Memory Organisation

Algorithms, Programming, and Data Representation · Module 92313 · Chapter 5

From Fundamentals to AI Apocalypse · personal study guide

Contents

Start from memory 2

How to use this guide 2

How the chapter fits together 3

5.1 Memory hierarchy and properties 4

5.2 RAM cells, addressing, and capacity 6

5.3 Endianness and aligned access 7

5.4 Memory chips and modules 8

5.5 Memory interleaving 9

Formula map and units 10

Derive it cold 10

Explain it 10

Mixed practice 11

Exam-style problems 12

Self-check 13

Answers 14

Made by Thanate Amornjiraporn (Focus)

This chapter explains how computers organise memory, identify locations, lay out multi-byte values, assemble memory modules, and distribute addresses across modules.

Start from memory

Try each question first. Mark it sure, unsure, or guessed.

1. Which is faster: CPU registers, RAM, or a hard disk?
2. What does a byte address select?
3. Which byte goes first in little-endian order?
4. How many bits select one of four memory modules?
5. What is the difference between high-order and low-order interleaving?

Answers: see "Start from memory" under Answers at the end.

How to use this guide

Study a section, hide it, then rebuild its rules and examples from memory. Attempt mixed practice before opening answers.

1. **Day 0:** Try the cold start and study §5.1–5.2; solve practice 1, 4, 6, 21.
2. **Day 1:** Reconstruct byte-order rules in §5.3; solve practice 2, 7, 12, 19.
3. **Day 3:** Work through chip sizing in §5.4; solve practice 3, 8, 11, 13, 17.
4. **About day 7:** Rebuild both interleaving schemes in §5.5; solve practice 5, 10, 14, 18.
5. **About day 14:** Complete both exam problems without notes; explain each answer aloud.
6. **Before the exam:** Tick only skills you can demonstrate cold; revisit sections named in missed answers.

When stuck: write the address width, addressable unit, row width, and byte order before calculating. Check units at the end.

Made by Phanuphat Anongjitaporn (Focus)

How the chapter fits together

Memory properties motivate a hierarchy. Addresses select locations, byte order explains multi-byte layout, chips build capacity and width, and interleaving distributes requests.

Measure · capacity and access

- **Memory hierarchy** §5.1

  speed · cost · capacity · volatility

  Rests on: identifying what each memory level trades away.

- **Addressing and capacity** §5.2

  locations = $2^{\text{address bits}}$

  Rests on: the addressable unit and memory row width.

Define · interpret layout

- **Byte ordering** §5.3

  MSB first / LSB first

  Rests on: addresses identifying successive bytes of one value.

- **Chips and modules** §5.4

  rows × width

  Rests on: multiplying row count by bits per row.

Derive · select and distribute

- **Interleaving** §5.5

  address = module bits + row bits

  Rests on: separating module selection from row selection.

Question	Method
How much memory?	Count addresses or multiply rows by width.
What byte appears first?	Identify endian convention, then order bytes.
Where does this address go?	Split module-selection and row bits.

Made by Thanate Amornjiraporn (Focus)

5.1 Memory hierarchy and properties

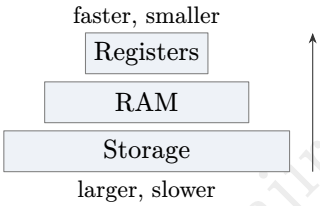
Hierarchy balances speed, capacity, cost, volatility, and flexibility. Registers are fastest and smallest; storage devices are largest and slowest.

> **In plain words:** memory is a ladder of trade-offs: the faster a memory is, the smaller and more expensive it is. Registers inside the processor are fastest; RAM is larger and slower; disks and SSDs are largest and slowest but keep their contents without power. SRAM and DRAM are two ways to build RAM: SRAM is faster, DRAM denser and cheaper.

Memory hierarchy

$$\text{fastest/smallest} \rightarrow \text{registers} \rightarrow \text{RAM} \rightarrow \text{storage} \rightarrow \text{largest/slowest}$$

Read it as: "from fastest and smallest to largest and slowest: registers, then RAM, then storage".
This is a qualitative ordering, not a universal timing specification.



