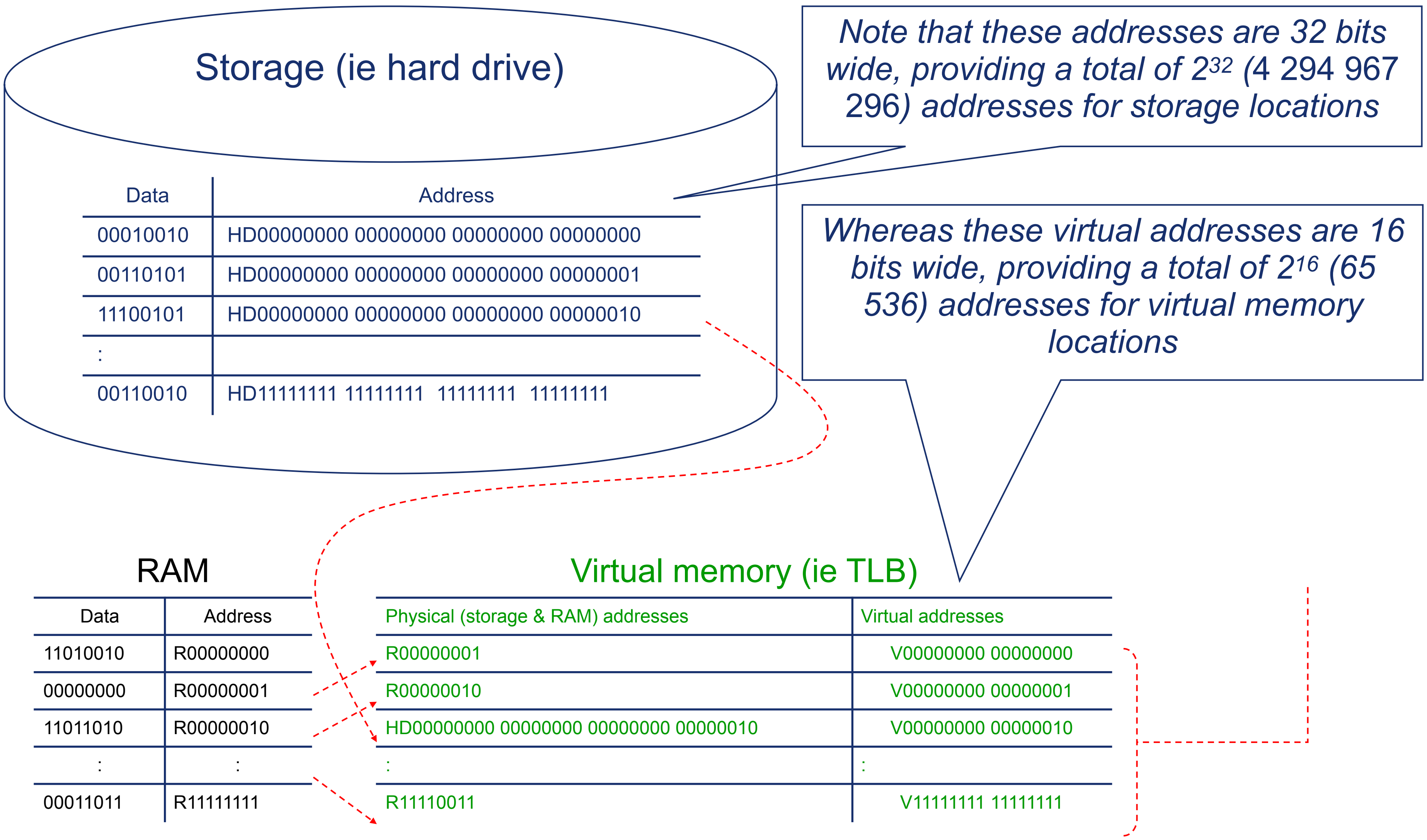
School of Information Technology and Mathematical Sciences

Virtual Memory

Virtual memory combines storage and memory, simulating memory of larger size than is physically available

The process is invisible to most software, and is indistinguishable from “real” memory.

To work, the microprocessor maintains a set of *virtual addresses* cross referenced to a set of *physical addresses* in the MMU.



