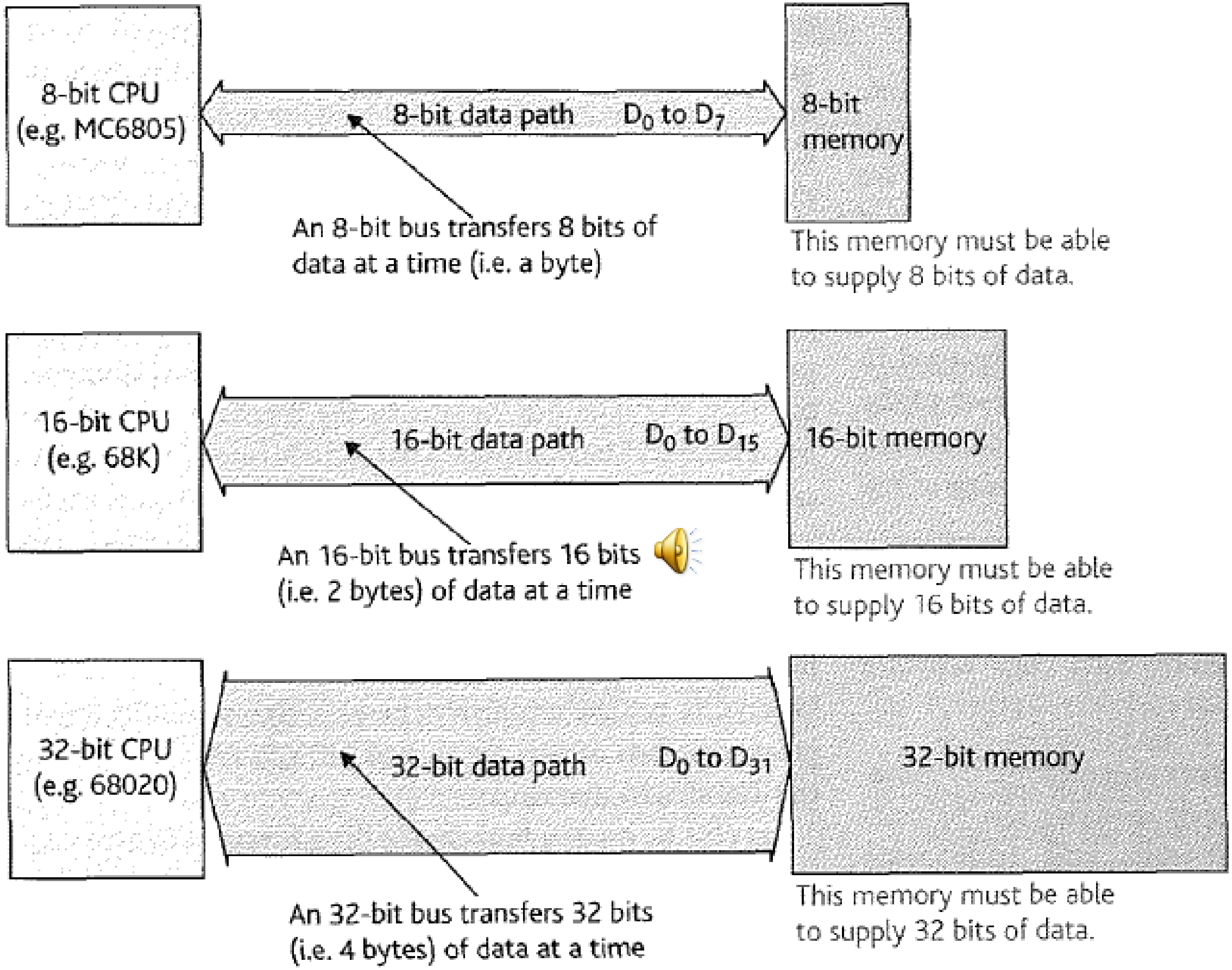
12.5 Interfacing memory to a CPU

12.5.1 Memory organization

- A microprocessor operates on a word of width *w bits and communicates with memory over a bus of width b bits.*
- *Memory components of width m bits are connected to the microprocessor via this bus.*
- *In the best case, the values of w, b, and m are all the same.*



An 8-bit bus transfers 8 bits of data at a time (i.e. a byte)

This memory must be able to supply 8 bits of data.

An 16-bit bus transfers 16 bits (i.e. 2 bytes) of data at a time

This memory must be able to supply 16 bits of data.

An 32-bit bus transfers 32 bits (i.e. 4 bytes) of data at a time

This memory must be able to supply 32 bits of data.

The organization of 8-bit, 16-bit, and 32-bit systems.

Memory organization

- Consider the 68K microprocessor, which has an internal 32-bit architecture and a 16-bit data bus interface. When you read a 32-bit value in memory, the processor automatically performs two 16-bit read cycles.
- If you use 4-bit-wide memory devices in a 68K system, you have to arrange them in groups of four because a memory block must provide the bus with 16 bits of data.
- A memory system must be as wide as the data bus. That is, the memory system must be able to provide an 8-bit bus with 8 bits of data, a 16-bit bus with 16 bits of data, and a 32-bit bus with 32 bits of data, etc.

Memory organization

Figure 12.14 demonstrates memory organization by showing how 16-bit-wide block of memory can be constructed from 4-bit-wide, 8-bit-wide, and 16-bit-wide memory components.

