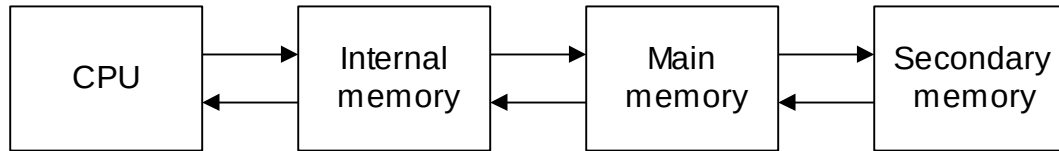


9. Memory System

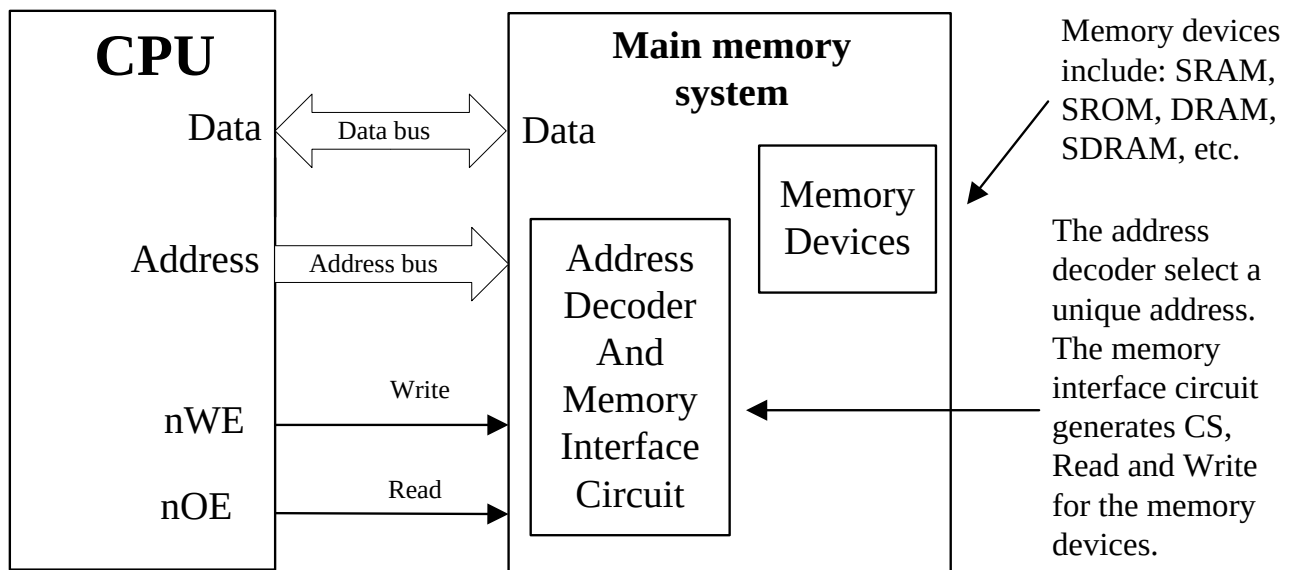
9.1 Introduction



The memory components of a computer system can be divided into four main groups:

1. **Internal processor memory:** high-speed registers and internal RAM (for fast access) and flash (for boot up).
2. **Cache memory:** It is a high speed memory to reduce the access time for retrieving information from main memory. It is usually resided on-chip and its operation is transparent to programmer.
3. **Main memory** (primary memory): This is a relatively large fast memory (static RAM, dynamic RAM, SDRAM) used for program and data storage. It can be accessed directly and rapidly by CPU.
4. **Secondary memory** (auxiliary or backing memory): Large capacity with access time slower than that of the main memory. Main functions include program and data files storage. It also serves as an overflow memory when the capacity of the main memory is exceeded, e.g, hard disk.

9.2 Basic Memory Interface



Basic memory interface for SRAM

9.3 Address Decoding

Address decoding is extremely important for memory system design. Each address **must be uniquely identified** when accessing.

Example 1:

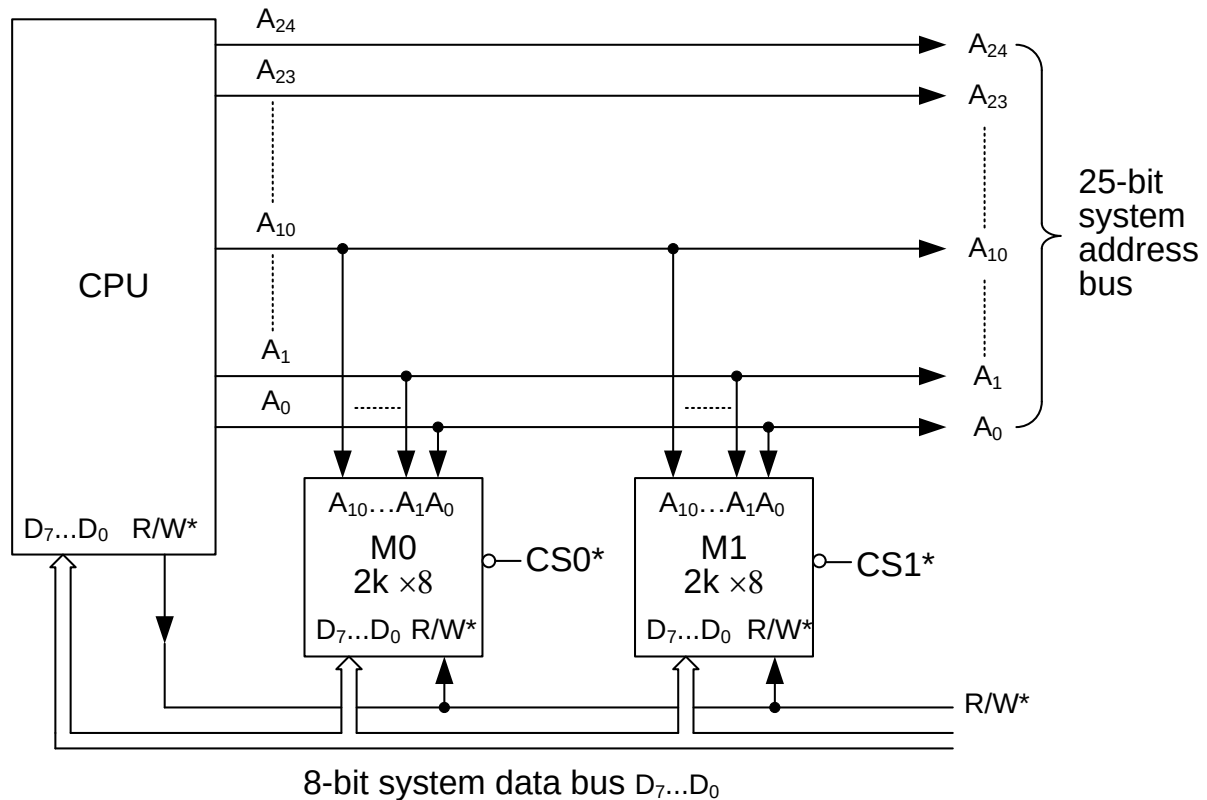
A microprocessor system, with 8-bit data bus, consists of two memory components, M0 and M1, each of 2K×8-bit (byte).

Note that the internal register length of this processor can be 8-bit, 16-bit or 32-bit.

If the register length is 8-bit, the CPU only needs to read the memory once to fetch the instruction.

If the register length is 16-bit, the CPU needs to read the memory twice to fetch the instruction.

Likewise, if the register length is 32-bit, the CPU needs to read the memory four times to fetch the instruction.



The 11 physical address lines, A₀ to A₁₀, from the CPU are connected to the corresponding address inputs of M0 and M1.

The data pins of both M0 and M1 are connected to the common data bus. The memory components must have drivers with **tristate** outputs to simultaneously access to the data bus. Otherwise, tristate transceiver should be used.

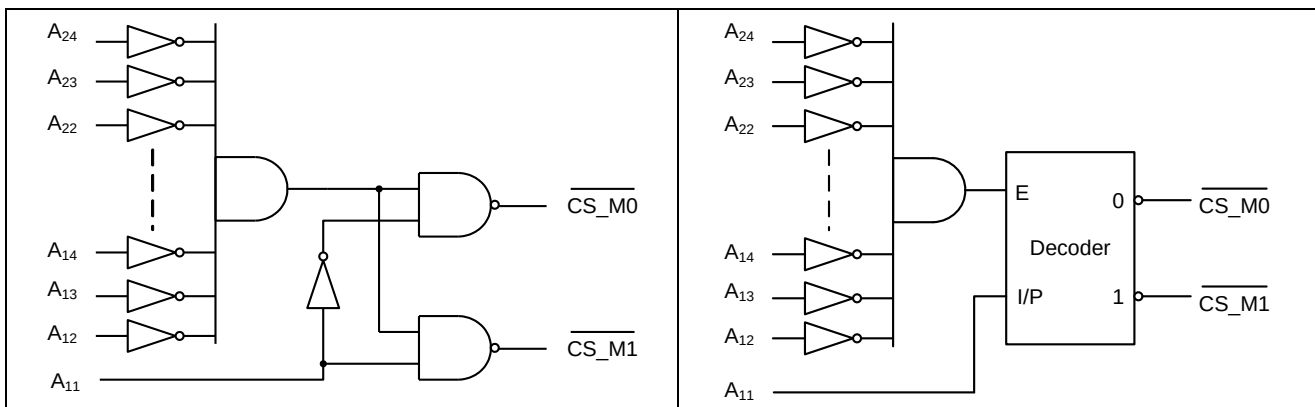
To avoid simultaneous access to the data bus, a unique chip select signal must be generated for each memory component to activate its tristate output. This chip select signal is derived from the remaining address lines of the CPU, i.e., CS0* = F(A₁₁ to A₂₄) and CS1* = F(A₁₁ to A₂₄) for this example. Three strategies to generate these chip select signals will be discussed: full address decoding, partial address decoding, and block address decoding.