Storage (ie hard drive)

Data	Address
00010010	HD00000000 00000000 00000000 00000000
00110101	HD00000000 00000000 00000000 00000001
11100101	HD00000000 00000000 00000000 00000010
:	
00110010	HD11111111 11111111 11111111 11111111

Note that these addresses are 32 bits wide, providing a total of 2^{32} (4 294 967 296) *addresses for storage locations*

Whereas these virtual addresses are 16 bits wide, providing a total of 2^{16} (65 536) *addresses for virtual memory locations*

RAM

Data	Address
11010010	R00000000
00000000	R00000001
11011010	R00000010
:	:
00011011	R11111111

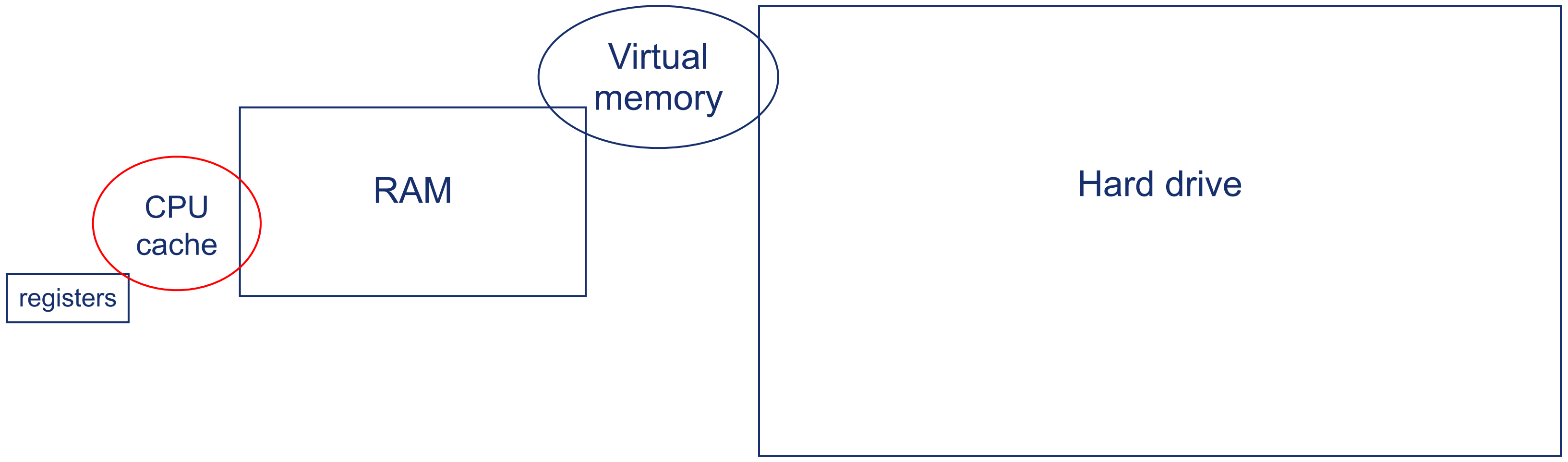
Virtual memory (ie TLB)

Physical (storage & RAM) addresses	Virtual addresses
R00000001	V00000000 00000000
R00000010	V00000000 00000001
HD00000000 00000000 00000000 00000010	V00000000 00000010
:	:
R11110011	V11111111 11111111

