Lecture 2: More on Memory and I/O Modules

Prof. Xiangzhong FANG
xzfang@sjtu.edu.cn

William Stallings
Computer Organization and Architecture

Chapter 4

Internal Memory

Characteristics

- Location
- Capacity
- Unit of transfer
- Access method
- Performance
- Physical type
- Organisation

Location

- CPU
  - registers
- Internal
  - Cache, main memory
- External
  - Disk, tape, dvd

Capacity

- Word size
  - The natural unit of organisation
- Number of words
  - or Bytes
- E.g., 2M 8-bit, 16M 1-bit
  - Same capacity but different organization

Unit of Transfer

- Internal
  - Usually governed by data bus width
- External
  - Usually a block which is much larger than a word
- Addressable unit
  - Smallest location which can be uniquely addressed
  - Word internally
  - Cluster on disks

Access Methods (1)

- Sequential
  - Start at the beginning and read through in order
  - Access time depends on location of data and previous location
  - e.g. tape
- Direct
  - Individual blocks have unique address
  - Access is by jumping to vicinity plus sequential search
  - Access time depends on location and previous location
  - e.g. disk

Access Methods (2)

- Random
  - Individual addresses identify locations exactly
  - Access time is independent of location or previous access
  - e.g. RAM
- Associative
  - Data is located based on a portion of its contents rather than its address
  - Access time is independent of location or previous access
  - e.g. cache

Memory Hierarchy

- Registers
  - In CPU
- Internal or Main memory
  - May include one or more levels of cache
  - “RAM”
- External memory
  - Backing store

Performance

- Access time
  - Time between presenting the address and getting the valid data
- Memory Cycle time
  - Time may be required for the memory to “recover” before next access
  - Cycle time is access + recovery
- Transfer Rate
  - Rate at which data can be moved

Physical Types

- Semiconductor
  - RAM
- Magnetic
  - Disk & Tape
- Optical
  - CD & DVD
- Others
  - Bubble
  - Hologram

Organisation

- Physical arrangement of bits into words
- Not always obvious
- e.g. interleaved

Hierarchy List

- Registers
- L1 Cache
- L2 Cache
- Main memory
- Disk cache
- Disk
- Optical
- Tape

So you want fast?

- It is possible to build a computer which uses only static RAM (see later)
- This would be very fast
- This would need no cache
  - How can you cache cache?
- This would cost a very large amount

Semiconductor Memory

- RAM
  - Misnamed as all semiconductor memory is random access
  - Read/Write
  - Volatile
  - Temporary storage
  - Static or dynamic

Dynamic RAM

- Bits stored as charge in capacitors
- Charges leak
- Need refreshing even when powered
- Simpler construction
- Smaller per bit
- Less expensive
- Need refresh circuits
- Slower
- Main memory

Static RAM

- Bits stored as on/off switches
- No charges to leak
- No refreshing needed when powered
- More complex construction
- Larger per bit
- More expensive
- Does not need refresh circuits
- Faster
- Cache

Read Only Memory (ROM)

- Permanent storage
- Microprogramming (see later)
- Library subroutines
- Systems programs (BIOS)
- Function tables