9.3.1 Full Address Decoding

Full address decoding means that each addressable location within a memory component responds only to a single, **unique** address on the system address bus. In other words, **all the address lines** must be used to access each physical memory location, either to specify a given device or to specify an address within it.

One of the possible solutions for example 1 is:

Device	ADDRESS LINES																	
	24	23	22	...	13	12	11	10	09	08	07	06	05	04	03	02	01	00
M0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X
M1	0	0	0	0	0	0	1	X	X	X	X	X	X	X	X	X	X	X



000 0000

000 07FF

000 0800

000 0FFF

000 1000

1FF FFFF

M0

M1

Example 2

Using full address decoding to design a microprocessor system with the following components:

2K bytes (8-bit) ROM (ROM1)

2K bytes RAM (RAM)

8K bytes ROM (ROM2)

2 byte for peripheral (PERI1)

2 byte for peripheral (PERI2)

Address table:

Device	ADDRESS LINES																	
	24	...	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ROM1	0	...	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X
RAM	0	...	0	0	0	0	1	X	X	X	X	X	X	X	X	X	X	X
ROM2	0	...	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X
PERI1	0	...	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	X
PERI2	0	...	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	X

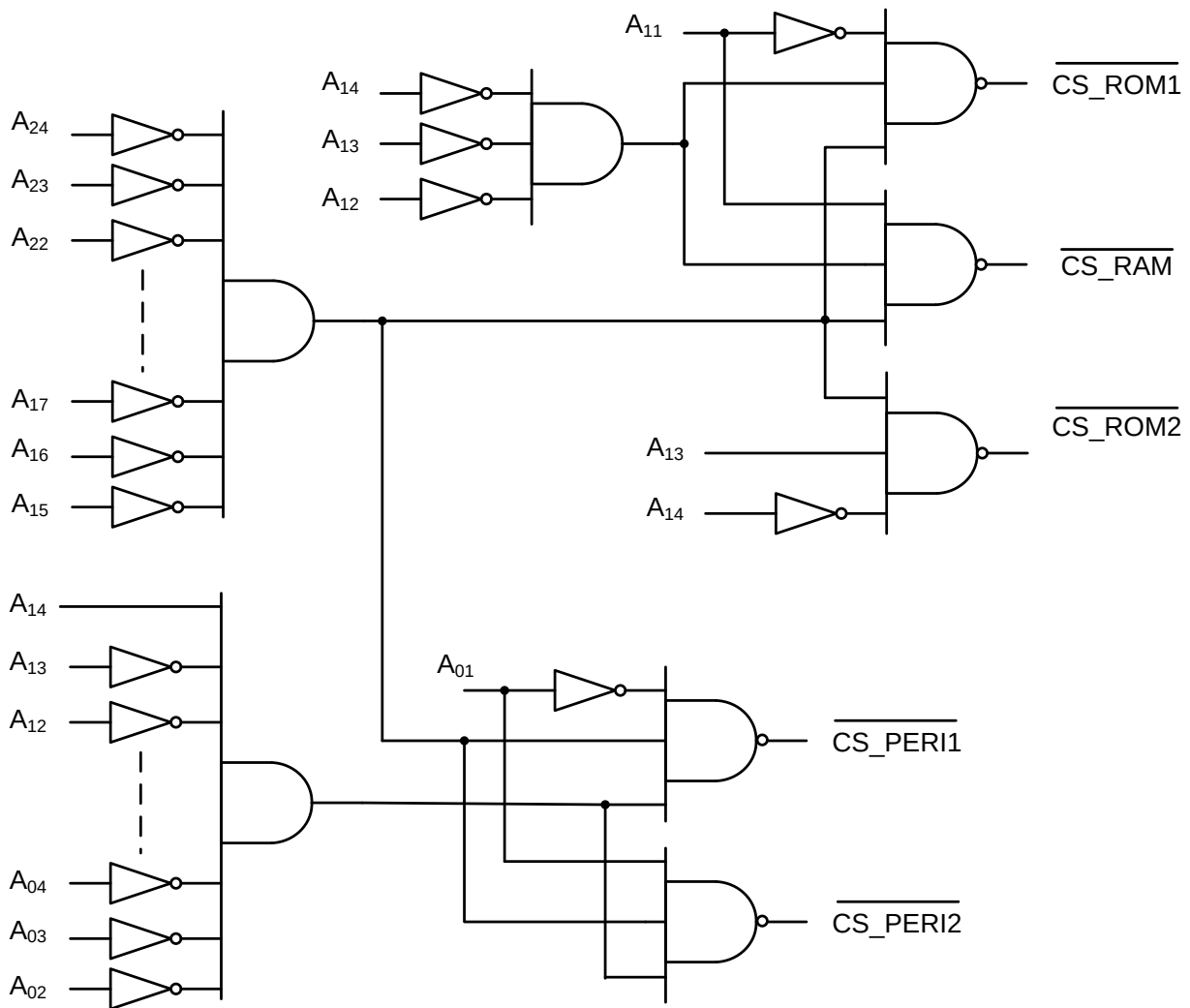
ROM1: \$000 0000 - \$000 07FF

RAM: \$000 0800 - \$000 0FFF

ROM2: \$000 2000 - \$000 3FFF

PER1: \$000 4000 - \$000 4001

PER2: \$000 4002 - \$000 4003



Full addressing decoding for example 2

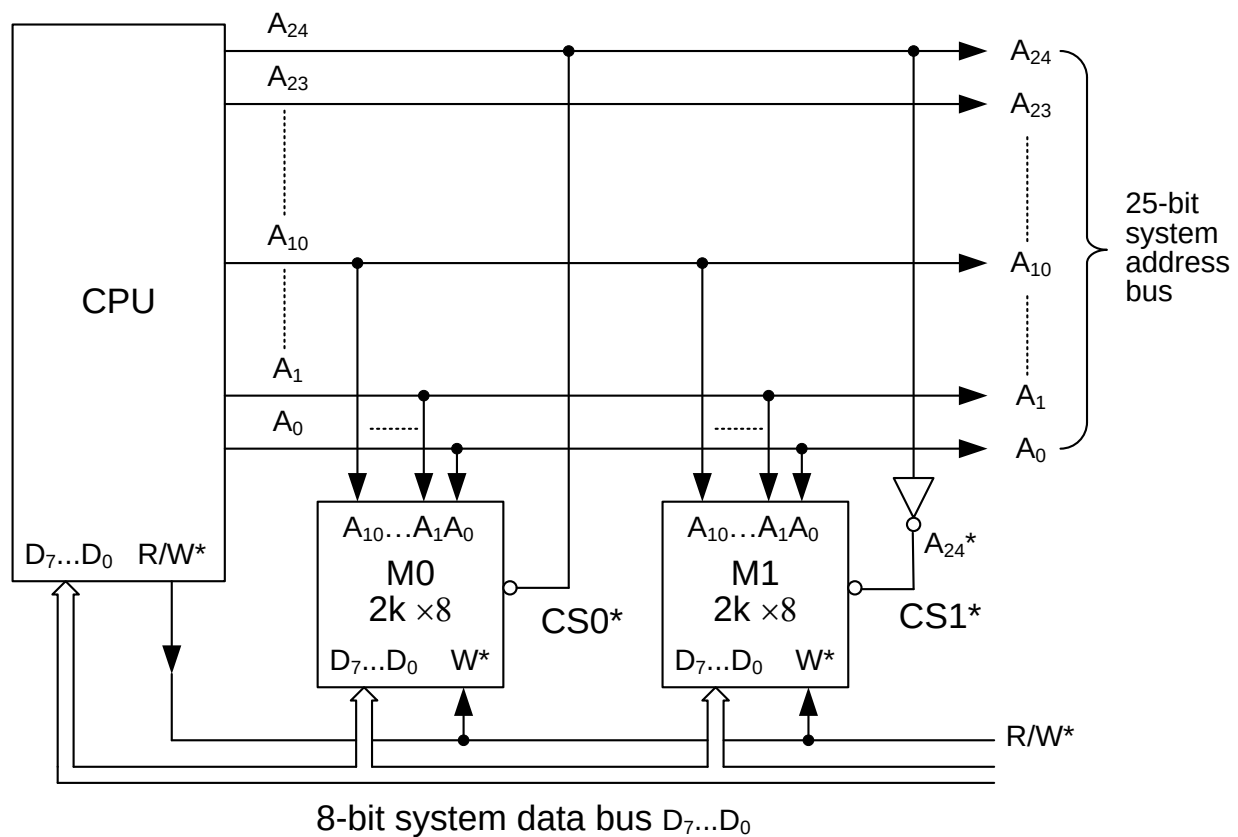
It is a relatively complex design with high chip counts, hence more expensive to implement.

OR use decoder based design – an exercise for you !!!