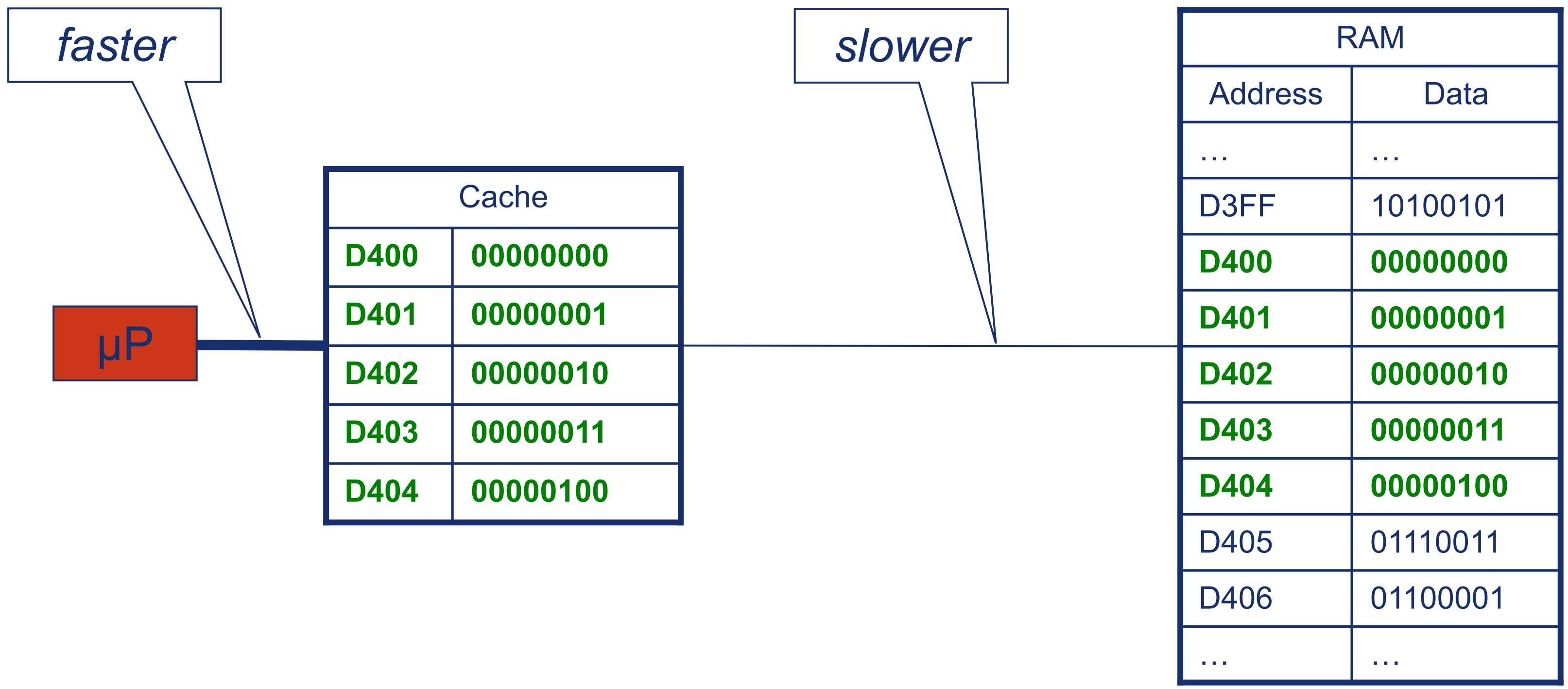
Virtual memory structure

The *CPU cache* decreases latency by exploiting locality of reference.

If we can make an educated guess about which memory locations are likely to be required soon, then we can read them from memory early, and store them in faster memory.



Cache in the memory hierarchy.



Cache	
D400	00000000
D401	00000001
D402	00000010
D403	00000011
D404	00000100

RAM	
Address	Data
…	…
D3FF	10100101
D400	00000000
D401	00000001
D402	00000010
D403	00000011
D404	00000100
D405	01110011
D406	01100001
…	…

Cache is faster to read than main memory.

Caches are organised differently to main memory.

Cache space is organised into *blocks* (also called *cache lines*) instead of page frames (generally 8 to 512 bytes in size)

Each block has a *tag* identifying which memory locations are currently held in a block

When the microprocessor needs data, it checks the cache first.
A hit is when the data is there, a miss is when it is not.

Misses are bad for performance.
The best performance has a high hit ratio.

```
number of hits / number of requests = hit ratio
```

Replacement Policies:
If there is a miss, we need to work out which block to replace.

Generally we use Least Recently Used.

If the data hasn’t changed since it was loaded into cache,
it can be overwritten.

If it has changed, we need an eviction policy.

Write through (simple and slow)

Write back (faster but requires more complex circuitry)

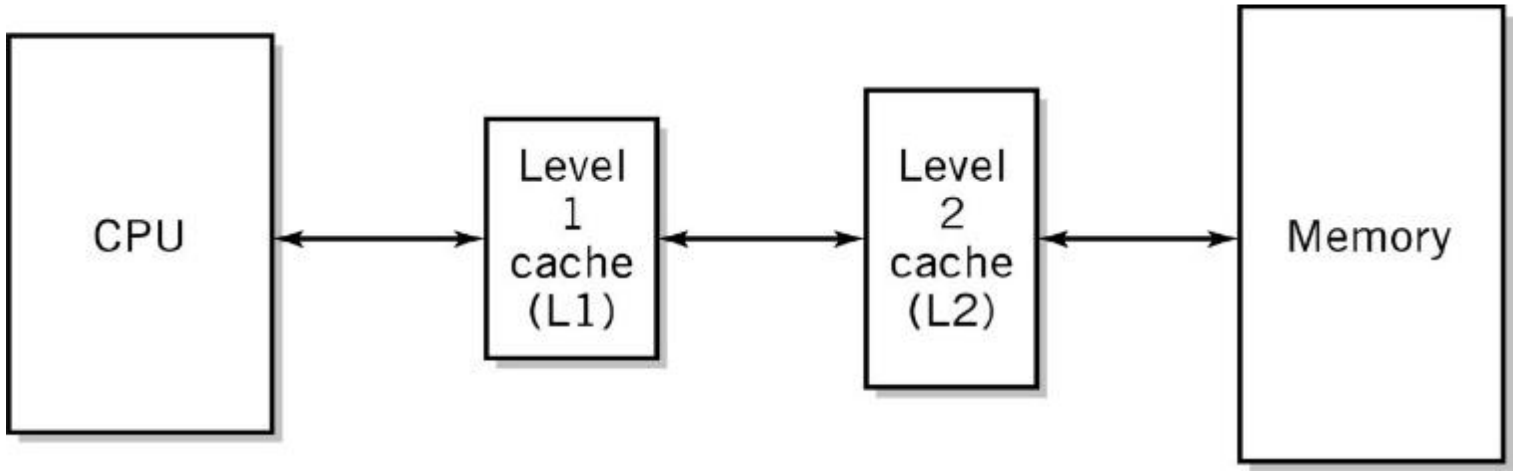
Does cache work?