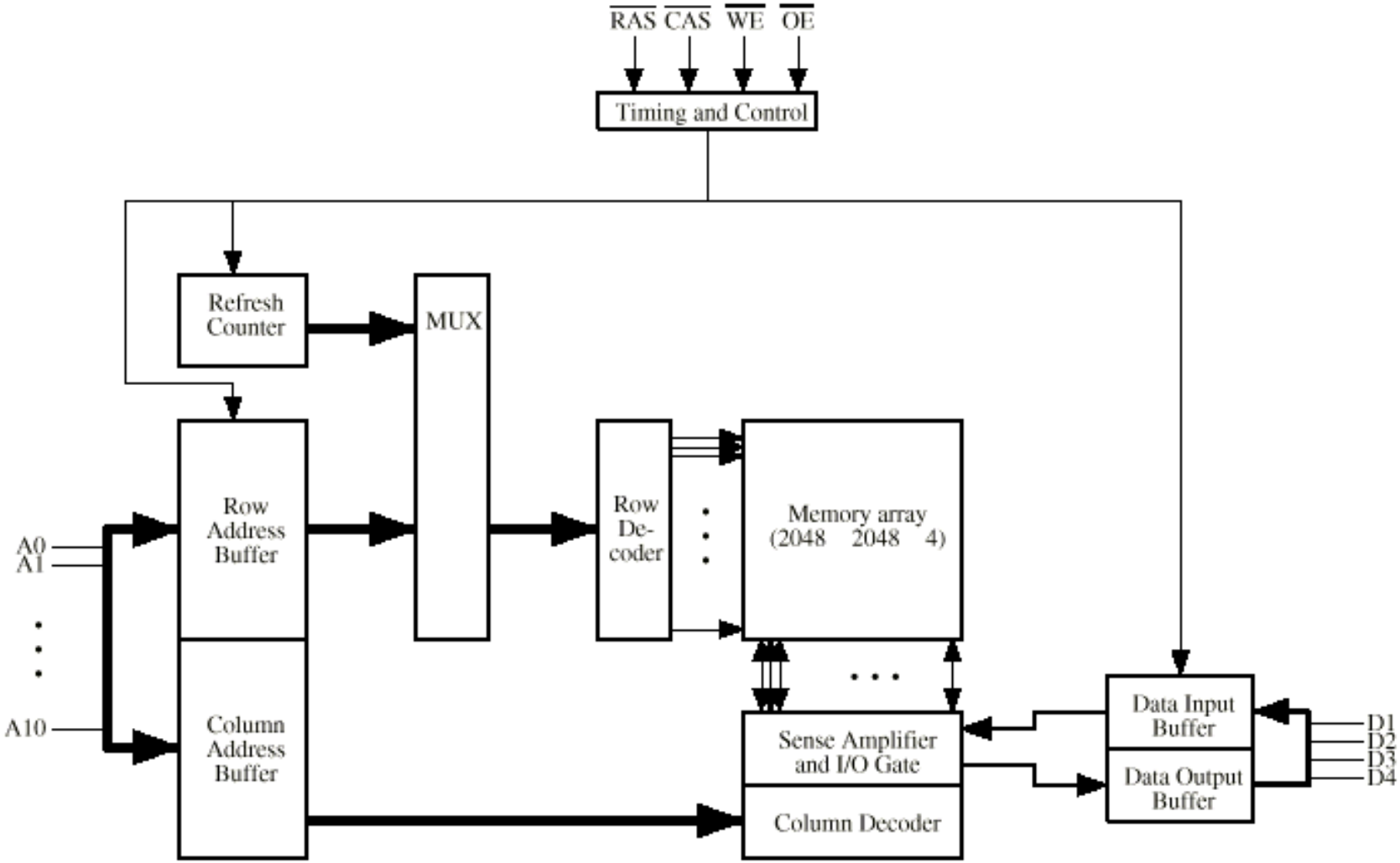
Types of ROM

- Written during manufacture
  - Very expensive for small runs
- Programmable (once)
  - PROM
  - Needs special equipment to program
- Read “mostly”
  - Erasable Programmable (EPROM)
    - Erased by Ultraviolet radiation
  - Electrically Erasable (EEPROM)
    - Takes much longer to write than read
  - Flash memory
    - Erase whole memory electrically

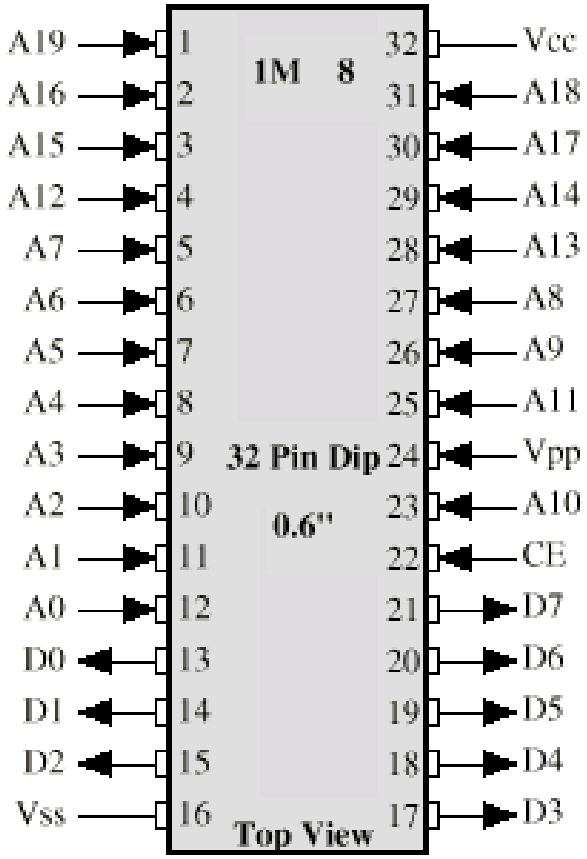
Organisation in detail

- A 16Mbit chip can be organised as 1M of 16 bit words
  - E.g., a bit per chip system has 16 lots of 1Mbit chip with bit 1 of each word in chip 1 and so on
- A 16Mbit chip can be organised as a 2048 x 2048 x 4bit array
  - Reduces number of address pins
    - Multiplex row address and column address
    - 11 pins to address (2^{11}=2048)
    - Adding one more pin doubles range of values so x4 capacity

