The x86 PC

assembly language, design, and interfacing

fifth edition

Prentice Hall

Dec Hex Bin
22 16 00010110

ORG ; TWENTY-TWO

HIGH-SPEED MEMORY DESIGN AND CACHE

The x86 PC

assembly language, design, and interfacing

fifth edition

MUHAMMAD ALI MAZIDI
JANICE GILLISPIE MAZIDI
DANNY CAUSEY

OBJECTIVES

this chapter enables the student to:

- Explain how the introduction of wait states is implemented in the IBM PC to coordinate memory cycle times of x86 CPUs and high-speed memory.
- Define terms used in memory design, such as memory cycle time and memory access time.
- Describe the various types of DRAM: standard mode, page mode, and static column mode.
- Describe how the interleaving method is implemented to solve the problem of back-to-back DRAM access and the required precharge time.

OBJECTIVES (*cont*)

this chapter enables the student to:

- Discuss the advantages of using DRAM for main memory and SRAM for cache.
- Diagram the three types of cache organization:
  - Fully associative; Direct mapped; Set associative.
- Explain the write-back and write-through methods of updating main memory as cache data is altered.
- Describe cache replacement policies LRU & FIFO.
- Contrast and compare EDO and FPM DRAM.
- Describe the operation and purpose of SDRAM
- Explain components/function of *Rambus* technology

22.1: MEMORY CYCLE TIME OF THE x86

- In the 8088/86 microprocessor, the memory cycle time consists of 4 clocks.
  - For the 286, 386, 486, and Pentium®, memory cycle time consists of only two **T** clocks.

22.1: MEMORY CYCLE TIME OF THE x86

- As CPU frequency is increased, maximum amount of time allowed to access memory is decreased.

Table 22-1: Memory Cycle Times for x86

Mem. Cycle	CPU	Bus Frequency (MHz)	Clock Cycle (ns)	Memory Cycle (ns)
4 clocks	8088/86	5	200	800
	8088/86	8	125	500
	8086	10	100	400
2 clocks	80286	6	166.6	333.3
	80286	8	125	250
	80286	10	100	200
	80286	16	62.5	125
	80286	20	50	100
2 clocks	80386DX	16	62.5	125
	80386DX	[illegible]	50	100

See the entire table on page 561 of your textbook.

22.1: MEMORY CYCLE TIME OF THE x86 wait introducing wait states to memory cycle

- When the memory timing requirement of the CPU cannot be met, an option is to introduce wait states.
  - All x86 microprocessors have the READY pin.
- When the processor initiates the memory cycle, the time at which it must have data at the pins of the data bus is fixed, extended by activating pin READY.
  - Every time the READY pin is activated, the CPU adds one extra clock to the memory cycle

22.1: MEMORY CYCLE TIME OF THE x86 wait introducing wait states to memory cycle

- If allocated memory cycle time is not enough, more wait states are needed, making the memory cycle time longer.

Example 22-1

Find the effective memory performance of a 25-MHz 386 CPU with one wait state.

Solution:

Since the 0 WS memory cycle is 80 ns (1/25 MHz = 40 and 2 × 40 = 80 ns), for 1 WS we have a memory cycle time of 120 ns. That means that the memory performance is the same as that of a 16.6-MHz 80386 (120 ns/2 = 60 ns, then 1/60 ns = 16.66 MHz) as far as memory accessing is concerned. This is 67% performance of the 80386 with zero wait states.

22.2: PAGE AND STATIC COLUMN DRAMS

- To understand the interface of memory to high-performance computers, different types of available RAM must first be understood.
  - **SRAM**s are fast, expensive & consume a lot of power due to use of flip-flops in memory cell design.
  - **DRAM** is cheaper but slow (compared to CPU speed) and needs to be refreshed periodically.
- Refresh overhead with the long DRAM access time is a major problem in high-performance computers.
  - A common solution is a combination of a small amount of SRAM, (cache), with a large amount of DRAM.
    - Achieving a goal of near zero wait states.

22.2: PAGE AND STATIC COLUMN DRAMS memory cycle time

- Memory data sheets call $\mathbf{t_{AA}}$ (address access time) the time interval between the moment addresses are applied to the memory chip address pins & the time data is available at the memory's data pins.
- Time interval $\mathbf{t_{CA}}$ (access time from CS) is measured from when the chip select pin of memory is activated to the time the data is available.
- In some cases, notably EEPROM, $\mathbf{t_{OE}}$ is the time interval between moment OE (READ) is activated to the time data is available.
  - Memory access time $\mathbf{t_{AA}}$ is the one most often advertised.

22.2: PAGE AND STATIC COLUMN DRAMS memory cycle time

- *Memory cycle time* is the time interval between two consecutive accesses to the memory chip.
  - While the SRAM memory cycle time is equal to memory access time, this is not so in DRAM memory.

22.2: PAGE AND STATIC COLUMN DRAMS
types of DRAM

- There are different types of DRAM, categorized by their mode of data access:
  - Standard mode.
  - Page mode.
  - Static column mode.
  - Nibble mode.
- Often two of the above modes exist on the same DRAM chip.

22.2: PAGE AND STATIC COLUMN DRAMS
DRAM standard mode

- **Standard mode** (random access) DRAM, has the longest memory cycle time, requires the row address to be provided first, then the column address for each cell.
  - Each group is latched by activation of RAS (row address select) and CAS (column address select) inputs.
  - The memory cycle time for memory chips is the minimum time interval between two back-to-back read/write operations.
  - In SRAM and ROM, the access time &memory cycle time are always equal, but that is not the case for DRAMs.

22.2: PAGE AND STATIC COLUMN DRAMS
DRAM standard mode

- In DRAM is the approximate relationship between memory access time and memory cycle time.

$t_{RC} = t_{RAC} + t_{RP}$ (This is for standard mode)
read cycle time = RAS access time + RAS precharge time

Table 22-2: DRAM Access Time vs. Cycle Time (4M × 1)

DRAM	RAS Access (t_{RAC}) (ns)	Read Cycle (t_{RC}) (ns)	RAS Precharge (t_{RP}) (ns)
MCM44100-60	60	110	45
MCM44100-70	70	130	50
MCM44100-80	80	150	60

Tables **22-2** and 22-3 show **DRAM** and SRAM memory cycle times, respectively.

22.2: PAGE AND STATIC COLUMN DRAMS
DRAM standard mode

- In DRAM is the approximate relationship between memory access time and memory cycle time.

$t_{RC} = t_{RAC} + t_{RP}$ (This is for standard mode)
read cycle time = RAS access time + RAS precharge time

Table 22-3: SRAM Access Time vs. Cycle Time

SRAM (IDT Product)	Address Access (t_{AA}) (ns)	Read Cycle (t_{RC}) (ns)
IDT71258S25	25	25
IDT71258S35	35	35
IDT71258S45	45	45
IDT71258S25	70	70

Tables 22-2 and **22-3** show DRAM and **SRAM** memory cycle times, respectively.

22.2: PAGE AND STATIC COLUMN DRAMS
DRAM standard mode

- Read cycle time not equal to access time is a major difference between SRAM and DRAM.
  - DRAM standard mode read cycle time is about twice the access time normally advertised. ($\mathbf{t_{ACC}}$)
    - This could make a difference in the total time spent by the CPU to access memory.

22.2: PAGE AND STATIC COLUMN DRAMS
DRAM standard mode

Example 22-2

Compare the minimum CPU time needed to read 150 random memory locations of a given bank in each of the following.
(a) DRAM with T_{ACC} = 100 ns and T_{RC} = 190 ns
(b) SRAM of T_{ACC} = 100 ns

Solution:
(a) DRAM requires 190 ns to access each location. Therefore, a total of 150 × 190 = 28,500 ns would be spent by the CPU to access all those 150 memory locations.
(b) In the case of SRAM, the CPU spends only 150 × 100 ns = 15,000. This would have been needed since T access = T read cycle ($t_{ACC} = t_{RC}$).

Example 22-3

Calculate the time to access 1024 random bits of a 1M × 1 chip if t_{RC} = 85 ns and t_{RAC} = 165 ns.

Solution:
For standard mode (also called random) we have the following for reading 1024 bits:

time to read 1024 random bits = 1024 × t_{RC} = 1024 × 165 ns = 168,960 ns

22.2: PAGE AND STATIC COLUMN DRAMS
DRAM standard mode

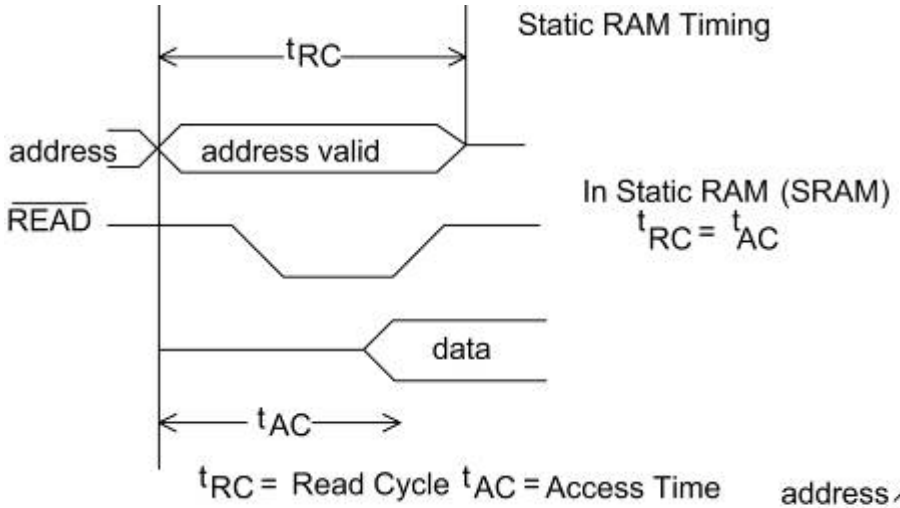
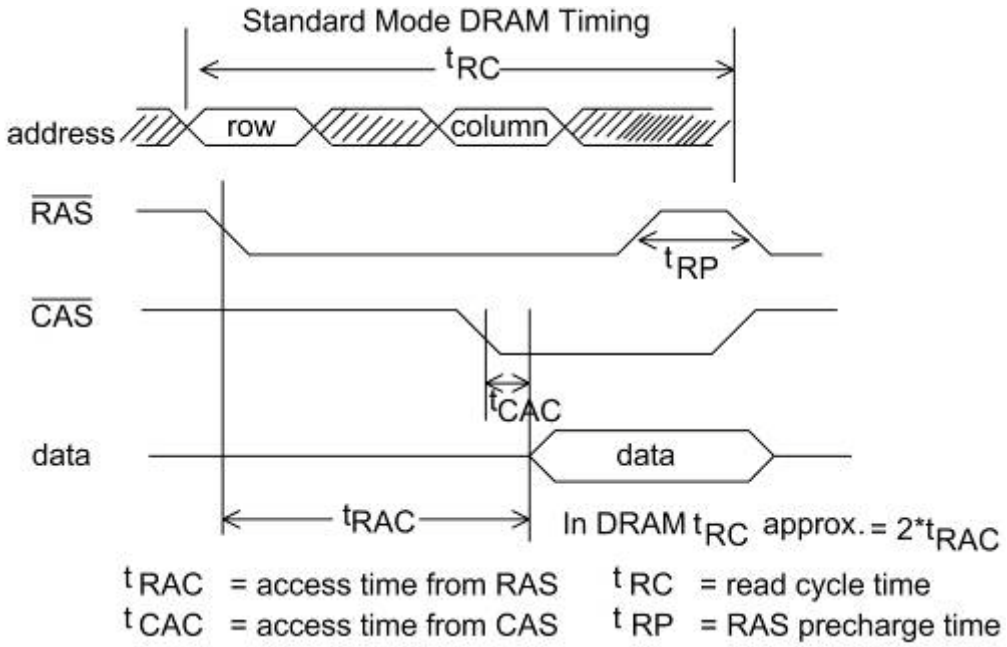