Cache hit ratios of 90% can be observed with a small amount of cache.

Performance improvements of over 50% are attainable.

Larger caches have better hit ratios, but are slower.

We can improve things further by *interleaving* or by increasing the number of *cache levels*.



Two levels of cache.

There are two approaches:

Inclusive caches: improve cache coherence, frees up one level for writing.

Exclusive caches: allows for more data.

Can we add more levels of cache?

Note that this depends on locality of reference.

What prevents locality of reference from working?

* Widely dispersed data, such as Object Oriented code or M-dimensional data structures.

* Insufficient RAM.

Registers

Registers are single storage locations within the microprocessor.

Each register has one specific job to do.

Today, that means the MAR and MDR.

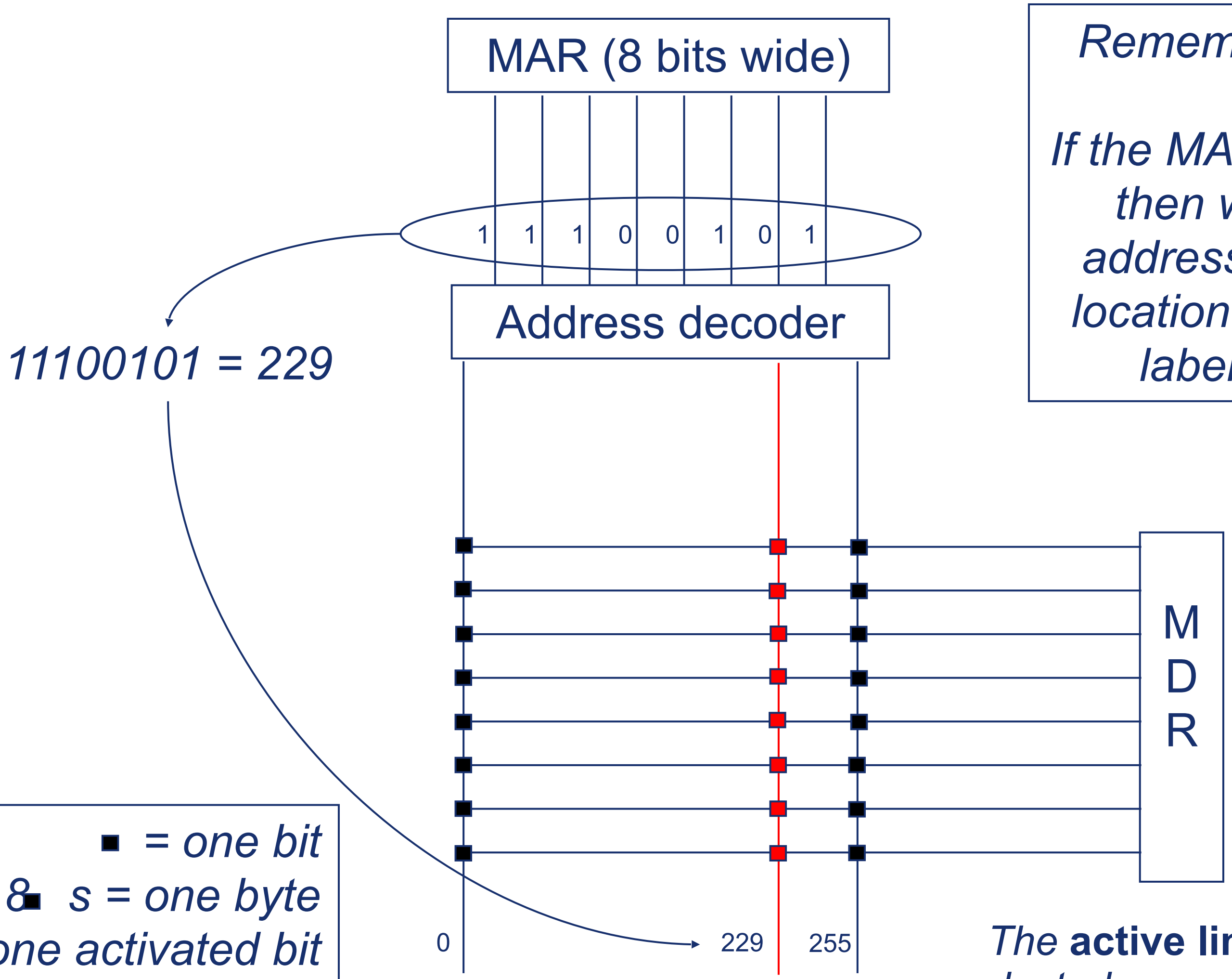
The MAR and MDR are part of the simplest, minimal memory addressing configuration that works.

