UNIVERSITY of WASHINGTON

CSS 422 Hardware and Computer Organization

Bus organization and memory design

Ver. 1

Professor: Wooyoung Kim

The slides are re-produced by the courtesy of Dr. Arnie Berger

Before we start

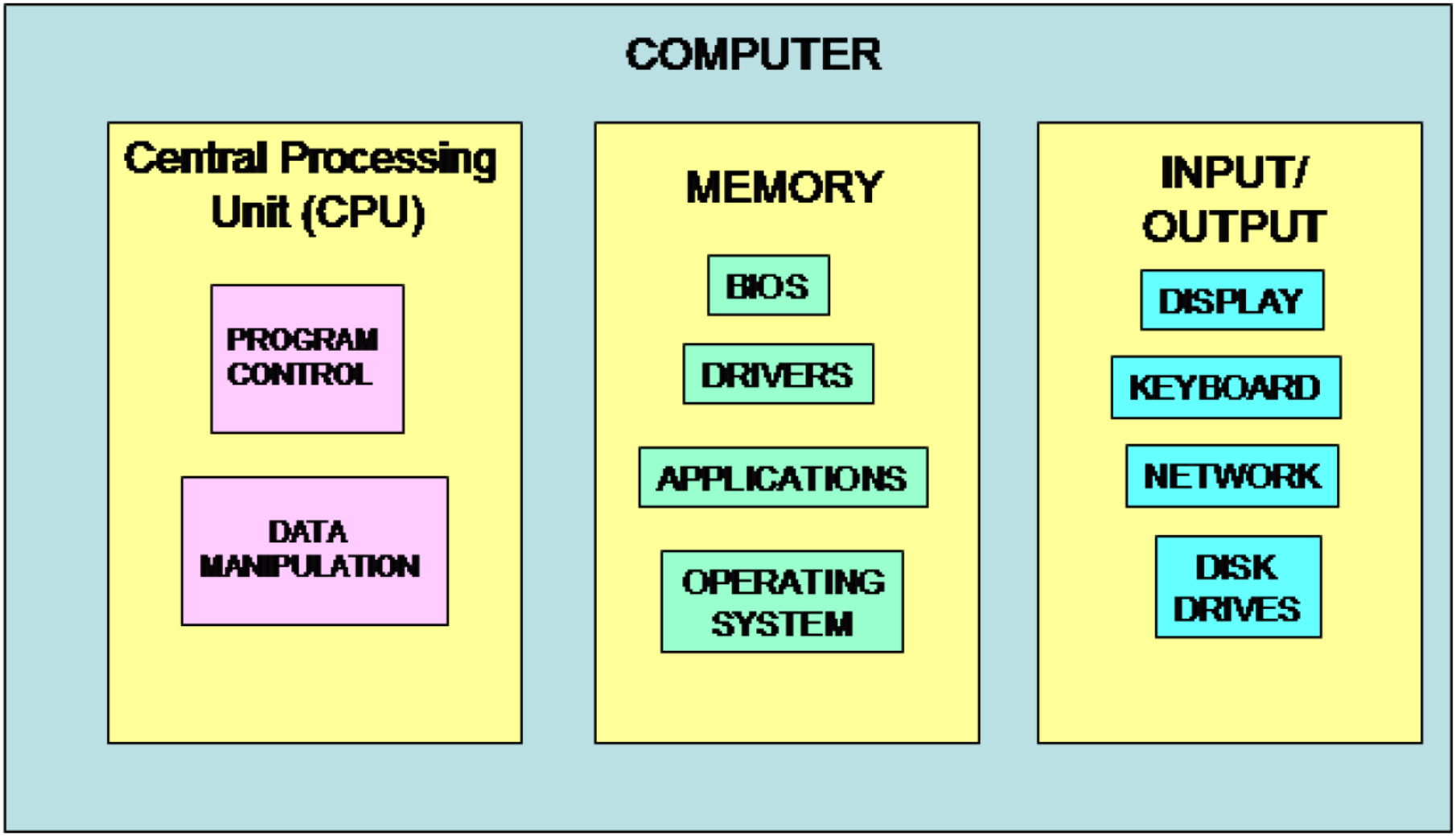
- Turn off your phone or at least sound off
- Do not open your laptop, or turn on your computer unless you are asked to do
  - Computers for class notes ONLY. Otherwise, I will consider you are distracting the class, and ask you to leave
  - I tend to ask the students with the laptop open, just to see if you are following my lecture
- No talk with your friends unless I ask to discuss
- If I think you are distracting this class, I will ask you to leave the class

Warning: You will see this slide in EVERY class, because you keep forgetting to do so

Topic

- Bus organization and memory design
  - Chapter 6 by Berger
  - Chapter 4 by Null

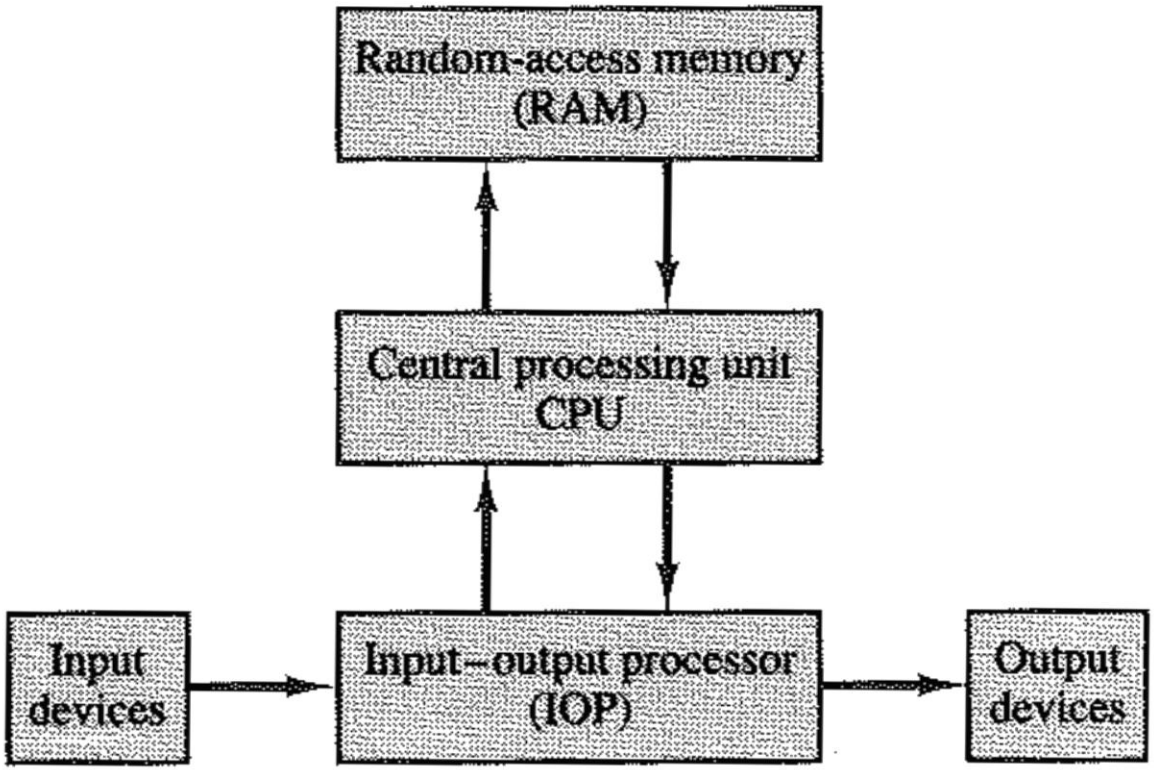
Review: Two views for Today's Computer



Hardware designer's point of view

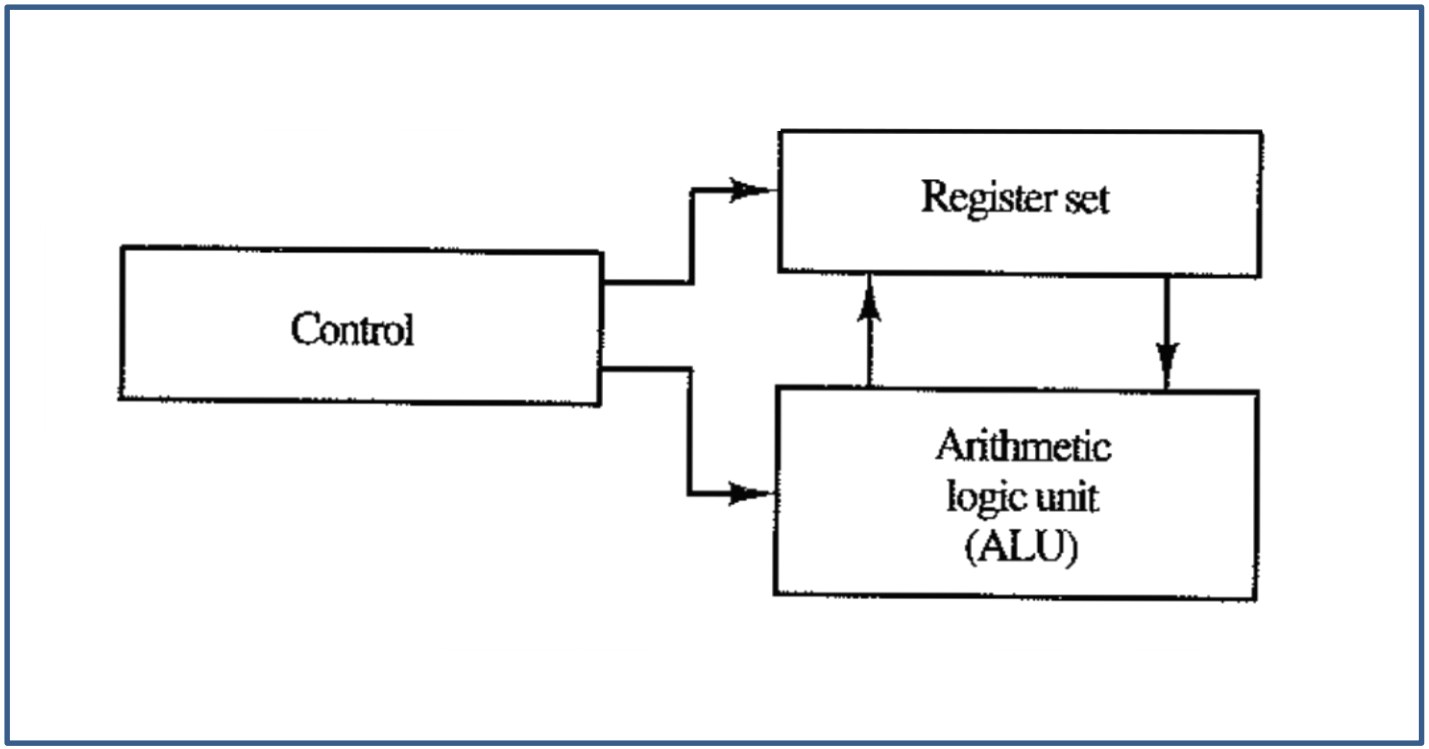
Diagram of Digital Computer

- The components are connected with ***external*** bus.



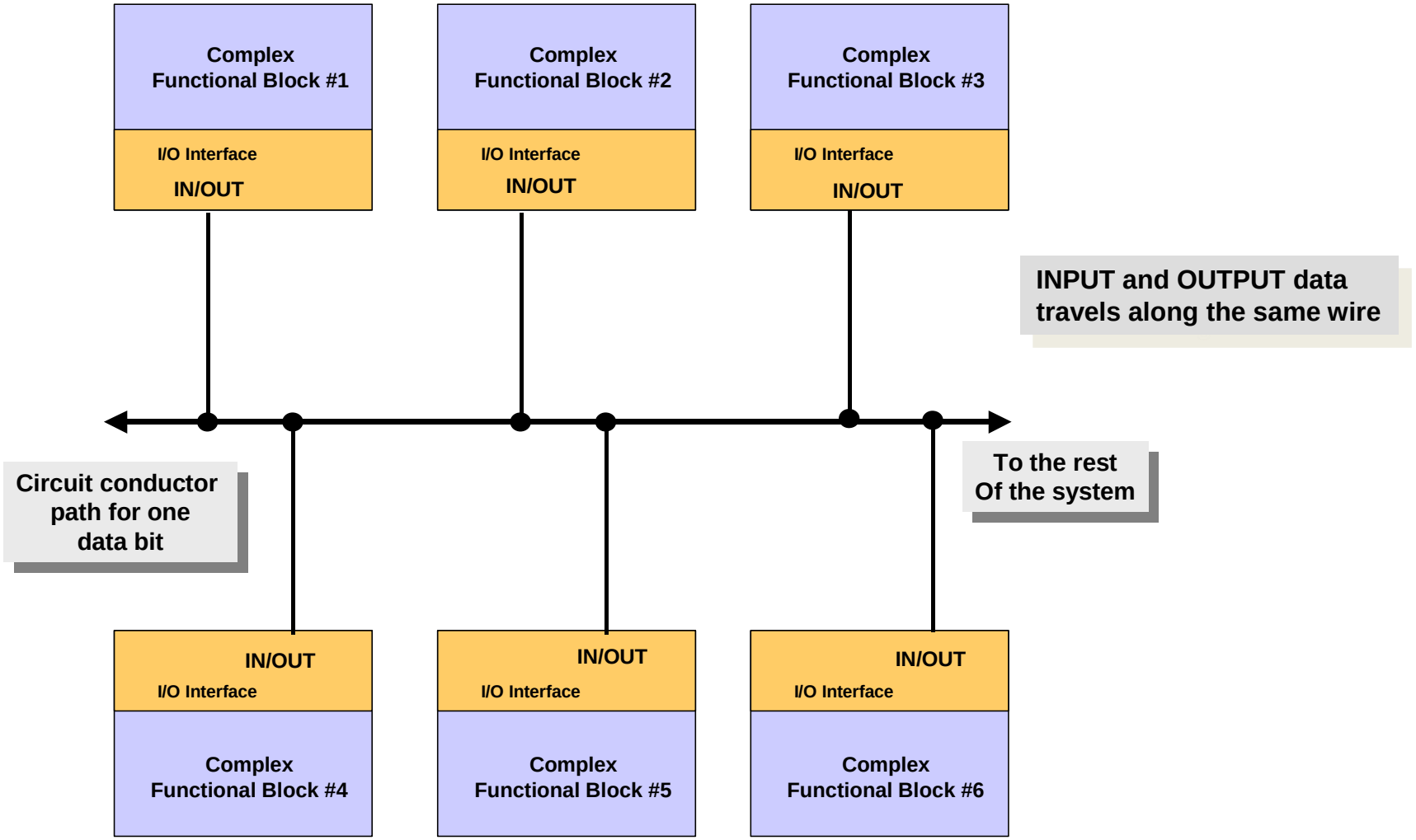
CPU

- The components in CPU is connected with ***internal*** bus.



