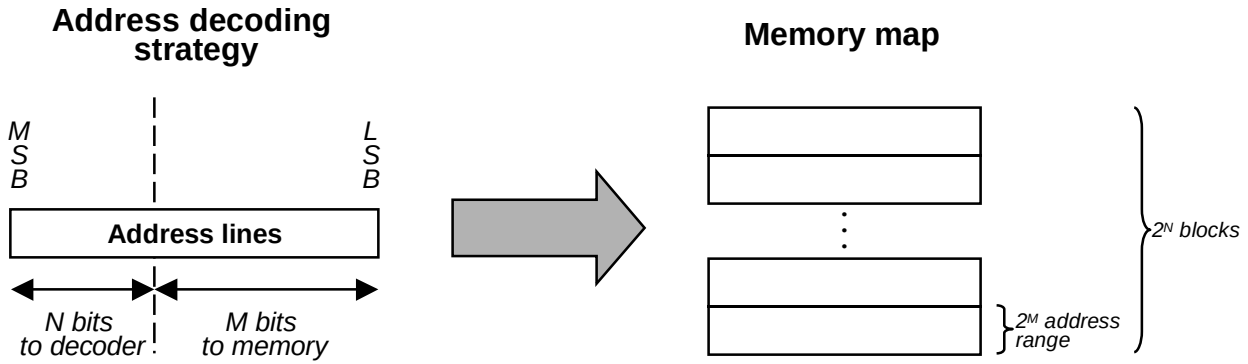
Lecture 16: Address decoding

- **Introduction to address decoding**
- **Full address decoding**
- **Partial address decoding**
- **Implementing address decoders**
- **Examples**