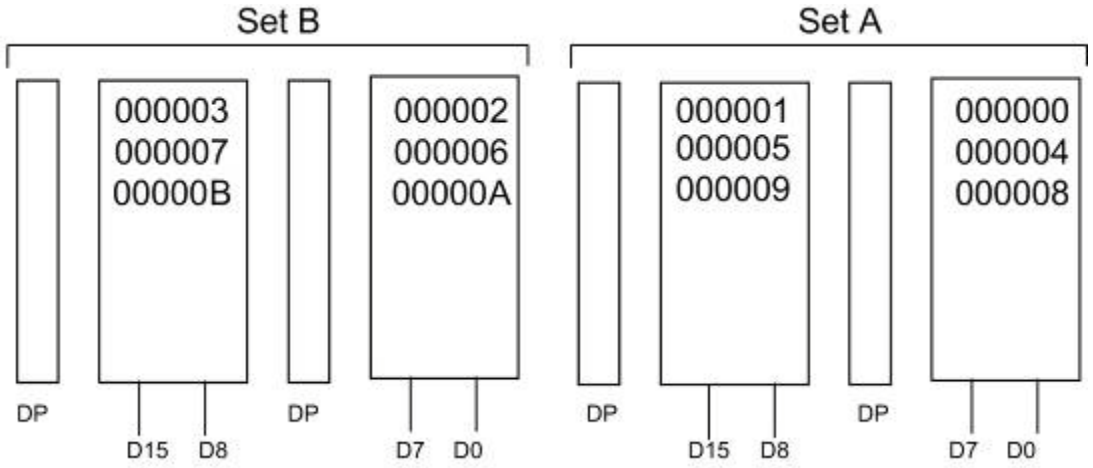
Fig. 22-1 DRAM vs. SRAM Timing

For successive accesses of random locations inside DRAM, the CPU must spend a minimum of $\mathbf{t_{RC}}$ time on each access.



22.2: PAGE AND STATIC COLUMN DRAMS
DRAM interfacing using interleaving

- A methods used to overcome precharge time in DRAMs is interleaving of DRAM interfacing.
  - Two sets of banks are placed next to each other and the CPU accesses each set of banks alternately.
    - Precharge time of one set of banks is hidden behind the access time of the other one.



While the CPU is accessing one set of banks, the other set is being precharged.

Fig. 22-2 Interleaved DRAM Organization

22.2: PAGE AND STATIC COLUMN DRAMS
DRAM interfacing using interleaving

- A major drawback in expanding interleaved memory is that a minimum of two sets of banks must be added every time additional memory is required.
  - Many inexpensive embedded systems use interleaved memory to avoid expensive cache memory.

Example 22-4

Show the time needed to access all 1024 memory locations of Example 22-3 if the interleaved method of memory interfacing is used.

Solution:

In the interleaved method, since the precharge time of one bank is hidden behind the access time of the other bank, each memory location is accessed in t_{RAC} as far as the CPU is concerned; therefore, $1024 \times 85 = 87{,}040$ ns is the total amount of time spent by the CPU to access 1024 locations.

22.2: PAGE AND STATIC COLUMN DRAMS
DRAM interfacing using interleaving

Example 22-5

Assume that we are using 1M × 1 DRAM organization in Figure 22-2. If each set is 4 megabytes, find the following.
(a) the chip count (b) the minimum memory addition and the chip count

Solution:

(a) Assuming 1M × 9 for each bank where each bank takes care of 8 bits of data, there are 9 chips for every byte. That means a total of 36 DRAM chips for each set, or a total of 72 1M × 1 chips for the first 8 megabytes of interleaved memory.
(b) From then on, any memory addition must be in multiples of 4 megabytes since each set needs 2M; therefore, we need another 36 1M × 1 DRAM chips to raise total memory of the system to12M

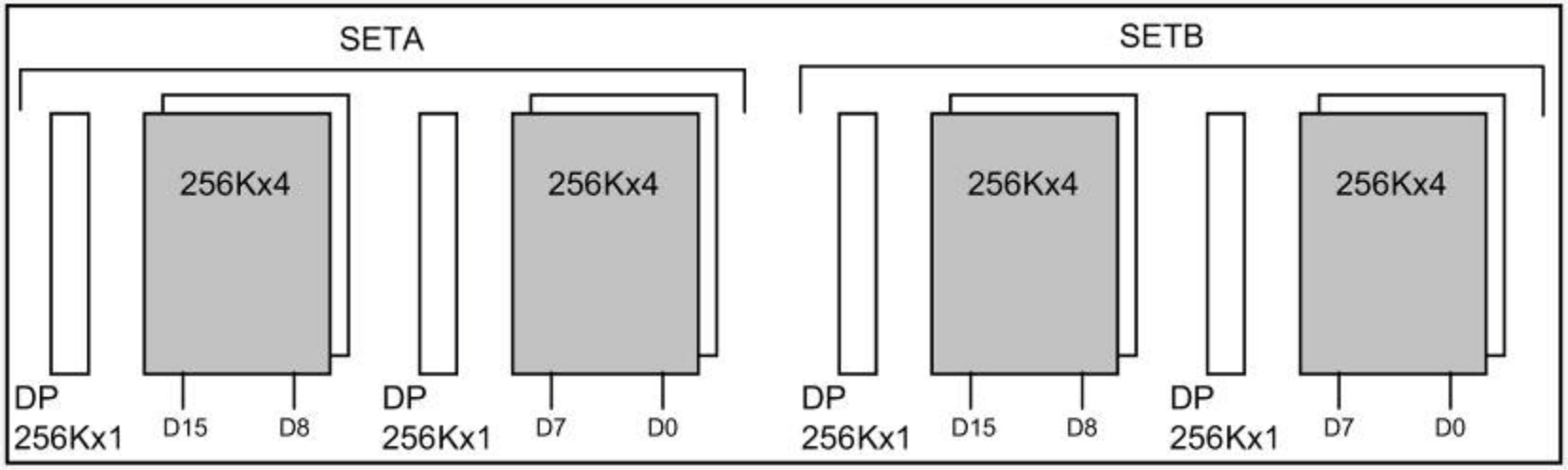
22.2: PAGE AND STATIC COLUMN DRAMS
DRAM interfacing using interleaving

Example 22-6

A 386 embedded system has 1M of DRAM installed using the interleaved design method. Show the memory organization and DRAM chip count assuming that only 256K × 1 and 256K × 4 DRAM chips are used.

Solution:

Since the 386SX has a 16-bit data bus, it uses 512K bytes for each set of A and B, or four banks of 256K × 9, where each set consists of two banks of 256K × 9. Therefore, the total chip count is 12 since each bank uses three chips (two 256K × 4 and one 265K × 1 for parity bit). This is shown as follows.



22.2: PAGE AND STATIC COLUMN DRAMS
DRAM interfacing using interleaving

Example 22-7

Show the minimum memory addition and the chip count for Example 22-6. Assume that the available DRAM chips are 256K × 1 and 256K × 4.

Solution:

The minimum memory addition is 1M. Since we have two banks for each set of interleaved memory, we have two 256K × 4 and one 256K × 1 for parity, which means three chips for each bank. Therefore, the minimum memory addition requires 12 chips, eight of which are 256K × 4 and four are 256K × 1 for parity bits, resulting in 1 megabyte.

22.2: PAGE AND STATIC COLUMN DRAMS
page mode DRAM

- Storage cells inside DRAM are organized in a matrix of *N* rows & *N* columns, so in reading a cell:
  - The address for the *row* (A1–An) is provided first, and **RAS** is activated.
  - The address for the *column* (A1–An) is then provided and **CAS** is activated.
- In DRAM literature the term *page* refers to a number of column cells in a given row.
  - The row address is provided first, latched by RAS.
  - The column addresses are provided, and CAS toggles.
    - Latching column addresses until the last column of a given page is accessed.

22.2: PAGE AND STATIC COLUMN DRAMS
page mode DRAM

- Since memory locations are accessed consecutively in most situations, there is no need to provide both the row and column address for each location.
  - As was the case in DRAM with standard timing.

Example 22-8

Show how memory storage cells are organized in each of the following DRAM chips.
(a) 256K × 1 (b) 1M × 1 (c) 4M × 1

Solution:

(a) The 256K × 1 has 9 address pins (A0–A8); therefore, cells are organized in a matrix of $2^9 \times 2^9$ = 512 × 512, giving 512 rows, each consisting of 512 columns of cells.
(b) 1024 × 1024
(c) 2048 × 2048

22.2: PAGE AND STATIC COLUMN DRAMS
page mode DRAM

- While the access time of the first cell is the standard access time using both row and column (t_{RAC}), the access time in accessing the second cell on the last cell of the same page (row) is much shorter.
  - Often referred to as t_{CAC}. (T of column access)

Example 22-9

Assuming that the DRAMs in Example 22-8 are of page mode, show how each chip is organized into pages. Find the number of columns per page for (a), (b), and (c).

Solution:

(a) For 1M × 1 we have 512 pages, where each page has 512 columns of cells.
(b) 1024 pages, where each page has 1024 bits (columns).
(c) 2048 pages each of 2048 bits

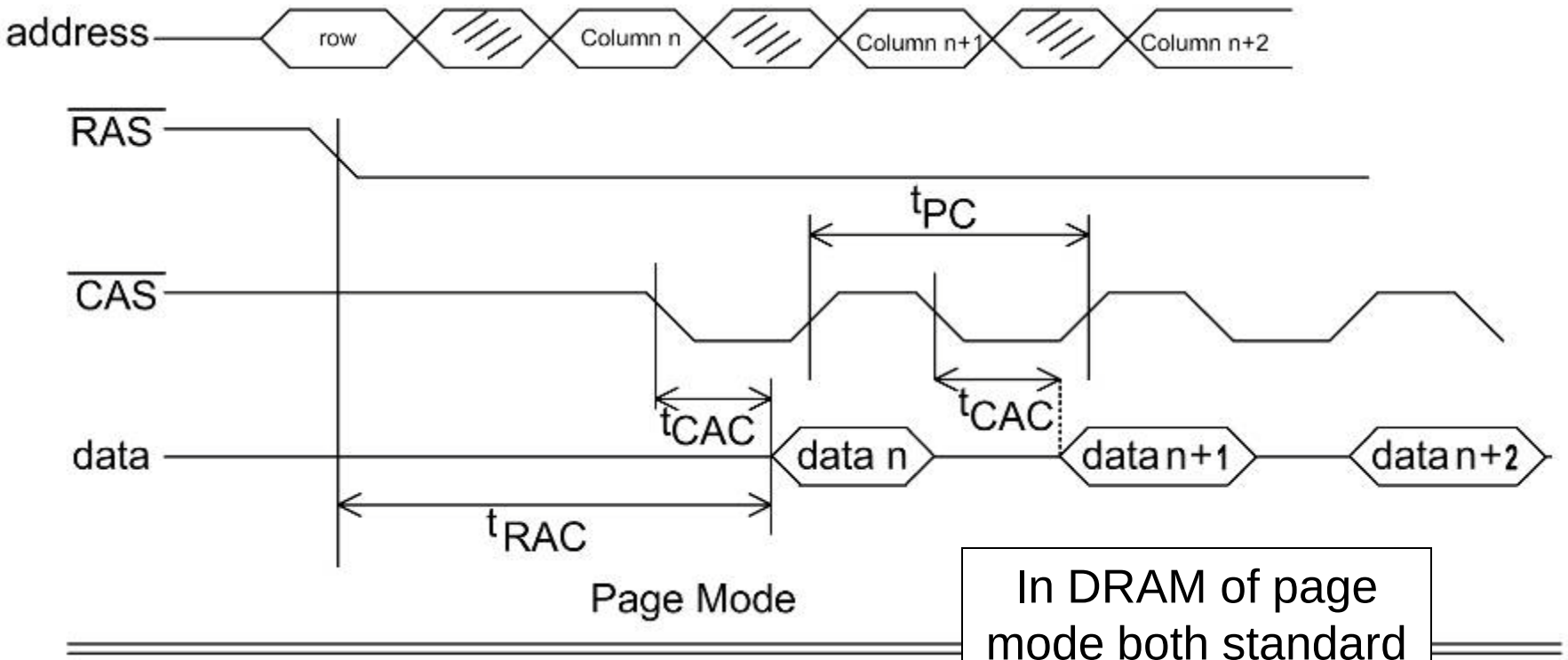
22.2: PAGE AND STATIC COLUMN DRAMS
page mode DRAM

- In page mode DRAM when in a given page, each successive cell can be accessed no faster than $\mathbf{t_{PC}}$ (page cycle time).