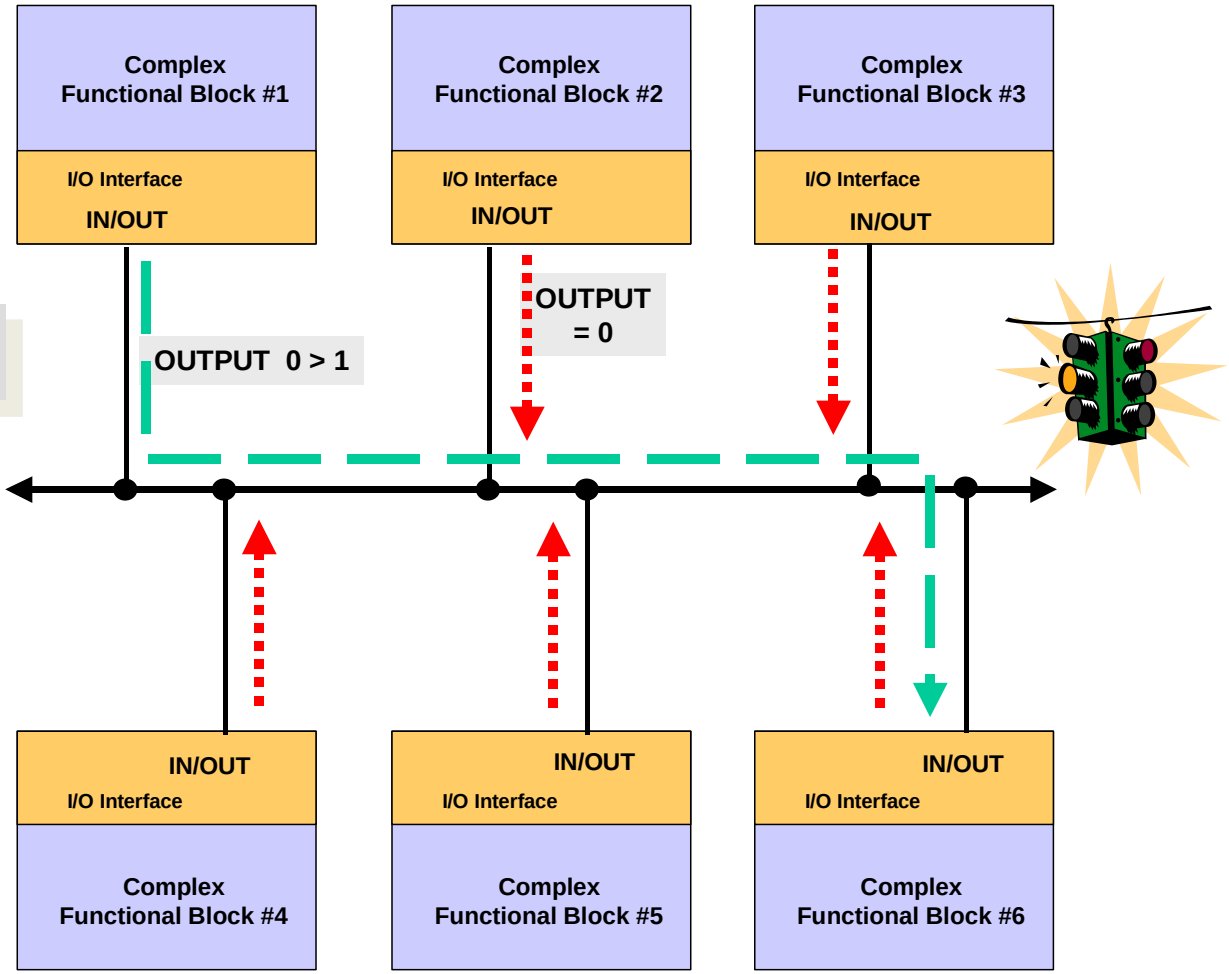
Major components of CPU

Bus organization



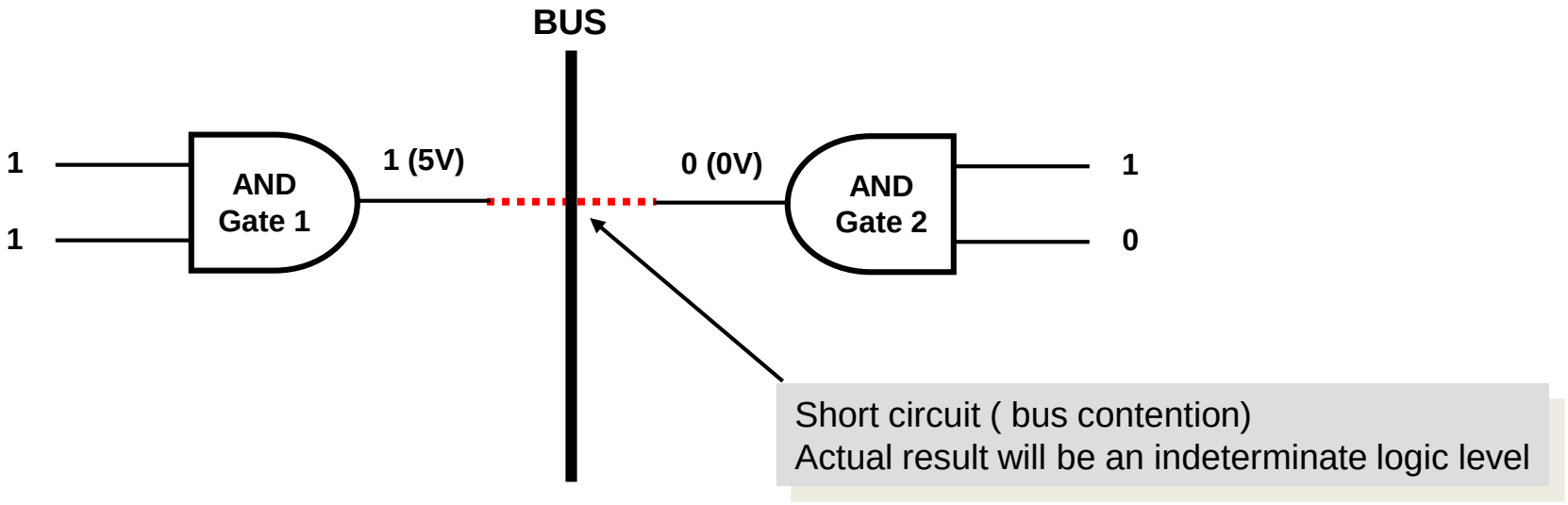
Bus organization

How do we manage the flow of information?



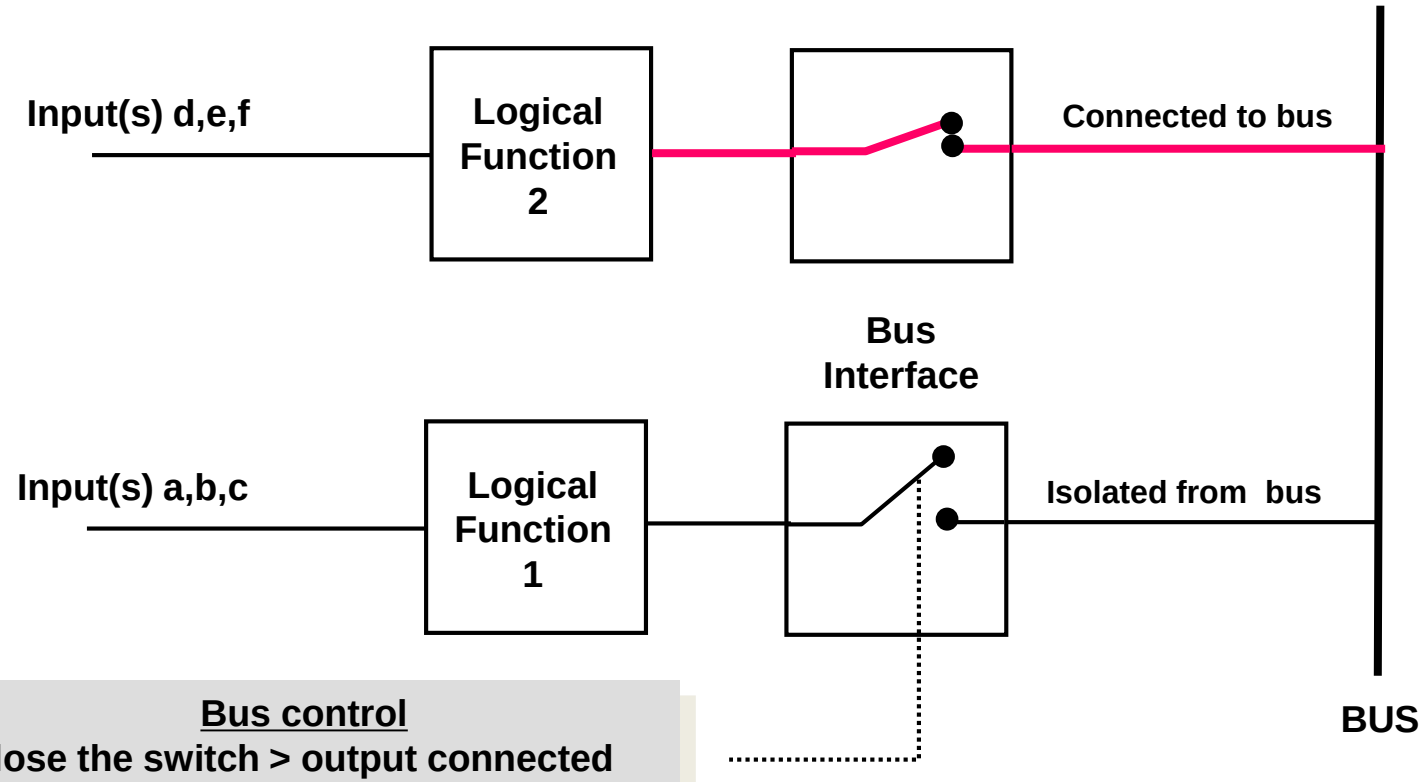
Bus organization

- Busses were invented in order to simplify the organization and flow of data within computer systems
  - Busses allow many devices to connect to the same data path
    - Allow for efficient exchange of data between devices
- Question: How do I connect outputs together and not get a short circuit?



Bus organization(2)

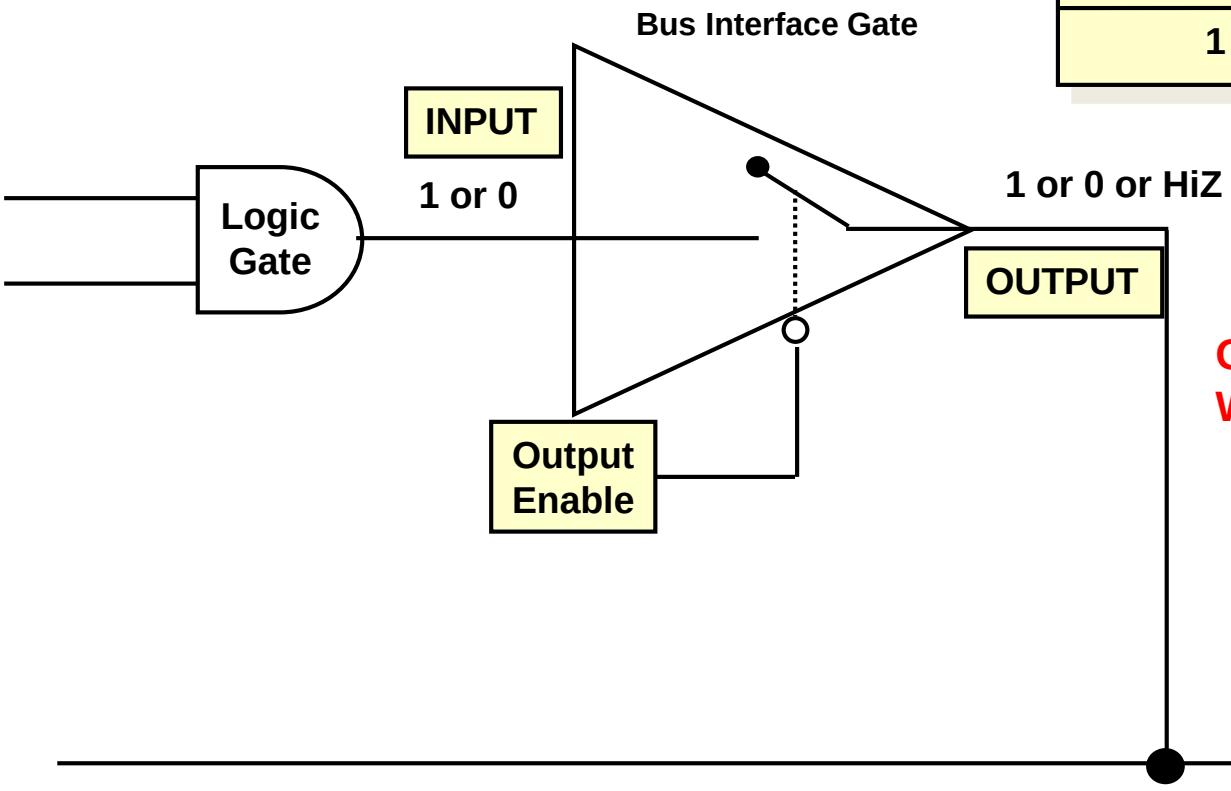
- Answer: All logic devices that connect to a bus are actually divided into two parts: Logic functional block and bus interface unit