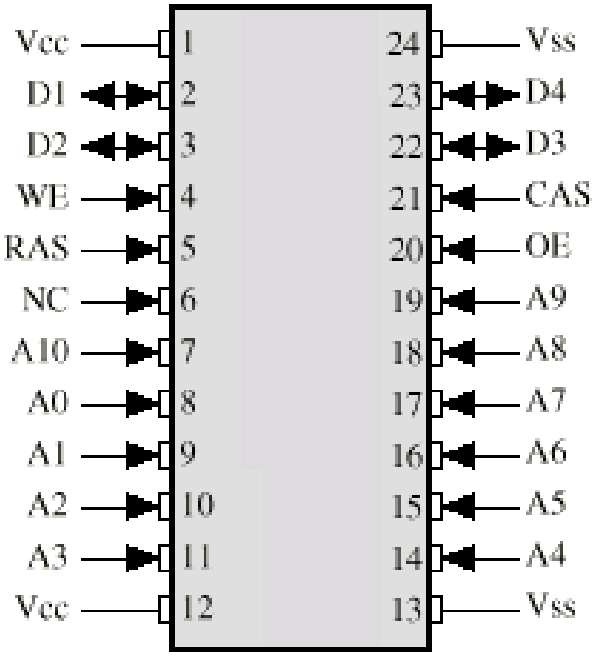
Typical 16 Mb DRAM (4M x 4)