9.3.2 Partial Address Decoding

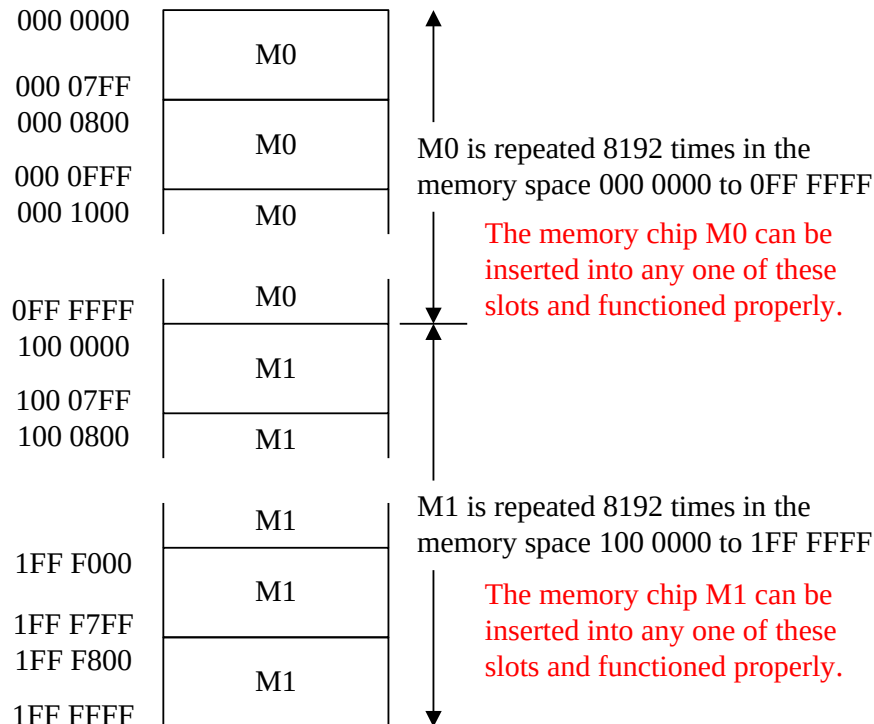
Partial decoding is the simplest and least expensive form for address decoding since only **a subset of address lines** are taking part in the decoding process. **An address may point to different memory locations at the same time.**

The following figure shows a partial decoding scheme satisfy the requirement for example 1.



The CS0* and CS1* of M0 and M1 are connected A₂₄ and A₂₄*, respectively. M0 is selected when A₂₄ = 0 and M1 is selected when A₂₄ = 1.

Since A₂₃ to A₁₁ are not used in the decoding process, M0 and M1 appear 8192 (2¹³) times in the lower and upper half of the memory map.



The memory map of example 1 using partial address decoding.

A simple explanation using a CPU with 8 address lines (A7-A0)

	A7	A6	A5	A4	A3	A2	A1	A0
Address 1:	1	1	0	0	0	1	0	1
Address 2:	1	1	1	0	1	1	0	1
Address 3:	1	1	0	1	1	1	0	1

If the address decoder only uses A7, A6, A2, A1 & A0 to design the decoding circuit, it will generate CS signal for all these addresses even though the actual addresses are different. Hence each address **cannot** be uniquely decoded. In fact, these 3 different addresses will be treated as one single common address.

Partial address decoding for example 2

Device	ADDRESS LINES																		
	24	23	22	21	...	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ROM1	0	0	0				0	0	X	X	X	X	X	X	X	X	X	X	X
RAM	0	0	1				0	0	X	X	X	X	X	X	X	X	X	X	X
ROM2	0	1					X	X	X	X	X	X	X	X	X	X	X	X	X
PERI1	1	0																	X
PERI2	1	1																	X

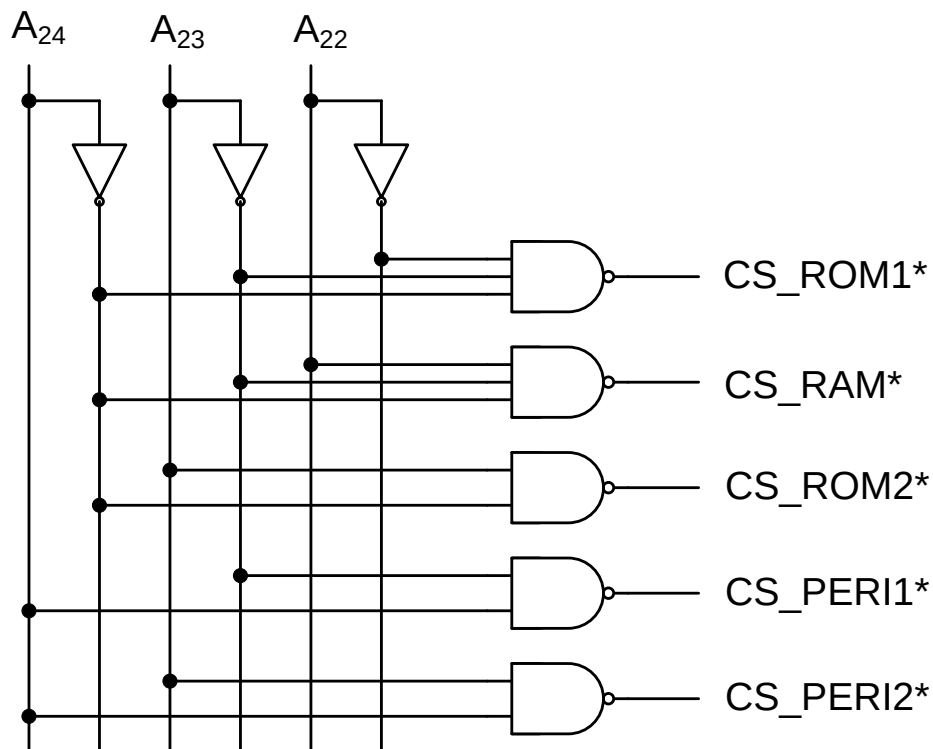
ROM1: \$000 0000 - \$000 07FF

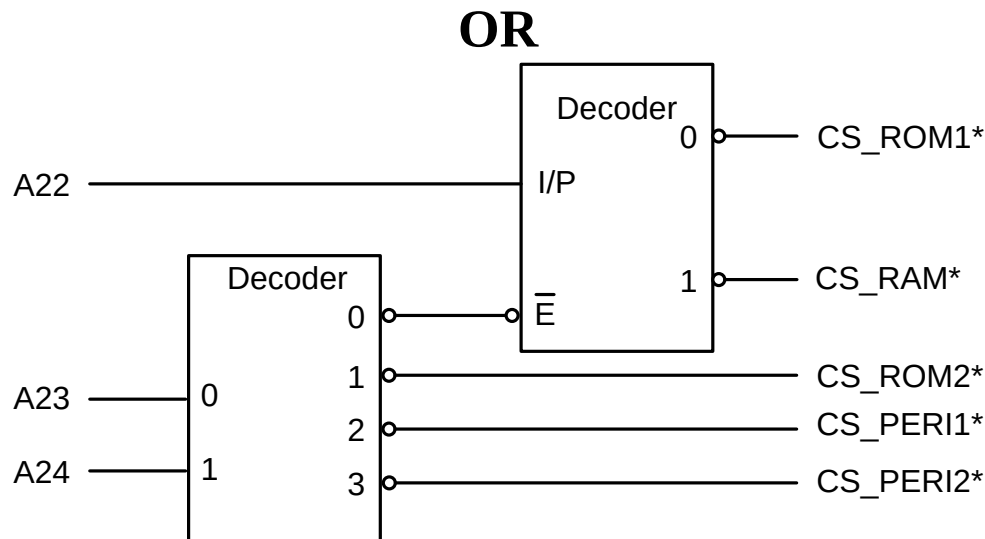
RAM: \$040 0000 - \$040 07FF

ROM2: \$080 0000 - \$080 1FFF

PERI1: \$100 0000 - \$100 0001

PERI2: \$180 0000 - \$180 0001





However, this decoding scheme cannot be expanded. Further memory devices cannot be added since the existing devices fill the entire 32 M-byte of memory space.

Alternative decoding scheme:

Device	ADDRESS LINES			
	24	23	22	21
ROM1	0	0	0	0
RAM	0	0	0	1
ROM2	0	0	1	
PERI1	0	1	0	
PERI2	0	1	1	
SPACE	1			

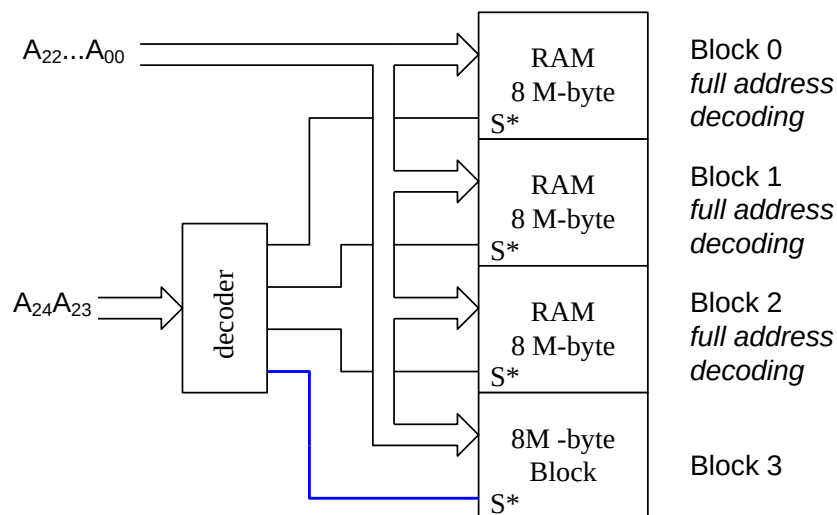
In this case, only the lower half 16 M-byte are occupied, leaving the upper half 16 M-byte address space for future expansion.

Disadvantage: partial decoding prevents full use of the available memory space and makes it difficult to expand the memory system.