Table 22-4: Page Mode DRAM Timing Parameters (4M x 1)

Page Mode DRAM	Access Time from RAS, t_{RAC} (ns)	Read Cycle Time, t_{RC} (ns)	Access Time from CAS, t_{CAC} (ns)	Page Cycle Time, t_{PC} (ns)
MCM44100-60	60	110	15	40
MCM44100-70	70	130	20	45
MCM44100-80	80	150	20	50

22.2: PAGE AND STATIC COLUMN DRAMS
page mode DRAM



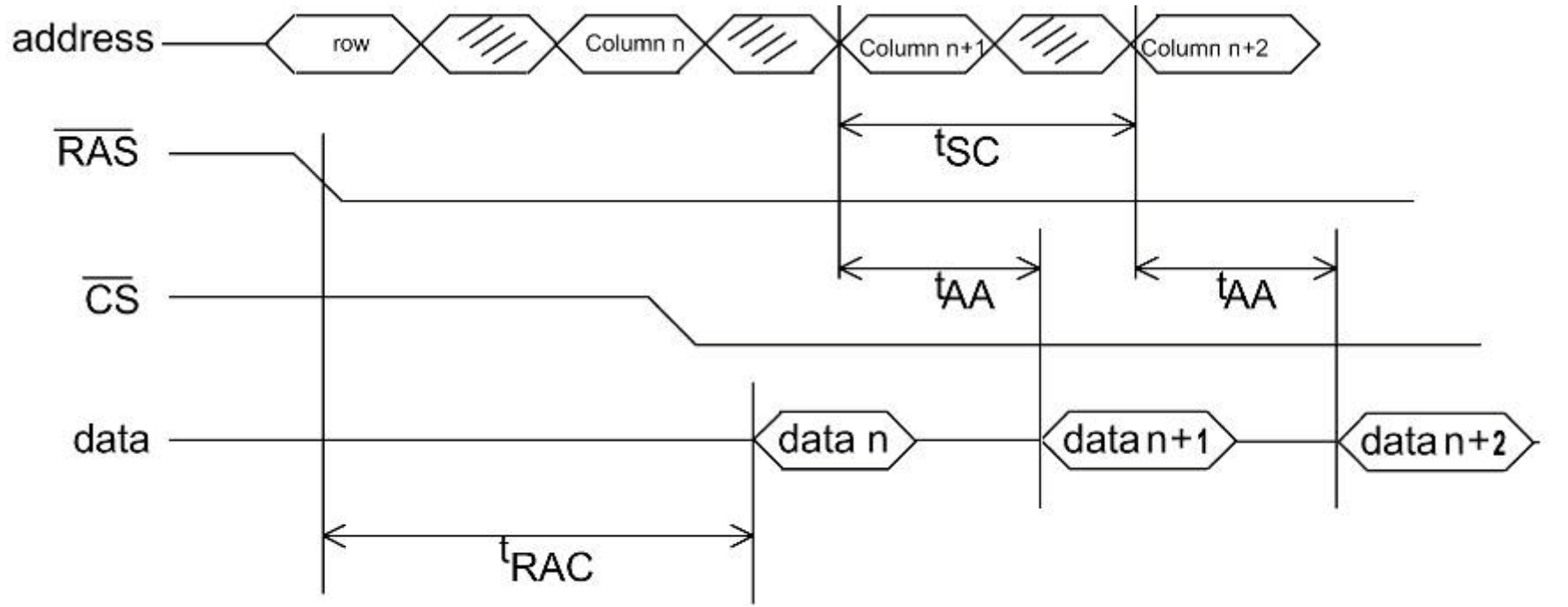
In DRAM of page mode both standard and page mode are supported.

Fig. 22-3a DRAM Page Mode

22.2: PAGE AND STATIC COLUMN DRAMS
static column mode

- Static column mode makes accessing columns of a given row simpler by eliminating the need for CAS.
  - The first location is accessed with a standard read cycle where the row address is latched by RAS.
  - It is followed by the column address, and the CS (chip select) clock.
    - From then on, CS is incremented internally.
- While RAS & CS remain *low*, the contents of successive cells appear at the DRAM data output pin until the last column of a given row is accessed.
  - Then the process is moved to the next row.

22.2: PAGE AND STATIC COLUMN DRAMS
static column mode



RAS & CS (chip select) need to be kept *low* to access successive cells.

Static Column Mode

Fig. 22-3b DRAM Static Column Modes

22.2: PAGE AND STATIC COLUMN DRAMS
static column mode

- Initial access time of the first cell is the standard access time ($\mathbf{t_{RAC}}$), but each subsequent column in that row is accessed in a time called $\mathbf{t_{AA}}$. (access time from column address).

Table 22-5: Static Column DRAM Timing Parameters (4M × 1)

Static Column DRAM	T RAS Access, t_{RAC} (ns)	T Read Cycle, t_{RC} (ns)	T Column Access, t_{AA} (ns)	Cycle Time, t_{SC} (ns)
MCM54102A-60	60	110	30	35
MCM54102A-70	70	130	35	45
35MCM54102A-80	80	150	40	45

A large percentage of 80386 & higher computers use static column DRAM.

22.2: PAGE AND STATIC COLUMN DRAMS
static column mode

- If the time spent by the CPU is the same for both the page mode and static column mode…
  - *What is the advantage of static column mode?*

Example 22-10

Calculate the total time spent by the CPU to access an entire page of memory if the memory banks are page mode DRAM of 1M × 1 with t_{RC} =165 ns, t_{RAC} = 85 ns, and t_{PC} = 50 ns.

Solution:

For page mode we have the following for reading 1024 bits:

Time to read 1024 bits of the same page $= t_{RAC} + 1023 \times t_{PC}$

$= 85 \text{ ns} + 1023 \times 50 \text{ ns} = 51{,}235 \text{ ns}$

22.2: PAGE AND STATIC COLUMN DRAMS
static column mode

- Static-column-mode DRAM design is simpler since there is no circuit or timing requirement for pin CAS.

Example 22-11

Calculate the total time spent by the CPU to access the entire page of memory if the memory banks are static-column-mode DRAMs of 1M × 1 with t_{RC} = 165 ns, t_{RAC} = 85 ns, and t_{SC} = 50 ns.

Solution:

For static column mode we have the following for reading 1024 bits:

time to read 1024 bits of the same page $= t_{RAC} + 1023 \times t_{SC}$

$= 85 \text{ ns} + 1023 \times 50 \text{ ns} = 51{,}235 \text{ ns}$

22.3: CACHE MEMORY

- The most widely used memory design uses DRAMs for main memory & a small amount of cache SRAM.
  - Cache is placed **between** the CPU and main memory.

When the CPU initiates memory access, it first asks cache for the information. (data or code)

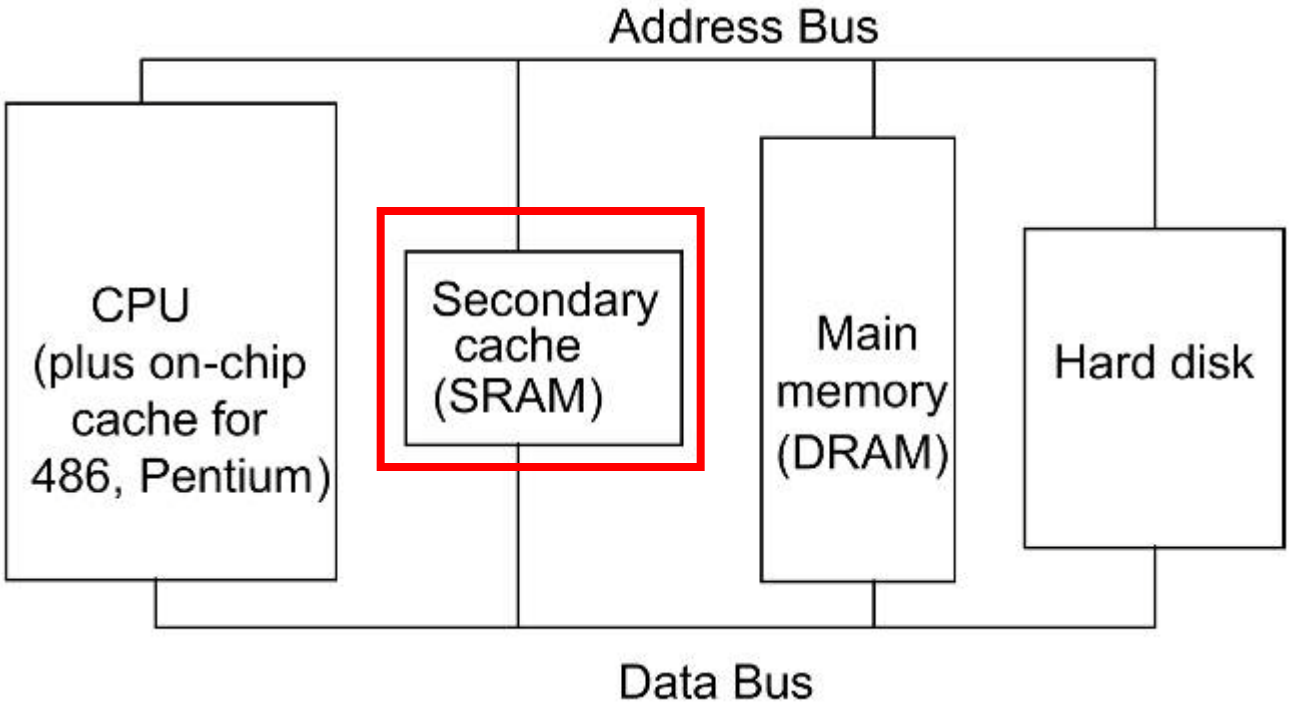


Fig. 2-4
CPU and its Relations to Various Memories

22.3: CACHE MEMORY

- If the requested data is there, it is provided to the CPU with zero wait states.
  - Called a *hit*.
- If the data is not in cache, the memory controller will transfer the data from main memory to the CPU
  - Called a *miss*.
- The memory controller gives a copy of the data to cache memory to allow any subsequent request for the same information to result in a hit and provide it to the CPU with zero wait states.
  - In most computers with cache, hit rate is 85% and higher.