Packaging

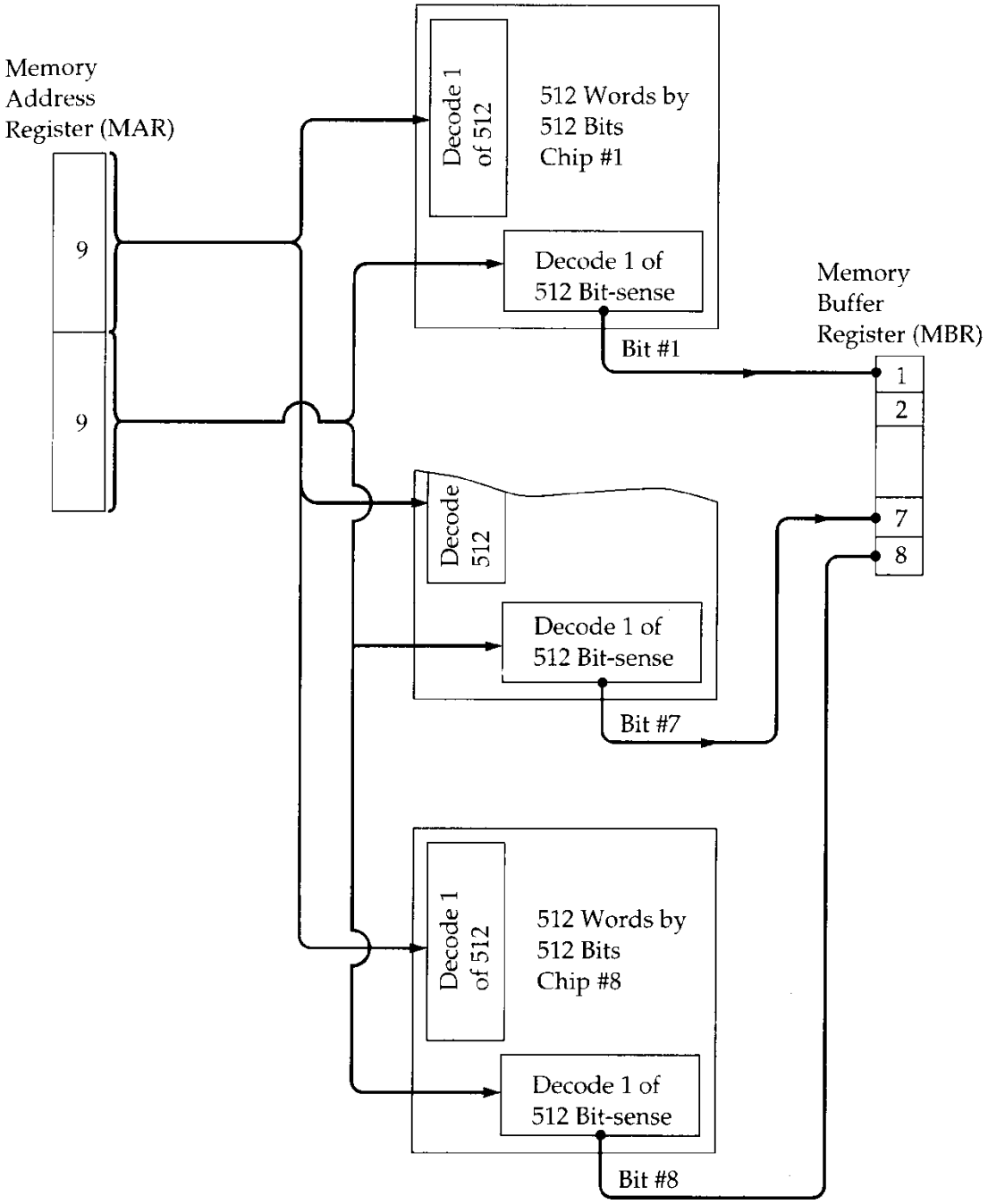


(a) 8 Mbit EPROM

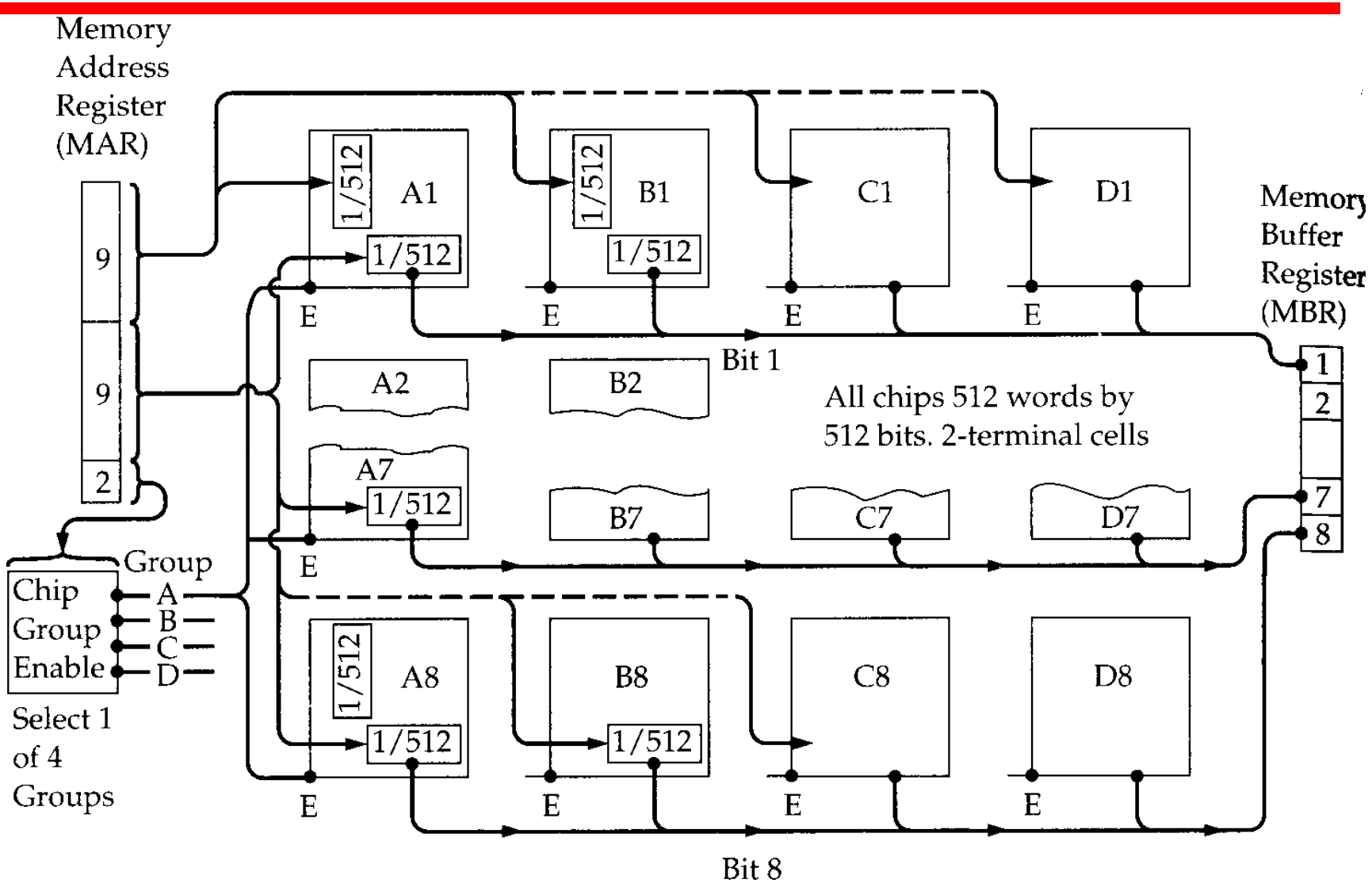


(b) 16 Mbit DRAM