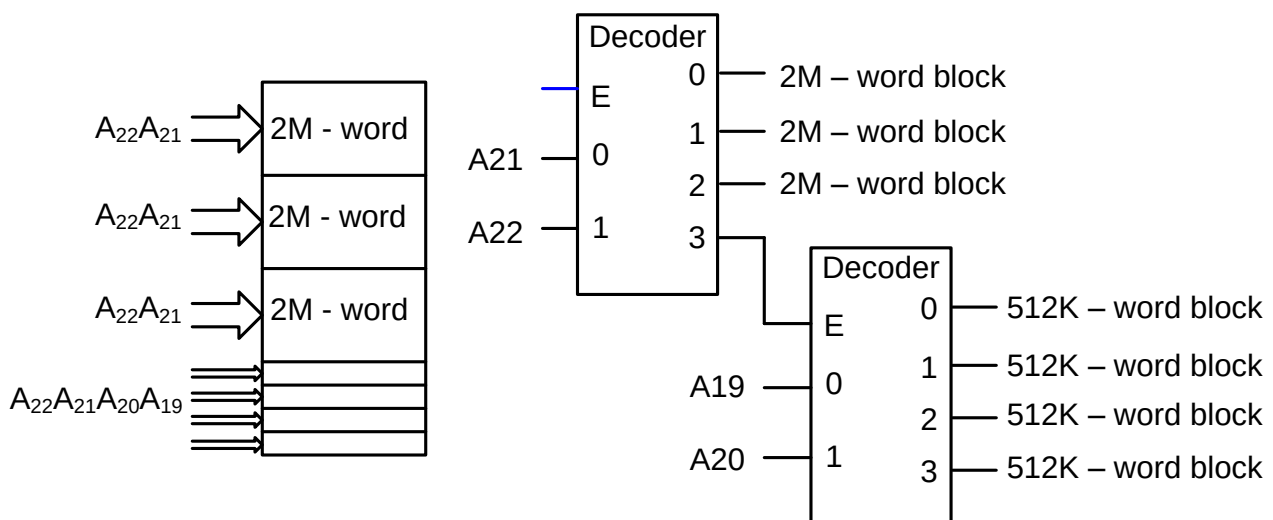
9.3.3 Block Address Decoding

Block address decoding is a compromise between full and partial address decoding. The memory space is divided into **a number of blocks**, for example, 4 blocks of 8 M-byte. The memory blocks may be decoded using full address decoding scheme.

In fact, each block can further be divided into smaller blocks. Then, partial address decoding can be applied to sufficiently small block for peripheral interfacing in order to avoid wastage of memory space.



Block 3 can further be divided into smaller as follows:



9.4 Memory Features of S3C44B0X

- Little/Big endian (selectable by an external pin)
- 25 physical address lines, ADDR0 to ADDR24.
- Fixed 16 data lines, DATA0-DATA15
- Additional 16 data lines (shared with functions) also available, DATA16-DATA31 port C
- 8 memory banks selected by nGCS₀₋₇
- Address space: 32 Mbytes per bank (total 256 Mbytes)
- Programmable access size (8/16/32-bit) for all banks
- 6 memory banks for ROM, SRAM etc.
- 2 memory banks for ROM, SRAM, FP/EDO/SDRAM etc.
- 7 fixed memory bank start address and programmable bank size
- 1 flexible memory bank start address and programmable bank size
- Programmable access cycles for all memory banks
- External wait to extend the bus cycles
- Supports self-refresh mode in DRAM/SDRAM for power-down

9.5 Hardware Signals for Memory Interface

The **hardware signal pins** of S3C44B0X for interfacing memory are listed below:

Signal	I/O	Description
BUS CONTROLLER		
OM[1:0]	I	OM[1:0] sets S3C44B0X in the TEST mode, which is used only at fabrication. Also, it determines the bus width of nGCS0. The logic level is determined by the pull-up/down resistor during the RESET cycle. 00:8-bit 01:16-bit 10:32-bit 11:Test mode
ADDR[24:0]	O	ADDR[24:0] (Address Bus) outputs the memory address of the corresponding bank.
DATA[31:0]	IO	DATA[31:0] (Data Bus) inputs data during memory read and outputs data during memory write. The bus width is programmable among 8/16/32-bit.
nGCS[7:0]	O	nGCS[7:0] (General Chip Select) are activated when the address of a memory is within the address region of each bank. The number of access cycles and the bank size can be programmed.
nWE	O	nWE (Write Enable) indicates that the current bus cycle is a write cycle.
nWBE[3:0]	O	Write Byte Enable
nBE[3:0]	O	Upper Byte/Lower Byte Enable(for 16-bit and 32-bit devices)
nOE	O	nOE (Output Enable) indicates that the current bus cycle is a read cycle.
nWAIT	I	nWAIT requests to prolong a current bus cycle. As long as nWAIT is L, the current bus cycle cannot be completed.
ENDIAN	I	It determines whether or not the data type is little endian or big endian. The logic level is determined by the pull-up/down resistor during the RESET cycle. 0:little endian 1:big endian
DRAM/SDRAM		
nRAS[1:0]	O	Row Address Strobe
nCAS[3:0]	O	Column Address strobe
nSRAS	O	SDRAM Row Address Strobe
nSCAS	O	SDRAM Column Address Strobe
nSCS[1:0]	O	SDRAM Chip Select
DQM[3:0]	O	SDRAM Data Mask
SCLK	O	SDRAM Clock
SCKE	O	SDRAM Clock Enable

- The data bus width of the system can be configured to be 8-bit, 16-bit or 32-bit.
- Generally speaking, most systems still use byte for quantifying the addressing space, even if the data bus width is 16-bit or 32-bit.
- 8 signal pins nGCS[7:0] are used to access 8 banks of memory space.
- Each bank consists of 32 Mbytes since 25 address lines ($A_0 - A_{24}$) are available ($2^{25} = 32 \text{ M}$).

- nWE, nWBE[3:0], nBE[3:0], nOE are read/write control signals for SRAM/ROM.
- nWAIT is to prolong the memory cycle for slow device.
- The rest of the signals are for DRAM and SDRAM.

9.6 Memory Map of S3C44B0X

nGCS[7:0] can be considered the decoded outputs of 3 extra **unseen** address lines A25, A26 and A27 (A25-A27 are **NOT** available from the CPU for external connection).

Address table:

Bank	nGCS[7:0]			Address lines (A24-A0)						
	A27	A26	A25	A24	A23-A20	A19-A16	A15-A12	A11-A8	A7-A4	A3-A0
0	0	0	0	0	0000	0000	0000	0000	0000	0000
	0	0	0	1	1111	1111	1111	1111	1111	1111
1	0	0	1	0	0000	0000	0000	0000	0000	0000
	0	0	1	1	1111	1111	1111	1111	1111	1111
2	0	1	0	0	0000	0000	0000	0000	0000	0000
	0	1	0	1	1111	1111	1111	1111	1111	1111
3	0	1	1	0	0000	0000	0000	0000	0000	0000
	0	1	1	1	1111	1111	1111	1111	1111	1111
4	1	0	0	0	0000	0000	0000	0000	0000	0000
	1	0	0	1	1111	1111	1111	1111	1111	1111
5	1	0	1	0	0000	0000	0000	0000	0000	0000
	1	0	1	1	1111	1111	1111	1111	1111	1111
6	1	1	0	0	0000	0000	0000	0000	0000	0000
	1	1	0	1	1111	1111	1111	1111	1111	1111
7	1	1	1	0	0000	0000	0000	0000	0000	0000
	1	1	1	1	1111	1111	1111	1111	1111	1111

Address range for each bank:

Bank 0: 0x0000 0000 – 0x01FF FFFF

Bank 1: 0x0200 0000 – 0x03FF FFFF

Bank 2: 0x0400 0000 – 0x05FF FFFF

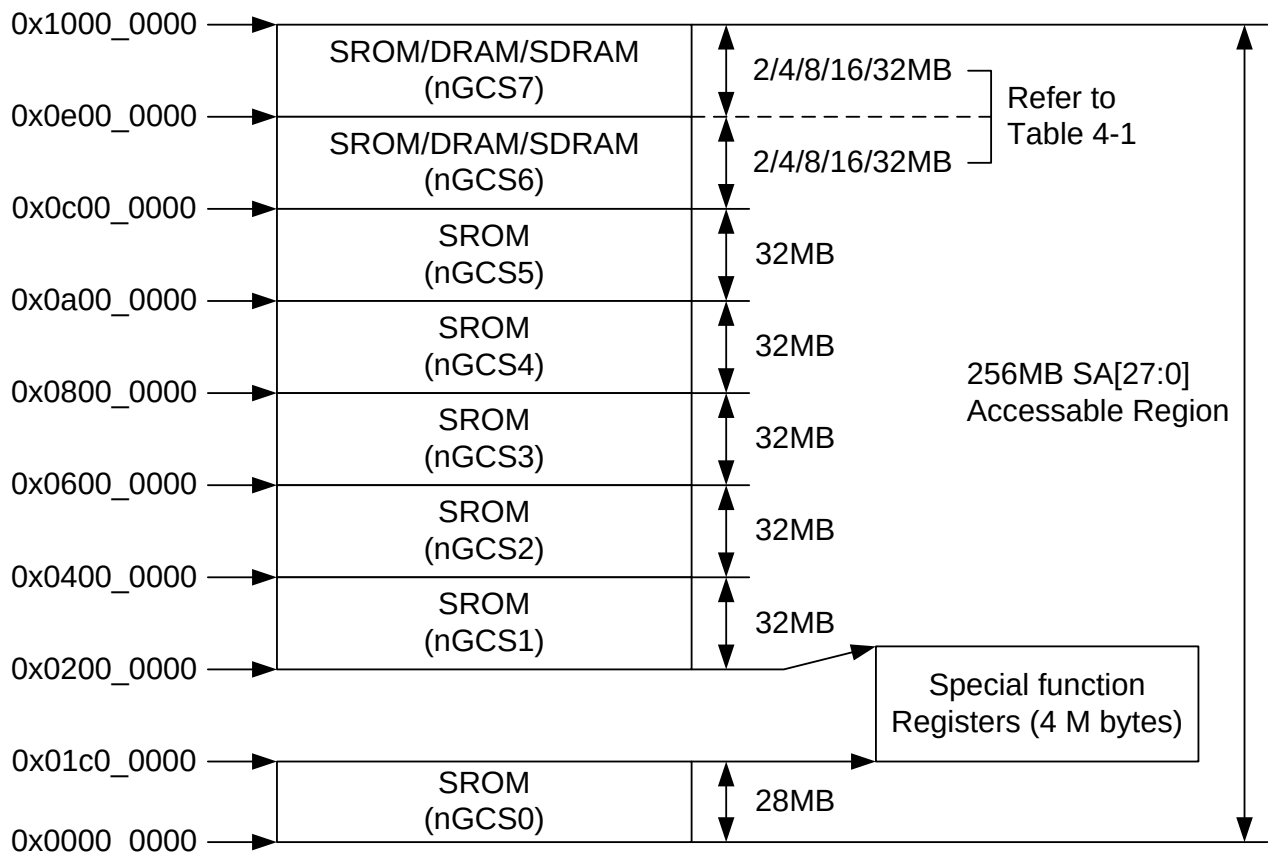
Bank 3: 0x0600 0000 – 0x07FF FFFF

Bank 4: 0x0800 0000 – 0x09FF FFFF

Bank 5: 0x0A00 0000 – 0x0BFF FFFF

Bank 6: 0x0C00 0000 – 0x0DFF FFFF

Bank 7: 0x0E00 0000 – 0x0FFF FFFF



S3C44B0X Memory Map after Reset
 (NOTE: SROM means ROM or SRAM type memory)

- 4 M bytes memory space in Bank 0 (**Boot Bank**) is used by the processor to address the special registers (memory-mapped I/O) related to the on-chip devices.
- Other hardware devices such as ROM, Flash, or peripheral I/O are normally connected to Banks 1 – 5.