22.3: CACHE MEMORY

- By combining SRAM and DRAM, cache memory's access time matches the memory cycle of the CPU.
  - Absolutely essential in the 80386/486 33 MHz & above.
- When the CPU accesses memory, it is likely to access the information in the vicinity of the same addresses, at least for a time.
  - Called the *principle of locality of reference*.
- The *hit rate*, the number of hits divided by the total number of tries, depends on the size of the cache, how it is organized (cache organization), and the nature of the program.

22.3: CACHE MEMORY
cache organization

- There are three types of cache organization:
  - Fully associative; Direct mapped; Set associative.

22.3: CACHE MEMORY
fully associative cache

- In fully associative cache, only a limited number of bytes from main memory are held by cache along with their addresses.
  - SRAMs holding ***data*** are called *data cache*.
  - SRAMs holding ***data addresses*** are called *tag cache*.
- When the information is brought into cache, the contents of the memory locations & their associated addresses are saved in the cache
  - The more data that is kept, the higher the hit rate.

22.3: CACHE MEMORY
fully associative cache

- The problem with fully associative cache is that if depth is increased to raise the hit rate, the number of comparisons is too time consuming & inefficient.

A cache with a depth of 1024 requires 1024 comparisons, too time consuming even for fast comparators.

At a depth of 16, the CPU waits for data too often.

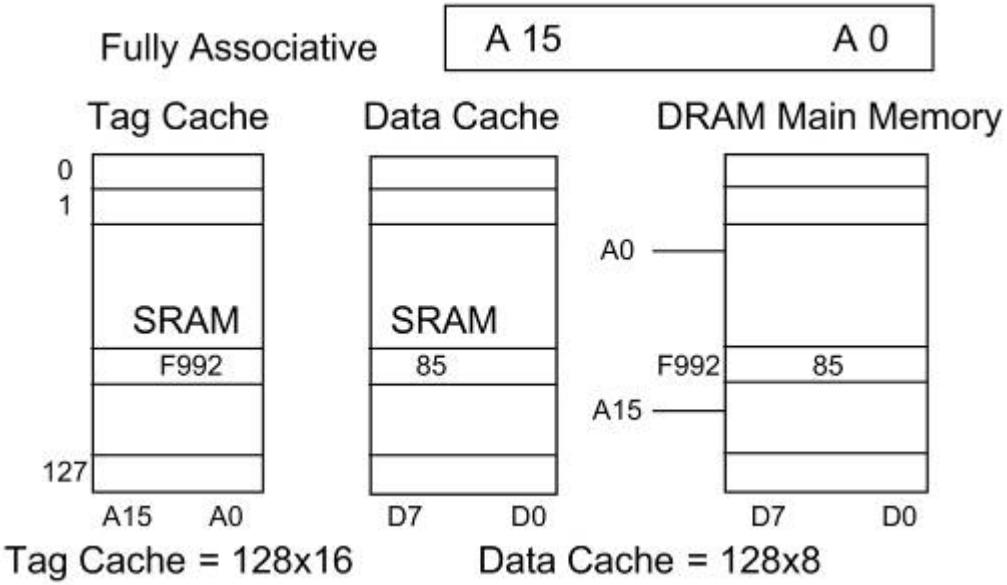


Fig. 22-5 Fully Associative Cache

22.3: CACHE MEMORY
direct-mapped cache

- Direct-mapped cache is the opposite extreme of fully associative, requiring only one comparison.
  - The address is divided into the *index* and the *tag*.

The index is the lower part of the address, directly mapped into SRAM—**A0** to **A10**.

The upper part of the address is held by the tag SRAM—**A11** to **A15**.

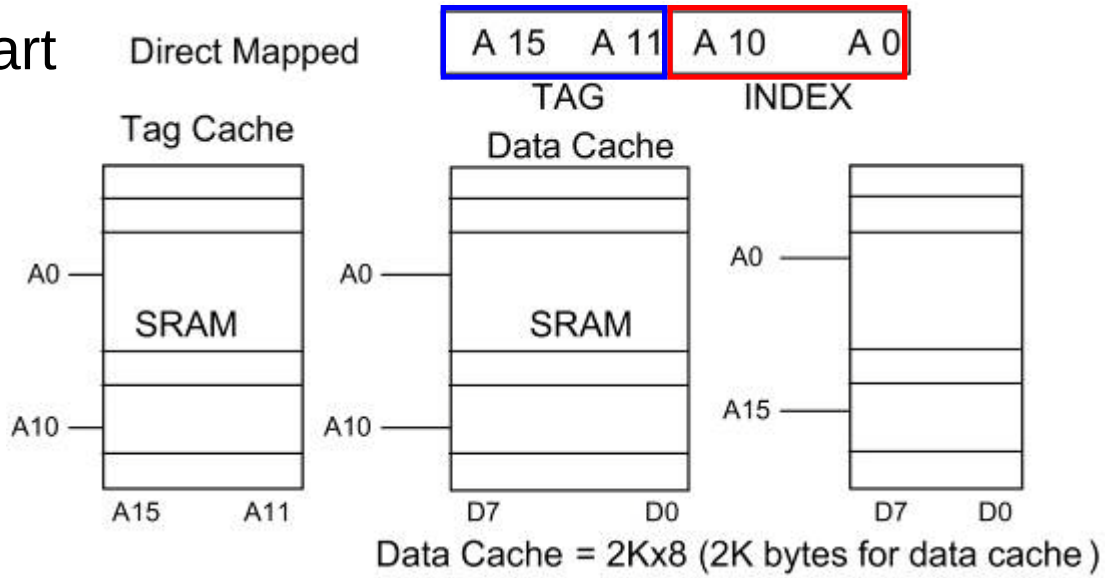


Fig. 22-6 Direct-Mapped Cache

22.3: CACHE MEMORY
direct-mapped cache

- Assuming the CPU addresses location F7A9H, the 7A9 goes to the index, but data is not read until tag location 7A9 is compared with 11110B.
  - If it matches (content is 11110), data is read to the CPU.
  - Otherwise, the processor must wait until the contents of F7A9 are brought from memory DRAM to the CPU.
    - While a copy of it is issued to cache for future reference.
- While the number of comparisons is reduced to one, the problem of accessing information from locations with the same index but different tag, such as F7A9 and 27A9, is a drawback.

22.3: CACHE MEMORY
set associative

- In set associative cache, the number of tags for each index is increased, increasing the hit rate.
  - In **2-way** set associative, **two tags** for each index.

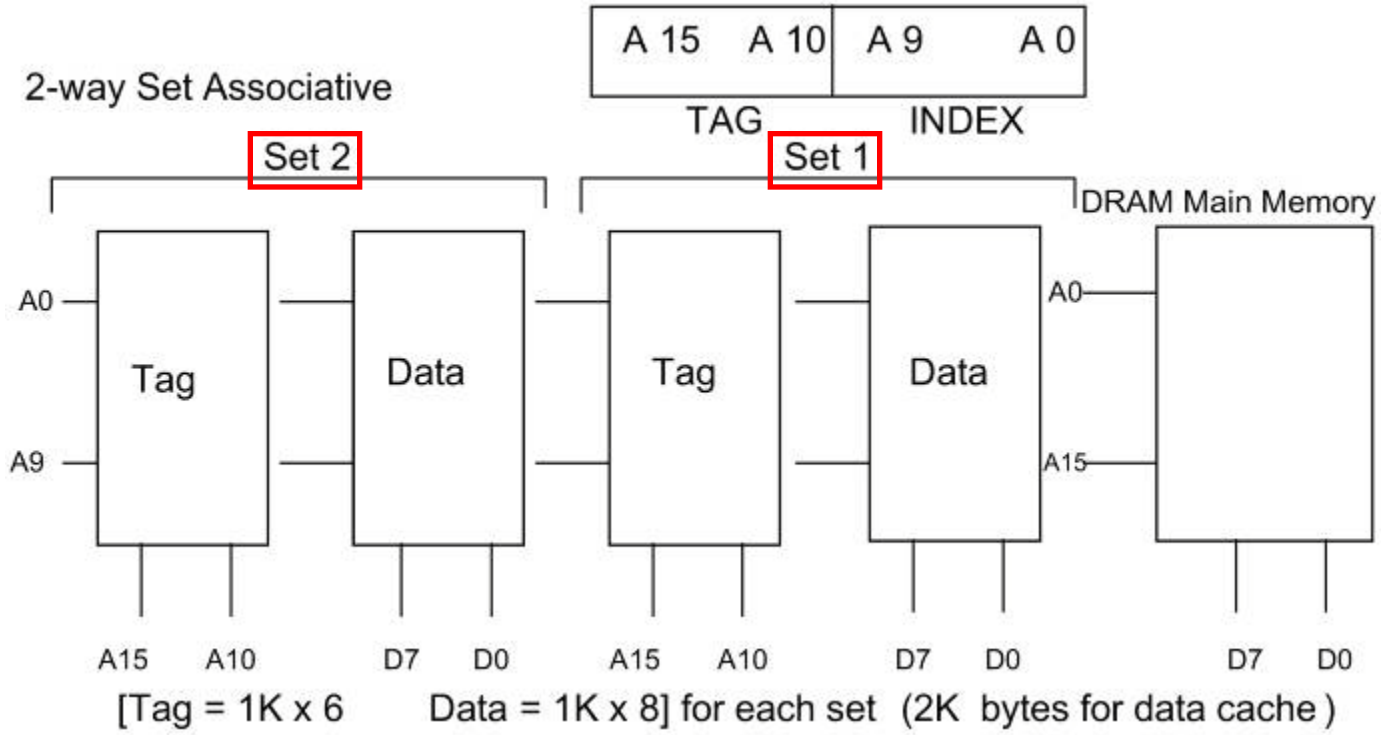


Fig. 22-7
Two-Way Set Associative

22.3: CACHE MEMORY
set associative

- In set associative cache, the number of tags for each index is increased, increasing the hit rate.
  - In **4-way**, there are **4 tags** for each index.