The memory hierarchy. Each level down holds more and costs less per bit, but takes longer to reach.

Memory holds data, instructions, and addresses as binary values. Contents change only when overwritten. Registers are fixed in the CPU; RAM is expandable main memory; disks and other devices provide expandable storage. The lecture's comparison is qualitative: registers are expensive, small, fast, volatile; RAM is intermediate; disks are cheap, large, slower, and non-volatile.

Made by Thanadol Amornjiraporn

SRAM and DRAM

Static RAM (SRAM) is fast, less dense because each bit needs more transistors, and used mainly for cache; the slides also say it consumes less power. Dynamic RAM (DRAM) stores a bit in a transistor and capacitor. The capacitor leaks its charge, so DRAM must be refreshed periodically. It is cheaper but slower than SRAM and is used mainly for main memory.

Type	Cell and behaviour	Main use in the lecture
SRAM	More transistors per bit; faster, less dense, lower power	Cache
DRAM	Transistor and capacitor; needs periodic refresh	Main memory

Worked example

A program, its data, and pointers to both can all be represented as binary values.

1. CPU instructions, data values, and addresses all use binary storage. §5.1

Trap: state the addressable unit, width, and byte order before calculating. Do not treat a qualitative lecture comparison as a fixed hardware specification.

Near miss: DRAM holds data for as long as the computer runs, so it can seem permanent, but it is volatile: it needs power and constant refreshing, and switching off loses its contents, unlike flash storage.

Recall check · answer without looking

1. Why can a computer use a small amount of fast memory together with much larger, slower storage?
2. Name three things memory can store.
3. Which RAM type is used mainly for cache, and which for main memory?
4. Why does DRAM need refreshing, and how does its speed compare with SRAM?

Answers: see "5.1 Memory hierarchy and properties" under Answers at the end.

Made by

5.2 RAM cells, addressing, and capacity

A memory needs a way to identify locations and a unit that each address selects.

> **In plain words:** think of memory as a table of numbered rows. The **address** is the row number, and each row holds a fixed number of bits, the addressable unit (often one byte). With k address bits you can number 2^k rows, so the capacity is the number of rows times the bits per row.

Memory size

$$R \text{ rows} \times W \text{ bits per row} = R \times W \text{ bits}$$

Read it as: "R rows times W bits per row is R × W bits".
Address count depends on address width and the addressable unit.



16 address bits

16 address bits number $2^{16} = 65\,536$ rows: each extra bit doubles the count.

RAM can be modelled as R rows of W bits: total capacity is R×W bits. Byte-addressable memory assigns one address to each byte; word-addressable memory assigns one address to each processor word. For n address bits there are 2^n distinct addresses. If each address selects one byte, capacity is 2^n bytes. The slides use binary capacity units: 1 KB=2^{10} bytes, 1 MB=2^{20}, 1 GB=2^{30}, 1 TB=2^{40}.

Worked example
The lecture's 1GB example has 256M rows × 32 bits = 8,192M bits = 1,024M bytes = 1 GiB (binary units).

1. 256M rows times 4 bytes per row equals 1,024M bytes. §5.2

Trap: state the addressable unit, width, and byte order before calculating. Do not treat a qualitative lecture comparison as a fixed hardware specification.

Near miss: a 16-bit address looks as if it should give 16 rows, but it gives $2^{16} = 65\,536$ rows: each extra address bit doubles the count.

Recall check · answer without looking

1. Why does byte addressing make the address increment by one byte?
2. A RAM has 256M rows, 32 bits per row. How many bytes is this?

Answers: see "5.2 RAM cells, addressing, and capacity" under Answers at the end.

5.3 Endianness and aligned access

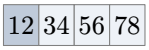
When a value spans multiple bytes, the byte order determines what appears at increasing addresses.

> **In plain words:** a number wider than one byte is spread over several addresses, so the order of its bytes needs a rule. **Big-endian** puts the most significant byte first, at the lowest address, the way we write numbers; **little-endian** puts the least significant byte first.