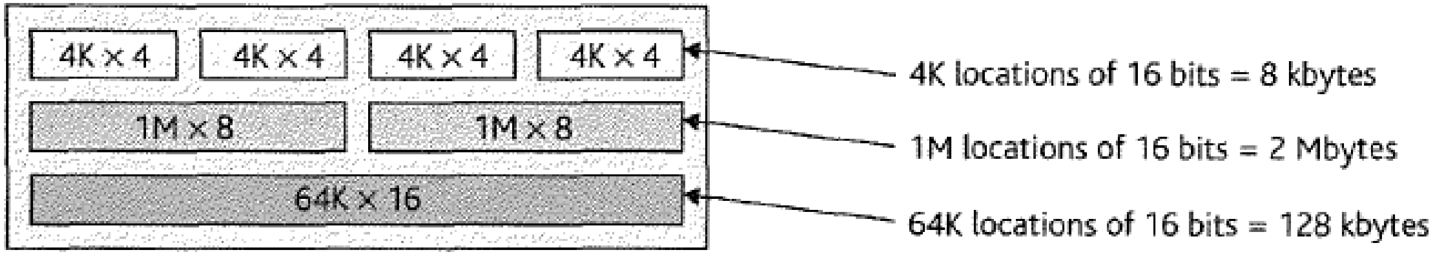

4K × 4
4K × 4
4K × 4
4K × 4
1M × 8
1M × 8
64K × 16
4K locations of 16 bits = 8 kbytes
1M locations of 16 bits = 2 Mbytes
64K locations of 16 bits = 128 kbytes

Figure 12.14 16-bit memory organization.

Memory organization

- The amount of data in a block of memory, in bytes, is equal to the width of the data bus (in bytes) multiplied by the number of locations in the block of memory.
- **Example 1** An **8-bit** computer with an **8-bit** bus uses memory components that are **4** bits wide. **Two** of these devices are required to supply 8 bits of data; each chip supplies 4 bits.
- **Example 2** A **16-bit** computer with a **16-bit** bus uses memory components that are **1 bit** wide. **Sixteen** of these devices are required to supply 16 bits of data at a time.

Memory organization

- **Example 3** An **8-bit** computer uses memory components organized as **64K X 4** bits; that is, there are 64K = 2^{16} different addressable locations in the chip. **Two** of these chips are required to provide the CPU with 8 data bits. The **total size** of the memory is **64 kbytes**.
- **Example 4** A **16-bit** computer uses memory components that are **64K X 4** bits. **Four** of these chips must be used to provide the CPU with 16 bits of data. Each of the 4 chips provides 4 of the 16 bits. The total size of the memory is 2 bytes X 64K = 128 kbytes

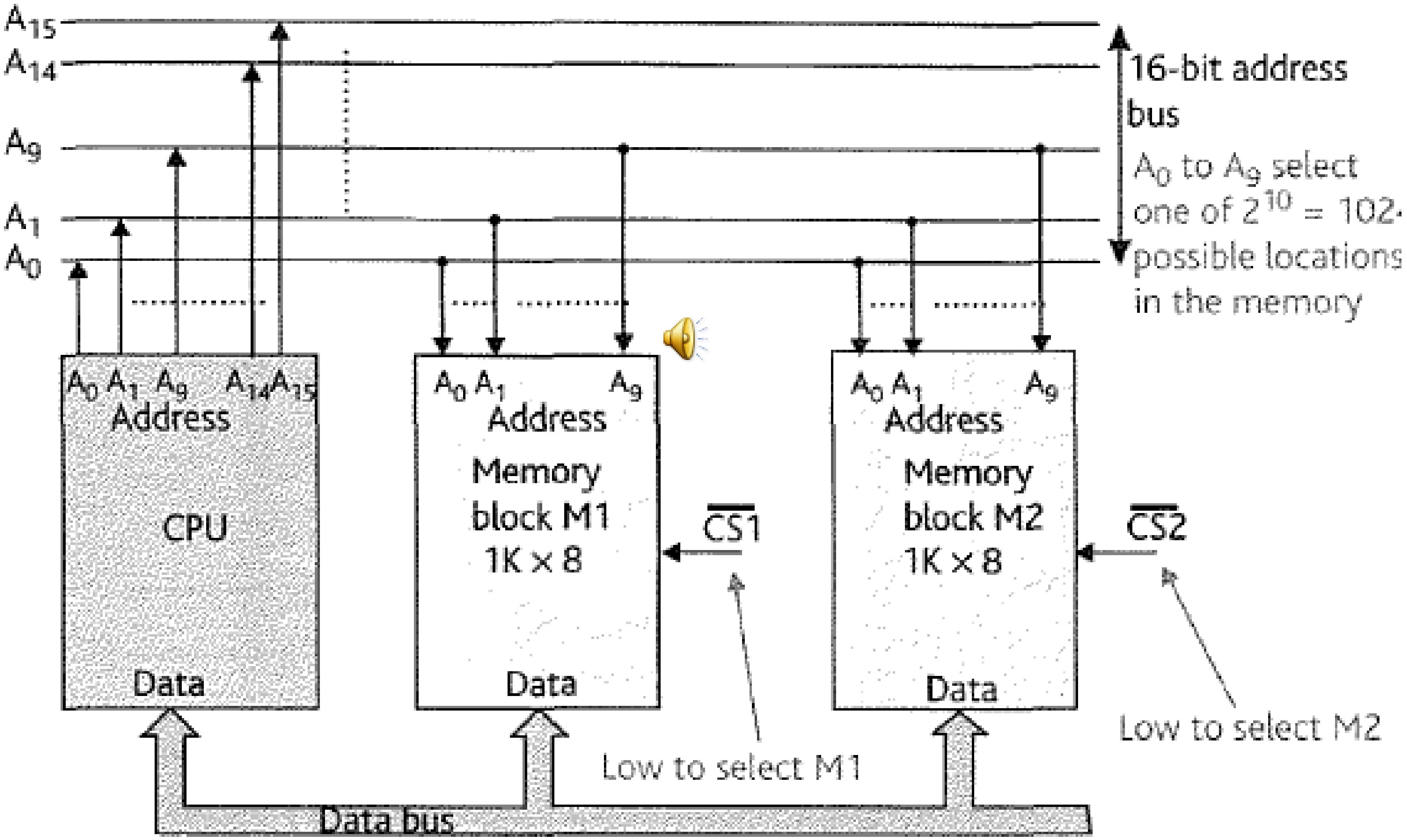
Memory organization

Example 5 A **16-bit** computer uses **64K X 16** memory components, Only **one** of these chips is required to provide 16 bits of data (2 bytes). Therefore, the chip provides 2 X 64K= 128 kbytes.