Bus Structure: Tristate

Truth table for Bus Interface Gate →

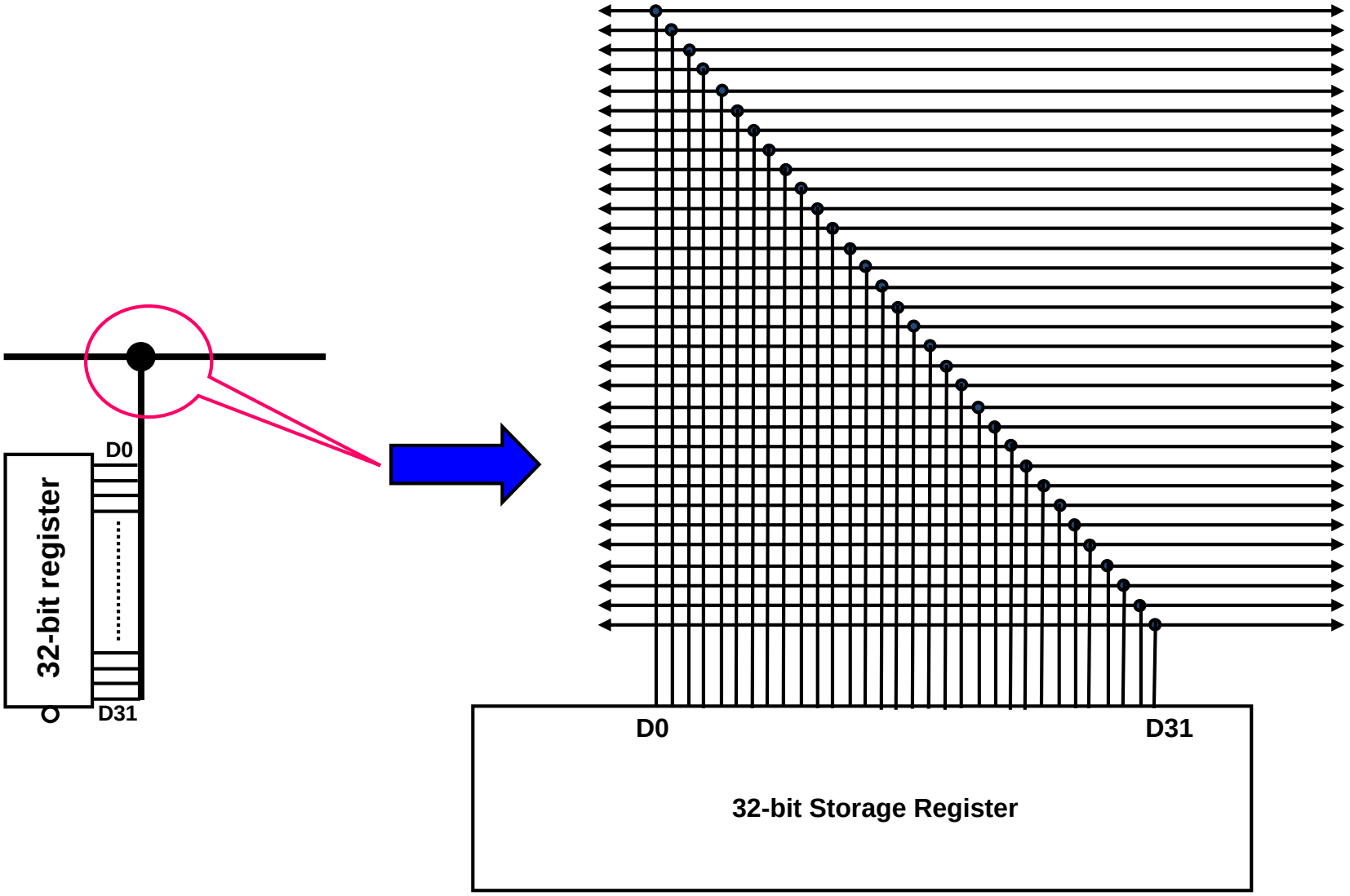
Output Enable	INPUT	OUTPUT
0	1	1
0	0	0
1	1	Hi Z
1	0	Hi Z



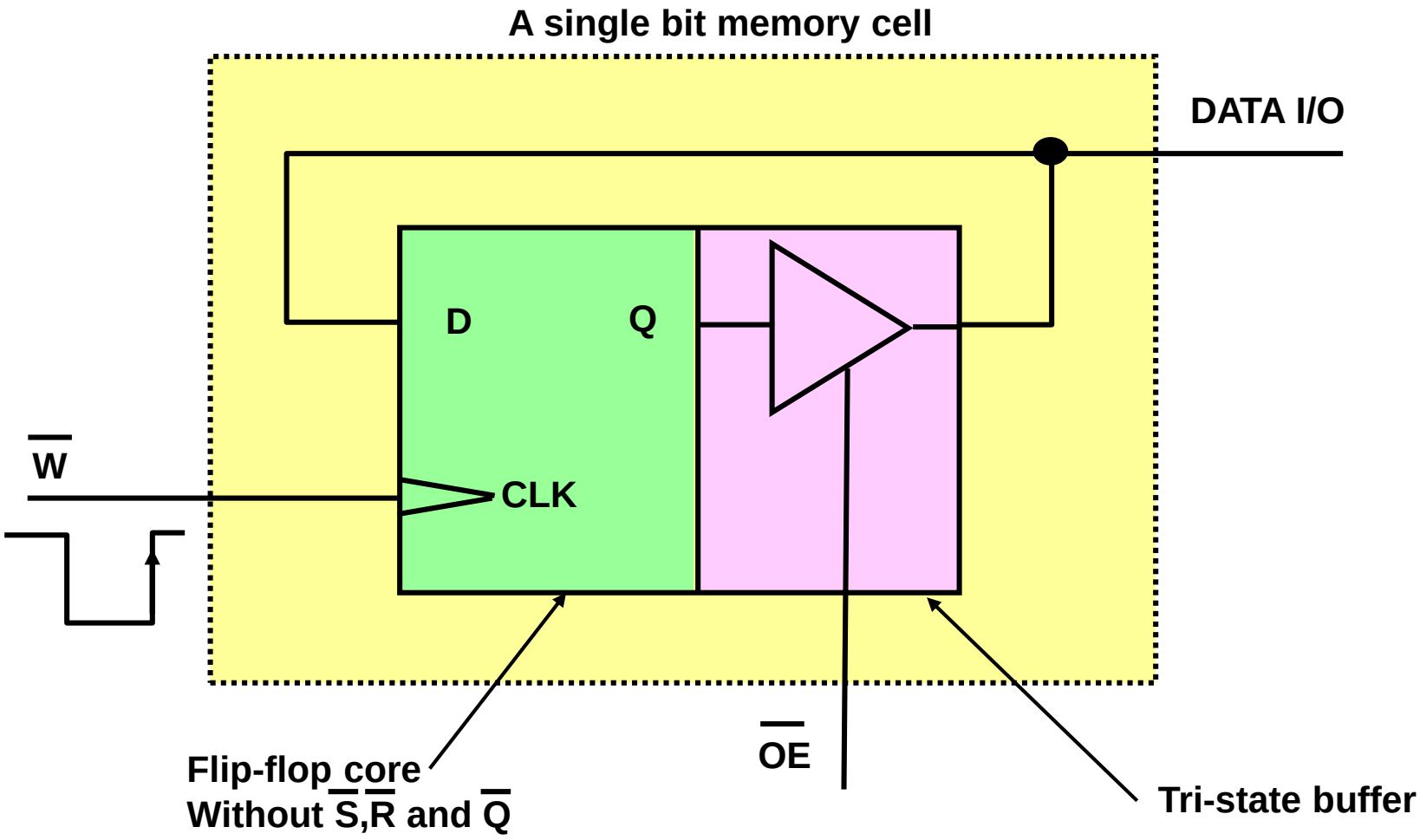
Hi Z: removing the gate from the circuit

Output Enable is 0 to activate. Why?

Bus organization



A “D” flip-flop memory cell



Data path (input/output) width

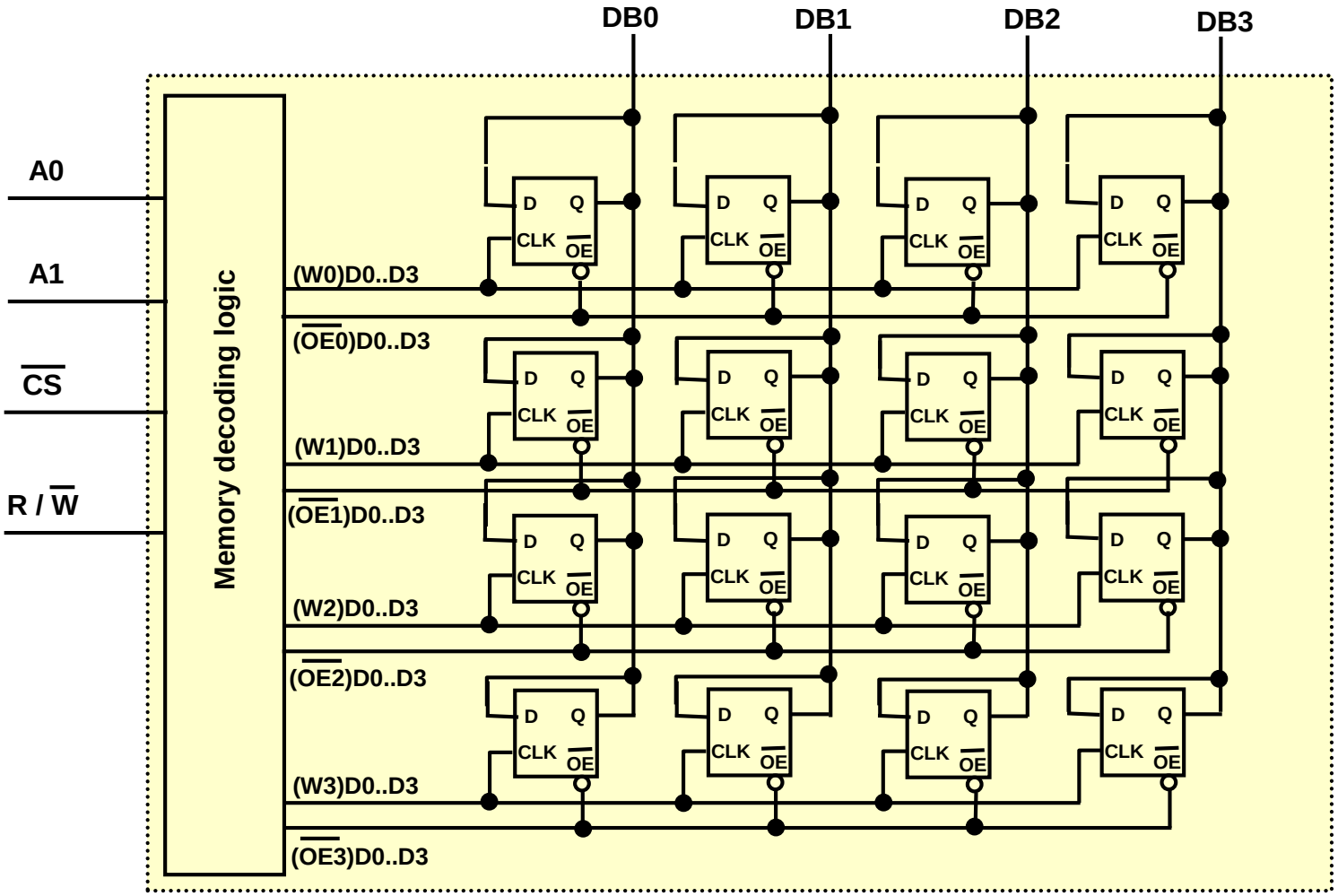
- Summary of where the various bus widths are most common:
  - 4,8 bits - Appliances, modems, simple applications
  - 16 bits - Industrial controllers, automotive
  - 32 bits - Telecomm, laser printers, high-performance apps
  - 64 bits - PC's, UNIX workstations, games
  - 128, 256 bits (VLIW) - Next generation

- Internal and external data paths may differ in size
  - Narrower memory is more economical
  - MC68000: 32-bit internal/16-bit external
  - 80C188: 16-bit internal/8-bit external

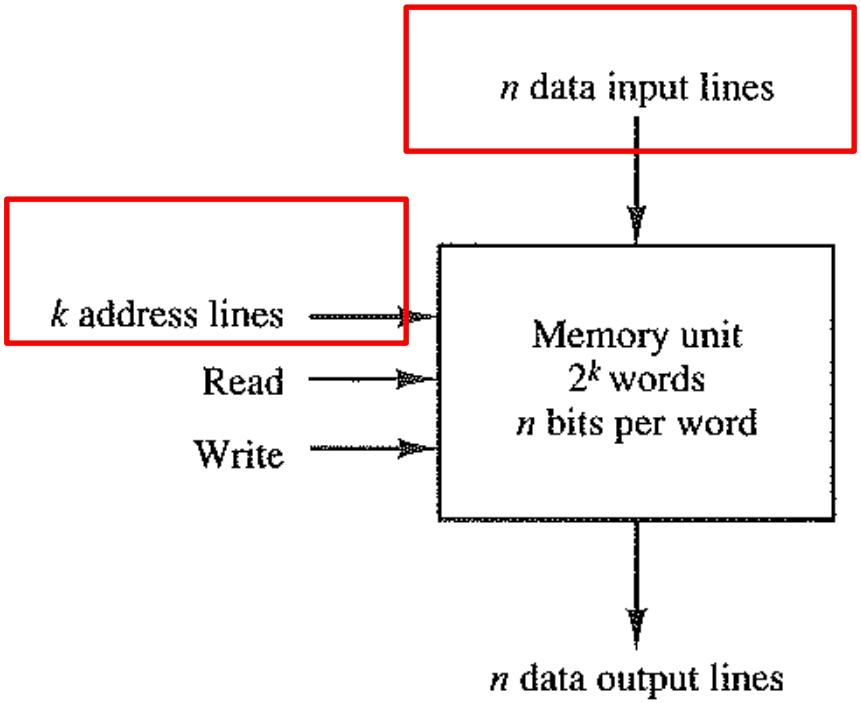
Addressable memory

- The amount of externally accessible memory is defined as the ***Address Space*** of the processor
- If we represent memory as a matrix, the size of the address space is same as the number of rows in a matrix
- Address space informs the number of data we can store, but the data size can be vary
- Memory capacity is determined by the data width as well as the addressable memory
- ***Address bits are connected to decoder:*** So k address bits can have 2^k address spaces

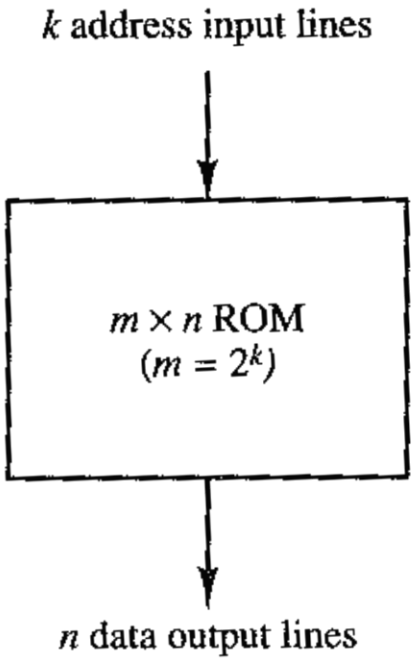
Memory organization



Memory Unit



RAM



ROM

Exercise

- Each memory chip can be expressed with its capacity (addressable spaces x data bits) in bits
- *m X n* capacity of memory indicates *m* addressable spaces for *n-bits* data

- How many address lines and input-output data lines are needed for each of the following capacity of memories?