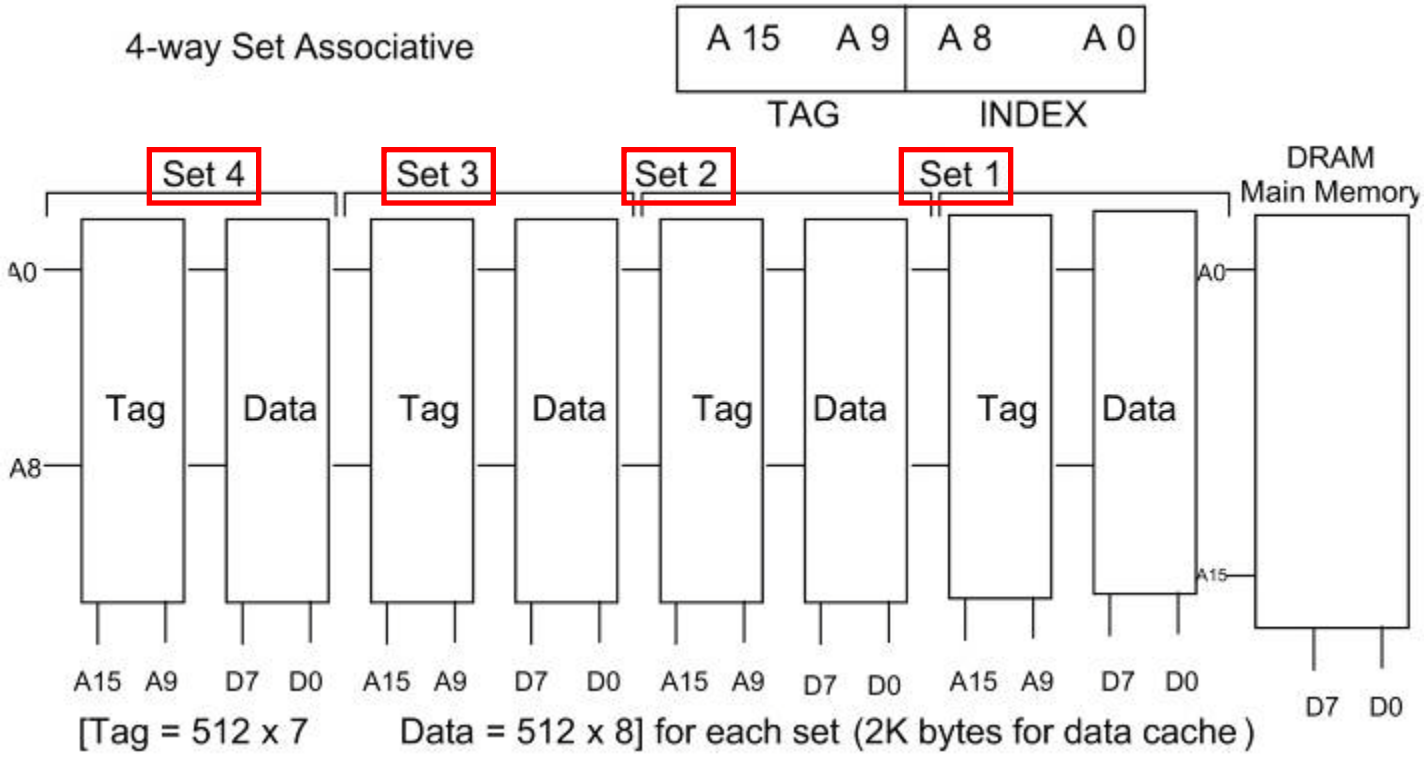


Fig. 22-8
Four-Way Set Associative

22.3: CACHE MEMORY
set associative

- In this organization, if the processor is requesting the contents of memory location 41E6H, there are 2 possible tags that could hold it.
  - If any of them matches it, the data of index location 1E6 is read to the CPU.
  - If none of the tags matches "0100 00", the miss will force the controller to bring the data from DRAM to cache.
    - With a copy of it is provided to the CPU at the same time.
- In 4-way set associative, the search is initiated by comparing the 4 tags with "0100 000“.
  - Which will increase the chance of having the data in the cache by 50%, compared with 2-way set associative.

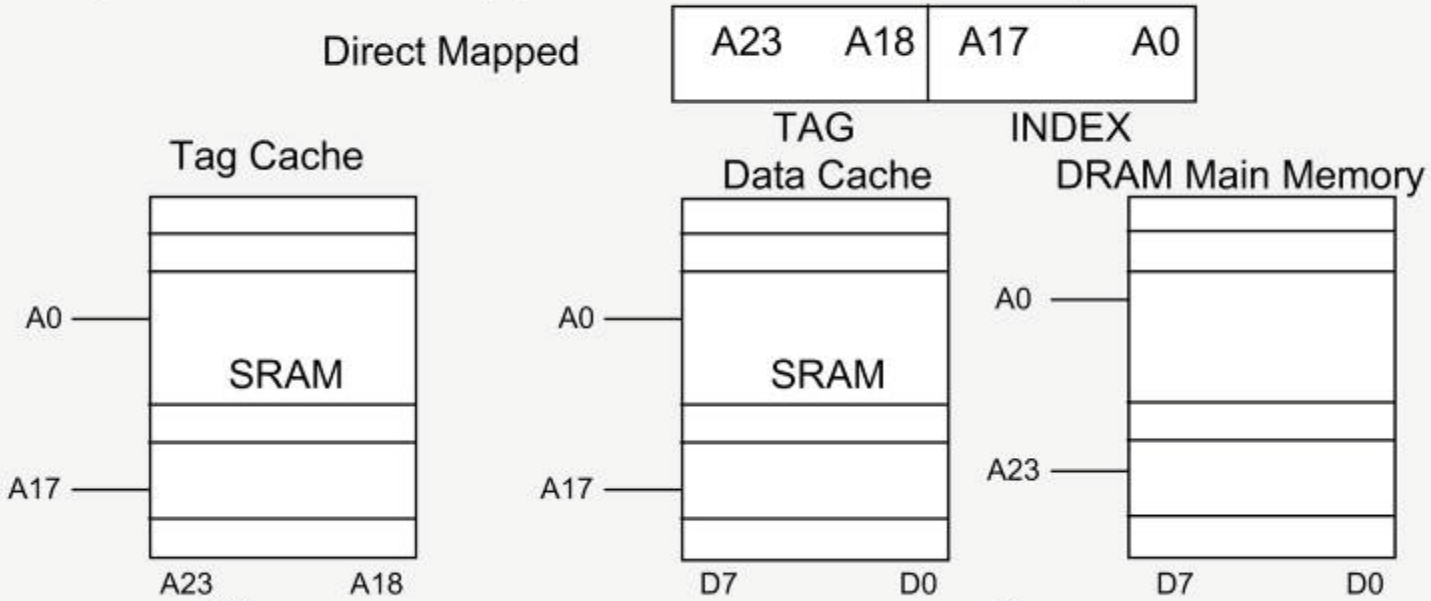
22.3: CACHE MEMORY
set associative

- The number of comparisons in set associative depends on the *degree* of associativity.
  - It is 2 for 2-way set associative.
  - 4 for 4-way; 8 for 8-way; *n* for *n*-way set associative
  - In the thousands for fully set associative.
- The higher the set, the better the performance, but the more SRAM required for tag cache
  - Making 8- and 16-way associatives' increased costs unjustifiable, compared to the small increase in hit rate.

22.3: CACHE MEMORY
set associative

Example 22-12

This example shows directed-mapped cache for 16M main memory.



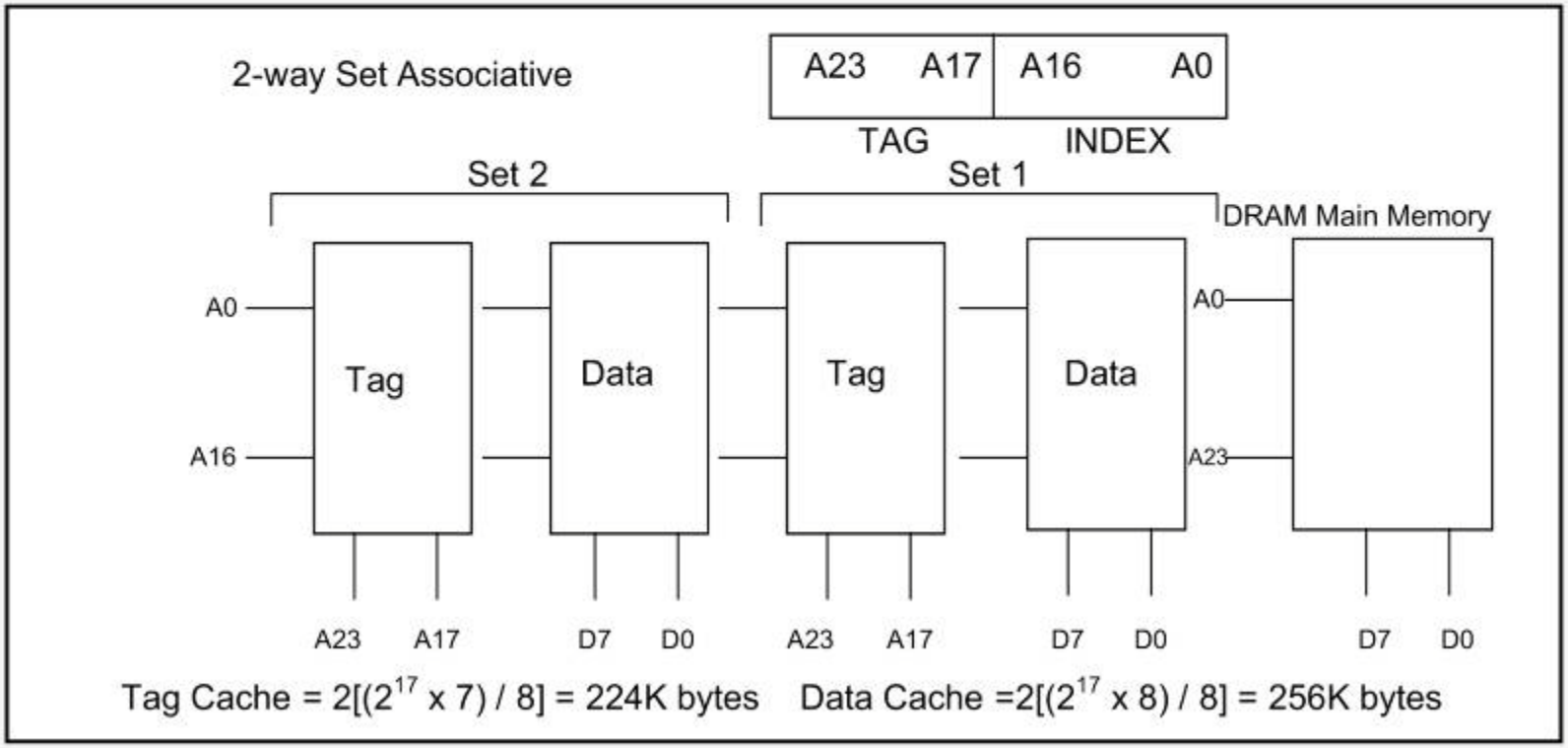
Tag Cache = $(2^{18} \times 6) / 8 = 192K$ bytes

Data Cache = $(2^{18} \times 8) / 8 = 256K$ bytes

22.3: CACHE MEMORY
set associative

Example 22-13

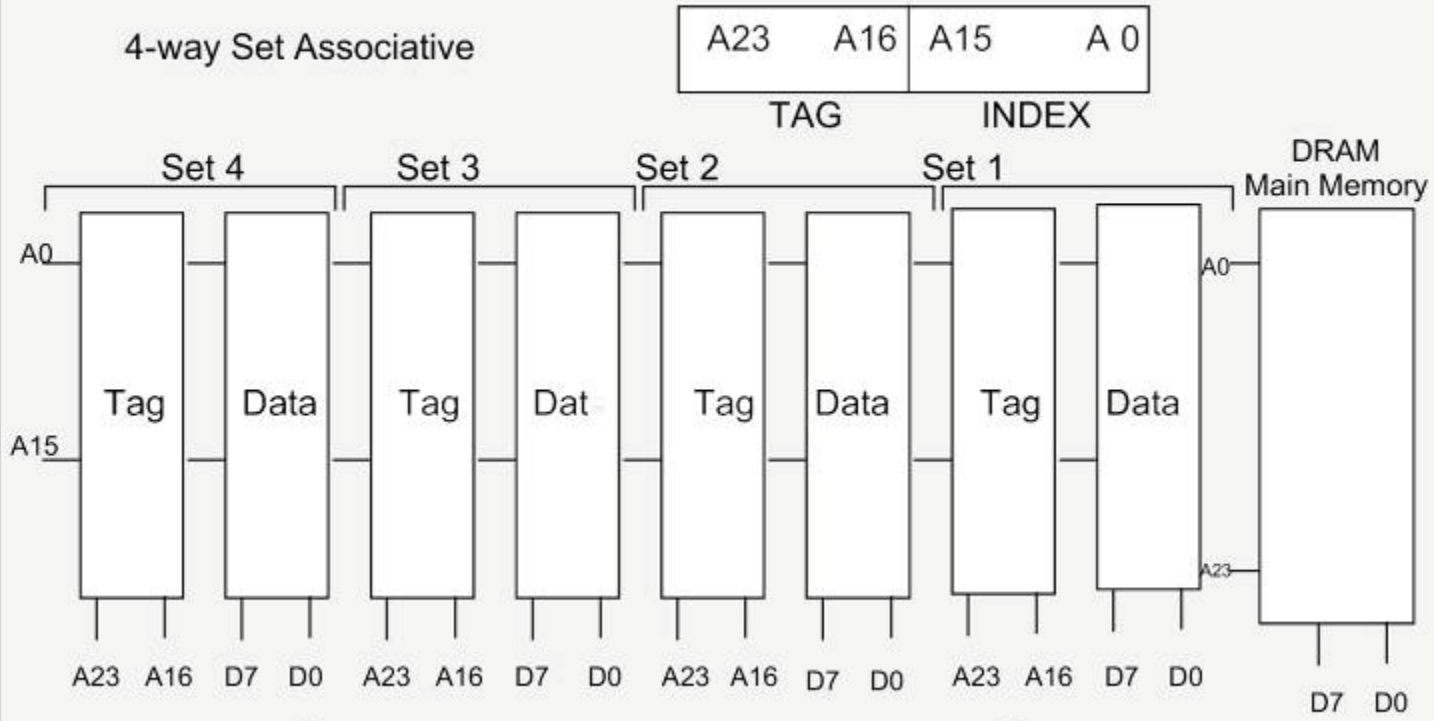
This example shows 2-way set associative mapped cache for 16M main memory.