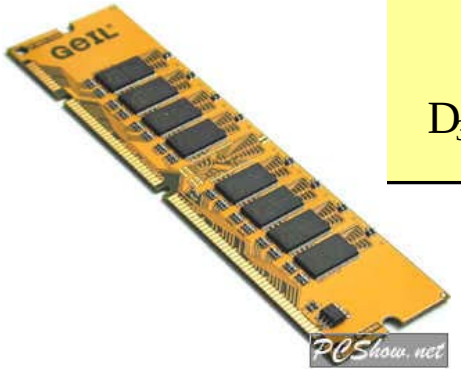
Module Organisation



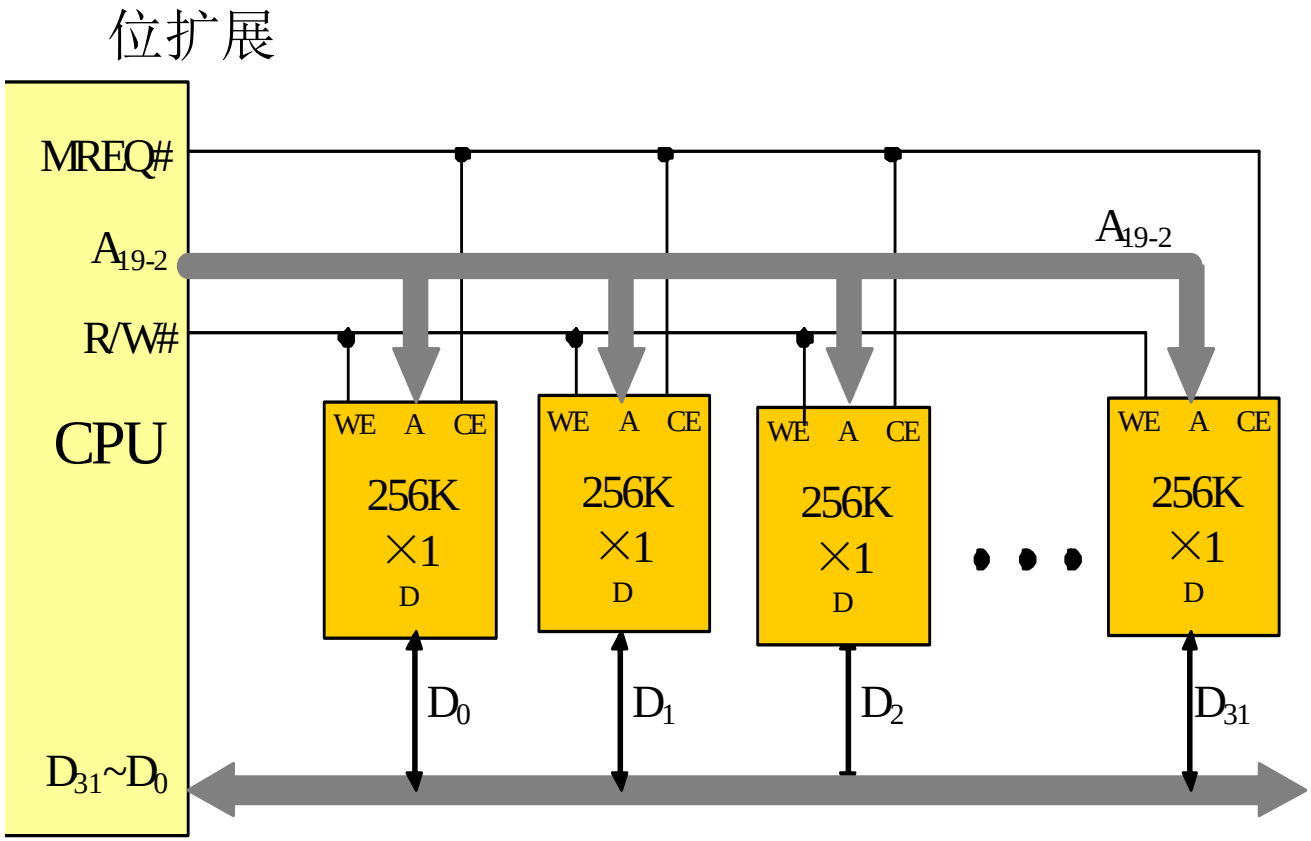
Module Organisation (2)