1) 2K X 16

2) 64K X 8
3) 16M X 32
4) 4G X 64

Exercise

- Each memory chip can be expressed with its capacity (addressable spaces x data bits) in bits
- *m X n* capacity of memory indicates *m* addressable spaces for *n-bits* data

- How many address lines and input-output data lines are needed for each of the followings capacity of memories?

1) 2K X 16

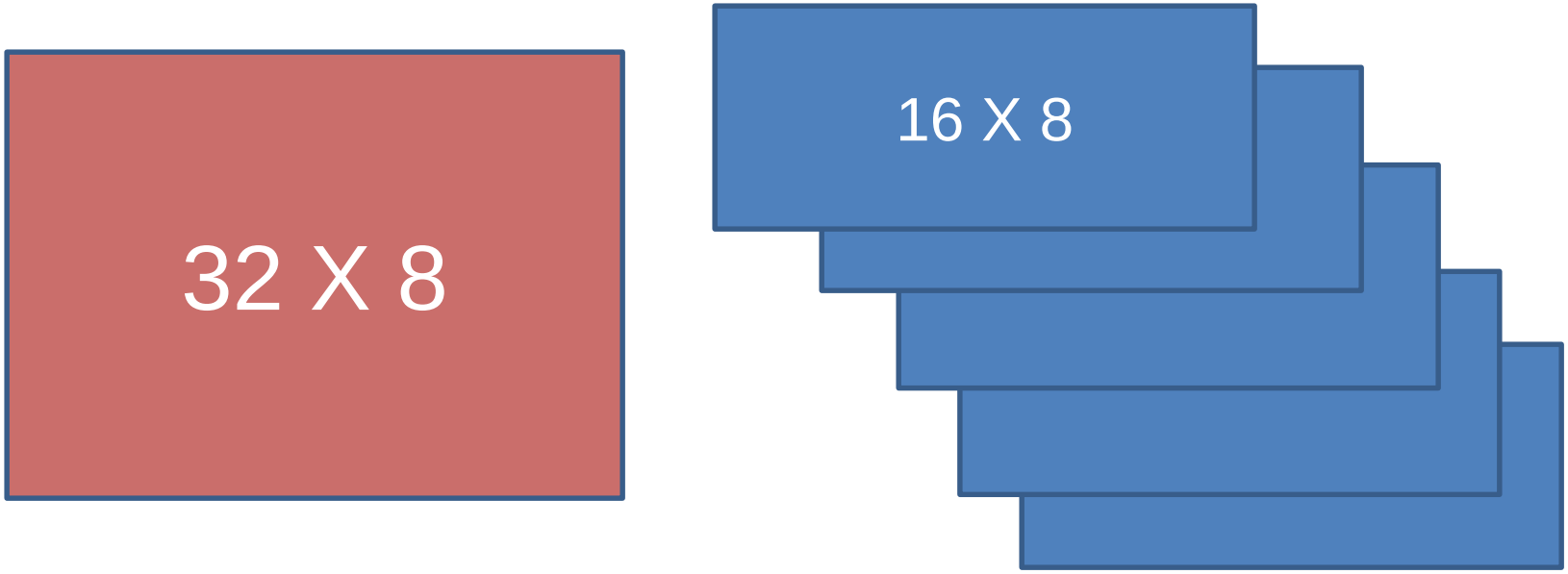
→ 2K = 2^{11}. Therefore it has 11 address lines, and 16 data lines

1) 64K X 8
2) 16M X 32
3) 4G X 64

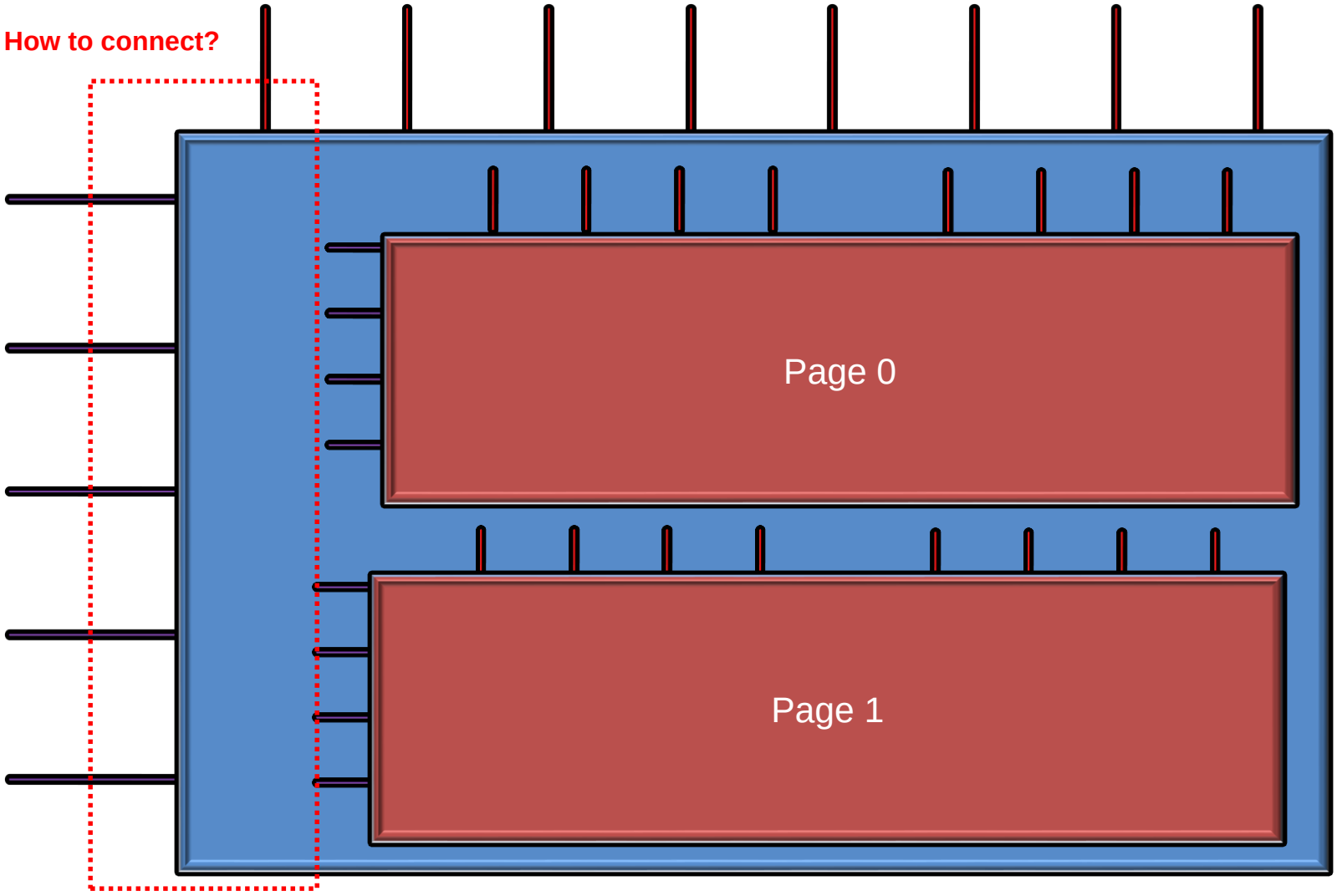
Memory extension

- Large memory chip can be constructed with a number of smaller size of memory chips
- Suppose you want to have a memory with the capacity of 32 X 8
- But, you have a number of 16 X 8 memory chips
- How to arrange the small chips to have a 32 X 8 memory?

How many pins each chip has?

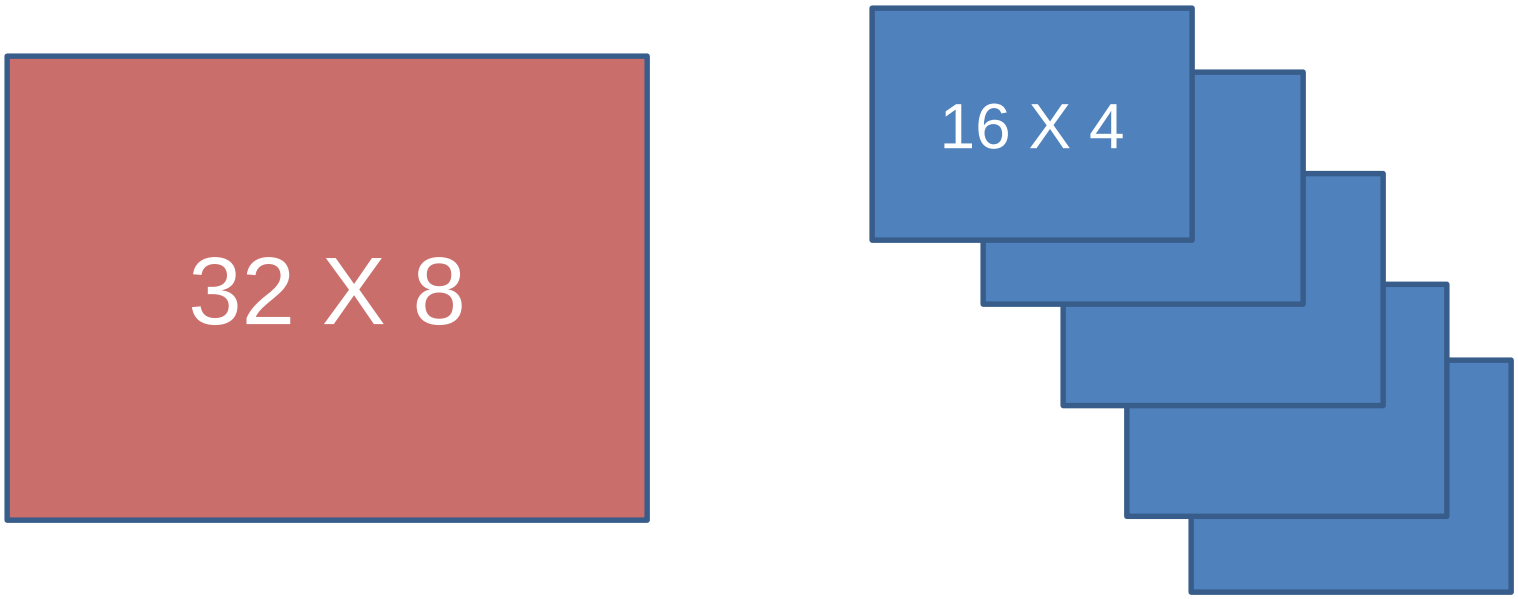


Increase address spaces



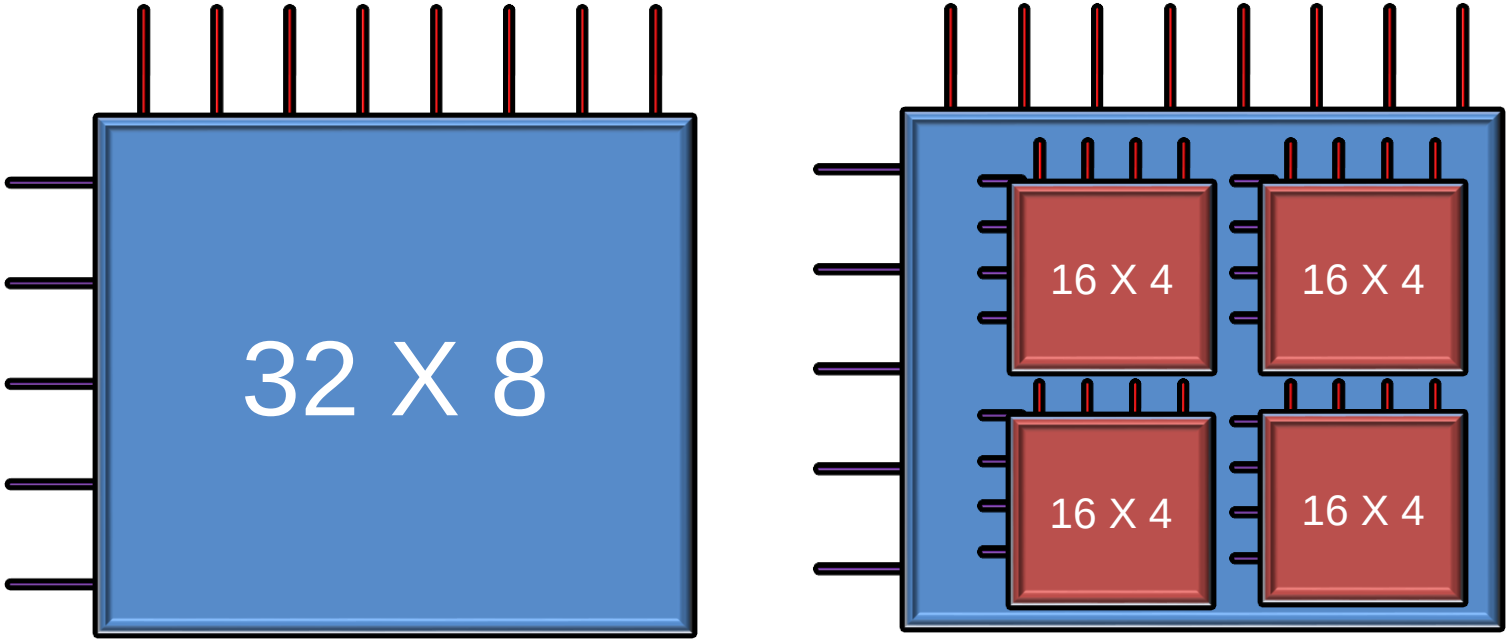
Memory extension (2)

- Suppose you want to have a memory with the capacity of 32 X 8 bits
- But, you have a number of 16 X 4 memory chips
- How to arrange the small chips to have a 32 X 8 memory?