Table 4-1. Bank 6/7 Address

Address	2MB	4MB	8MB	16MB	32MB
Bank 6					
Start address	0xc00_0000	0xc00_0000	0xc00_0000	0xc00_0000	0xc00_0000
End address	0xc1f_ffff	0xc3f_ffff	0xc7f_ffff	0xcff_ffff	0xdff_ffff
Bank 7					
Start address	0xc20_0000	0xc40_0000	0xc80_0000	0xd00_0000	0xe00_0000
End address	0xc3f_ffff	0xc7f_ffff	0xcff_ffff	0xdff_ffff	0xff_ffff

NOTE: Bank 6 and 7 must have the same memory size and the starting address of Bank 7 is the immediate address after the end address of Bank 6.

9.7 Hardware Setting

Little Endian /Big Endian

While nRESET is L (CPU is in reset state), the **hardware ENDIAN pin** defines which endian mode should be selected.

If the ENDIAN pin is connected to Vss with a pull-down resistor (logic “0”), the little endian mode is selected.

If the pin is connected to Vdd with a pull-up resistor (logic “1”), the big endian mode is selected.

ENDIAN Input @Reset	ENDIAN Mode
0	Little endian
1	Big endian

Bank0 Bus Width

The data bus width of BANK0 (nGCS0) should be configured as one of 8-bit, 16-bit and 32-bit. Because the BANK0 is the booting ROM bank (map to 0x0000_0000), the bus width of BANK0 should be determined before the first ROM access, which will be **determined by the logic level of OM[1:0] at Reset.**

OM1 (Operating Mode 1)	OM0 (Operating Mode 0)	Booting ROM Data width
0	0	8-bit
0	1	16-bit
1	0	32-bit
1	1	Test Mode

9.8 Memory Device

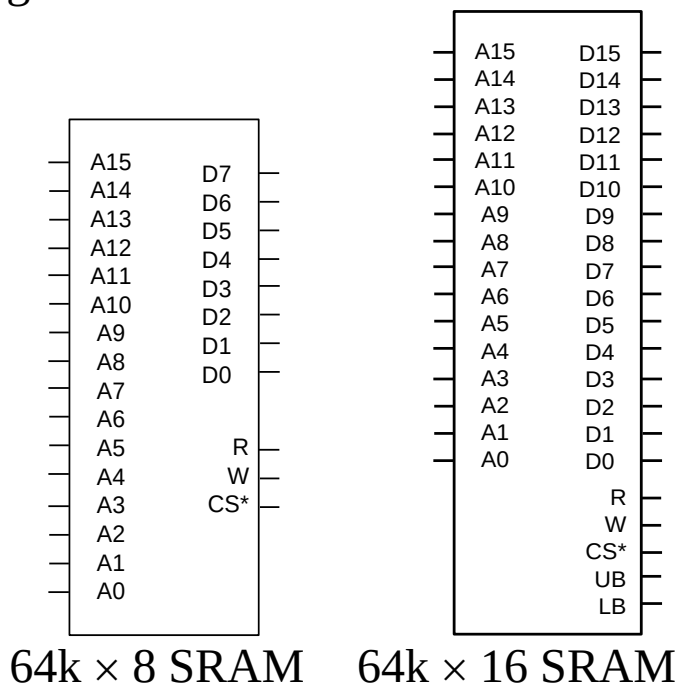
For the time being, let us consider SRAM (static random access memory) only. DRAM (dynamic RAM) and SDRAM (synchronous DRAM) will be considered in another chapter.

SRAM are available in 4-bit, 8-bit, 16-bit and 32-bit.

Device	Minimum number of device required		
	8-bit data bus	16-bit data bus	32-bit data bus
4-bit	2	4	8
8-bit	1	2	4
16-bit	1	1	2
32-bit	1	1	1

Important note: all data lines of the memory devices must be connected to the data lines of the CPU.

The following shows two SRAM:



$$2^6, k=10$$

9.9 Address Decoder Design

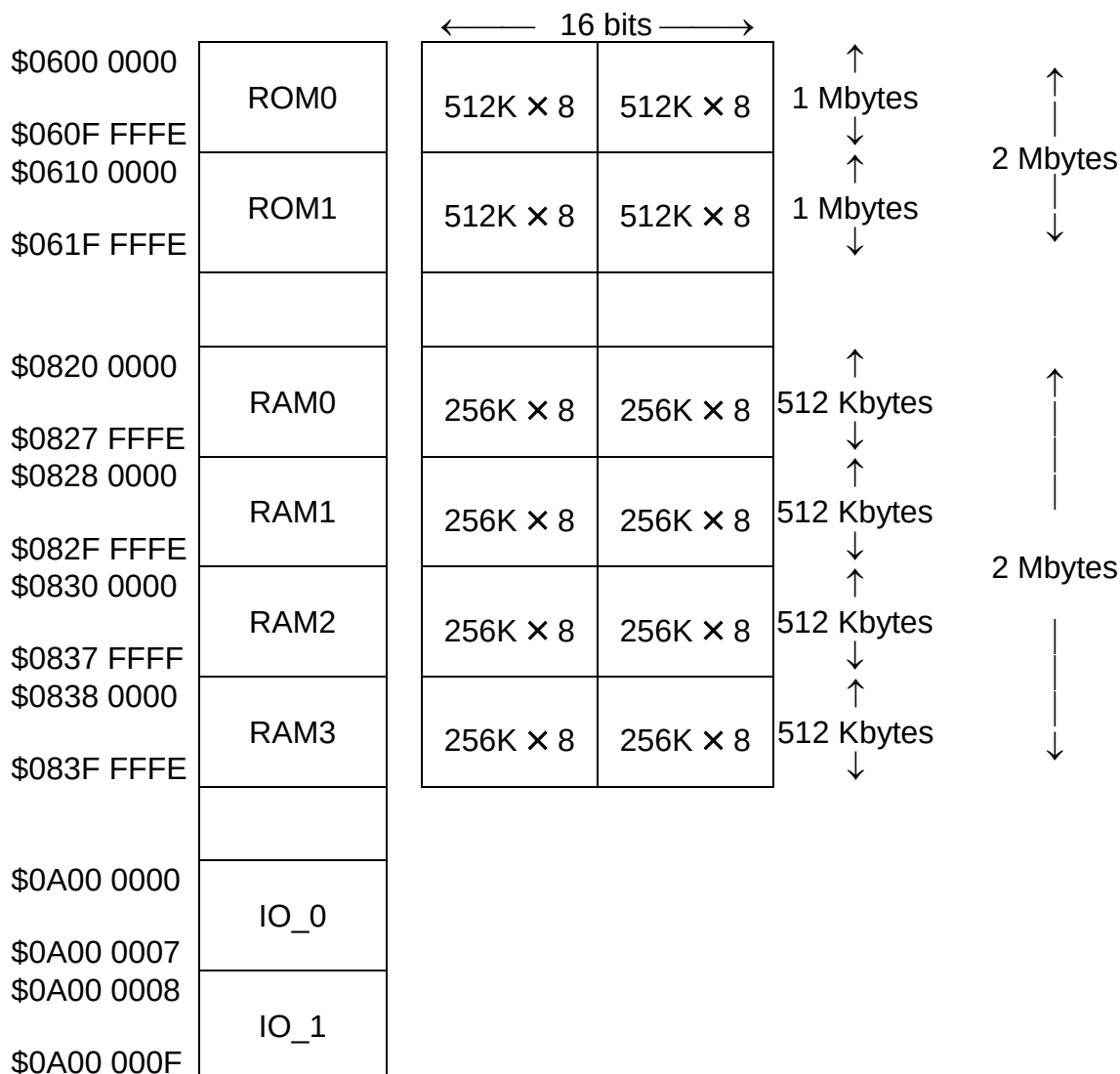
- ◆ Suppose we will design a memory system for S3C44B0X with 16-bit data bus width
- ◆ 2 Mbytes of ROM with starting address at \$0x0600 0000 using 512K × 8 chips
- ◆ 2 Mbytes of RAM with starting address at \$0x0820 0000 using 256K × 8 chips
- ◆ IO_0 from \$0x0A00 0000 to \$0x0A00 0007
- ◆ IO_1 from \$0x0A00 0008 to \$0x0A00 000F

Step 1: chip count

No. of ROM chips required: $2\text{M}/512\text{K} = 4$

No. of RAM chips required: $2\text{M}/256\text{K} = 8$

Step 2: memory map



Step 3: address decoding table

		Address lines										
Device	nGCS	24	23	22	21	20	19	18 - 04	03	02	01	00
ROM 0	3(011)	0	0	0	0	0	X	X	X	X	X	-
ROM 1	3(011)	0	0	0	0	1	X	X	X	X	X	-
RAM 0	4(100)	0	0	0	1	0	0	X	X	X	X	-
RAM 1	4(100)	0	0	0	1	0	1	X	X	X	X	-
RAM 2	4(100)	0	0	0	1	1	0	X	X	X	X	-
RAM 3	4(100)	0	0	0	1	1	1	X	X	X	X	-
IO_0	5(101)	0	0	0	0	0	0	0	0	X	X	X
IO_1	5(101)	0	0	0	0	0	0	0	1	X	X	X

Note that for 16-bit data bus system, address line **A00** is not connected since memory access must be in even memory boundary.