22.3: CACHE MEMORY
set associative

Example 22-14

This example shows 4-way set associative mapped cache for 16M main memory.



Tag Cache = $4[(2^{16} \times 8) / 8] = 256K$ bytes Data Cache = $4[(2^{16} \times 8) / 8] = 256K$ bytes

22.3: CACHE MEMORY
updating main memory

- To prevent data inconsistency between cache & main memory, there are two major methods of updating the main memory:
  - Write-through & Write-back.

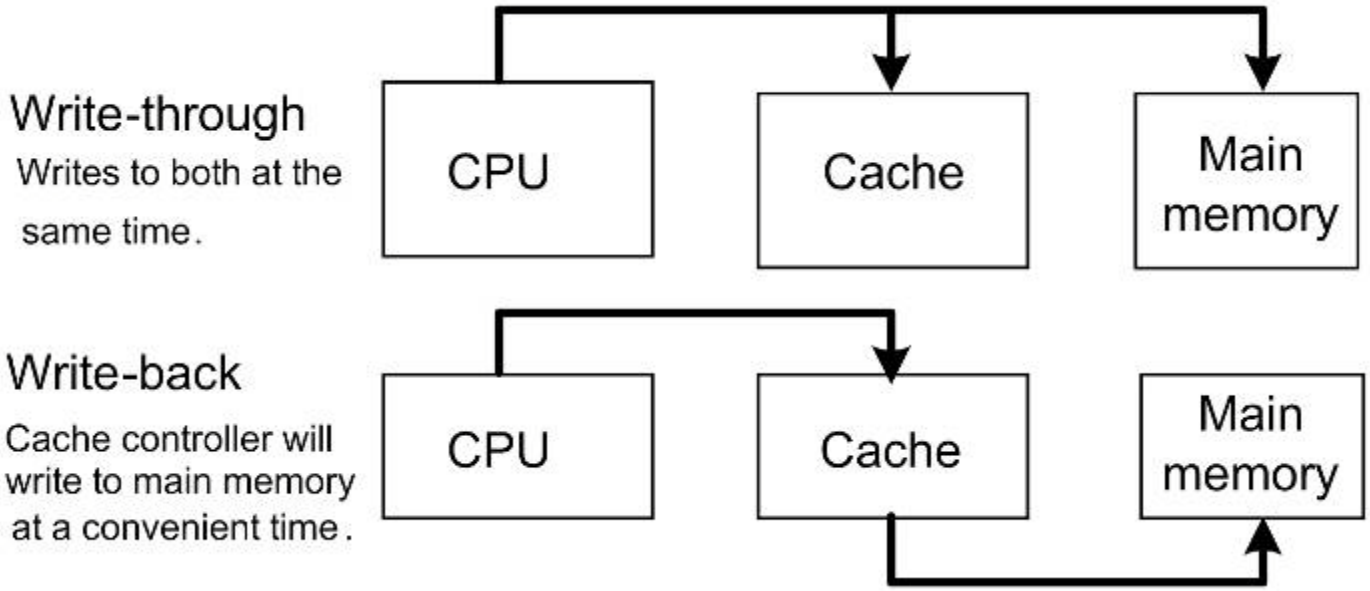
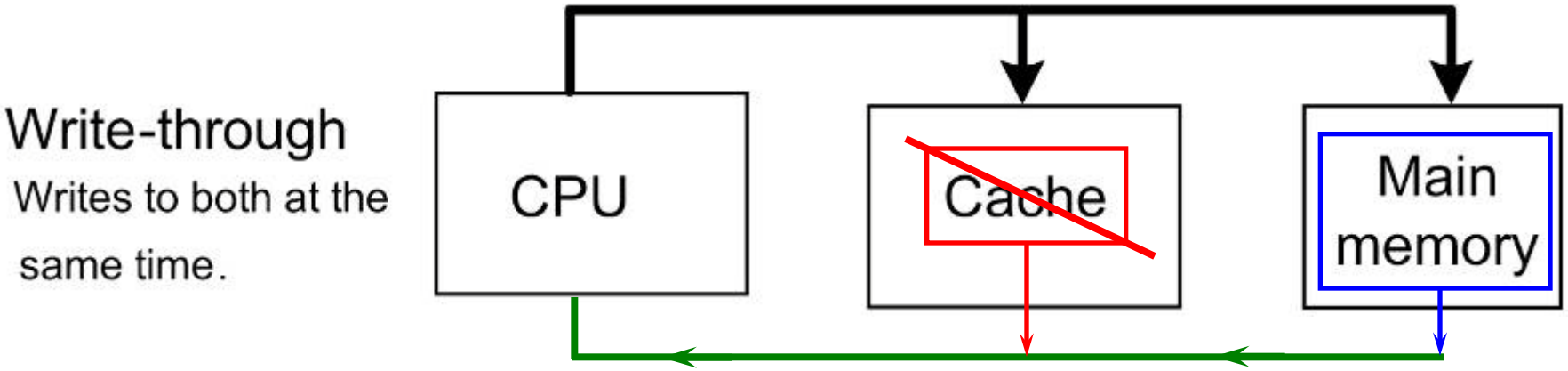


Figure 22-9 Method of Updating Main Memory

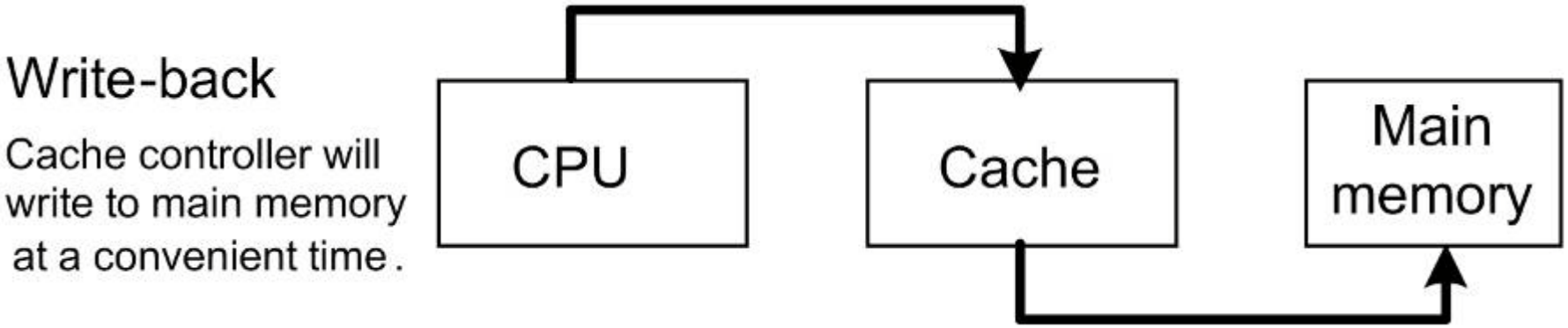
22.3: CACHE MEMORY
write-through

- In write-through, the data will be written to cache and to main memory at the same time.
  - At the cost of increasing bus traffic to main memory, this ensures main memory always has valid data,
    - If the **cache** is overwritten, the **copy** of the latest valid data can be accessed from main memory.



22.3: CACHE MEMORY
write-back (copy-back)

- In the write-back (called copy-back) policy, a copy of the data is written to cache by the processor.
  - The data will be written to main memory by the cache controller *only* if cache's copy is about to be altered.



  - If cache data has not been altered & is the same as main memory, there is no need to write it again & increase the bus traffic.

22.3: CACHE MEMORY
write-back (copy-back)

- The cache has an extra bit called the dirty bit, also called the altered bit.
  - If data is written to cache, the dirty bit is set to 1.
    - To indicate cache data is data that exists only in cache and not in main memory.
    - At a later time, the cache data is written to main memory and the dirty bit is cleared.
- When the dirty bit is *high,* the cache controller will make sure before erasing the new data in cache, a copy of it is given to main memory.
  - Getting rid of information in cache is often referred to as *cache flushing*.

22.3: CACHE MEMORY
cache coherency

- In systems in which memory is accessed by more than one processor (DMA or multiprocessors), it must be ensured that cache always has the most recent data and not old (or stale) data.
  - If data in main memory is changed by one processor, the cache of *that* processor has the copy of the latest data.
- In cases where there is more than one processor and all share a common set of data, there must be a way to ensure that no processor uses stale data.
  - This is called *cache coherency*.

22.3: CACHE MEMORY
cache replacement policy

- In the **LRU** (least recently used) algorithm…
  - The controller keeps account of which block of cache has been accessed (used) the least number of times, and this block will be swapped out to main memory
    - Or flushed if a copy of it already exists in main memory.
- Other replacement policies are to overwrite the blocks of data in cache sequentially or randomly.
  - Or use the **FIFO** (*first in, first out*) policy.

22.3: CACHE MEMORY
cache fill block size

- *How many bytes of data are brought in when there is a miss?*
  - If the block size is too large, it will be too slow since the main memory is accessed normally with 1 or 2 WS.
  - If the block is too small, there will be too many misses.
- Block size transfer from main memory to CPU varies between 32 and 512 bytes.
  - 32 bytes block size is called the 8-line cache refill policy.
    - Each line is 4 bytes of the 32-bit data bus.

22.3: CACHE MEMORY
level 1, 2, and 3 caches

- Advances in IC fabrication has allowed placing some caches on the CPU chip itself.
  - Cache embedded into the CPU die, is called **L1** (*level 1*) cache.
  - If the cache is on-chip, inside the package but *outside* of the CPU die, then it is called **L2** (*level 2*) cache.
  - Cache outside the CPU, residing on the motherboard is called **L3** (*level 3*).

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES

- Three very high-speed DRAMs:
  - EDO (extended data-out)
  - SDRAM (synchronous DRAM)
  - RDRAM (Rambus DRAM)

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
EDO DRAM: origin and operation

- Page mode DRAM has been modified and now is referred to as fast page mode DRAM, and has led to EDO DRAM.
  - 1. The row address is provided & latched when RAS falls.
    - This opens the page.
  - 2. The column address is latched in when CAS falls and data shows up after $\mathbf{t_{CAC}}$ has elapsed.
    - However, the next column of the same row (page) cannot be accessed faster than $\mathbf{t_{PC}}$ (page cycle time).
  - The $\mathbf{t_{PC}}$ timing itself is influenced by how long CAS has to stay low before it goes up.