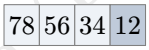
Byte order

$$\text{Big} - \text{endian} : \text{MSB at lowest address} \cdot \text{Little} - \text{endian} : \text{LSB at lowest address}$$

Read it as: "big-endian: most significant byte at the lowest address; little-endian: least significant byte at the lowest address".
The starting address is the address of the first byte of the item.



Big-endian: 0x12345678 with its most significant byte, 12, at the lowest address.



Little-endian: the same value with its least significant byte, 78, at the lowest address.

Big-endian stores the most significant byte first; little-endian stores the least significant byte first. Endianness changes the order of bytes in one multi-byte item, not the bit order inside each byte. A character string treated as individual ASCII bytes therefore retains its character order. Some architectures prohibit unaligned multi-byte accesses; the lecture identifies this as architecture-dependent.

> **Worked example**
> A 16-bit value A9A1 beginning at byte address 2 is aligned in the slide's example; F021 beginning at address 7 is unaligned.
>
> 1. Determine alignment from the access width and starting address; the lecture's examples contrast 2 with 7. §5.3

Trap: state the addressable unit, width, and byte order before calculating. Do not treat a qualitative lecture comparison as a fixed hardware specification.

Near miss: 0x12345678 stored little-endian from address 100 puts 0x78 at 100 and 0x12 at 103, so the bytes in address order read 78 56 34 12. That looks backwards, but it is the same value read with the other rule.

> **Recall check · answer without looking**
>
> 1. How would the bytes of 0x54BCFE30 appear from low to high addresses in each format?
> 2. Store the 16-bit value 0x0005 at address 24. What is at address 25 in each format?
>
> *Answers: see "5.3 Endianness and aligned access" under Answers at the end.*

5.4 Memory chips and modules

Chip organisation connects a chip's row count and data width to the capacity and width of a module.

> **In plain words:** a memory chip is a grid of rows, each a few bits wide. Larger memories combine chips: side by side to widen the data path, or added rows to hold more. A module's capacity is simply the total of its chips' bits.

Chip capacity

$$\text{rows} \times \text{bits per row} = \text{bits}$$
$$\text{divide by 8 for bytes}$$

Read it as: "rows times bits per row gives bits; divide by 8 for bytes".
Here M means 2^{20}, as in the lecture's capacity convention.

A chip described as 128M×8 stores 128M rows of 8 bits, or 128 MB using the lecture's binary convention. Chips can be combined in parallel to widen a module, and multiple banks/modules can build larger capacity. The slides show a 1GB (256M×32-bit) memory assembled from alternative combinations of chips and modules.

Modules	Chips in each module	Module capacity	Total
2	2 × 128M×16	512 MiB	1 GiB
2	4 × 128M×8	512 MiB	1 GiB
4	2 × 64M×16	256 MiB	1 GiB
4	4 × 64M×8	256 MiB	1 GiB

Worked example

Four 128M×8 chips provide 128M rows of 32 bits, or 512 MiB. Two such modules provide 1 GiB.

1. Width combines in parallel; module capacity is rows times total width. §5.4

Trap: state the addressable unit, width, and byte order before calculating. Do not treat a qualitative lecture comparison as a fixed hardware specification.

Near miss: a 1M × 4 chip looks like 4 MB, but it holds 1M rows of 4 bits, which is 4 Mbit = 512 KB, because a byte is 8 bits. Divide by 8 before writing bytes.

Recall check · answer without looking

1. How can four 128M×8 chips provide a 128M×32-bit module?
2. How many 64M×16 chips provide 256 MiB?

Answers: see "5.4 Memory chips and modules" under Answers at the end.

5.5 Memory interleaving

Interleaving assigns addresses across modules so independent modules can serve requests in parallel.

> **In plain words:** interleaving divides the addresses between several memory modules, and which address bits choose the module decides how. Take the module bits from the low end of the address (low-order interleaving) and consecutive addresses are dealt round the modules like cards round a table: while one module is still busy, the next address is already in another that can start straight away. Take them from the high end and each module holds one large block of consecutive addresses.