12.5.2 Address decoders

- If the memory in a microprocessor system were constructed from memory components with the same number of uniquely addressable locations as the processor, the problem of address decoding would not exist.
- Microprocessor systems often have memory components that are smaller than the addressable memory space. Moreover, there are different types of memory: read/ write memory, read-only memory, and memory-mapped peripherals.
- How memory components are interfaced to a microprocessor?

Address decoders



Address decoders

- Let CSl =f1(A15,A14,A13,A12,A11,A10). Similarly, let CS2 =f2(A15,A14,A13,A12,A11,A10).
- Suppose we choose functions f1 and f2 subject to the constraint that there are no values of A15,A14,A13,A12,A11,A10 *that cause both CSl and CS2 to be low simultaneously.*
- *There are several different strategies for decoding A10 to A15 (i.e. choosing functions f1 andf2).*
- *These strategies may be divided into three groups: partial address decoding, full address decoding, and block address decoding.*

Address decoders

EXAMPLE 1

An 8-bit microprocessor with a 16-bit address bus accesses addresses in the range 101xxxxxxxxxxxxx^2 (where bits A15, A14, A13 marked 101 are selected by the address decoder and the x's refer to locations within the memory block).

- – *What range of addresses does this block correspond to?*

- **The lowest address is 1010000000000000^2 and the highest address is 1011111111111111^2.This corresponds to the range A000 to BFFF**

- – *How big is this block?*

- **Three address lines are decoded to divide the address space spanned by A0 to A15 into eight blocks. The size of one block is 64K/8 = 8K.**
- **You could also calculate the size of the block because you know it is spanned by 13 address lines and 2^{13} = 8K.**

Address decoders

EXAMPLE 2

An 8-bit microprocessor with a 16-bit address bus addresses a block of 32 kbytes of ROM

(a) How many memory components are required if the memory is composed of 8 kbyte chips?

(a) The number of chips required is (memory block)/ (chip size) = 32K/8K = 4.

(b) What address lines from the processor select a location in the 32 kbyte ROM?

(b) Each chip has 8K = 2^{13} locations, which are accessed by the 13 address lines A0 to A12 from the processor.

(c) What address lines have to be decoded to select the ROM?

(c) Address lines A0 to A12 from the CPU select a location in the chip leaving A13 to A15 to be decoded.

Address decoders

EXAMPLE 2 cont.

(d) What is the range of memory locations provided by each of the chips (assuming that the memory blocks are mapped contiguously in the region of memory space starting at address 0000)?

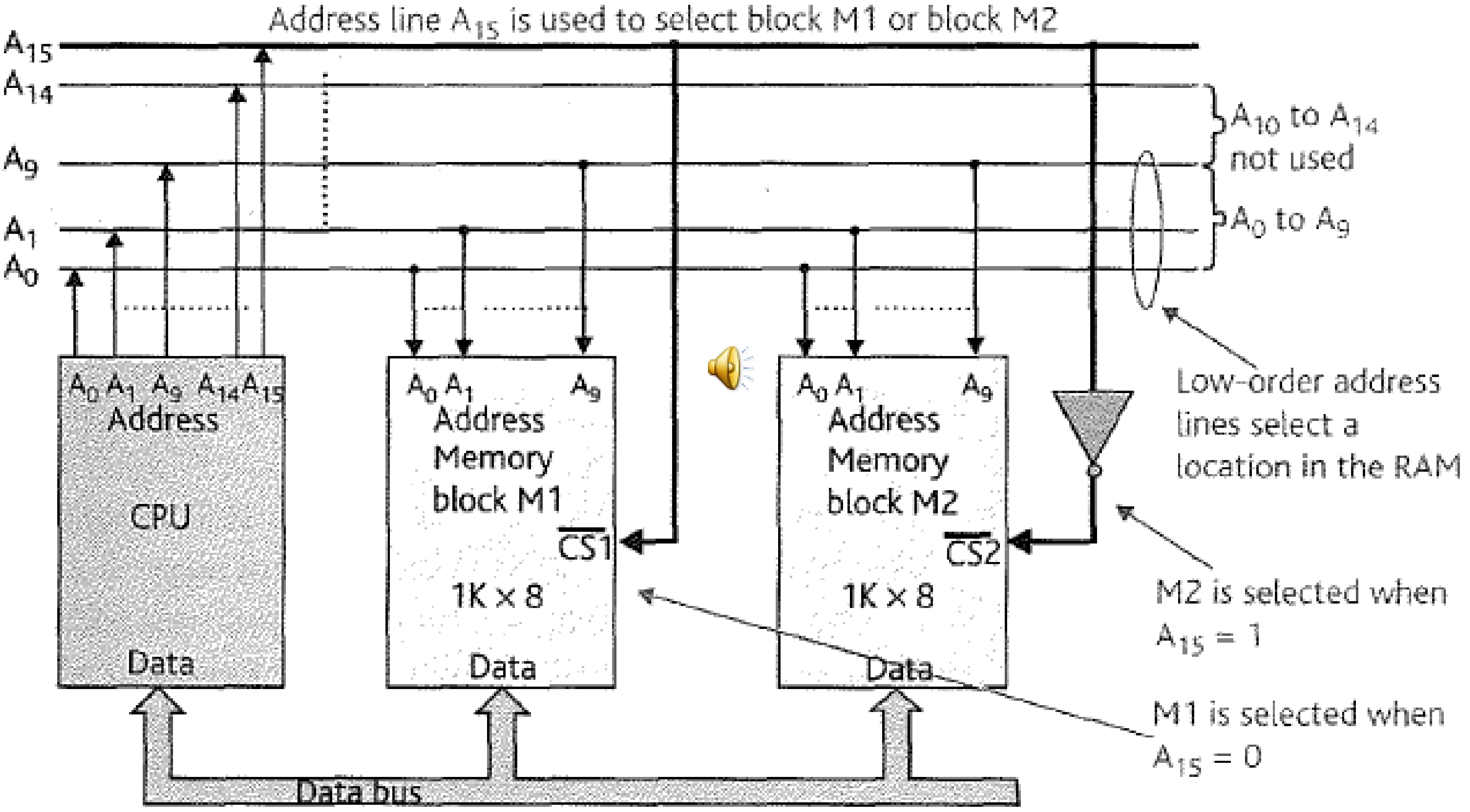
(d) For first chip, the left 3 bits will be 000 and the right 13 bit ranges from 000….0 to 111….1