存储器的构成


GEIL
PCShow.net

位扩展


MREQ#
A19-2
R/W#
CPU
D31~D0
WE A CE
256K
×1
D
WE A CE
256K
×1
D
WE A CE
256K
×1
D
WE A CE
256K
×1
D
A19-2
D0
D1
D2
D31

位扩展

芯片的地址线数：18

存储器的结构256Kx32

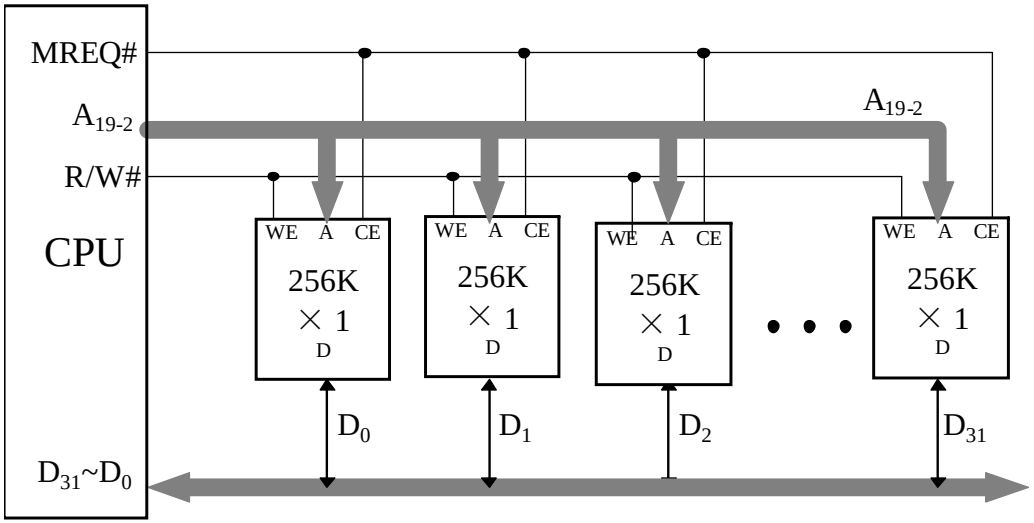
容量：1MB

CPU的有效地址位数：20位字节地址

每个芯片的地址范围相同

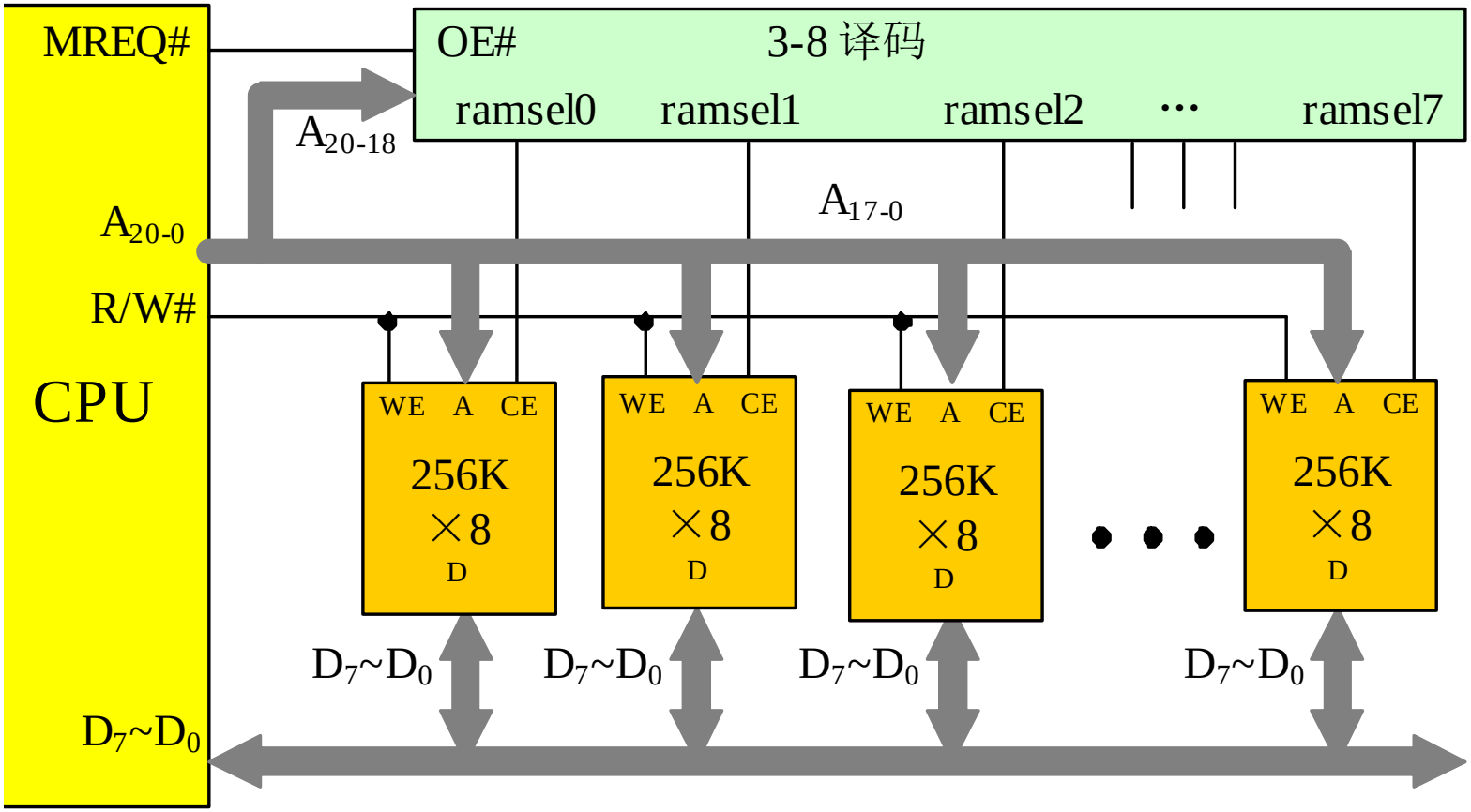
0	3	2	1	0
1	7	6	5	4
2	11	10	9	8

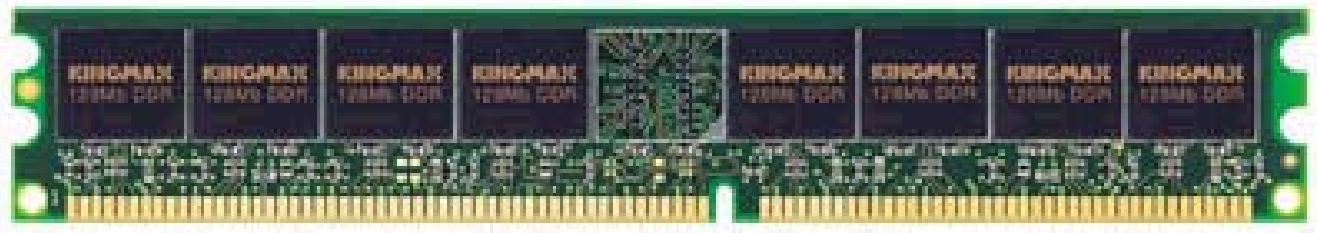
位扩展



256kx1 256kx1 …… 256kx1

字扩展


MREQ#
OE#
3-8 译码
ramsel0
ramsel1
ramsel2
…
ramsel7
A20-18
A17-0
A20-0
R/W#
CPU
WE A CE
256K
×8
D
D7~D0
D7~D0