Address split for four modules

$$22\,\text{address bits} = 2\,\text{module bits} + 20\,\text{row bits}$$

Read it as: "22 address bits are 2 bits that choose the module plus 20 bits that choose the row inside it".
The 2 module bits select one of four modules; the 20 row bits select a row.



Low-order interleaving: the lowest 2 bits choose the module, so consecutive addresses go to different modules.



High-order interleaving: the top 2 bits choose the module, so each module holds one long run of addresses.

The lecture's example has 4M words, each 32 bits, built from four 1M×32 modules. A 22-bit word address splits into 2 module bits and 20 row bits. High-order interleaving places module bits at the high end: consecutive addresses stay in one module for a large region. Low-order interleaving places module bits at the low end: consecutive addresses cycle across modules, supporting parallel access to adjacent data.

Worked example

Low-order interleaving sends adjacent addresses to different modules; high-order interleaving groups a large contiguous range in one module.

1. The address-bit positions determine whether nearby addresses are distributed or grouped. §5.5

Trap: state the addressable unit, width, and byte order before calculating. Do not treat a qualitative lecture comparison as a fixed hardware specification.

Near miss: high-order interleaving also splits memory across four modules, but with the module bits at the top of the address, a run of consecutive addresses stays inside one module, so reading them gets no parallel speed-up. For that, the module bits must come from the low end: low-order interleaving.

Recall check · answer without looking

1. For low-order interleaving over four modules, where do addresses 8, 9, 10, and 11 go?
2. For high-order interleaving, which module contains addresses 0–3?

Answers: see "5.5 Memory interleaving" under Answers at the end.

Formula map and units

Relation	Unit / meaning
Capacity = rows × bits per row	bits; divide by 8 for bytes
n address bits → 2^n addresses	locations
Byte-addressable capacity = 2^n bytes	bytes
Address bits = module bits + row bits	bits
1 KB=2^{10} bytes; 1 MB=2^{20}; 1 GB=2^{30}; 1 TB=2^{40}	binary units used in the lecture

Check every answer: distinguish bits from bytes; state the addressable unit; confirm the number of module and row bits sums to the address width.

Derive it cold

1. Derive the byte capacity of an R×W-bit memory. §5.2
2. Derive the maximum capacity of n-bit byte addressing. §5.2
3. Derive module and row bit counts for four 1M×32 modules. §5.5
4. Derive the capacity of four parallel 128M×8 chips. §5.4

Explain it

1. Why are several memory types used instead of only the fastest one?
2. When does endianness affect the representation of a value?
3. Why do four 8-bit chips combine to make a 32-bit data path?
4. Why can low-order interleaving improve sequential access?
5. When does high-order interleaving fit the lecture's use case?
6. Why must a capacity calculation state whether it counts bits or bytes?

Answers: see "Explain it" under Answers at the end.

Made by Thanate Anornjiraporn (Focus

Mixed practice

Try all 21 problems before showing the answers. The order mixes hierarchy, addressing, byte layout, chip sizing, and interleaving; choose the method yourself.

1. A four-bit address selects how many locations?
2. A 32-bit value is 54 BC FE 30. Give its byte order at increasing addresses in little-endian memory.
3. A 16M×8 chip stores how many bytes?
4. Name two volatile memories from the lecture's hierarchy.
5. For low-order interleaving across four modules, which module receives word address 15?
6. A 64-bit byte-addressed machine has how many byte addresses and what capacity do they represent?
7. Write the bytes of 0x0005 at addresses 24 and 25 in big-endian order.
8. How many 128M×8 chips are needed for a 128M×32 module?
9. Which is generally faster in the lecture's hierarchy: registers or a hard disk?
10. For high-order interleaving with four modules, give module and row for address 2^{20}.
11. A memory has 2K rows of 16 bits. What is its capacity in bytes?
12. In big-endian storage, which byte is at the lowest address: MSB or LSB?
13. What is the capacity of a 64M×8 chip?
14. Why can neighbouring instruction fetches benefit from low-order interleaving?
15. Does removing power preserve RAM contents according to the slides?
16. What capacity is represented by 2^{30} bytes using the lecture's units?
17. What does 256M×32 mean in the 1GB memory example?
18. For low-order interleaving, where do addresses 0, 1, 2, 3 go?
19. Does an ASCII string's character order reverse in little-endian memory in the lecture example? Explain.
20. Which is larger: 1 TB or 1 GB, and by what factor under the lecture's definitions?
21. Which statements distinguish SRAM from DRAM in the lecture?