Which is 000 0 0000 0000 0000

To 000 1 1111 1111 1111 → 0000 to 1FFF in hexa

The other memory blocks are 2000 to 3FFF
4000 to 5FFF
6000 to 7FFF

Partial address decoding

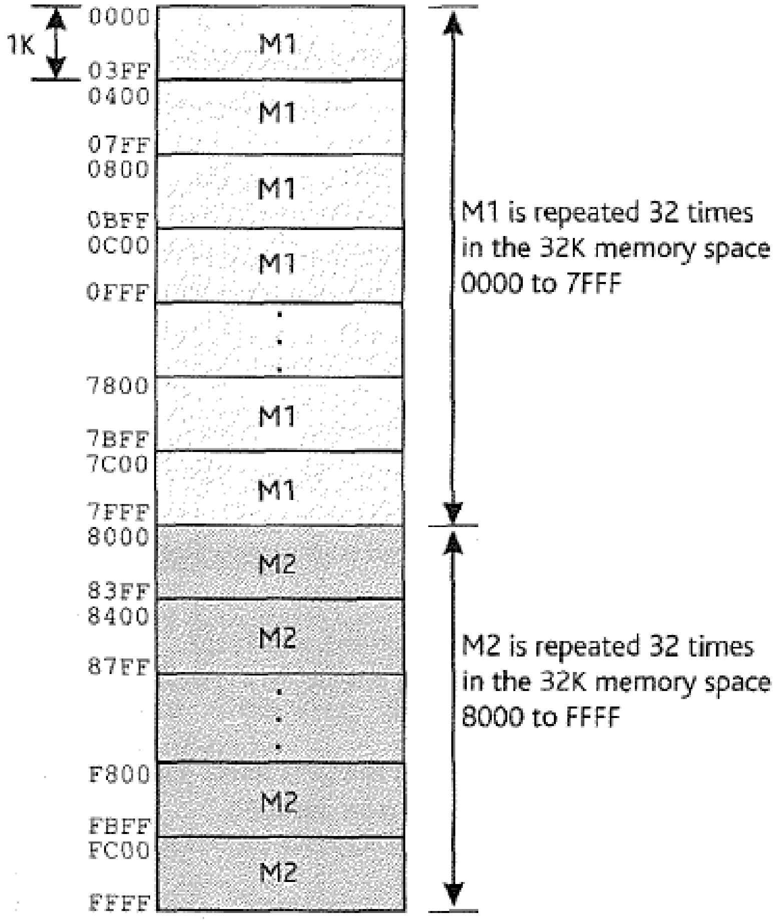

Address line A_{15} is used to select block M1 or block M2
A_{15}
A_{14}
A_9
A_1
A_0
A_{10} to A_{14} not used
A_0 to A_9
A_0 A_1 A_9 A_{14} A_{15}
Address
CPU
Data
A_0 A_1 A_9
Address
Memory block M1
$\overline{CS1}$
1K × 8
Data
A_0 A_1 A_9
Address
Memory block M2
$\overline{CS2}$
1K × 8
Data
Data bus
Low-order address lines select a location in the RAM
M2 is selected when $A_{15} = 1$
M1 is selected when $A_{15} = 0$

Partial address decoding

- Although we have distinguished between Ml and M2 for the cost of a **single inverter**, a heavy price has been paid.
- Because A15 = 0 selects Ml and A15 = 1 selects M2, it follows that either Ml or M2 will *always be selected. Although the system address bus can specify* 2^{16} *= 64K unique addresses, only 2K different* locations can be accessed.
- Address lines A10 to A14 take no part in the address-decoding process and consequently have no effect on the selection of a location within either Ml orM2.

Partial address decoding

- Memory block Ml is repeated 32 times in the lower half of the memory space and M2 is repeated 32 times in the upper half of the memory space because the five address lines A10 to A14 take no part in address decoding.
- The penalty paid when a partial address-decoding scheme is employed is that it prevents full use of the microprocessor's address space and frequently makes it difficult to expand the memory system at a later date.


1K
0000
03FF
0400
07FF
0800
0BFF
0C00
0FFF
7800
7BFF
7C00
7FFF
8000
83FF
8400
87FF
F800
FBFF
FC00
FFFF
M1
M1
M1
M1
M1
M1
M2
M2
M2
M2
M1 is repeated 32 times in the 32K memory space 0000 to 7FFF
M2 is repeated 32 times in the 32K memory space 8000 to FFFF

Full address decoding

- A microprocessor system has *full address decoding when each addressable location within a memory component is* accessed by a **single address** on the system's address bus; that is, **all** the microprocessor's address lines are used to access each physical memory location, either by specifying a given memory device or by specifying an address within it.
- Full address decoding represents the ideal but is sometimes impractical because it may require an **excessive** quantity of **hardware** to implement it.

Full address decoding

- We will design an address decoder for the pervious example of a system with two IK blocks of memory. Address lines A0 to A9 select a location in one of the memory components, leaving A10 to A15 to be decoded.
- Suppose we select Ml when A15,….,A10= 0,0, 0,0,0,0 and M2 when A15,….,A10 = 0,0,0,0,0, 1. These address values correspond to the IK address blocks 0000 to 03FF and 0400 to 07FF. Figure 12.18 (next slide) demonstrates how we might perform the address decoding with random logic.