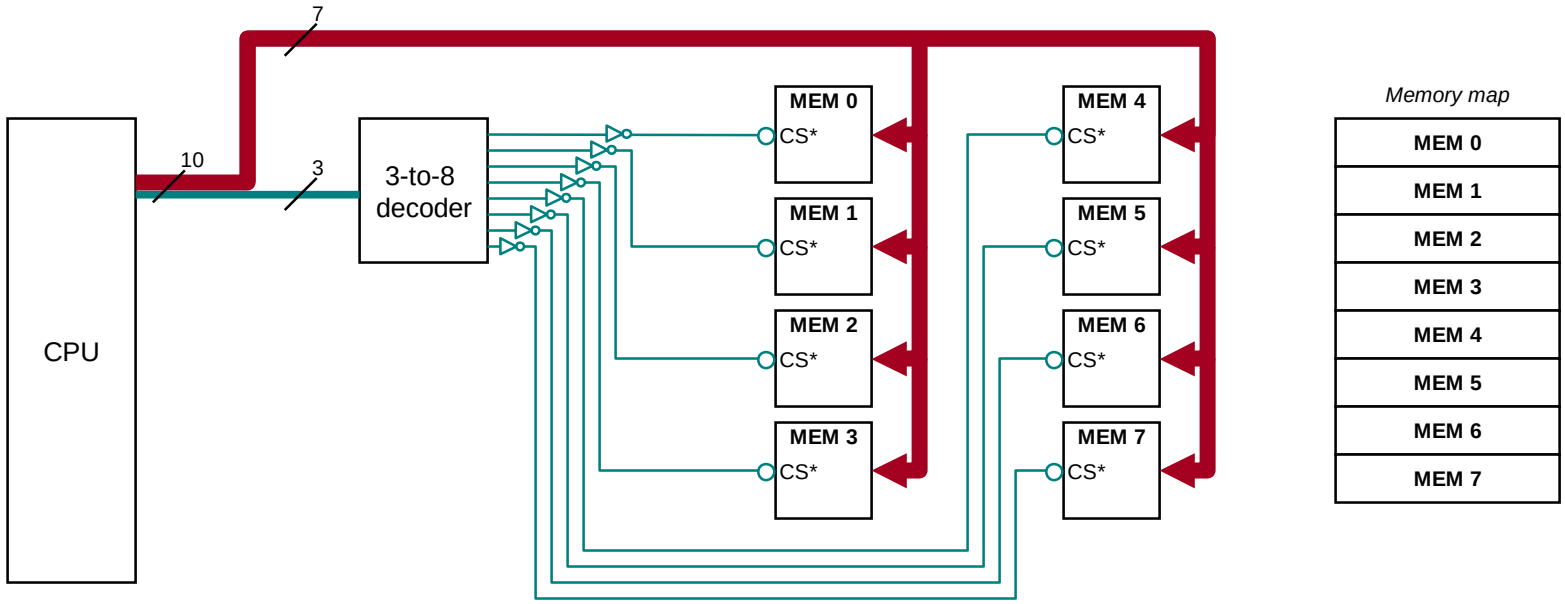
Introduction to address decoding

- **Although the memory space in the 68000 is said to be flat, it does not mean that the physical implementation of memory is homogeneous**
  - Different portions of memory are used for different purposes: RAM, ROM, I/O devices
  - Even if all the memory was of one type, we still have to implement it using multiple ICs
  - This means that for a given valid address, one and only one memory-mapped component must be accessed
- **Address decoding is the process of generating chip select (CS*) signals from the address bus for each device in the system**
- **The address bus lines are split into two sections**
  - the N most significant bits are used to generate the CS* signals for the different devices
  - the M least significant signals are passed to the devices as addresses to the different memory cells or internal registers



A very simple example

- **Let's assume a very simple microprocessor with 10 address lines (1KB memory)**
- **Let's assume we wish to implement all its memory space and we use 128x8 memory chips**
- **SOLUTION**
  - We will need 8 memory chips (8x128=1024)
  - We will need 3 address lines to select each one of the 8 chips
  - Each chip will need 7 address lines to address its internal memory cells



Address decoding methods

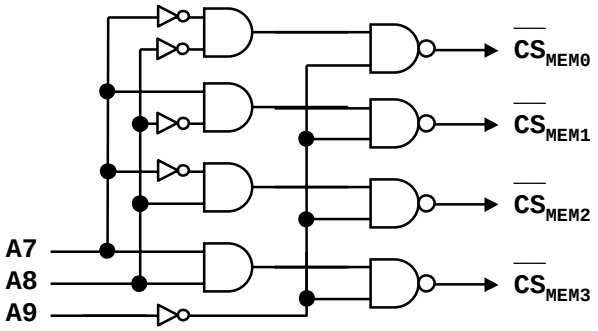
- **The previous example specified that all addressable memory space was to be implemented but**
  - There are some situations where this requirement is not necessary or affordable
- **If only a portion of the addressable space is going to be implemented there are two basic address decoding strategies**
  - Full address decoding
    - All the address lines are used to specify a memory location
    - Each physical memory location is identified by a unique address
  - Partial address decoding
    - Since not all the address space is implemented, only a subset of the address lines are needed to point to the physical memory locations
    - Each physical memory location is identified by several possible addresses (using all combinations of the address lines that were not used)

Full address decoding

- **Let's assume the same microprocessor with 10 address lines (1KB memory)**
  - However, this time we wish to implement only 512 bytes of memory
  - We still must use 128-byte memory chips
  - Physical memory must be placed on the upper half of the memory map
- **SOLUTION**

Used for Address Decoding — *Used to reference memory cells on each memory IC*

Device	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
MEM 0	0	0	0	X	X	X	X	X	X	X
MEM 1	0	0	1	X	X	X	X	X	X	X
MEM 2	0	1	0	X	X	X	X	X	X	X
MEM 3	0	1	1	X	X	X	X	X	X	X



Memory map

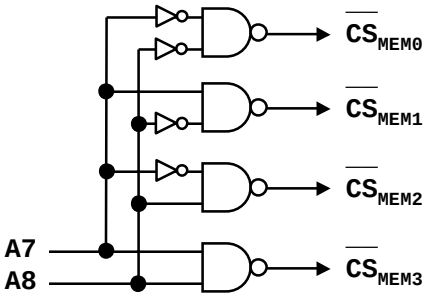
MEM 0
MEM 1
MEM 2
MEM 3
Not used
Not used
Not used
Not used