The MAR is the Memory Address Register.

It holds the address.

The MDR is the Memory Data Register.

It holds the data.



Remember: $2^8 = 256$.

If the MAR is 8 bits wide then we have 256 addressable memory locations and they are labelled 0 - 255

■ *= one bit*

8■ s = one byte

🟥 *= one activated bit*

The ***active line*** *(ie address) is selected according to the value in the address decoder*

MAR/MDR achitecture

The read/write line determines direction.

Thus memory is controlled by:

* The read/write line

* The active line

* The MAR decoder line

Memory access takes time, hence memory latency issues.

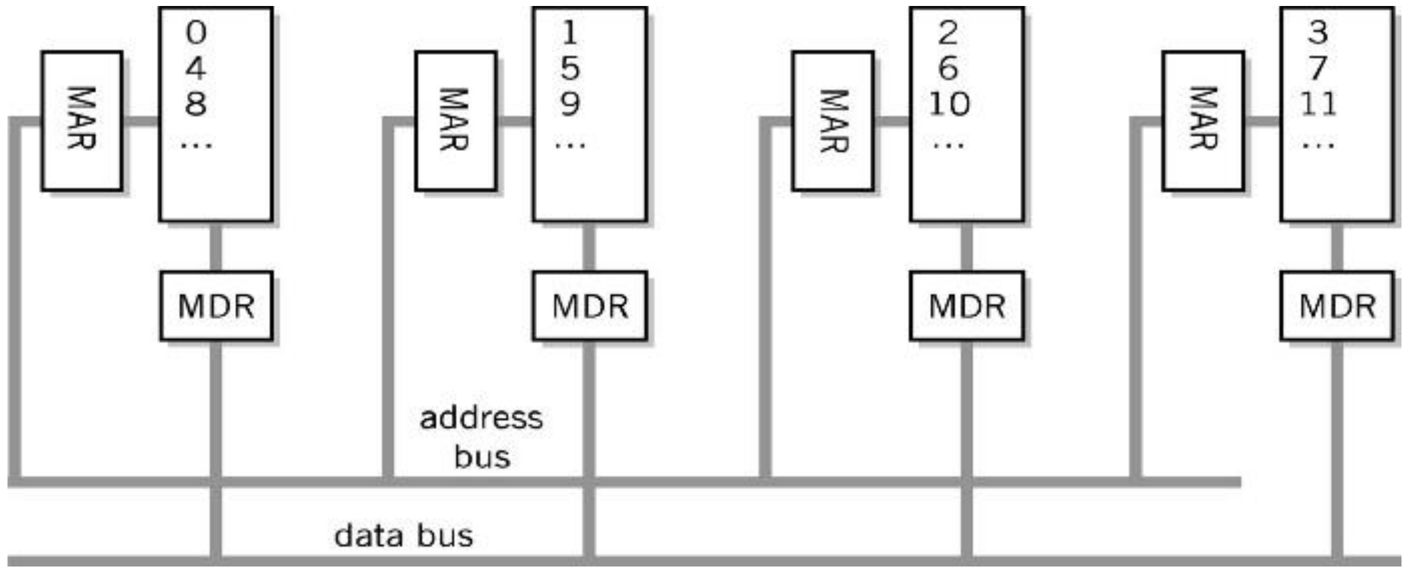
More bandwidth means less latency.

One solution is:

Interleaving

Interleaving divides memory into parts, each with its own MAR and MDR.

While one part is being written to, another part can be read.



N-Way interleaving.