Full address decoding

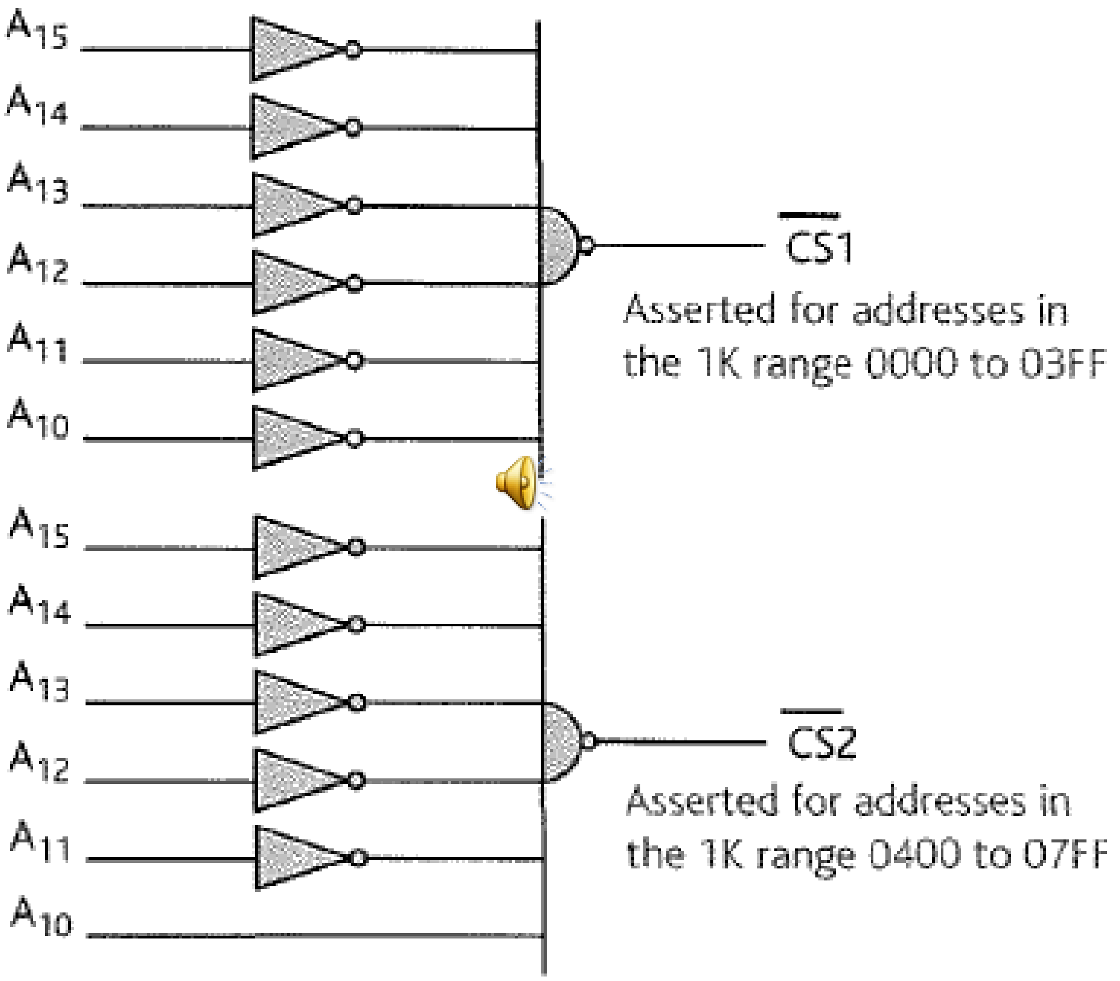


Figure 12.18 A full address decoder for two 1K memory blocks of Fig. 12.16.