Answers: see "Mixed practice" under Answers at the end.

Made by Thanate Amornjirapong (Focus)

Exam-style problems

1 · Build and read a memory module · 20 minutes

A system has 4M words of 32 bits, built from four 1M×32 modules. It is byte-addressable and uses low-order interleaving.

1. How many address bits are needed for all words?
2. How many module bits and row bits are used?
3. Which module and row receive word addresses 0 through 7?
4. Explain why sequential access may benefit.

Answers: see "Exam-style problems" under Answers at the end.

2 · Endian bytes and capacity · 20 minutes

The 32-bit value 0x54BCFE30 begins at byte address 24 in a byte-addressable memory.

1. List the four bytes at addresses 24–27 in big-endian order.
2. List them in little-endian order.
3. How many bits are needed to address 4 GiB of byte-addressable memory?
4. State the capacity of a 256M×32-bit memory.

Answers: see "Exam-style problems" under Answers at the end.

Made by Thanate Amornjiraporn (Focus)

Self-check

Tick a skill only in a later session, at least one day after studying, when you can demonstrate it without notes.

- ☐ Compare registers, RAM, and storage. (4, 9, 15)
- ☐ Explain the SRAM and DRAM trade-offs and their main uses. (21)
- ☐ Compute locations from address width. (1, 6)
- ☐ Convert row width and count to bytes. (3, 11, 16)
- ☐ Lay out a multi-byte value in either endian order. (2, 7, 12, 19)
- ☐ Explain why byte order does not reorder single-byte ASCII. (19)
- ☐ Combine chips to increase module width. (8, 17)
- ☐ Calculate chip capacity. (3, 13, 17)
- ☐ Map addresses under low-order interleaving. (5, 14, 18)
- ☐ Split high-order module and row fields. (10)
- ☐ Identify volatile memory and power-loss effect. (4, 15)
- ☐ Use binary KB/MB/GB/TB definitions. (6, 16, 20)
- ☐ Rebuild capacity and address-split derivations.
- ☐ Complete both chained exam problems.

Made by Thanate Amornjiraporn (Focus)

Answers

Start from memory

1. Registers are fastest, followed by RAM, then storage. §5.1
2. One byte. §5.2
3. The least significant byte. §5.3
4. Two bits, because 2^2=4. §5.5
5. High-order groups contiguous address ranges by module; low-order distributes adjacent addresses across modules. §5.5

5.1 Memory hierarchy and properties

1. Fast memory serves immediate work while larger, slower devices retain more data at lower cost. The hierarchy balances competing properties.
2. Data, CPU instructions, and memory addresses.
3. SRAM is used mainly for cache; DRAM is used mainly for main memory.
4. The DRAM capacitor leaks charge, so it must be refreshed. DRAM is cheaper but slower than SRAM.

5.2 RAM cells, addressing, and capacity

1. Each distinct address selects one byte, so consecutive addresses select consecutive bytes. A word may span several byte addresses.
2. 256M×32 bits = 256M×4 bytes = 1,024M bytes = 1 GiB using binary M.

5.3 Endianness and aligned access

1. Big-endian: 54 BC FE 30. Little-endian: 30 FE BC 54.
2. Big-endian: 05. Little-endian: 00.

5.4 Memory chips and modules

1. Place the four chips in parallel: their row addresses align and their widths add to 32 bits. Capacity is 128M×32 bits = 512 MiB.
2. Two chips make 64M×32 bits = 256 MiB.