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
EDO DRAM: origin and operation

- *Why don't DRAM designers pull up the* **CAS** *faster in order to shorten the* t_{PC}*?*
  - When the CAS goes high, the data output is turned off, so if CAS is pulled high too fast, the CPU is deprived of data.
- A solution is to change the internal circuitry of fast page DRAM to allow the data to be available longer.
  - The name **EDO** (*extended data-out*) was given to avoid confusion with fast page mode DRAM.
    - EDO is sometimes called hyper-page.

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
EDO DRAM: origin and operation

- A comparison of FPM and EDO DRAM timing.
  - In both cases all parameters are the same except t_{PC}.
  - For the EDO version of page mode, t_{PC} is 10 ns less than fast page mode.

Table 22-7:
70 ns 4M DRAM Timing

	FPM	EDO
Speed (ns)	70	70
tRAC (ns)	70	70
tRC (ns)	130	130
tPC (ns)	40	30

Table 22-8:
60, 50 ns 4M DRAM Timing

	FPM	EDO	EDO
Speed (ns)	60	60	50
tRAC (ns)	60	60	50
tRC (ns)	110	110	100
tPC (ns)	35	25	20

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
EDO DRAM: origin and operation

- A comparison of FPM and EDO DRAM timing.
  - In both cases all parameters are the same except $\mathbf{t_{PC}}$.
  - For the EDO version of page mode, $\mathbf{t_{PC}}$ is 10 ns less than fast page mode.

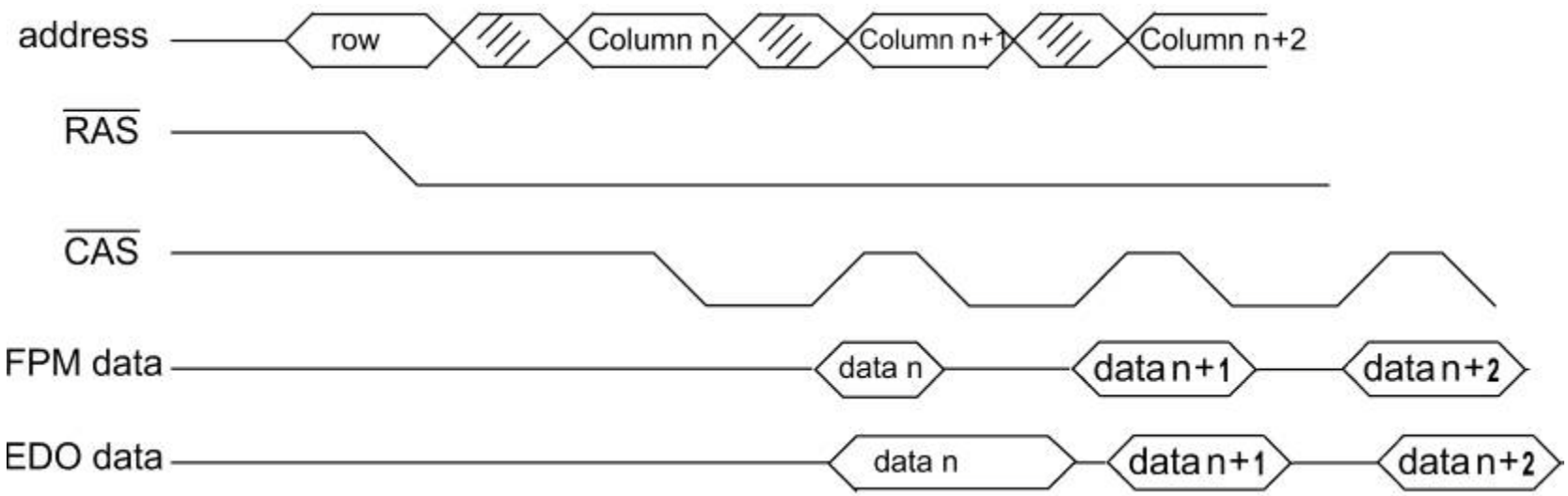


Figure 22-11 Comparison of FPO and EDM Timing

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
EDO DRAM: origin and operation

- Note that $\mathbf{t_{PC}}$ (page cycle time) has two portions:
  - $\mathbf{t_{CP}}$ (CAS precharge time) and $\mathbf{t_{CAS}}$ (CAS pulse width).
- The $\mathbf{t_{CP}}$ is similar across 70, 60, & 50 ns DRAMs of FPM and EDO (about 10 ns).
  - It is $\mathbf{t_{CAS}}$ that varies among these DRAMs.
    - In EDO this portion is made as small as possible.

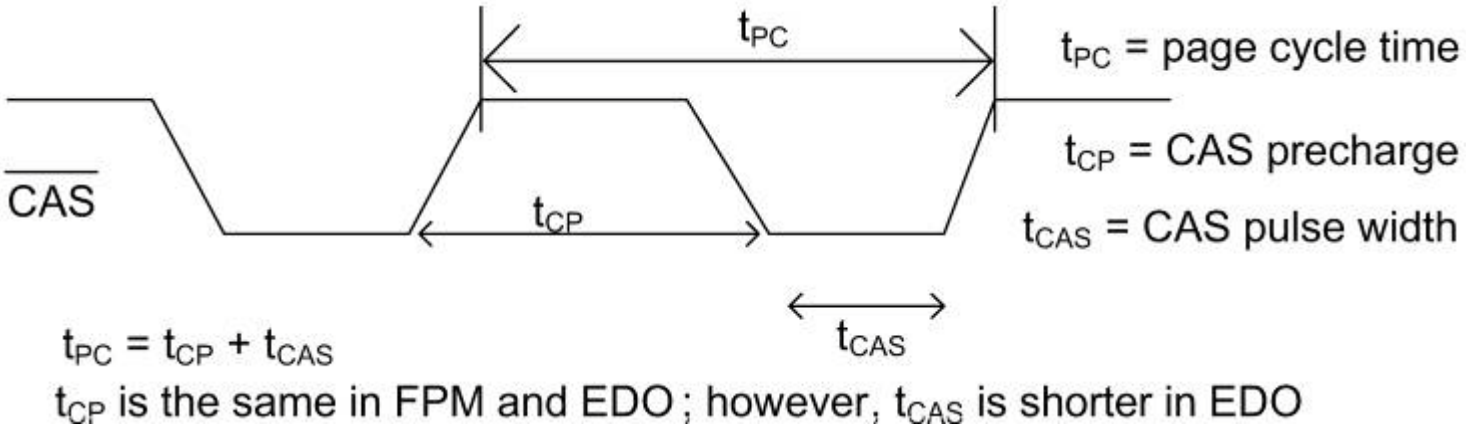


Figure 22-10
Timing in Page Mode DRAM

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
SDRAM (synchronous DRAM)

- In traditional DRAM, CPU timing is not synchronized with DRAM timing—there is no common clock between the CPU and DRAM for reference.
  - The DRAM is *asynchronous* with the processor.
- The CPU presents the address to DRAM & memory provides the data in the master/slave fashion.
  - If data cannot be provided on time, the CPU is notified with the NOT READY signal.
    - CPU bus timing is dependent upon the DRAM speed.

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
SDRAM (synchronous DRAM)

- In systems with SDRAM, there is a common clock (the system clock) between processor & SDRAM.
  - All bus activities (address, data, control) between the CPU & DRAM are synchronized with this common clock.
    - The clock is the point of reference for both CPU & SDRAM, there is no deviation from it, and hence no waiting by the CPU.

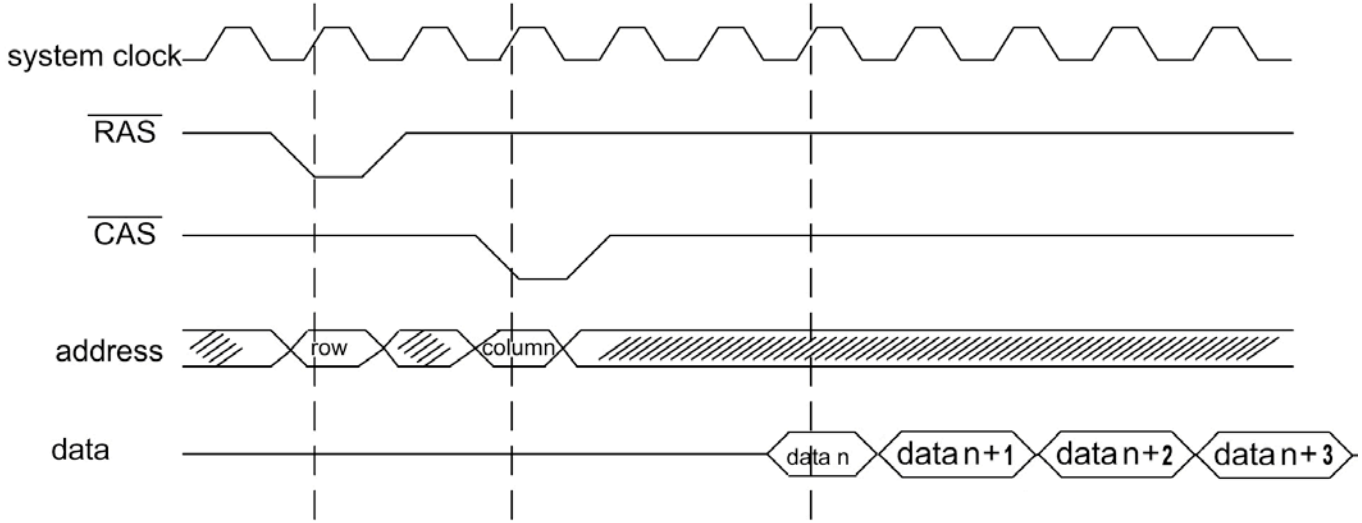


Figure 22-12
SDRAM Timing

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
SDRAM, DDR RAM, and burst mode

- The common system clock between the CPU and SDRAM lends itself to what is called burst I/O.
  - In burst read, the address of the first location is provided as normal—RAS is first, followed by CAS.
- Since several locations in the cache fill are read, the burst SDRAM is programmed with the consecutive locations needed according to cache design.
  - The number of burst reads can be 1, 2, 4, 8, 16, or 256, and burst SDRAM can be programmed in advance for any number of these reads.
    - The number of burst reads is referred to as *burst length*.

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
SDRAM and interleaving

- In order to increase performance, SDRAMs use the concept of interleaving, performed internally.
  - Inside the SDRAM itself, cells are organized so one bank is being refreshed while the other one is being accessed.
- Burst mode and interleaving can be used for a bus frequency as high as 125 MHz.
  - But not beyond.

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
SDRAM and interleaving

- *How many clocks after* **CAS** *will the data appear at the data pins?*
  - Called read latency, programmable as 1, 2, or 3 clocks.
  - The read latency shown here is 3, since the data appears at the data buses 3 clocks after **CAS**.