Likewise if the data bus system is 32-bit, address lines **A00 and A01** are not connected since memory access must be aligned with 4-byte memory boundary.

Step 4: full decoding logic

ROM 0: $\overline{\text{nGCS}}_3 \overline{A_{24}} \overline{A_{23}} \overline{A_{22}} \overline{A_{21}} \overline{A_{20}}$

ROM 1: $\overline{\text{nGCS}}_3 \overline{A_{24}} \overline{A_{23}} \overline{A_{22}} \overline{A_{21}} \overline{A_{20}}$

RAM 0: $\overline{\text{nGCS}}_4 \overline{A_{24}} \overline{A_{23}} \overline{A_{22}} \overline{A_{21}} \overline{A_{20}} \overline{A_{19}}$

RAM 1: $\overline{\text{nGCS}}_4 \overline{A_{24}} \overline{A_{23}} \overline{A_{22}} \overline{A_{21}} \overline{A_{20}} \overline{A_{19}}$

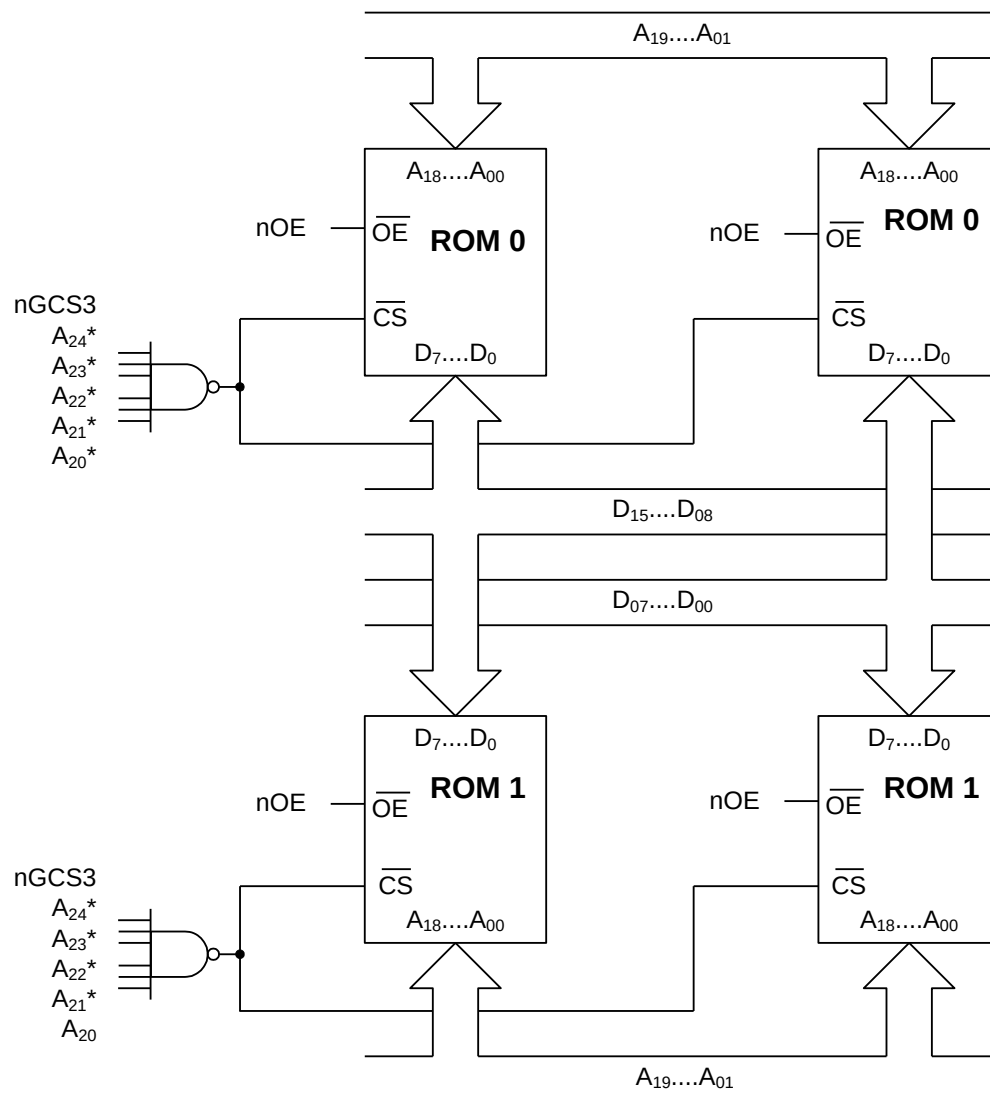
RAM 2: $\overline{\text{nGCS}}_4 \overline{A_{24}} \overline{A_{23}} \overline{A_{22}} \overline{A_{21}} \overline{A_{20}} \overline{A_{19}}$

RAM 3: $\overline{\text{nGCS}}_4 \overline{A_{24}} \overline{A_{23}} \overline{A_{22}} \overline{A_{21}} \overline{A_{20}} \overline{A_{19}}$

IO_0: $\overline{\text{nGCS}}_5 \overline{A_{24}} \overline{A_{23}} \overline{A_{22}} \dots \overline{A_5} \overline{A_{04}} \overline{A_{03}}$

IO_1: $\overline{\text{nGCS}}_5 \overline{A_{24}} \overline{A_{23}} \overline{A_{22}} \dots \overline{A_5} \overline{A_{04}} \overline{A_{03}}$

Step 5: Circuit implementation



Decoding circuit for ROM0 & ROM1

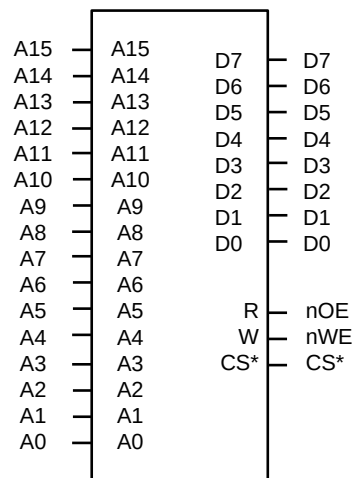
9.9.1 More Design Considerations

When designing memory system, the address lines' connections between CPU and memory chip are dependent on the number of CPU data bus width used and the bit-size of the memory chips.

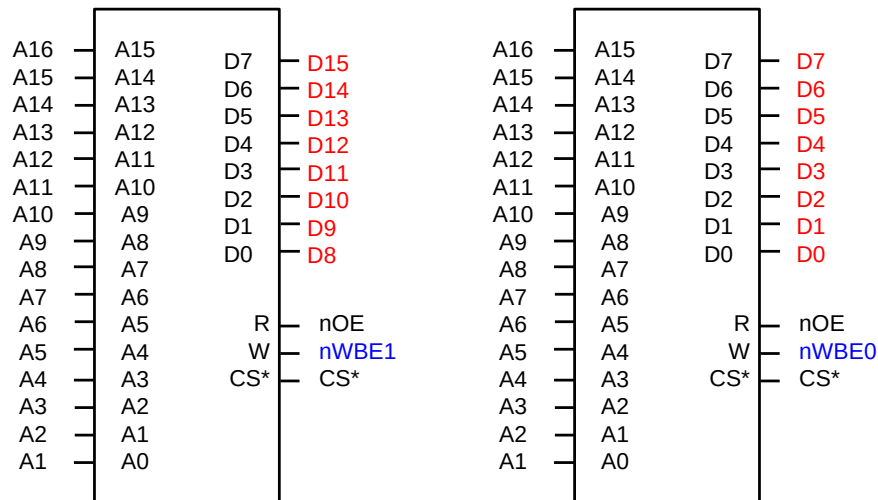
MEMORY (ROM/DRAM/SDRAM) ADDRESS PIN CONNECTIONS

MEMORY ADDR. PIN	S3C44B0X ADDR. @ 8-bit DATA BUS	S3C44B0X ADDR. @ 16-bit DATA BUS	S3C44B0X ADDR. @ 32-bit DATA BUS
A0	A0	A1	A2
A1	A1	A2	A3
A2	A2	A3	A4
A3	A3	A4	A5
...	...	...	...

For an 8-bit memory system with 8-bit memory device:



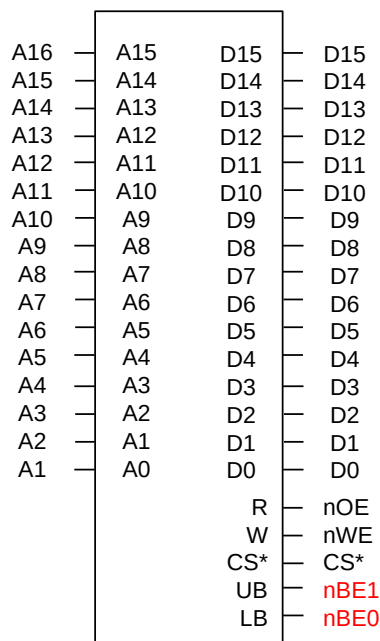
For 16-bit memory system with 8-bit memory devices:



16 -> 8 / 16
WBE0 -> A0 = 0
WBE1 -> A0 = 1

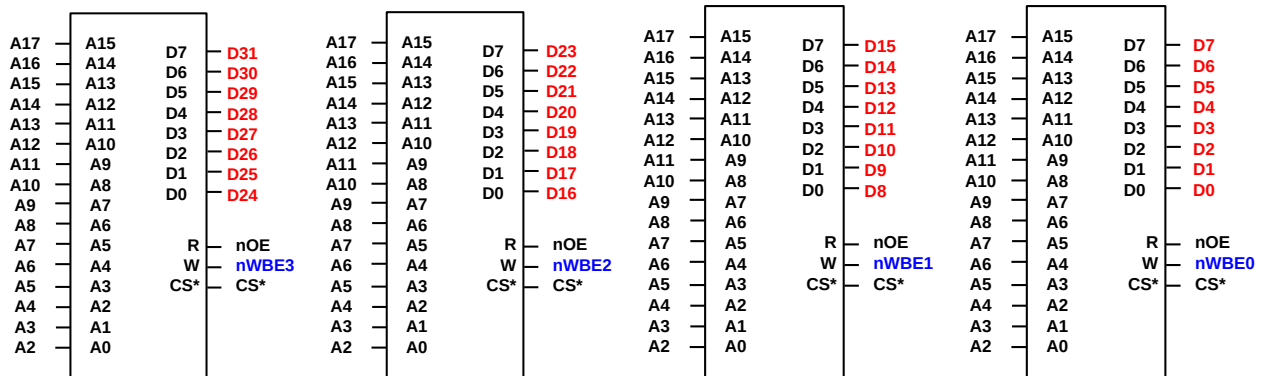
For 8-bit memory chip, nWBE0/1 to identify the byte data should be written into

For 16-bit memory system with 16-bit memory device:



For 16-bit memory chip, use nBE0/1 to identify the lower/upper byte inside the chip for R/W operation

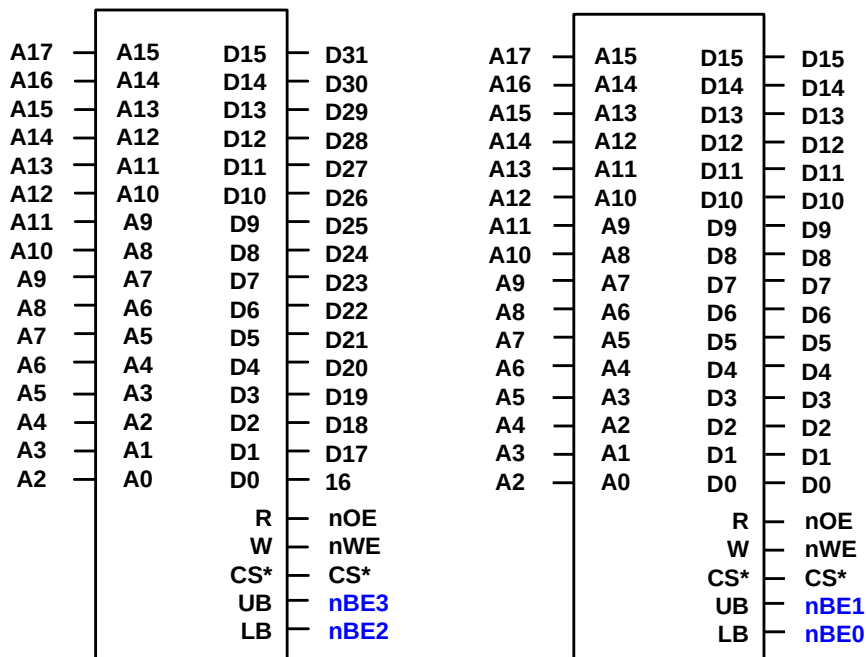
For 32-bit memory system with 8-bit memory devices:



For 8-bit memory chip, nWBE0/1/2/3 to identify the byte data should be written into

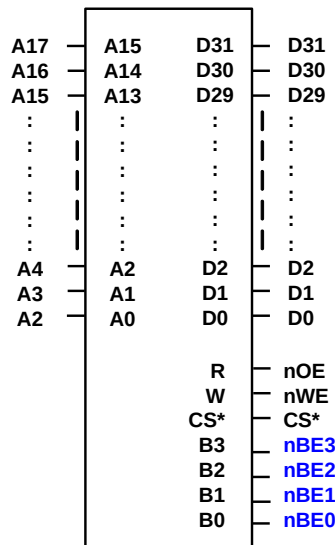
32 -> 8 / 16 / 32
 WBE0 -> A0A1 = 00
 WBE1 -> A0A1 = 01
 WBE2 -> A0A1 = 10
 WBE3 -> A0A1 = 11

For 32-bit memory system with 16-bit memory devices:



For 16-bit memory chip, use nBE0/1/2/3 to identify the lower/upper byte inside the chip for R/W operation

For 32-bit memory system with 32-bit memory device:



For 32-bit memory chip, use nBE0/1/2/3 to identify the lower/upper byte inside the chip for R/W operation

9.9.2 Another Example

A S3C44B0X system with 16-bit data bus width consists of the following memory components:

1M bytes of ROM (using 256K × 8 bits chips) with starting address \$000 0000

1M bytes of RAM0 (using 256K × 4 bits chips) with starting address \$200 0000

2M bytes of RAM1 (using 512K × 16 bits chips with starting address \$400 0000

8 16-bit PERI with starting address \$800 0000

Step 1: chip count

No. of ROM chips required: $1\text{M}/256\text{K} = 4$

No. of RAM0 chips required: $2 \times 1\text{M}/256\text{K} = 8$

No. of RAM1 chips required: $0.5 \times 2\text{M}/512\text{K} = 2$

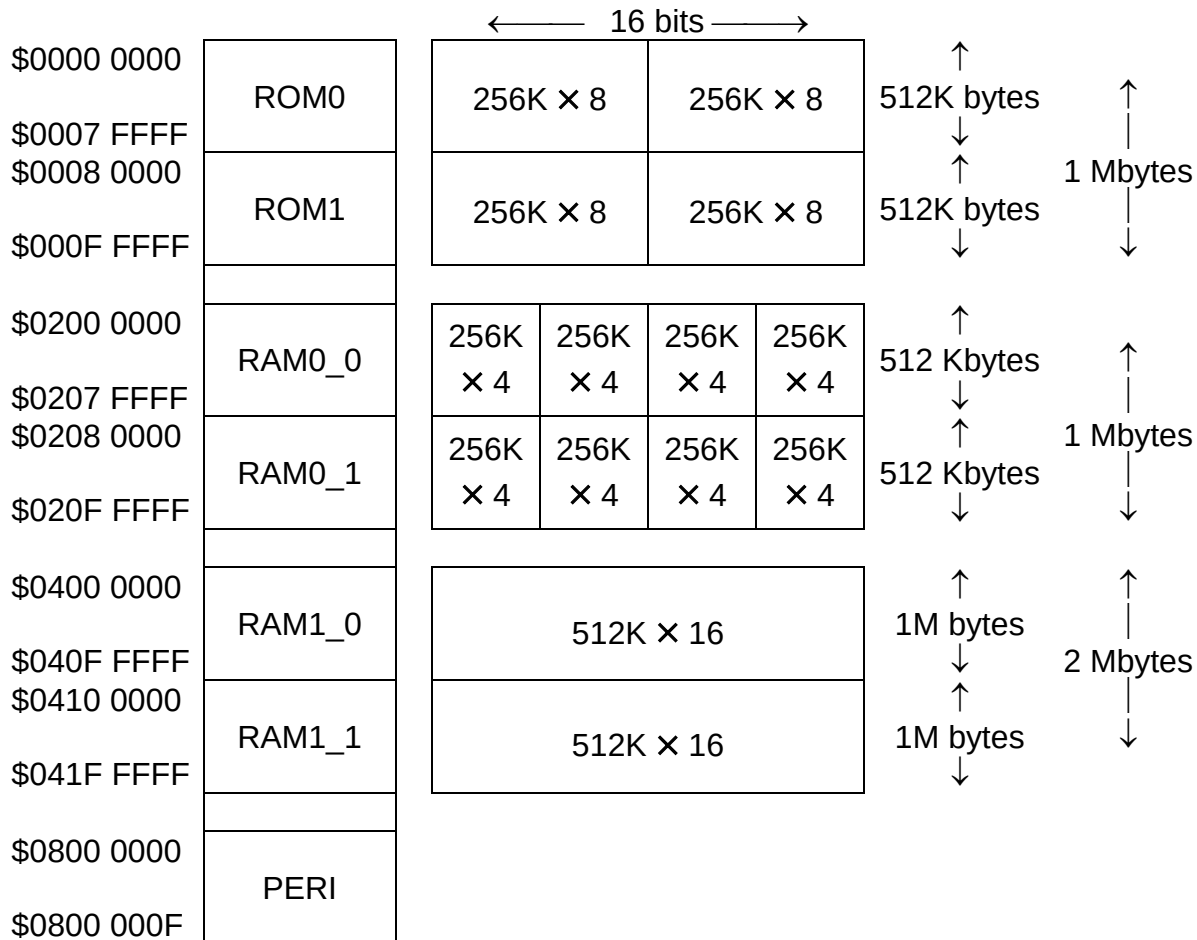
1 byte = 2 x 4 bits

Step 2: memory map

16-bit data bus width → **TWO** 8-bit chips per CS signal

16-bit data bus width → **FOUR** 4-bit chips per CS signal

16-bit data bus width → **ONE** 16-bit chips per CS signal



Step 3: address decoding table

		Address lines										
Device	nGCS	24	23	22	21	20	19	18 - 04	03	02	01	00
ROM 0	0(000)	0	0	0	0	0	0	X	X	X	X	-
ROM 1	0(000)	0	0	0	0	0	1	X	X	X	X	-
RAM0_0	1(001)	0	0	0	0	0	0	X	X	X	X	-
RAM0_1	1(001)	0	0	0	0	0	1	X	X	X	X	-
RAM1_0	2(010)	0	0	0	0	0	X	X	X	X	X	-
RAM1_1	2(010)	0	0	0	0	1	X	X	X	X	X	-
PERI	4(100)	0	0	0	0	0	0	0	X	X	X	-

Step 4: full decoding logic

$$\text{ROM0_CS} = \text{nGCS0} \overline{A_{24}} \overline{A_{23}} \overline{A_{22}} \overline{A_{21}} \overline{A_{20}} \overline{A_{19}}$$

$$\text{ROM1_CS} = \text{nGCS0} \overline{A_{24}} \overline{A_{23}} \overline{A_{22}} \overline{A_{21}} \overline{A_{20}} \overline{A_{19}}$$