5.5 Memory interleaving

1. Modules 0, 1, 2, and 3 respectively; each selects row 2.
2. Module 0, with rows 0–3.

Made by Thante Amorintapori (Focus)

Explain it

1. Fast memory is costly and limited in capacity, while slower storage is larger and cheaper. A hierarchy balances these properties. §5.1
2. When a value spans multiple bytes; the convention determines which byte is at the lowest address. §5.3
3. All chips use the same row address while each supplies a different 8-bit slice, giving 32 bits total. §5.4
4. Adjacent words map to different modules, which may serve their requests in parallel. §5.5
5. When independent units access different modules, so requests can proceed in parallel. §5.5
6. A bit count is eight times a byte count; omitting the unit can create an eightfold error. §5.2

Made by Thanate Amornjiraporn (Focus)

Mixed practice

1. $2^4 = 16$ locations. §5.2
2. 30 FE BC 54. The least significant byte is stored first. §5.3
3. 16M bytes (binary convention). §5.4
4. CPU registers and RAM. §5.1
5. Module 3, because the low two address bits select the module ($15 \bmod 4 = 3$). §5.5
6. 2^{64} byte addresses, corresponding to 2^{64} bytes = 16 EiB. This extrapolates the lecture's 32-bit example using the same rule. §5.2
7. Address 24: 00; address 25: 05. §5.3
8. Four chips in parallel; each contributes 8 of the 32 bits. §5.4
9. Registers; the slide gives under 1 ns for registers and 5–10 ms for a hard disk. These are illustrative lecture ranges. §5.1
10. Module 1, row 0: the top two of the 22 bits select module 1 and the remaining 20 bits are zero. §5.5
11. 2K×16÷8 = 4K bytes = 4 KiB. §5.2
12. The most significant byte (MSB). §5.3
13. 64M×8÷8 = 64 MiB. §5.4
14. Consecutive addresses map to different modules, allowing parallel service if the memory system supports it. §5.5
15. No. RAM contents are volatile and are lost when power is removed. §5.1
16. 1 GB (binary: 1 GiB). §5.2
17. 256M rows, each 32 bits wide; total 1,024M bytes = 1 GiB. §5.4
18. Modules 0, 1, 2, 3, each at row 0. §5.5
19. No. Each character is one byte, so the byte sequence remains the same; endianness affects byte order within multi-byte values. §5.3
20. 1 TB is larger by $2^{10} = 1{,}024$, since $2^{40}/2^{30} = 2^{10}$. §5.2
21. SRAM is faster, less dense, lower power, and used mainly for cache. DRAM is cheaper, slower, uses a transistor-capacitor cell that needs refresh, and is used mainly for main memory. §5.1

Exam-style problems

1. 4M=2^{22} words, so 22 word-address bits are needed. Each word is 4 bytes, so byte addresses would require 24 bits if byte-addressing the entire space. §5.2
2. For the lecture's word-addressed interleave example, 2 bits select one of four modules and 20 bits select the row. §5.5
3. Using word addresses 0–7: modules cycle 0,1,2,3,0,1,2,3, with row 0 for the first four and row 1 for the next four. §5.5
4. Adjacent words are in different modules, so a capable memory system may overlap accesses. §5.5

Made by Phanate Amornjirapom (Focus)

Exam-style problems

1. 54, BC, FE, 30. The most significant byte is first. §5.3
2. 30, FE, BC, 54. The least significant byte is first. §5.3
3. 4 GiB=2^{32} bytes, so 32 address bits are required. §5.2
4. 256M×32÷8=1,024M bytes=1 GiB. §5.4

Made by Thanate Amornjiraporn (Focus)

Personal study guide based on the supplied "05_memory_organisation.pdf" (42 slides, lecture dated 29 April 2024), written in my own words; no lecture pages or figures are reproduced. Not an official FAU publication; check course details against the current notes. Made by **Thanate Amornjiraporn (Focus)**.