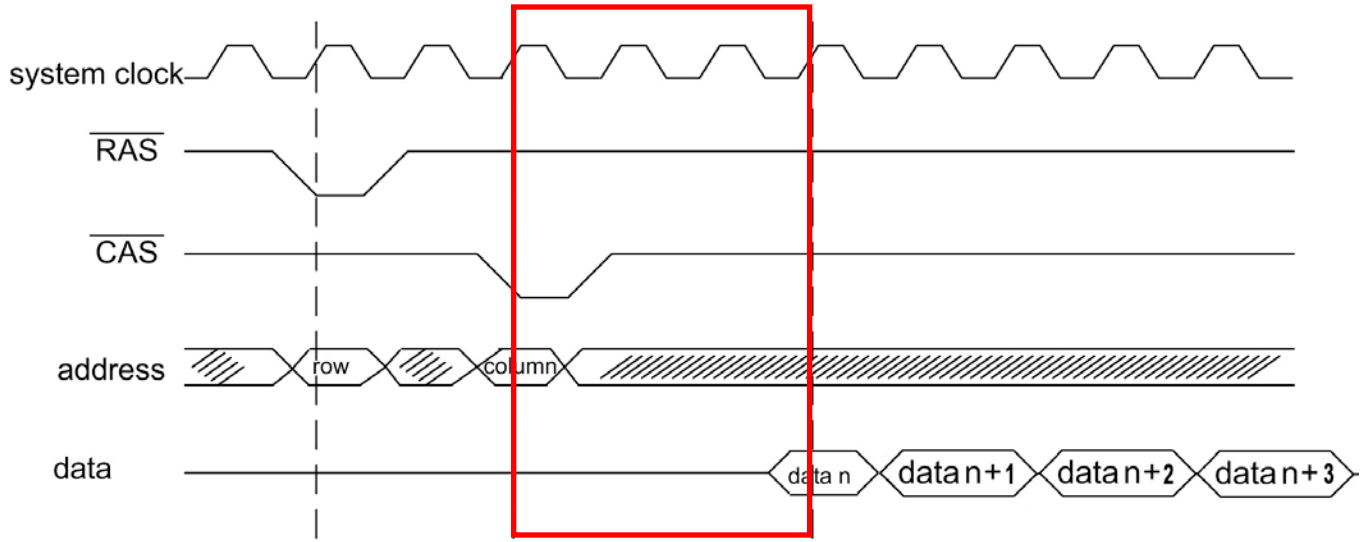


Figure 22-12
SDRAM Timing

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
SDRAM and interleaving

Example 22-15

Assume a bus frequency of 100 MHz. Discuss bus timing for (a) EDO of 50 ns speed where t_{CP} = 20, (b) SDRAM of t_{CK} = 10 ns.

Solution:

1/100 MHz = 10 ns is the system clock.

(a) In EDO when the page is opened, the fastest it can provide data is t_{PC}, which is 20 ns. Therefore, we need at least one wait state.

(b) In SDRAM of t_{CK} = 10 ns, the first address is strobed into the DRAM and subsequent data bursts are provided at 10 ns intervals. Therefore, no wait state is needed. Of course, for both of the above cases any bus overhead was ignored.

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
Overview of Rambus technology

- Rambus is proprietary DRAM architecture interface for chip-to-chip bus connection.
  - DRAM manufacturers license this technology from Rambus Inc., in exchange for royalty payments.

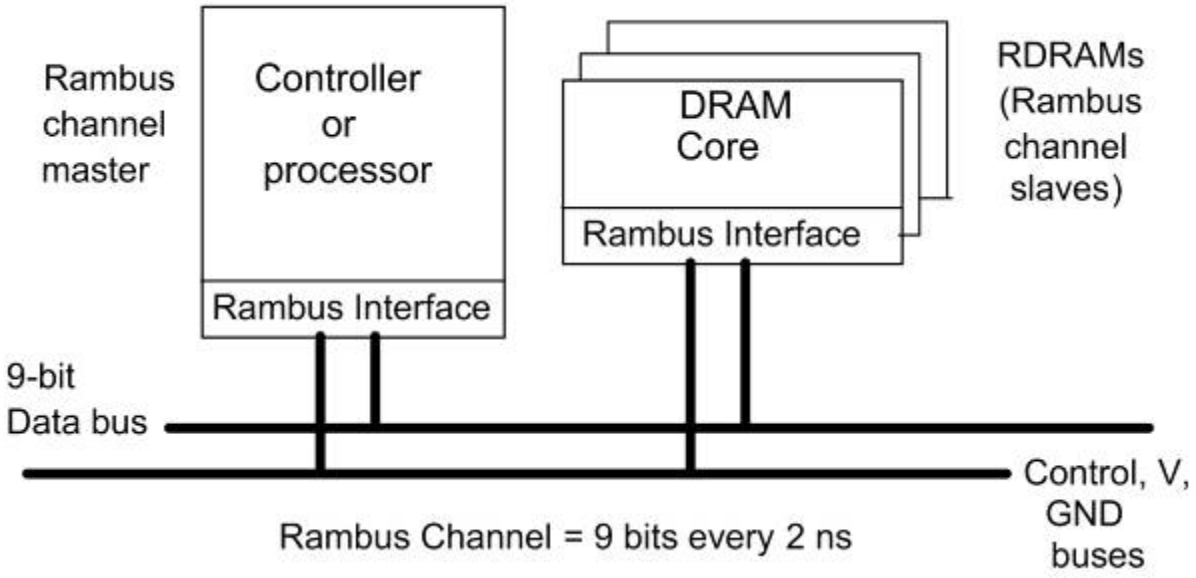


Figure 22-13
A Rambus Based System

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
Overview of Rambus technology

- The Rambus high-speed bus technology must be incorporated into both DRAM & CPU, and is composed of three sections:
  - A Rambus interface.
  - A Rambus channel
  - Rambus DRAM.
- Many DRAM makers are introducing DRAM with a Rambus interface. (called RDRAM)
  - Intel has indicated that it will equip future generations of the x86 with a Rambus interface.

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
Overview of Rambus technology

- If a given processor is not equipped with a Rambus interface, a controller with the Rambus interface can be placed it between the CPU and RDRAM.
  - Referred to as a Rambus *channel master* and the RDRAM is called a Rambus *channel slave*.

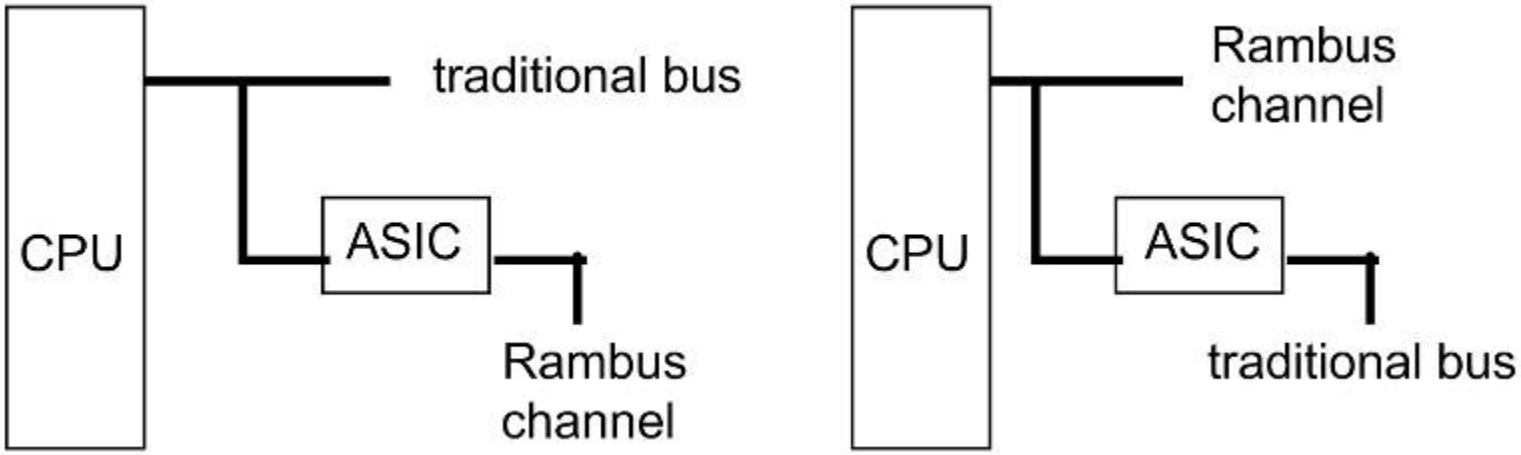


Figure 22-14 CPUs with and without Rambus Channel (*Courtesy of Rambus,Inc.*)

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
Overview of Rambus technology

- In Rambus technology, only the master can generate a request since it contains intelligence.
  - Slave devices such as RDRAM respond to requests.
  - Data transfers can happen only between master and slave and there is never any direct data transfer between slaves.
- Master capability can be added to devices other than the CPU.
  - Such as peripheral devices, graphic processors, and memory controllers.

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
Overview of Rambus technology

- To counter the impact of a limited bus size on bus bandwidth, Rambus employs block transfer, with a set of protocols in which packet types are defined:
  - Request; Acknowledge; Data.

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
Overview of Rambus technology

1. The master issues a request packet specifying the initial starting address of the needed data, plus the number of bytes needed to be transferred (the maximum for byte count is 256 bytes).
   - Considered one transaction.
2. RDRAM receives the request packet and decodes the addresses and byte count.
   - If it has the requested data, an acknowledge packet is sent back to the master.

Overview of Rambus technology

3. The acknowledge packet has three possibilities:
   - (a) The addressed data *does not* exist.
   - (b) The addressed data *does* exist, but it is too busy to transfer the data—try again later. (This is called *nack*)
   - (c) The addressed data does exist and it is ready to transfer them—this is called okay.
4. If the acknowledge packet has an okay in it, the RDRAM starts to transfer the data packet immediately.

22.4: SDRAM, DDR RAM, AND RAMBUS MEMORIES
Overview of Rambus technology

- The delay associated with receiving acknowledge, and sending the data packets can be programmed into configuration registers of both master and slave during BIOS system initialization.

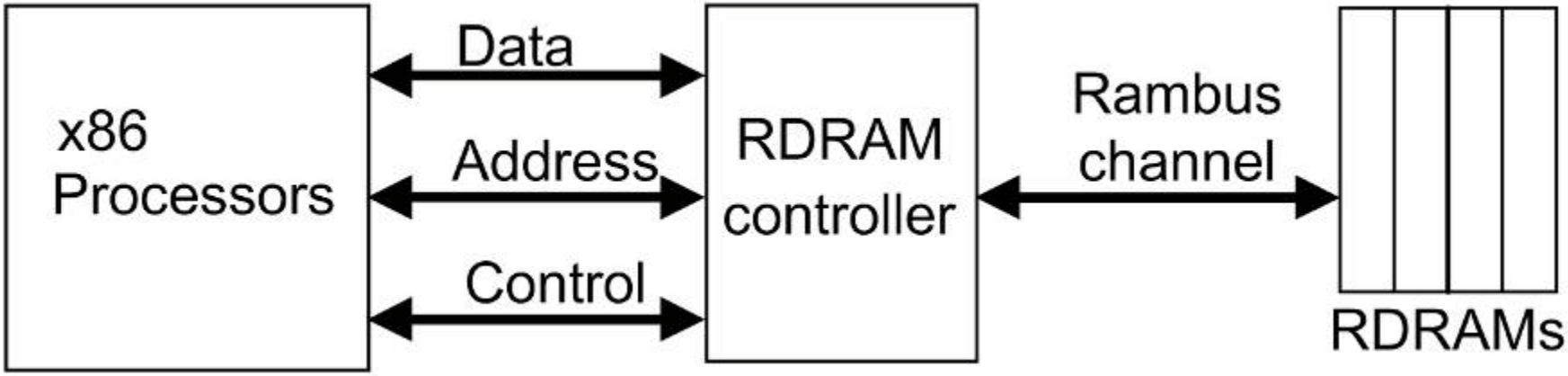


Figure 22-15 x86 System Using Rambus DRAM (*Courtesy of Rambus,Inc.*)

Dec Hex Bin
22 16 00010110
ENDS ; TWENTY-TWO

The x86 PC

assembly language, design, and interfacing

fifth edition

MUHAMMAD ALI MAZIDI
JANICE GILLISPIE MAZIDI
DANNY CAUSEY

The x86 PC
assembly language, design, and interfacing
fifth edition

Prentice Hall