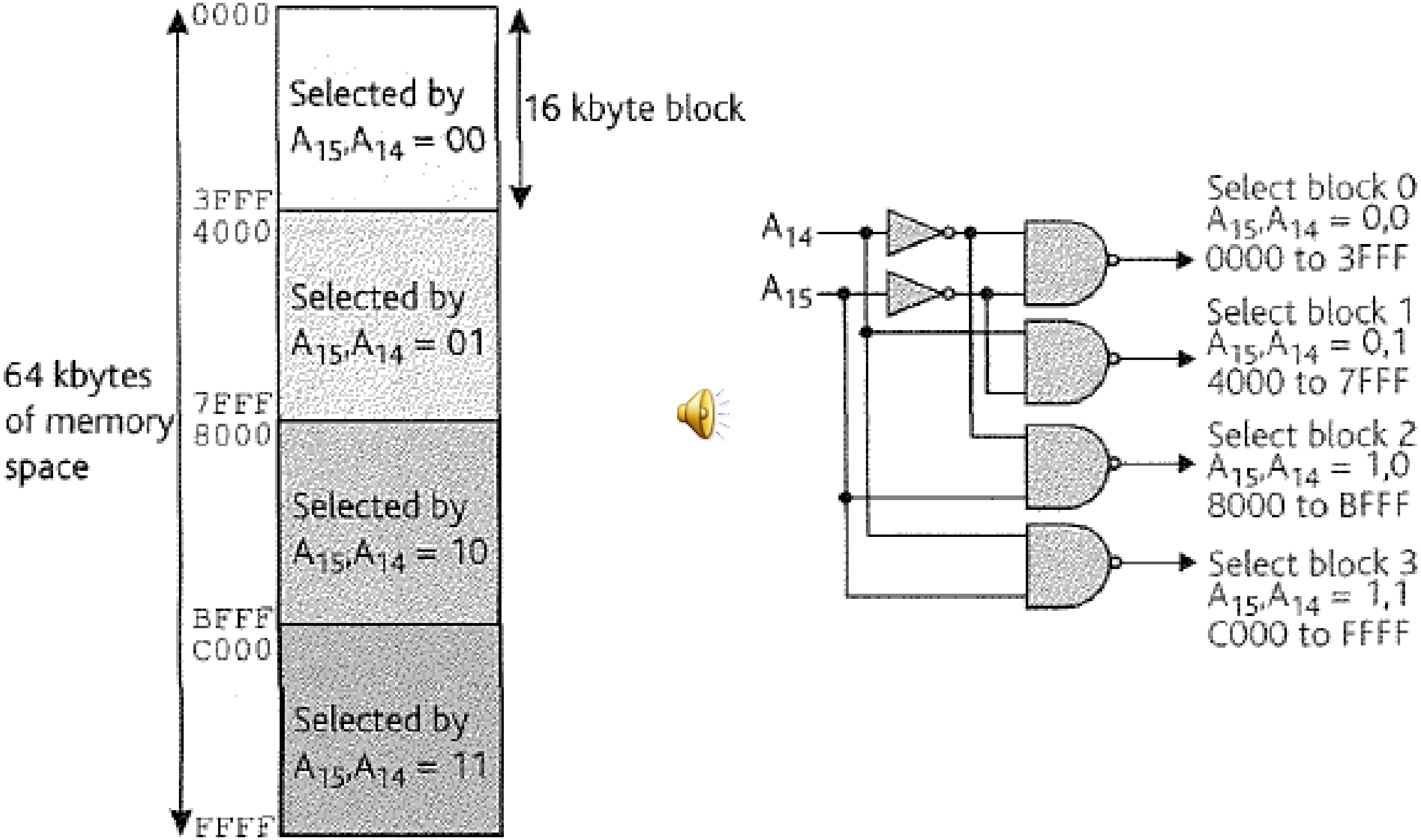
Block address decoding

- Block address decoding is a compromise between partial address decoding and full address decoding.
- It avoids the inefficient memory usage of partial address decoding, by dividing the memory space into blocks.
- Block address decoding is implemented by dividing the processor's address space into a number of equal-sized blocks.

Block address decoding

- Example: A microprocessor's 64K memory space is divided into four blocks of 16K.
- A 2-to-4 decoder converts the two high-order address lines, A15 and A14, into four lines.
- The four outputs of this address decoder are used as the chip-select inputs of memory components.
- In practice, real microprocessor systems often employ a combination of partial address decoding, full address decoding, and block address decoding.
- You can further decode these 16K blocks and divide the memory space between several peripheral devices. Figure 12.19 (next slide) describes how this arrangement might be implemented.

Block address decoding



(a) Memory map.

(b) Circuit of simple block address decoder.

Figure 12.19 Dividing 64K memory space into 4 blocks.

Block address decoding

Example

A microprocessor system is to be designed with

- 16 kbytes of ROM in the range 0000 to 3FFF using 4k-byte EPROMs and
- 8 kbytes of read/write memory in the range 4000 to 5FFF using a single 8-kbyte chip.
- Provision must be made for at least eight memory-mapped peripherals in the 256 byte range 6000 to 60FF.

What address lines have to be decoded to select each memory block?

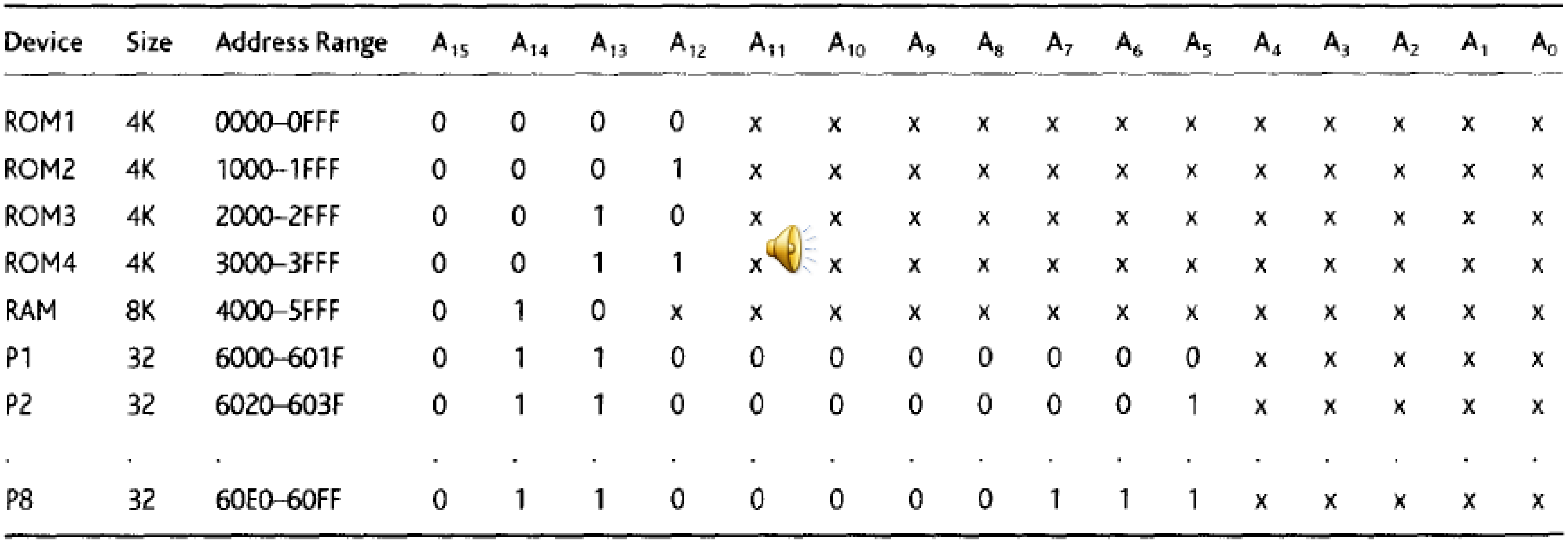
Consider the 8K RAM block. The memory space is 64K, so there are 64K/8K = 8 blocks. Because 8 = 2^3, the three high-order address lines have to be decoded.

Alternatively, we can write down the first and last addresses in the block and note which address values are common to all locations; that is,

4000 = **010**0000000000000 5FFF **= 010**1111111111111

Only the three high-order address lines are common to every location within this memory block.