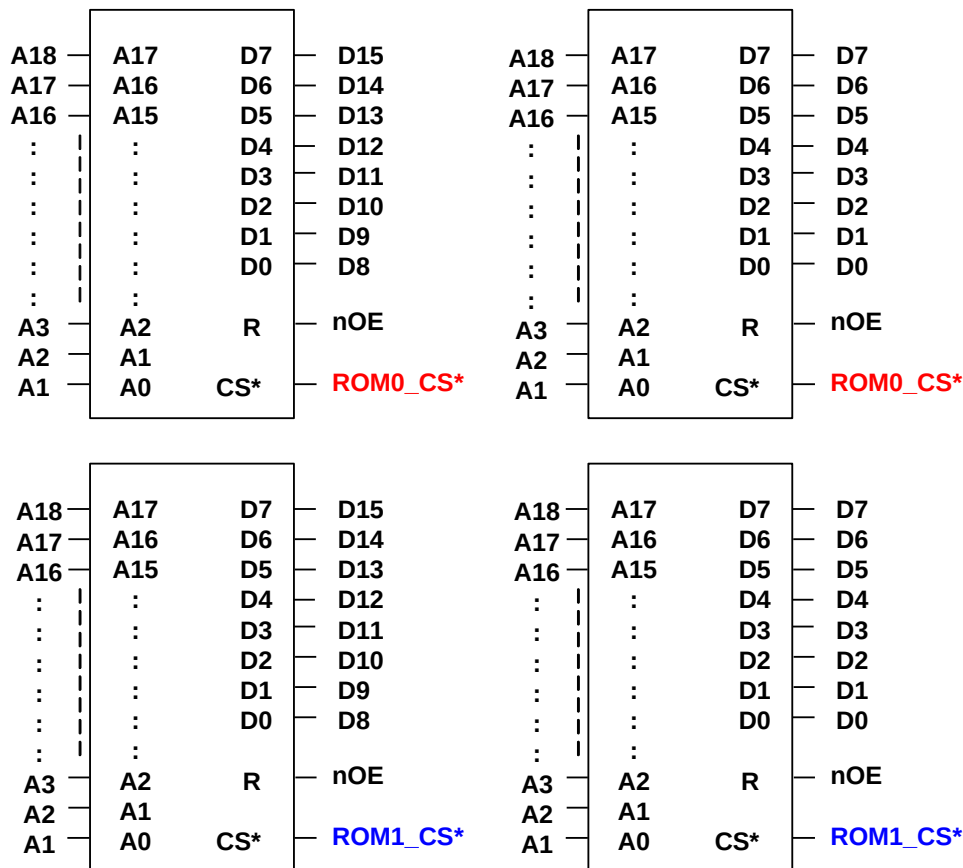
$$\text{RAM0_0_CS} = \text{nGCS1} \overline{A_{24}} \overline{A_{23}} \overline{A_{22}} \overline{A_{21}} \overline{A_{20}} \overline{A_{19}}$$

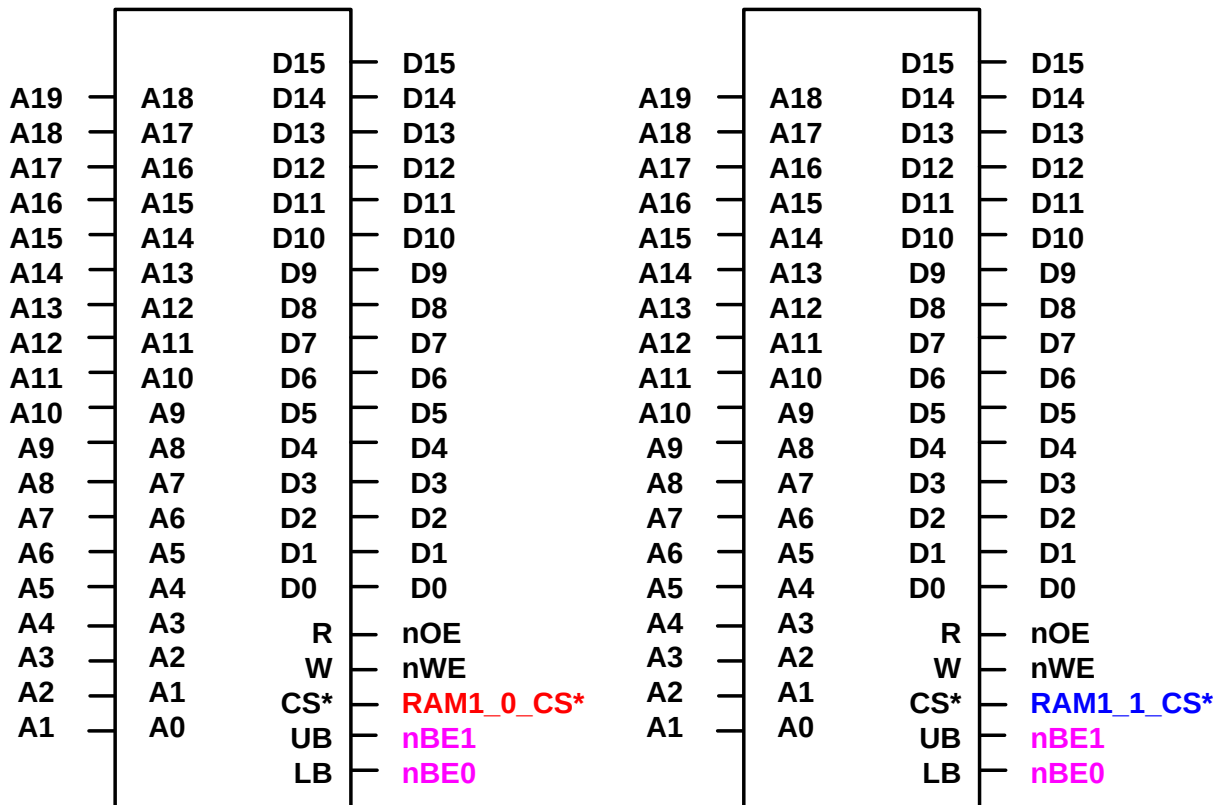
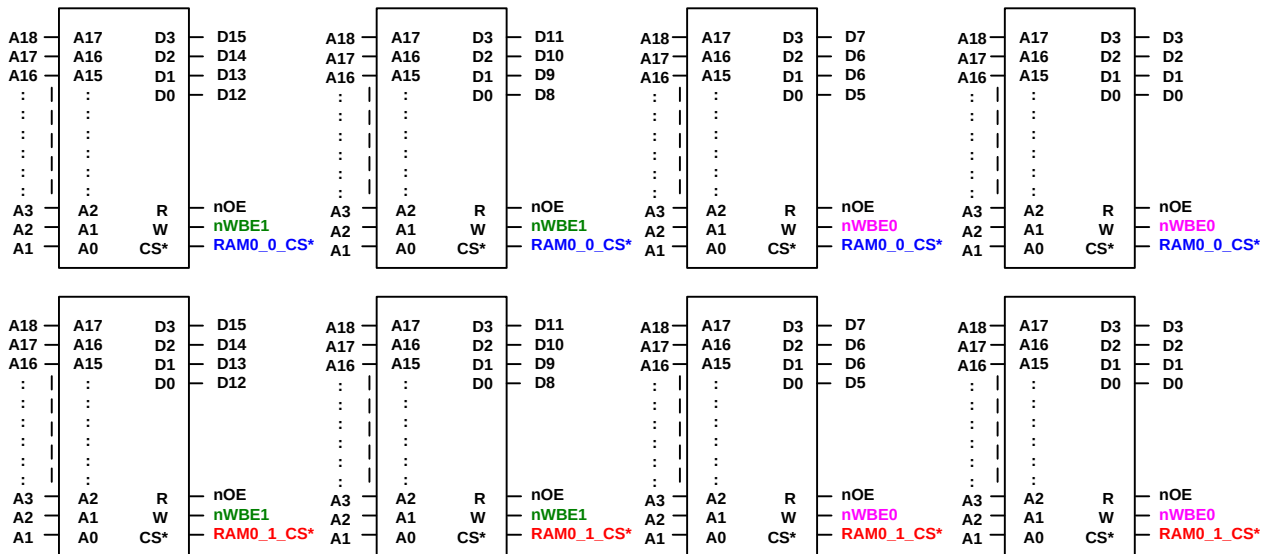
$$\text{RAM0_1_CS} = \text{nGCS1} \overline{A_{24}} \overline{A_{23}} \overline{A_{22}} \overline{A_{21}} \overline{A_{20}} \overline{A_{19}}$$

$$\text{RAM1_0_CS} = \text{nGCS2} \overline{A_{24}} \overline{A_{23}} \overline{A_{22}} \overline{A_{21}} \overline{A_{20}}$$

$$\text{RAM1_1_CS} = \text{nGCS2} \overline{A_{24}} \overline{A_{23}} \overline{A_{22}} \overline{A_{21}} \overline{A_{20}}$$

$$\text{PERI_CS} = \text{nGCS4} \overline{A_{24}} \overline{A_{23}} \overline{A_{22}} \overline{A_{21}} \overline{A_{20}} \overline{A_{19}} \cdots \overline{A_6} \overline{A_5} \overline{A_4}$$





9.10 Timing

In order to successfully write into or read from memory device, it is required to select memory device with appropriate timing characteristics that satisfy the CPU timing requirement.

For read cycle, the CPU timing will be dominated since CPU must read the data from memory devices successfully.

For write cycle, the memory device timing is important since the data must be written to memory successfully.

In nowadays technology, the timing requirement can easily be satisfied by programming the corresponding registers with appropriate values, instead of choosing the appropriate devices or adjusting the CPU clock frequency to fulfill delay requirement.

Detail setting of the memory controller for various memory devices will be discussed in next Chapter.