Memory extension (3)

- Increase the address space vertically
- Increase the data paths horizontally

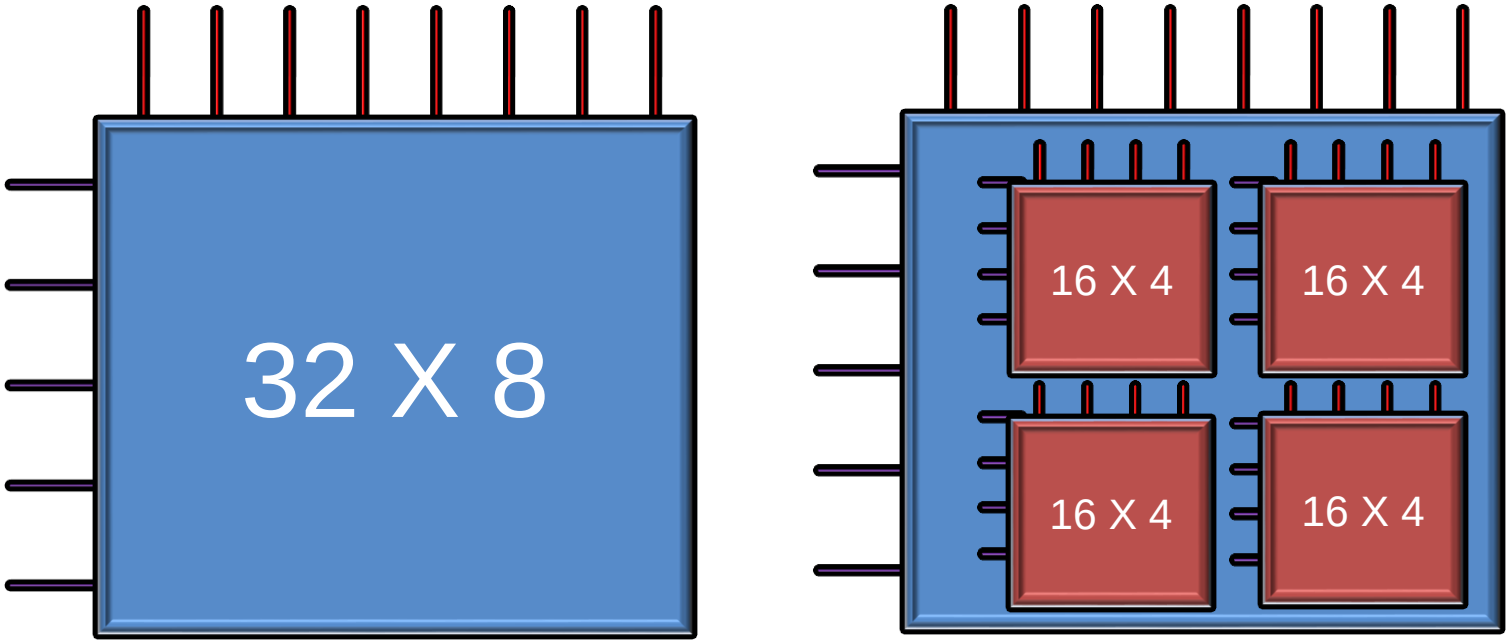


Exercise

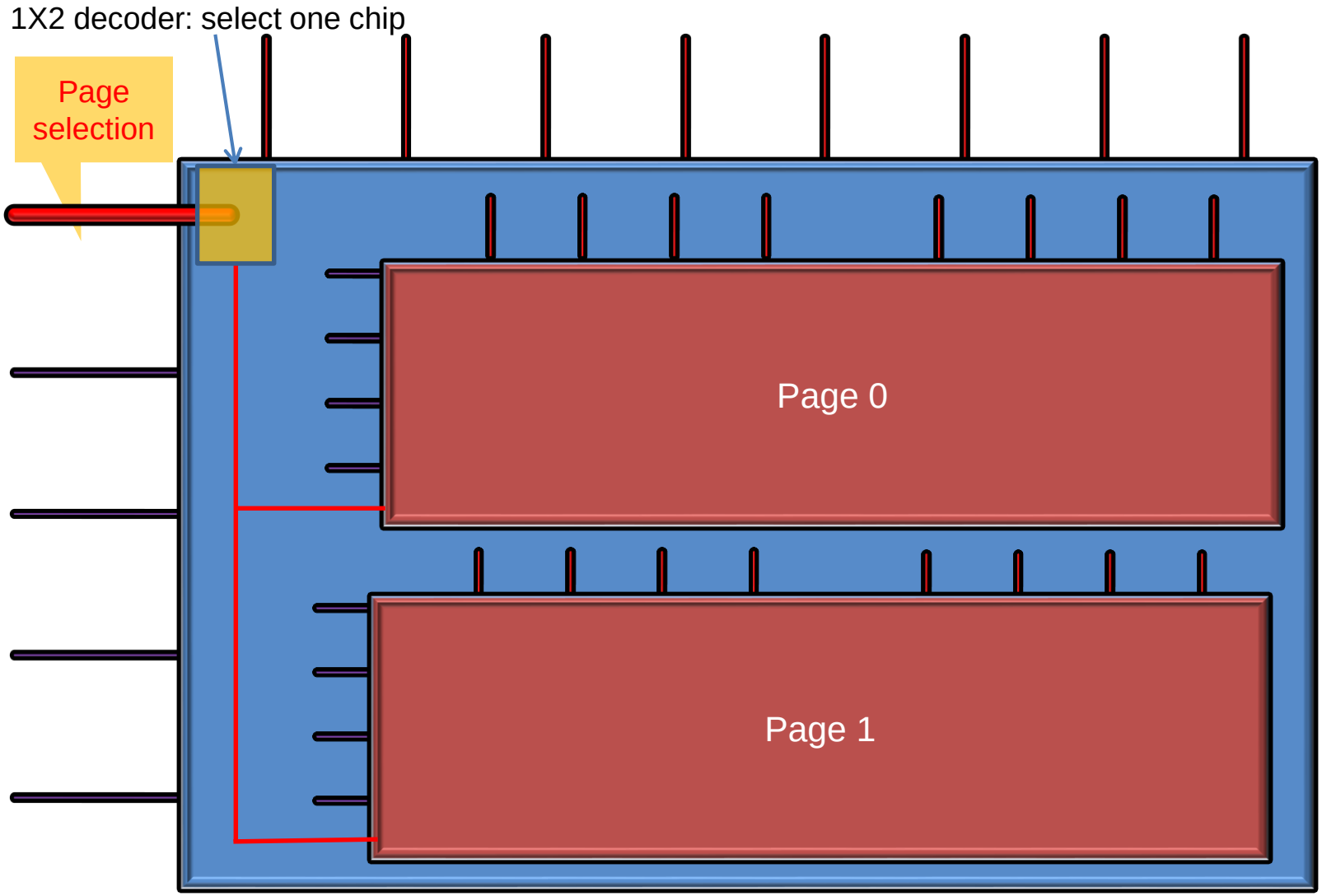
- Build a 4096 X 16 memory
- Use 128 X 8 memory chips.
  - How many 128 X 8 memory chips you should have?
  - How many address lines and data lines in the 128 X 8 memory chips?
  - How many address lines and data lines in the 4096 X 16 memory?

Memory extension (4)

- It is obvious how the two of 4 data bits can connect to 8 bits (How?)
- Question: How the two of 4 address lines can be 5 address lines?



Paging



Address space partitioning

- Build a 1GB memory (1G X 8 memory organization)
- Use 4 MB memory chips.
  - How many 4 MB memory chips are used?
  - How many pages you have?
  - How many page selection bits you need?
  - How many bits can be used for the page number?

Address space partitioning

- Build a 1GB memory (1G X 8 memory organization)
- Use 4 MB memory chips.
  - How many 4 MB memory chips are used?

  1GB / 4MB = 2^{30} /2^{22} = 2^{8} =256 chips
  - How many pages you have?

  256 chips = 256 pages

  - How many page selection bits you need?

  256 pages ➔ $\log_2 256 = \log_2 2^8 = 8$

  - How many bits can be used for the page number?

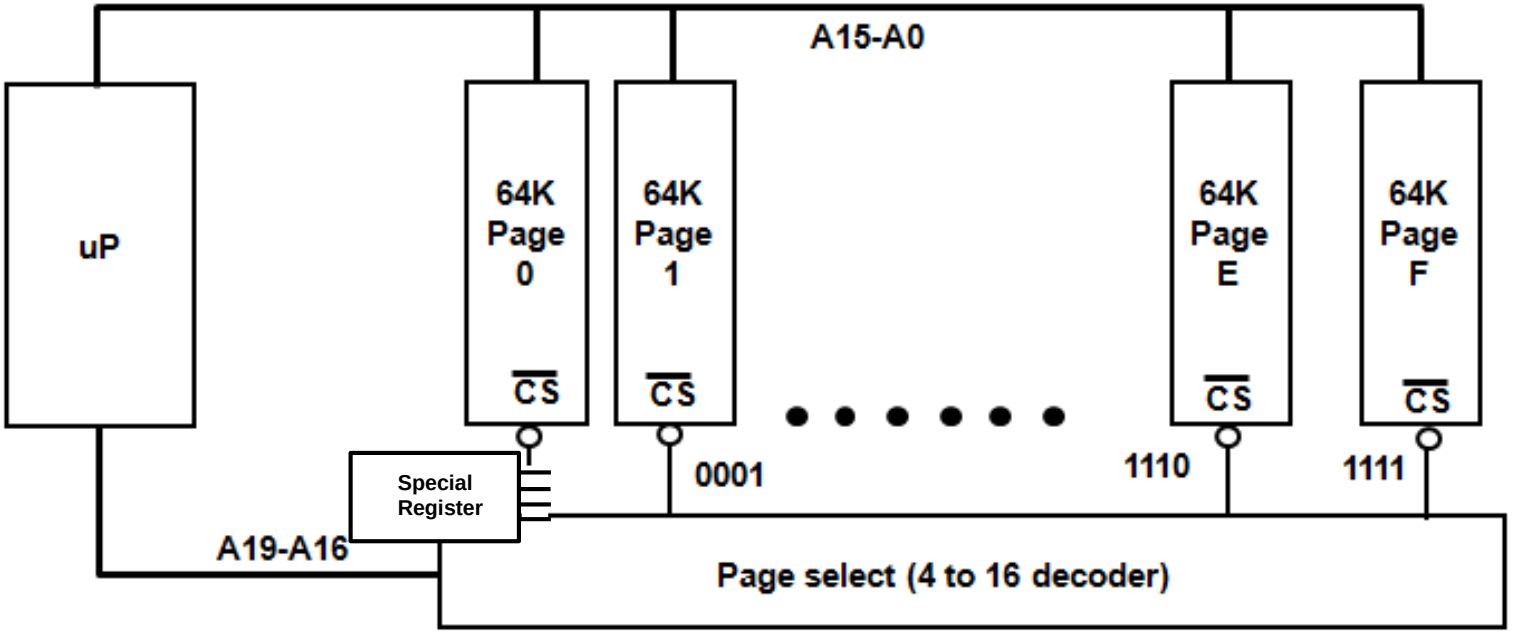
  Same as the previous one. 8

Exercise

- Build a 1MB memory (1M X 8 memory organization)
- Use 64 KB memory chips.
  - How many 64 KB memory chips are used?
  - How many pages you will have?
  - How many page selection bits you need?
  - How many bits can be used for the page number?

Paging

- Processors with smaller address spaces can still manipulate larger memory arrays with techniques such as ***Paging***
  - Special memory or I/O location used to swap in and out memory pages
- ***Hardware paging should not be confused with virtual addresses***
- In the figure below, writing a data value to the special register will cause a page swap to occur



Paging

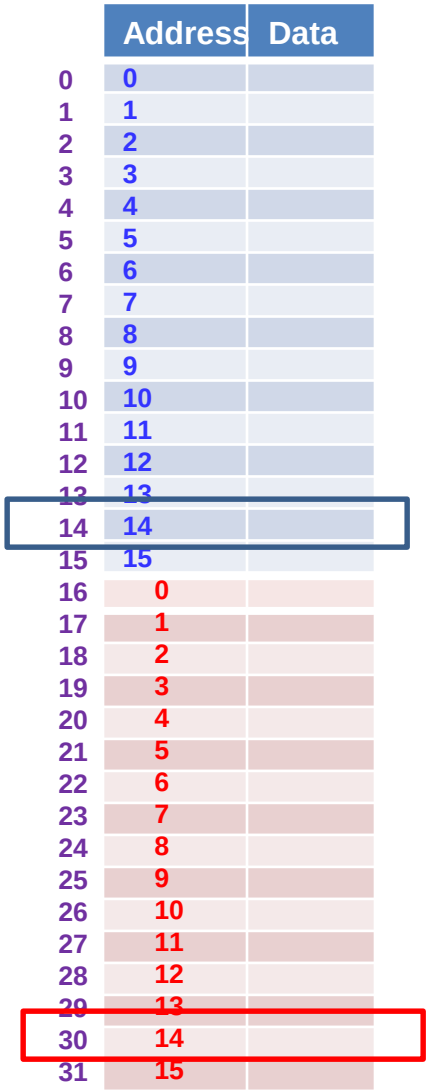
- Suppose a memory chip has 32 X 8 capacity. It can store up to 32 rows of 8 bits data. The memory chip has 2 pages, each of which has 16 addressable spaces
- How to address data at 14 and 30?
  - ***Method 1: Read an absolute address***

    Address 14 = 01110

    Address 30 = 11110
  - ***Method 2: Read with page number and offset, assuming that 0 to 15 at the 0 th page, the others at the next page.***

    Address 14 = 0 1110 (page 0, offset 1110)

    Address 30 = 1 1110 (page 1, offset 1110)



Address space partitioning

- Using 1 GB memories (30 bits) with a 4 MB chips (22 bits)
- 8 bits for page selection,22 bits for offset (capacity for each page)
- Page selector: (**8** bits), Offset (the # address lines for each chip/page): **22** bits

	Page Selector (8-bits. Binary)	Offset Address (22bits. Binary)	Comments	**Absolute Address (Hex)**
Page 0	**00 0000 00**	00 0000 0000 0000 0000 0000	First address: Page 0	**00000000**
	00 0000 00	11 1111 1111 1111 1111 1111	Last address: Page 0	
Page 1	**00 0000 01**	00 0000 0000 0000 0000 0000	First address: Page 1	
	00 0000 01	11 1111 1111 1111 1111 1111	Last address: Page 1	
Page 2		00 0000 0000 0000 0000 0000	First address: Page 2	
		11 1111 1111 1111 1111 1111	Last address: Page 2	
	. .	. .	. .	
Page 255		00 0000 0000 0000 0000 0000	First address: Page FF	
		11 1111 1111 1111 1111 1111	Last address: Page FF	