Block address decoding

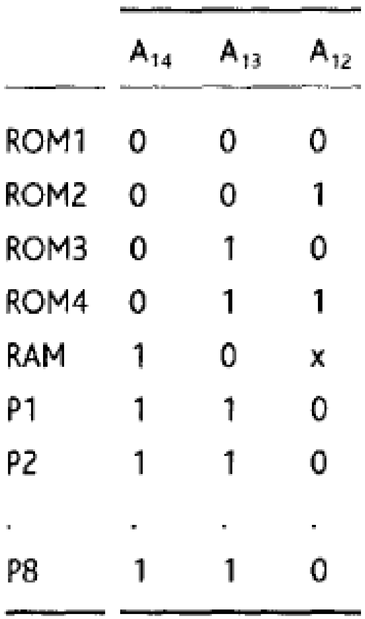


Device	Size	Address Range	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
ROM1	4K	0000–0FFF	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x
ROM2	4K	1000–1FFF	0	0	0	1	x	x	x	x	x	x	x	x	x	x	x	x
ROM3	4K	2000–2FFF	0	0	1	0	x	x	x	x	x	x	x	x	x	x	x	x
ROM4	4K	3000–3FFF	0	0	1	1	x	x	x	x	x	x	x	x	x	x	x	x
RAM	8K	4000–5FFF	0	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x
P1	32	6000–601F	0	1	1	0	0	0	0	0	0	0	0	x	x	x	x	x
P2	32	6020–603F	0	1	1	0	0	0	0	0	0	0	1	x	x	x	x	x
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
P8	32	60E0–60FF	0	1	1	0	0	0	0	0	1	1	1	x	x	x	x	x

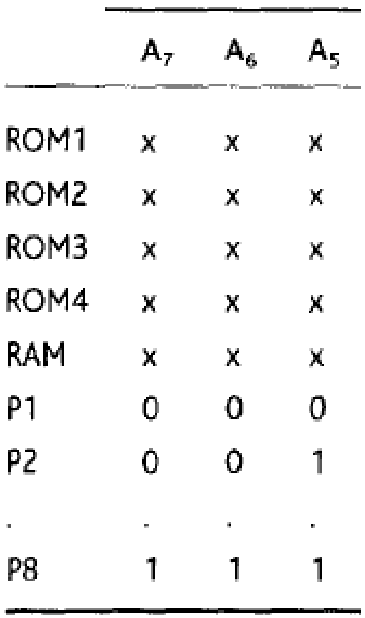
Table 12.3 Address table of a microprocessor system.

Block address decoding

- Table 12.3 (previous slide) shows how one 3-to-8 decoder (74LS138) divides the memory space into eight 4K blocks.
- A second decoder subdivides one of these blocks to provide memory space for the peripherals.
- Figure 12.20 (next slide) gives a circuit diagram of the address decoder

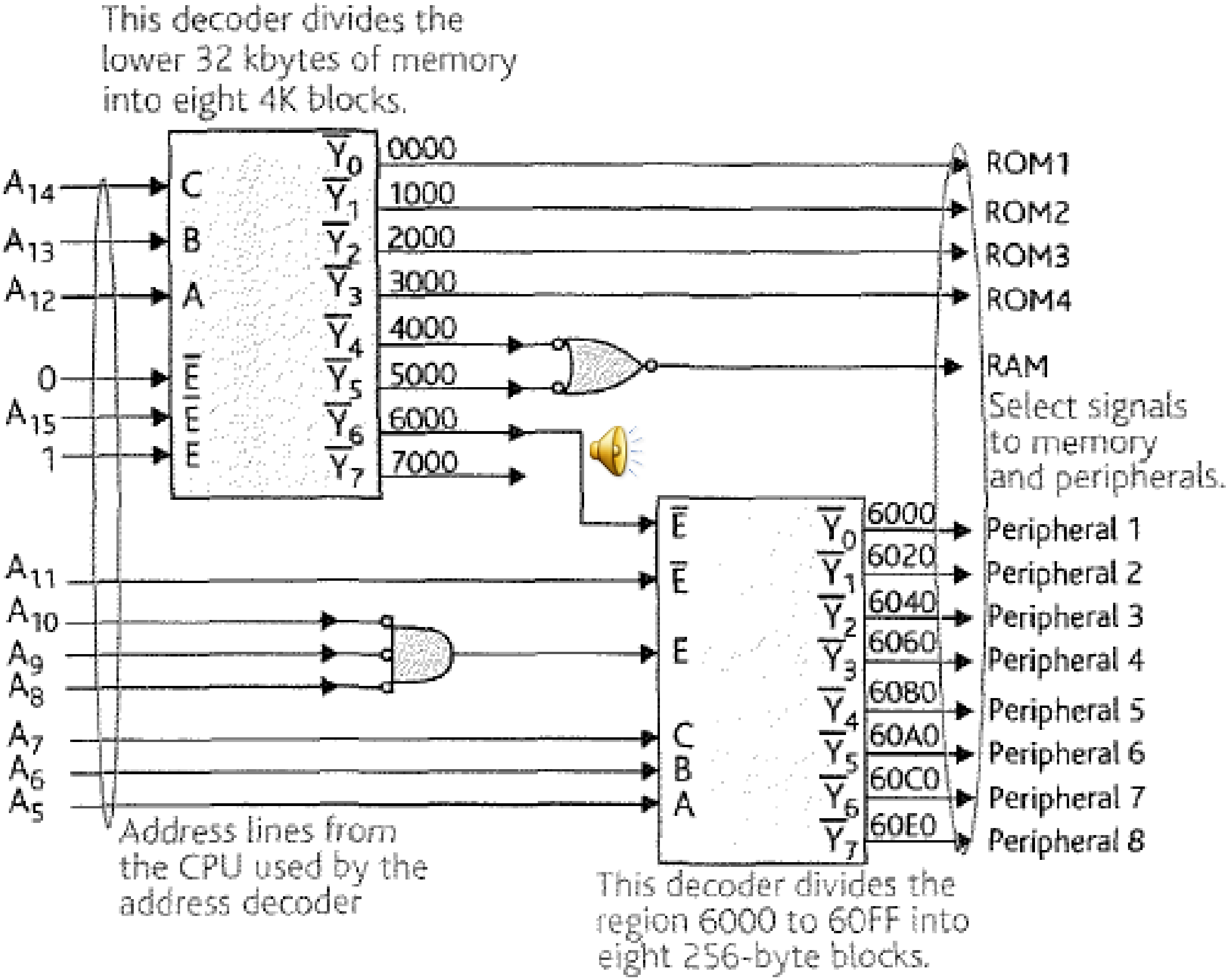


	A_{14}	A_{13}	A_{12}
ROM1	0	0	0
ROM2	0	0	1
ROM3	0	1	0
ROM4	0	1	1
RAM	1	0	x
P1	1	1	0
P2	1	1	0
.	.	.	.
P8	1	1	0



	A_7	A_6	A_5
ROM1	x	x	x
ROM2	x	x	x
ROM3	x	x	x
ROM4	x	x	x
RAM	x	x	x
P1	0	0	0
P2	0	0	1
.	.	.	.
P8	1	1	1

Block address decoding



This decoder divides the lower 32 kbytes of memory into eight 4K blocks.