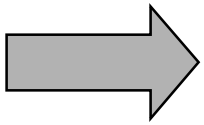
Partial address decoding

- **Let's assume the same microprocessor with 10 address lines (1KB memory)**
  - However, this time we wish to implement only 512 bytes of memory
  - We still must use 128-byte memory chips
  - Physical memory must be placed on the upper half of the memory map
- **SOLUTION**

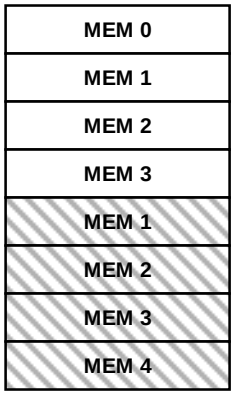
Not used — *Used for Address Decoding* — *Used to reference memory cells on each memory IC*

Device	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
MEM 0	X	0	0	X	X	X	X	X	X	X
MEM 1	X	0	1	X	X	X	X	X	X	X
MEM 2	X	1	0	X	X	X	X	X	X	X
MEM 3	X	1	1	X	X	X	X	X	X	X



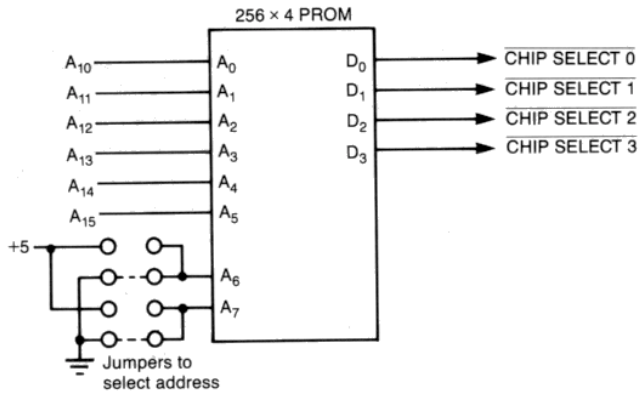


Memory map



Implementing address decoders

- **Discrete logic**
  - High speed (propagation signals)
  - High chip-count
  - Lacks flexibility
- **Data decoders**
  - More appropriate than random logic
  - The selection of devices is determined by the physical wiring
  - All the memory blocks must have the same size
- **Programmable Read Only Memory (PROM)**
  - Versatile, since the selection of devices is determined by the programming
  - Memory blocks can be of different sizes
  - Lookup tables can become very large for more than 8 address lines
- **Other methods (beyond the scope of the lecture) are**
  - Field Programmable Gate Arrays (FPGA) and
  - Programmable Address Decoders



Address	Contents	Address range	Block size
000000	1110	0000–03FF	1K
000001	1111	0400–07FF	1K, unused
000010	1101	0800–0FFF	2K
000011	1101		
000100	1011	1000–1FFF	4K
000101	1011		
000110	1011		
000111	1011		
001000	0111	2000–3FFF	8K
001001	0111		
001010	0111		
001011	0111		
001100	0111		
001101	0111		
001110	0111		
001111	0111		
010000	1111	4000–FFFF	48K, unused
.	.		
.	.		
.	.		
111111	1111		

Example 1

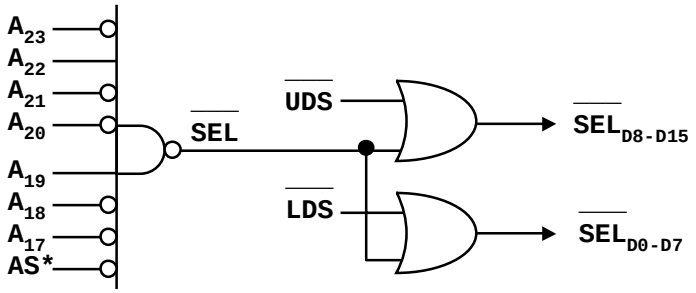
- **A circuit containing 64K words of RAM is to be interfaced to a 68000-based system, so that the first address of RAM (the *base* address) is at $480000**
  - What is the entire range of RAM addresses?
  - Design a FULL address decoder using two 64K×8 RAM ICs
- **Solution**
  - The address range for the RAM is from $480000 to $480000+(128K=$20000)=$4A0000-1=$49FFFF
  - The two ICs must be differentiated through UDS*/LDS* (since the 68000 DOES NOT have A_0)