Address space partitioning

- Using 1 GB memories (30 bits) with a 4 MB chips (22 bits)
- 8 bits for page selection,22 bits for offset (capacity for each page)
- Page selector: (**8** bits), Offset (the # address lines for each chip/page): **22** bits

	Page Selector (8-bits. Binary)	Offset Address (22bits. Binary)	Comments	**Absolute Address (Hex)**
Page 0	**00 0000 00**	00 0000 0000 0000 0000 0000	First address: Page 0	**00000000**
	00 0000 00	11 1111 1111 1111 1111 1111	Last address: Page 0	**003FFFFF**
Page 1	**00 0000 01**	00 0000 0000 0000 0000 0000	First address: Page 1	**00400000**
	00 0000 01	11 1111 1111 1111 1111 1111	Last address: Page 1	**007FFFFF**
Page 2	**00 0000 10**	00 0000 0000 0000 0000 0000	First address: Page 2	**00800000**
	00 0000 10	11 1111 1111 1111 1111 1111	Last address: Page 2	**00BFFFFF**
	. **.**	. .	. .	**.** **.**
Page 255	**11 1111 11**	00 0000 0000 0000 0000 0000	First address: Page FF	**3FC00000**
	11 1111 11	11 1111 1111 1111 1111 1111	Last address: Page FF	**3FFFFFFF**

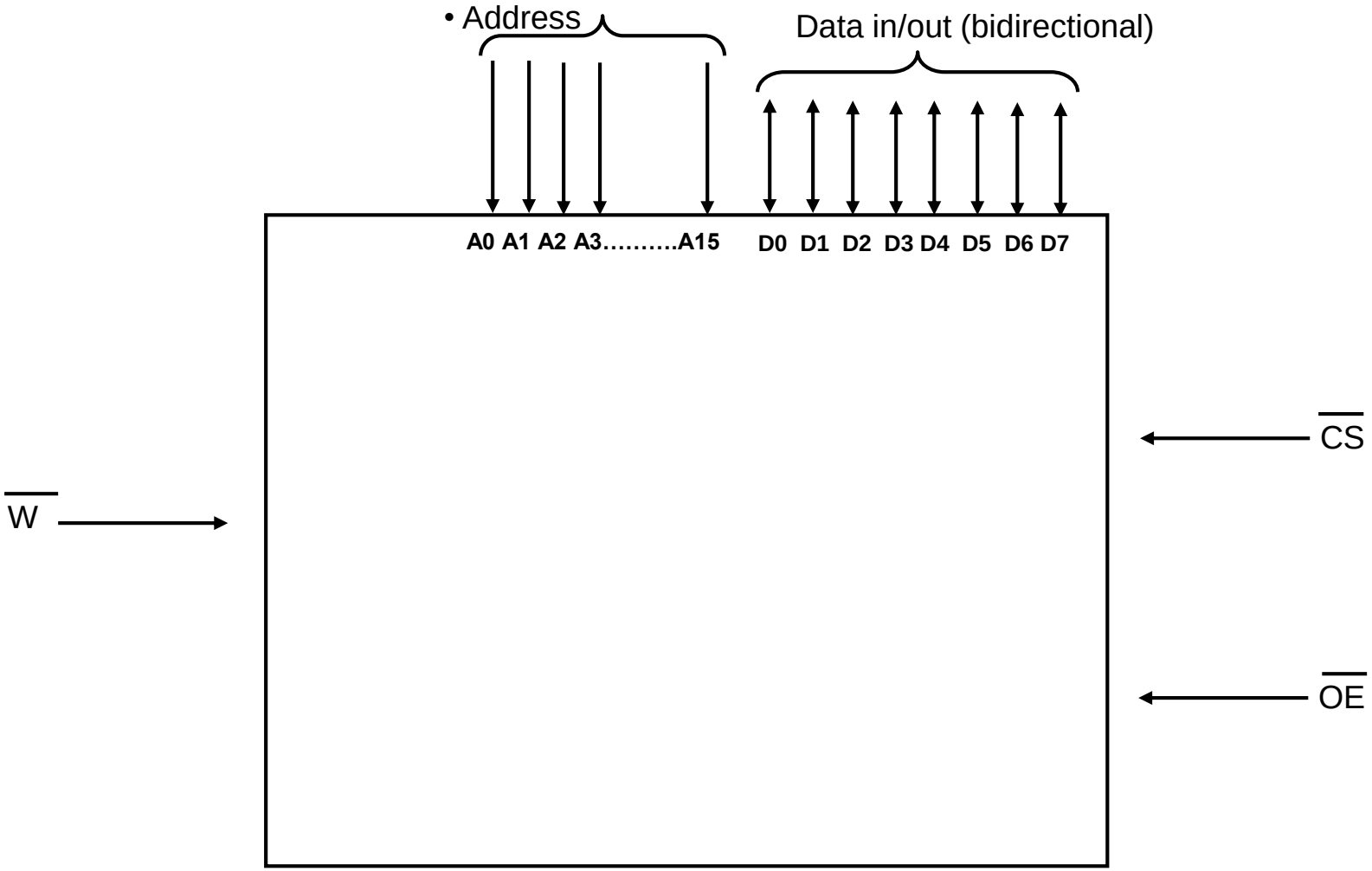
Exercise

- Using 1 KB memories with a 128 B chips
- ? bits for page selection. ? bits for offset (each page)

	Page Selector (?-bits. Binary)	Offset Address (? bits. Binary)	Comments	**Absolute Address (Hex)**
Page 0			First address: Page 0	
			Last address: Page 0	
Page 1			First address: Page 1	
			Last address: Page 1	
Page 2			First address: Page 2	
			Last address: Page 2	
Page 3			First address: Page 3	
			Last address: Page 3	
	…	…	…	…

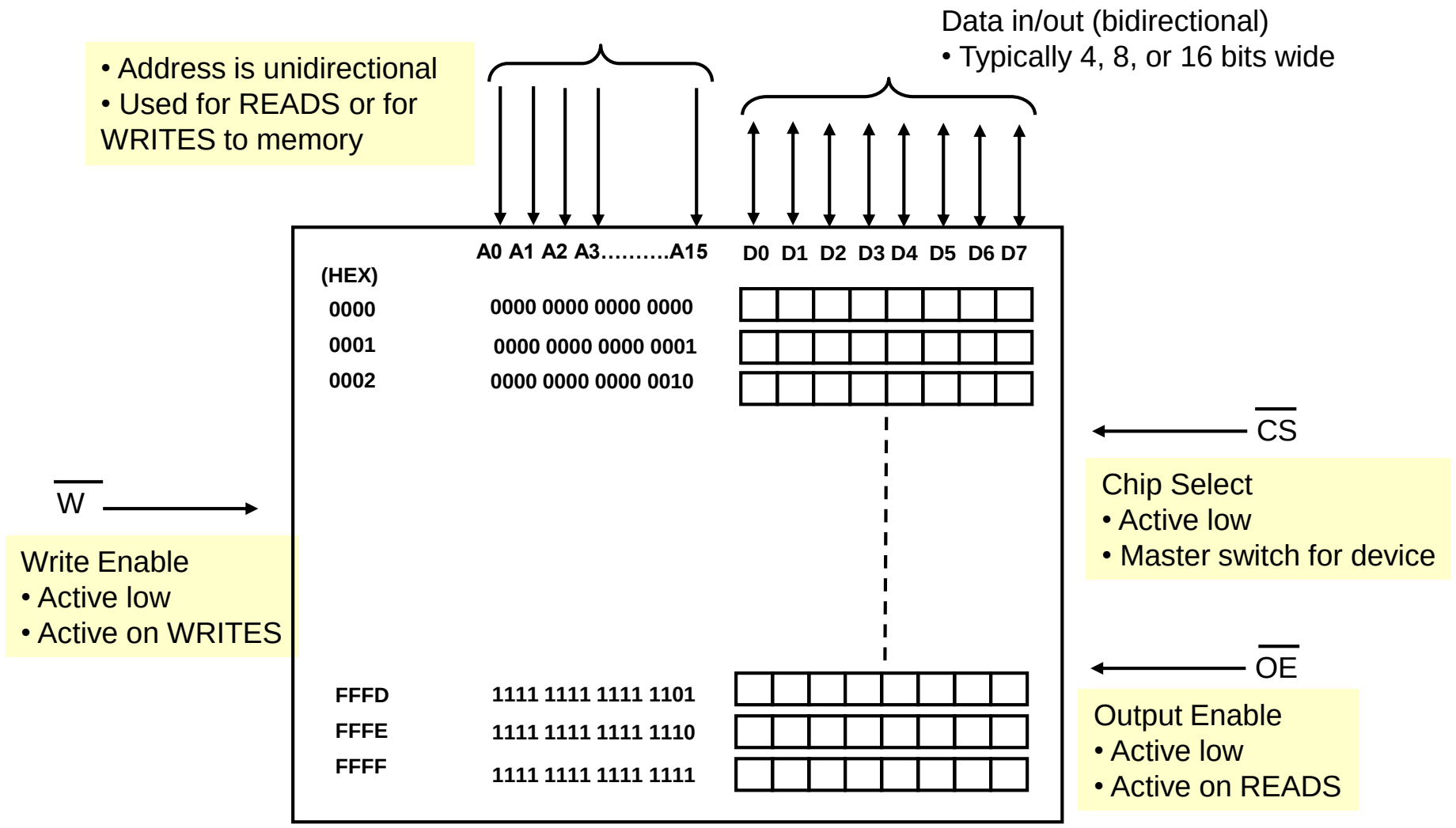
Characteristics of a memory device

Example: 512 Kbit device organized as 64K x 8



Characteristics of a memory device

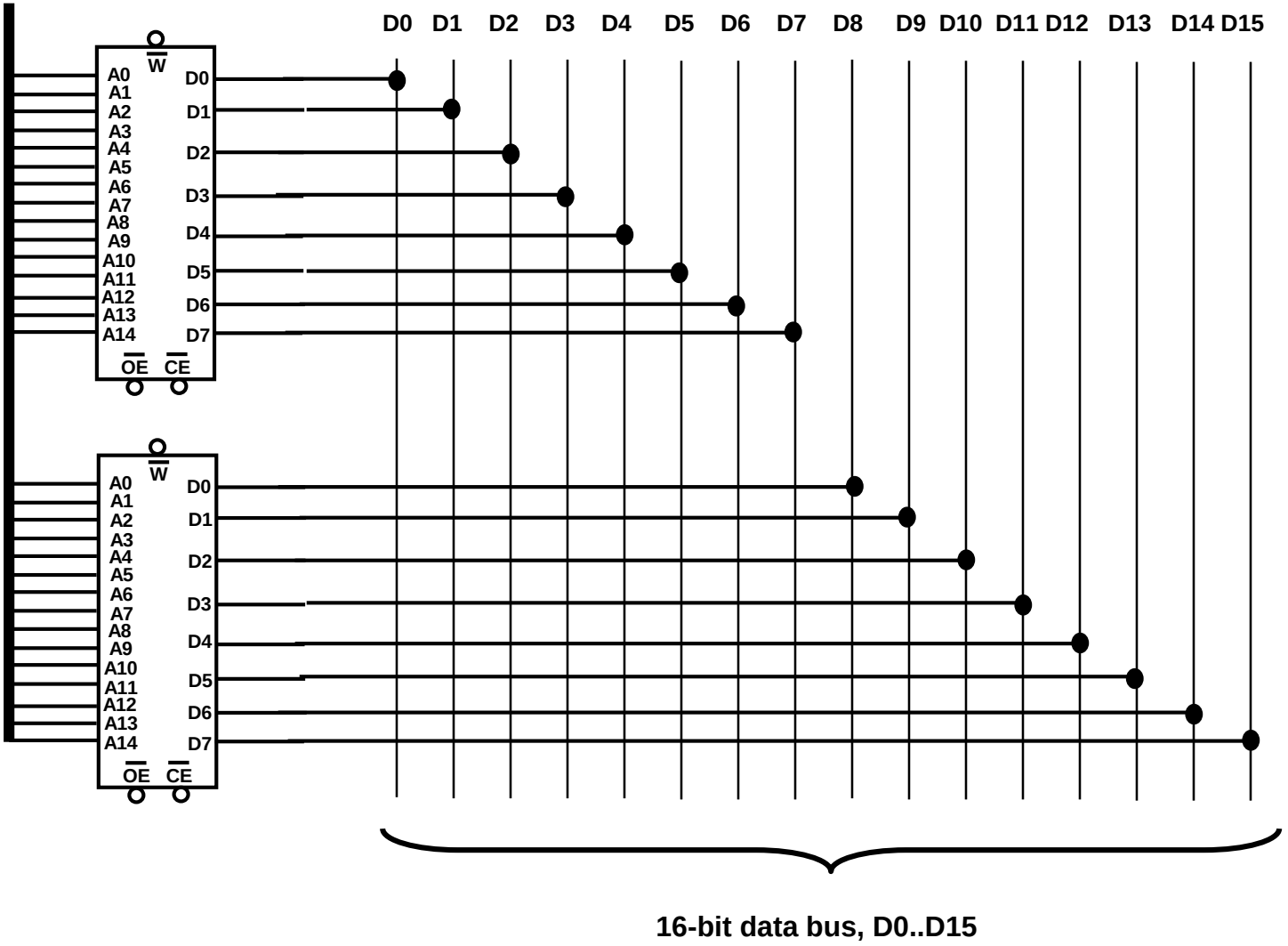
Example: 512 Kbit device organized as 64K x 8