Select signals to memory and peripherals.

Address lines from the CPU used by the address decoder

This decoder divides the region 6000 to 60FF into eight 256-byte blocks.

Figure 12.20 Circuit of an address decoder for Table 12.3.

Block address decoding

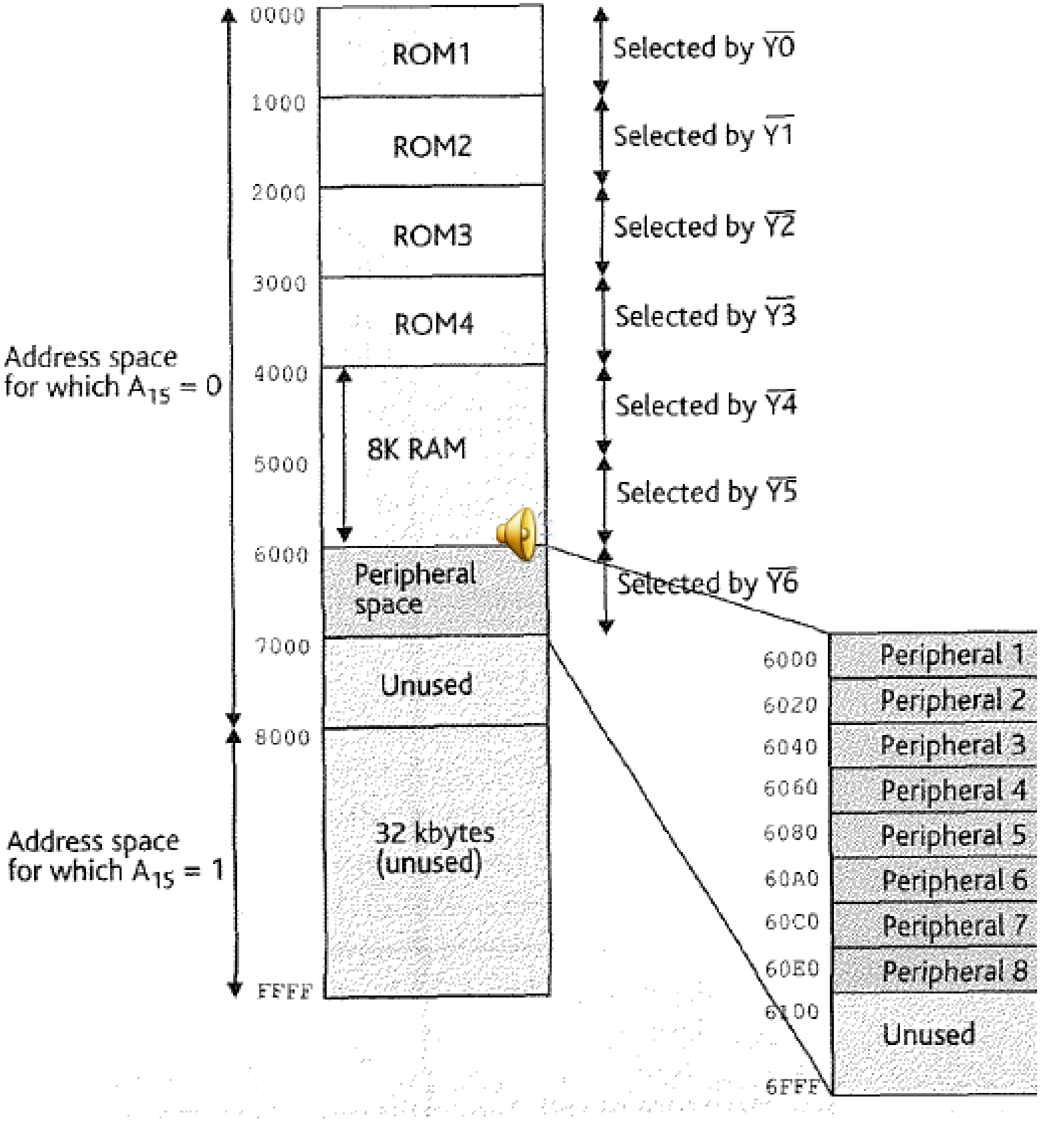

0000
ROM1
1000
ROM2
2000
ROM3
3000
ROM4
4000
8K RAM
5000
6000
Peripheral space
7000
Unused
8000
32 kbytes (unused)
FFFF
Address space for which $A_{15} = 0$
Address space for which $A_{15} = 1$
Selected by $\overline{Y0}$
Selected by $\overline{Y1}$
Selected by $\overline{Y2}$
Selected by $\overline{Y3}$
Selected by $\overline{Y4}$
Selected by $\overline{Y5}$
Selected by $\overline{Y6}$
6000
Peripheral 1
6020
Peripheral 2
6040
Peripheral 3
6060
Peripheral 4
6080
Peripheral 5
60A0
Peripheral 6
60C0
Peripheral 7
60E0
Peripheral 8
6100
Unused
6FFF

Figure 12.21 Memory map for the system of Table 12.3 and Fig. 12.20.

Block address decoding

- RAM, ROM, or peripheral devices can be added without further alterations to the address-decoding circuitry by employing the unused outputs of the decoders.
- Note how we've selected the 8K block of RAM

  Because the RAM is selected if either of the two 4K blocks selected by Y4 or Y5 is selected, we can OR (in negative logic terms) Y4 and Y5 to select the RAM.
- Because the peripherals don't occupy a 4K block, we have used address lines A8 to A11 to select a second 3-to-8 decoder that decodes the peripheral address space.