4				8 or 9				0 to F				0 to F				0 to F				0 to F			
A_{23}	A_{23}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17}	A_{16}	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
0	1	0	0	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

These 7 address lines set the base address of the memory

These 16 address lines will select one of the 2^{16} (64K) locations inside each RAM IC

This address line is implemented with UDS/LDS**

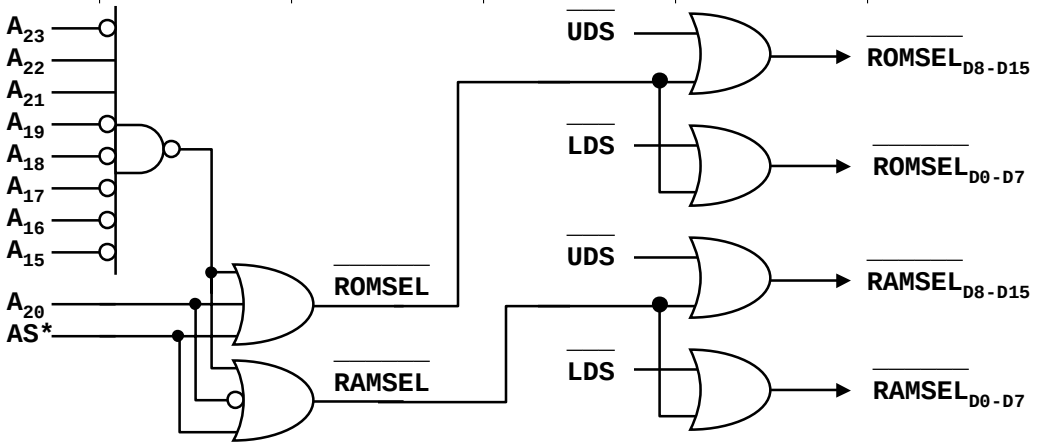


Example 2

- **A 68000-based system is to be built with these memory requirements**
  - a 16K word EPROM with a starting address of $60 0000
  - a 16K word RAM with a starting address of $70 0000
- **Design a FULL address decoder for this application using 16K×8 chips for both EPROM and RAM**
  - $60 0000 + (16KWord=32KB=$8000)-1=$60 7FFF
  - $70 0000 + (16KWord=32KB=$8000)-1=$70 7FFF

	6				0				0 to 7				0 to F				0 to F				0 to F			
ROM	A_{23}	A_{22}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17}	A_{16}	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
	0	1	1	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