Memory space notes

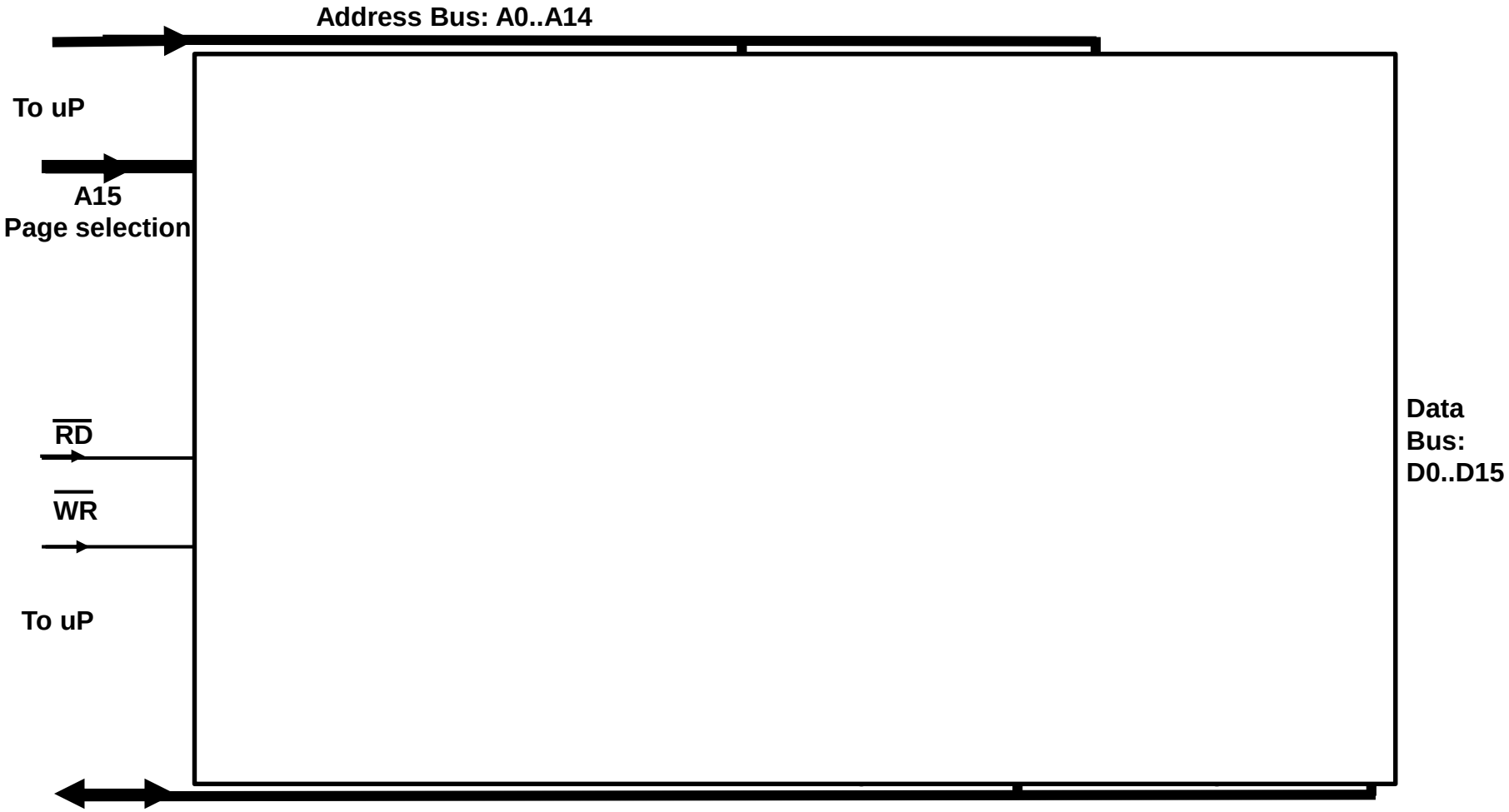
- Control signals are in negative symbol
- Memory read/write is activated only when the signals change from LOW to HIGH
- Therefore, in order to activate the signal, the signal should be in LOW
- $\overline{OE}$ or ~OE : symbol for output enable: read when it is true/active (output enable, OE, is LOW)
- $\overline{WR}$ or ~WR: symbol for writing: write to the memory when it is true/active (write enable, WR, is LOW)
- Logical equations: (Note that it is based on the circuit design in the previous slide)
  - **MEMORY READ = $\overline{OE}$ * $\overline{CE}$ * $\overline{WR}$**
  - **MEMORY WRITE = OE * $\overline{CE}$ * $\overline{WR}$**

Expanding memory by width



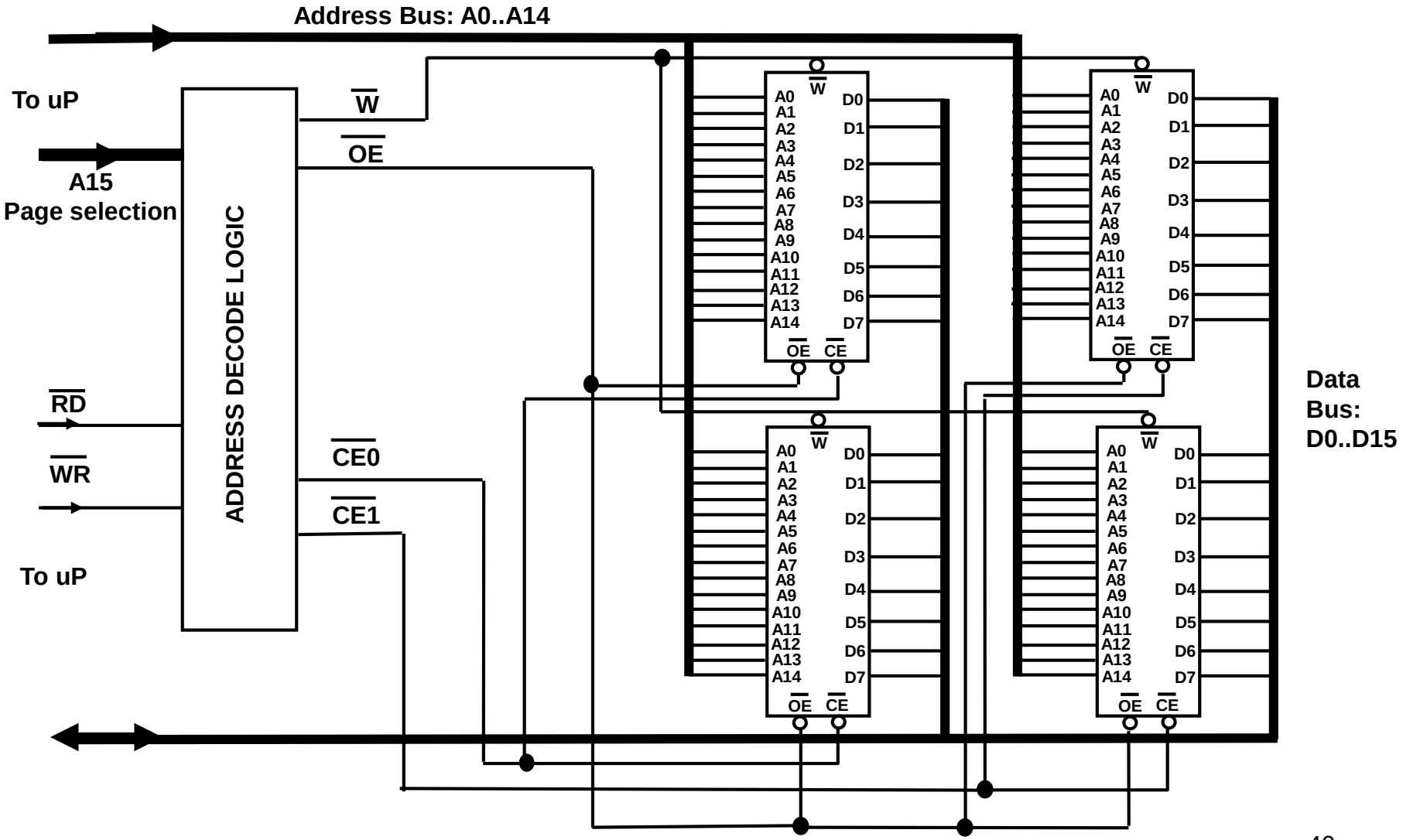
Decoding the memory space

- Design the SRAM (64K x 16) memory system using 256K bit (32K x 8) SRAM chips



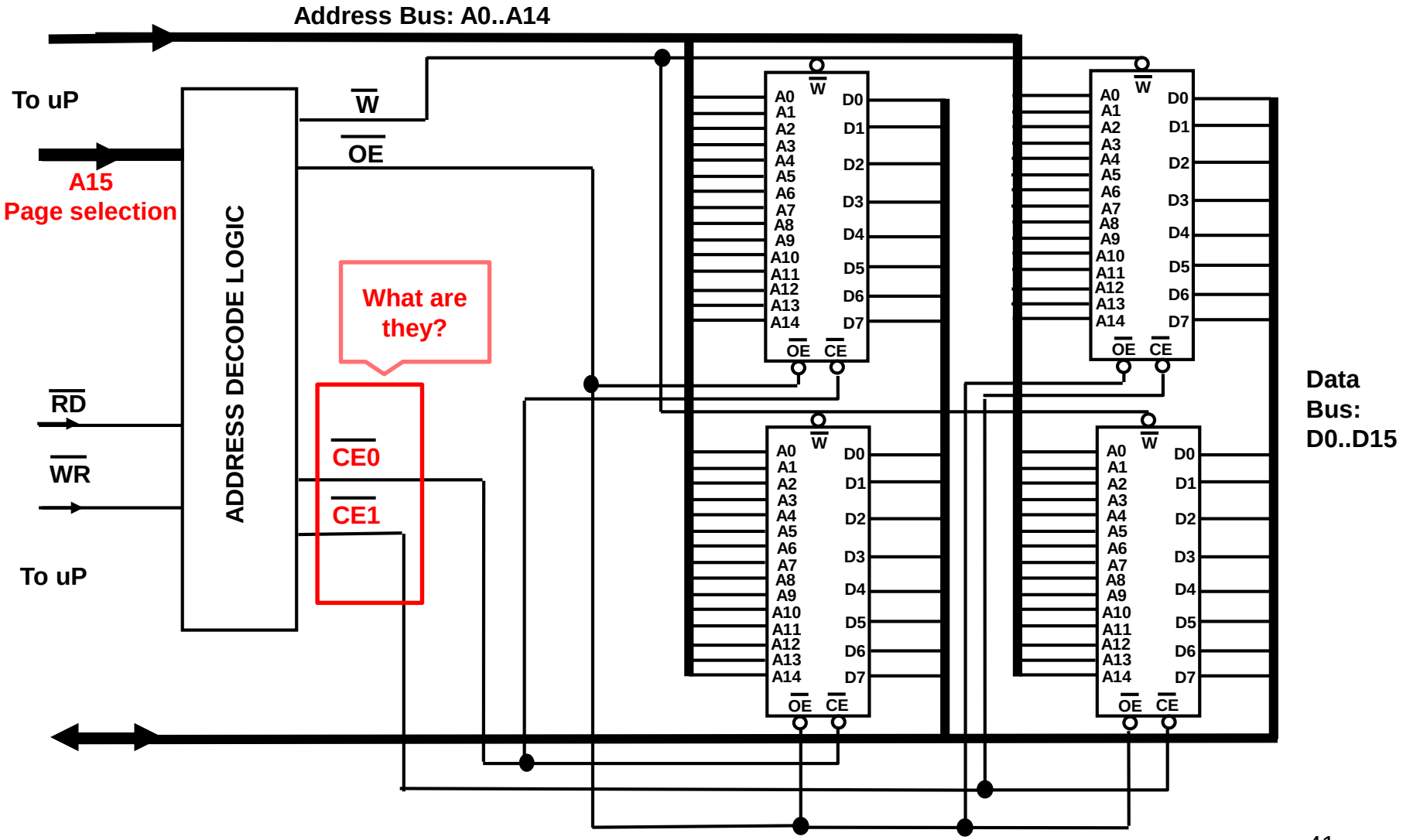
Decoding the memory space

- Design the SRAM (64K x 16) memory system using 256K bit (32K x 8) SRAM chips



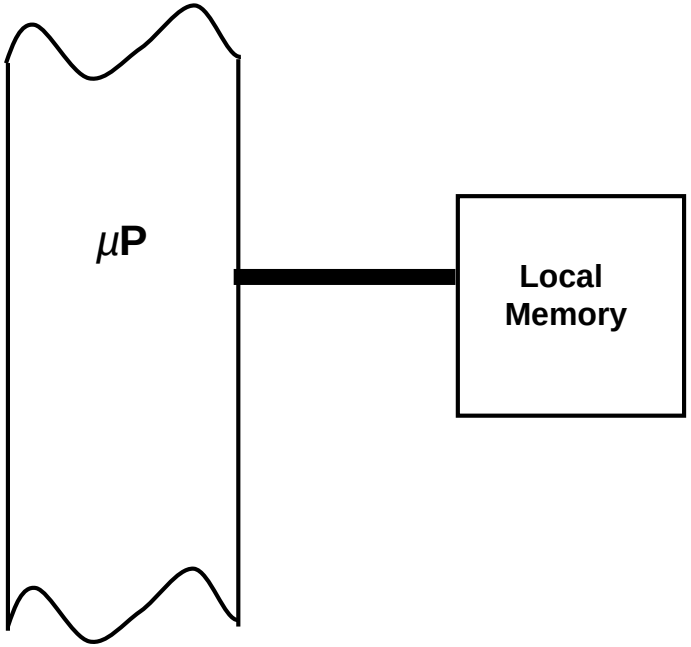
Decoding the memory space

- Design the SRAM (64K x 16) memory system using 256K bit (32K x 8) SRAM chips



Processor and memory

- Addressing is based on μP configuration
- You should understand the addressing in terms of uP, assuming that you do not know the physical memory configuration



Processor and memory

- Suppose a μP has a 17-bit address bus and 16-bit wide data bus, and it is byte-addressable addressing. And this μP is connected to a memory system M which is built with 32K x 8 devices

1) How many address lines and data lines at ***μP?***

17 address lines and 16 data lines

2) How many bytes the memory system M should have (The capacity of the memory system M)?

With 17 address bits, each byte should be accessible. Therefore, 2^{17} = 128Kbytes

3) How many address lines and data lines in memory system M?

The capacity of memory is the same as 128Kbytes = 2^{17} x2^{3}= 2^{20}bits. Since it has 16 data lines, the total number of addressable space is $2^{20}/2^{4}=2^{16}$

Therefore, 16 address lines and 16 data lines

4) How many page selection bits in memory system M?