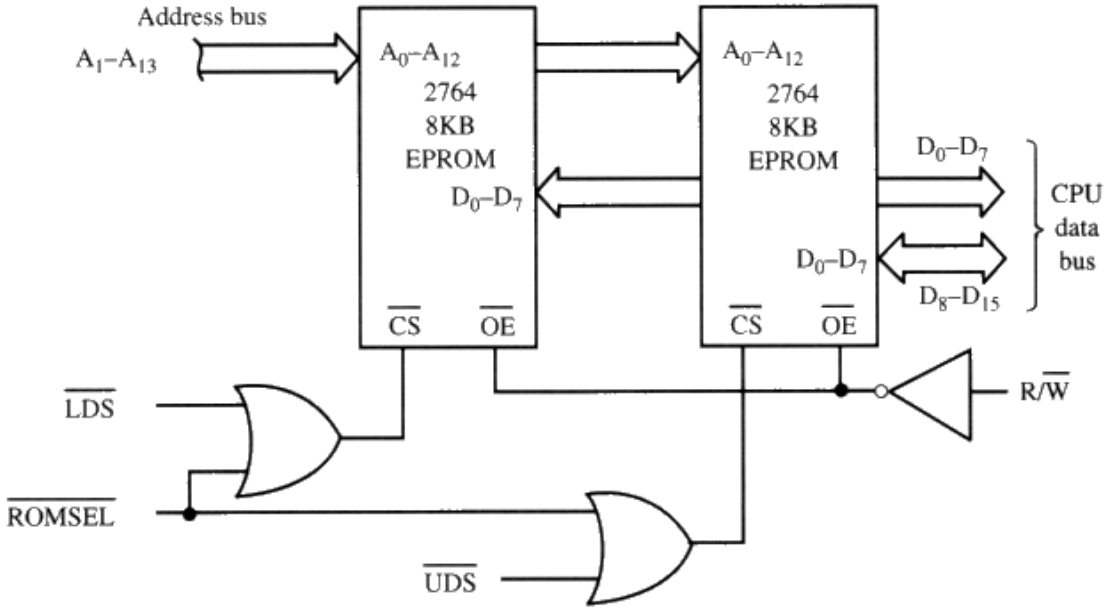
	7				0				0 to 7				0 to F				0 to F				0 to F			
RAM	A_{23}	A_{22}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
	0	1	1	1	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X



Example 3

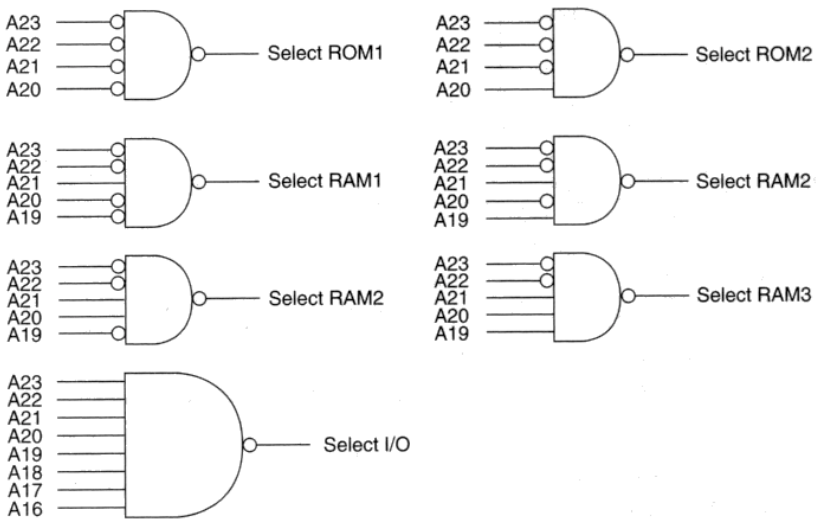
- **Design a PARTIAL address decoder for a 68000-based system with only 8K words of EPROM space, and a base address at $4000, using 8Kx8 memory chips**

0				0				4 to 7				0 to F				0 to F				0 to F			
A_{23}	A_{23}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17}	A_{16}	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
0	0	0	0	0	0	0	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X



Example 4

- **Design a partial address decoder for a 68000-based system that contains**
  - 2MB of EPROM at a starting address $00 0000 using 512Kx8 chips
  - 2MB of RAM at a starting address $10 0000 using 256Kx8 chips
  - 64KB I/O space starting at $FF0000
- **SOLUTION**
  - For the EPROM we will need 4 512Kx8 chips, organized as 2 pairs of 512x8 chips (in order to use UDS*/LDS*). We will call these pairs ROM1 and ROM2
  - For the RAM we will need 8 256Kx8 chips, organized as 4 pairs of 256Kx8: RAM1 to RAM4



	A_{23}	A_{22}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{00}
ROM1	0	0	0	0	X	X	X	X	X	X	X	X	X	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	X	X	X	X	X	X	X
RAM1	0	0	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RAM2	0	0	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RAM3	0	0	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RAM4	0	0	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
I/O	1	1	1	1	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X