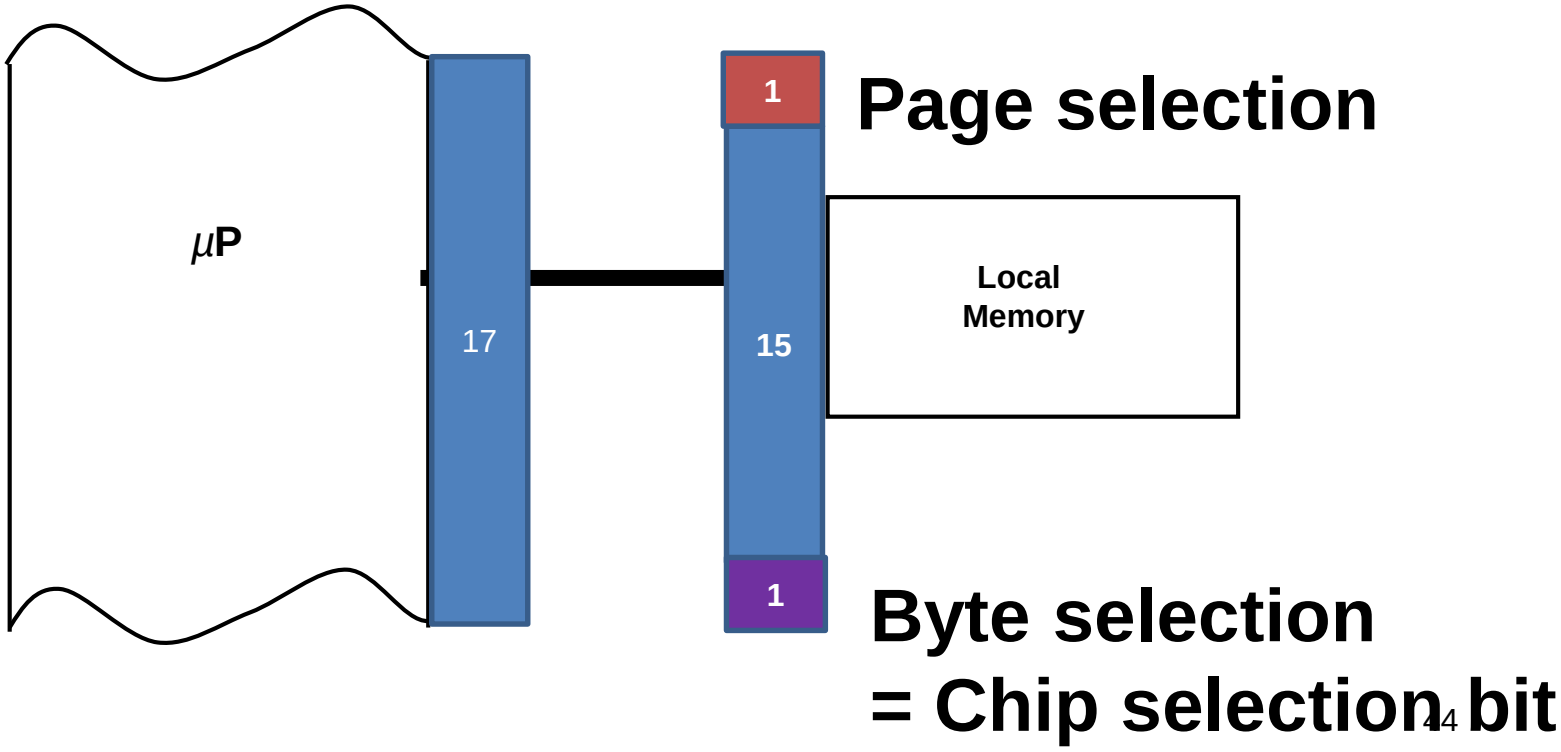
Since each page is 32K=2^{15}addressable spaces, the total number of pages in M is $2^{16}/2^{15}=2^{1}$. Therefore, 1 page selection bit.

5) How many chip selection bits in the memory system M?

Since the data is 16 bits, it should have 2 chips per page. Therefore, 1 chip selection bit.

Processor and memory

- Suppose a μP has a 17-bit address bus and 16-bit wide data bus, and it is byte-addressable addressing. And this μP is connected to a memory system M which is built with 32K x 8 devices



Processor and memory

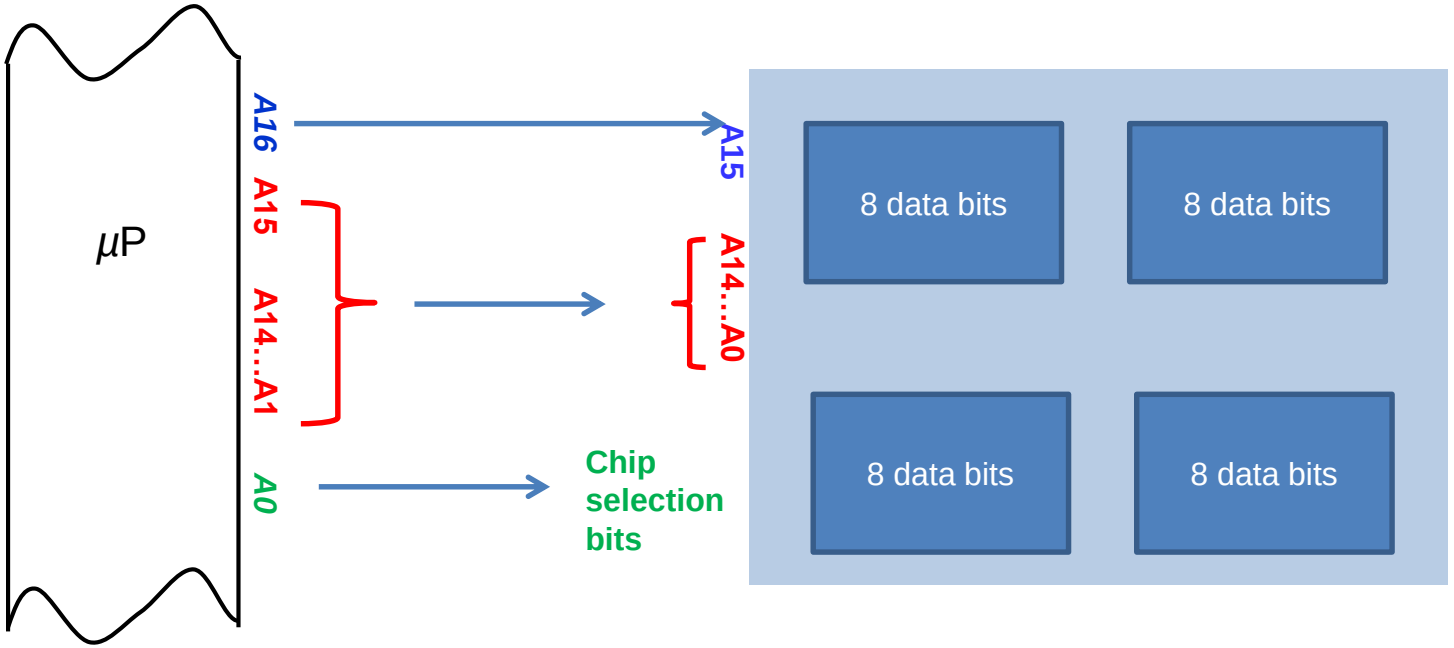
μP has a 17-bit address bus and 16-bit wide data bus, byte-addressable.

The memory system M is built with 32K x 8 devices

- What is the physical memory organization?

Address in uP	1	15	1
	Page	Offset	byte

- Page: selection the page bits in the memory
- Offset: the number of lines in a page in the memory
- Byte selection: the number of bytes in a page in the memory



Processor and address

- A μP has 10-bit address bus and 32-bit wide data bus. Assume that you are accessing a memory system that has been built using 128 x 8 memory chips. The μP is **byte-addressable.**
  - How many bits you need for byte selection?

  Since each row has 4 bytes (32-bits), you need two bits for byte selection
  - How many offset bits you need?

  Addressable space for each chip is 128, which is 7 bits. It is the same as the offset bits in uP
  - How many pages you have?

  10 bits = page bits + offset + byte selection = ? + 7 + 2. Therefore 1 bit is for page selection: Therefore, 2 pages.
  - Divide the 10-bit address lines into page number bits, offset bits and byte address bits.

1 (page)	7 (offset)	2 (byte)

  - How many 128 x 8 memory chips you need to completely fill the memory space of this processor?

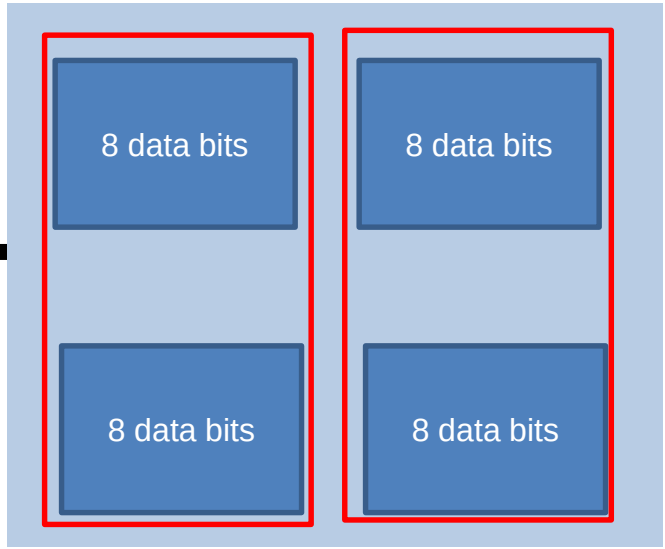
  There are 2 pages, and each page need 4 chips: 2 X 4 = 8 chips total

Byte addressing – big endian

- Where to start from each row? From MSB

1	15	1
Page	Offset	byte

16 data bits – 8 data bits/use



Page (A16)	Word Address (A16..A0)	D15…D8	D7..D0
0	0x00000	0x00000	0x00001
	0x00002	0x00002	0x00003
	0x00004	0x00004	0x00005
	..	…	…
	0x0FFFE	0x0FFFE	0x0FFFF
1	0x10000	0x10000	0x10001
	0x10002	0x10002	0x10003
	0x10004	0x10004	0x10005
	…	…	…
	0x1FFFE	0x1FFFE	0x1FFFF

Word address: should be always even number: **This is a big-endian processor**

Byte Packing – big endian

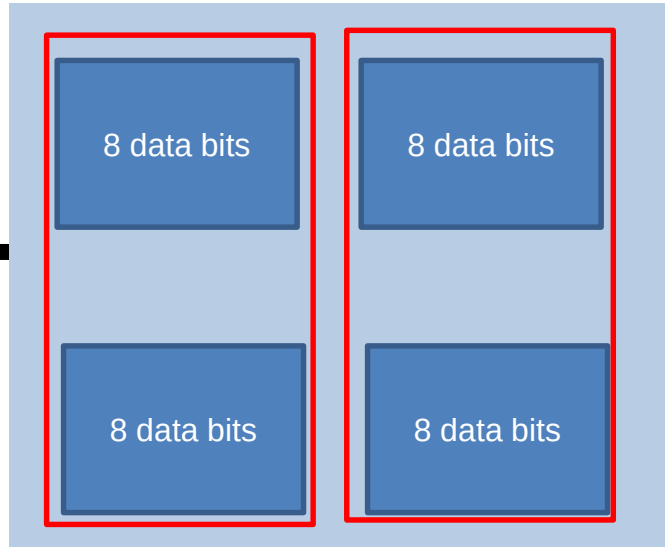
- When the size of the data element (byte) is smaller than the width of the available memory, we use byte packing to save space
- Big Endian- read from the MSB

longword Address				
000000	Byte 0 – Address 000000	Byte 1 – Address 000001	Byte 2 – Address 000002	Byte 3 – Address 000003
000004	Byte 0 – Address 000004	Byte 1 – Address 000005	Byte 2 – Address 000006	Byte 3 – Address 000007
000008	Byte 0 – Address 000008	Byte 1 – Address 000009	Byte 2 – Address 00000A	Byte 3 – Address 00000B
00000C	Byte 0 – Address 00000C	Byte 1 – Address 00000D	Byte 2 – Address 00000E	Byte 3 – Address 00000F
000010	Byte 0 – Address 000010	Byte 1 – Address 000011	Byte 2 – Address 000012	Byte 3 – Address 000013

FFFFF0	Byte 0 – Address FFFFF0	Byte 1 – Address FFFFF1	Byte 2 – Address FFFFF2	Byte 3 – Address FFFFF3
FFFFF4	Byte 0 – Address FFFFF4	Byte 1 – Address FFFFF5	Byte 2 – Address FFFFF6	Byte 3 – Address FFFFF7
FFFFF8	Byte 0 – Address FFFFF8	Byte 1 – Address FFFFF9	Byte 2 – Address FFFFFA	Byte 3 – Address FFFFFB
FFFFFC	Byte 0 – Address FFFFFC	Byte 1 – Address FFFFFD	Byte 2 – Address FFFFFE	Byte 3 – Address FFFFFF

Byte addressing – little endian

- Where to start from each row? From LSB



Page (A16)	Word Address (A16..A0)	D15…D8	D7..D0
0	0x00000	0x00001	0x00000
	0x00002	0x00003	0x00002
	0x00004	0x00005	0x00004
	..	…	…
	0x0FFFE	0x0FFFF	0x0FFFE
1	0x10000	0x10001	0x10000
	0x10002	0x10003	0x10002
	0x10004	0x10005	0x10004
	…	…	…
	0x1FFFE	0x1FFFF	0x1FFFE

Word address: should be always even number: **This is a little-endian processor**

Byte Packing – little endian

- When the size of the data element (byte) is smaller than the width of the available memory, we use byte packing to save space
- Little Endian- read from LSB

Longword Address				
000000	Byte 3 – Address 000003	Byte 2 – Address 000002	Byte 1 – Address 000001	Byte 0 – Address 000000
000004	Byte 3 – Address 000007	Byte 2 – Address 000006	Byte 1 – Address 000005	Byte 0 – Address 000004
000008	Byte 3 – Address 00000B	Byte 2 – Address 00000A	Byte 1 – Address 000009	Byte 0 – Address 000008
00000C	Byte 3 – Address 00000F	Byte 2 – Address 00000E	Byte1– Address 00000D	Byte 0 – Address 00000C
000010	Byte 3 – Address 000013	Byte 2 – Address 000012	Byte 1 – Address 000011	Byte 0 – Address 000010

FFFFF0	Byte 3 – Address FFFFF3	Byte 2 – Address FFFFF2	Byte 1 – Address FFFFF1	Byte 3 – Address FFFFF0
FFFFF4	Byte 3 – Address FFFFF7	Byte 2 – Address FFFFF6	Byte 1 – Address FFFFF5	Byte 3 – Address FFFFF4
FFFFF8	Byte 3 – Address FFFFFB	Byte 2 – Address FFFFFA	Byte 1 – Address FFFFF9	Byte 3 – Address FFFFF8
FFFFFC	Byte 3 – Address FFFFFF	Byte 2 – Address FFFFFE	Byte 1 – Address FFFFFD	Byte 3 – Address FFFFFC

Memory organization

- Memory organization usually depends upon the width of the processor data bus
  - The 68000 is a 32-bits for data internally, but interfaces to a 16-bit data bus: can read 16-bits at a time
  - All addresses are 32-bits internally, with 24-bits external (16M)