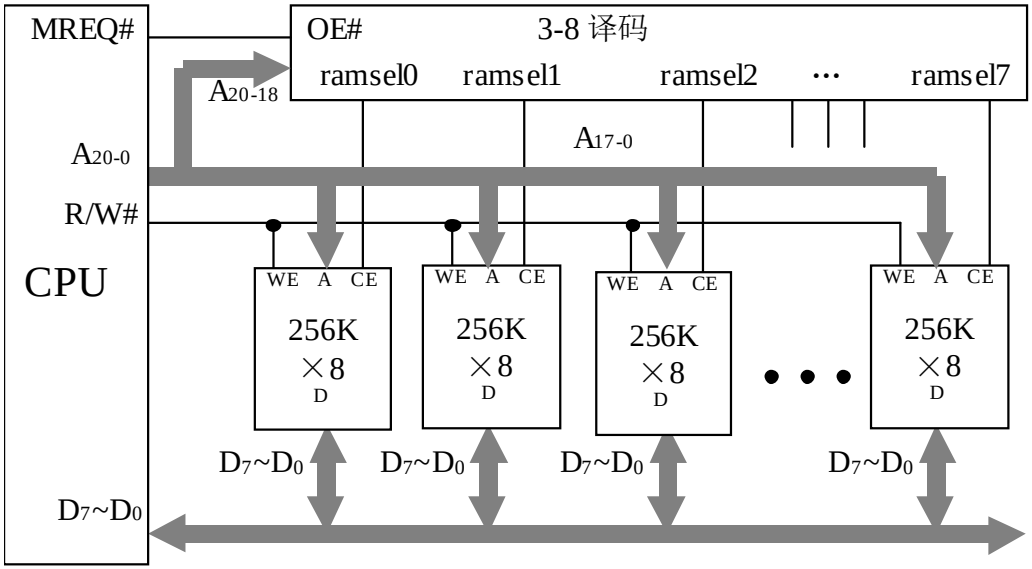
字扩展

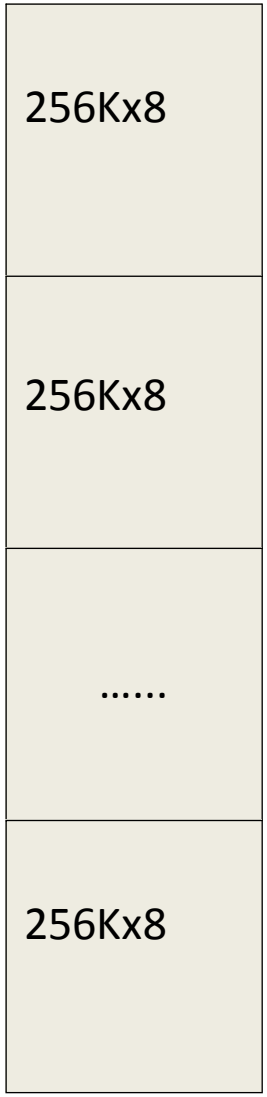
存储器芯片引脚数：18

存储器结构：2Mx8

CPU有效地址线数：21

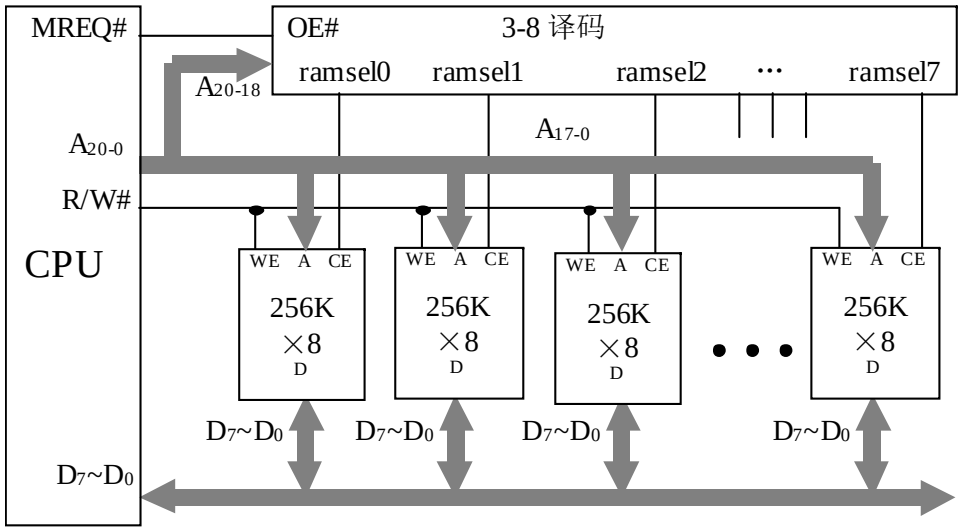
每个芯片的地址范围不同



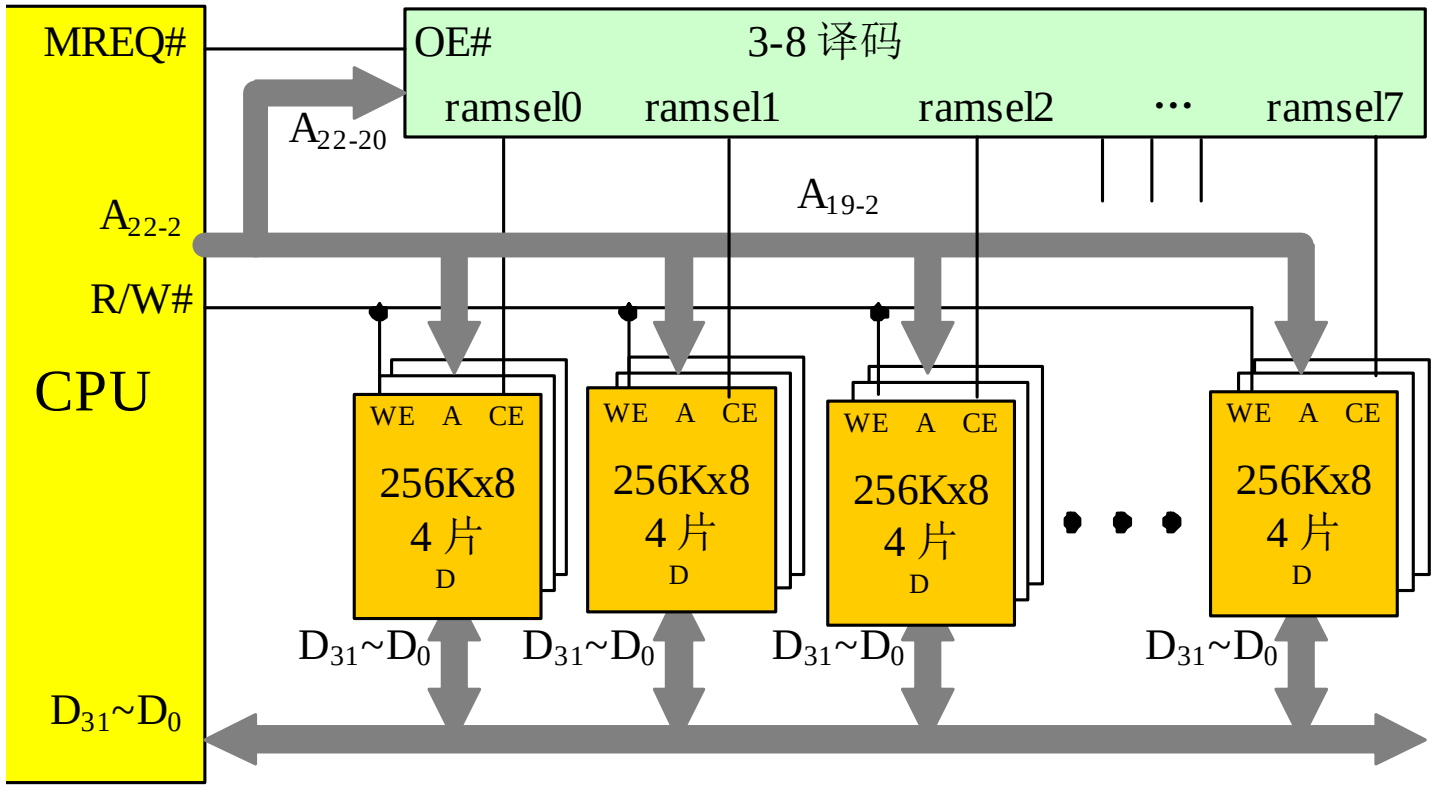


问题

- 字扩展中，求每个芯片的地址范围
- 字扩展中，求整个存储器的地址范围
- 如果用超出存储器地址范围的地址访存，会出现什么情况？



字位扩展


MREQ#
OE#
3-8 译码
ramsel0
ramsel1
ramsel2
…
ramsel7
A22-20
A19-2
A22-2
R/W#
CPU
WE A CE
256Kx8
4 片
D
D31~D0
D31~D0
D31~D0
D31~D0
D31~D0

例1 某计算机的主存地址空间中，从地址0000_{16}到$3FFF_{16}$为ROM存储区域，从4000_{16}到$5FFF_{16}$为保留地址区域，暂时不用，从6000_{16}到$FFFF_{16}$为RAM地址区域。RAM的控制信号为CS#和WE#，CPU的地址线为A15~A0，数据线为8位的线路D7~D0，控制信号有读写控制R/W#和访存请求MREQ#，要求：

(1) 画出地址译码方案。

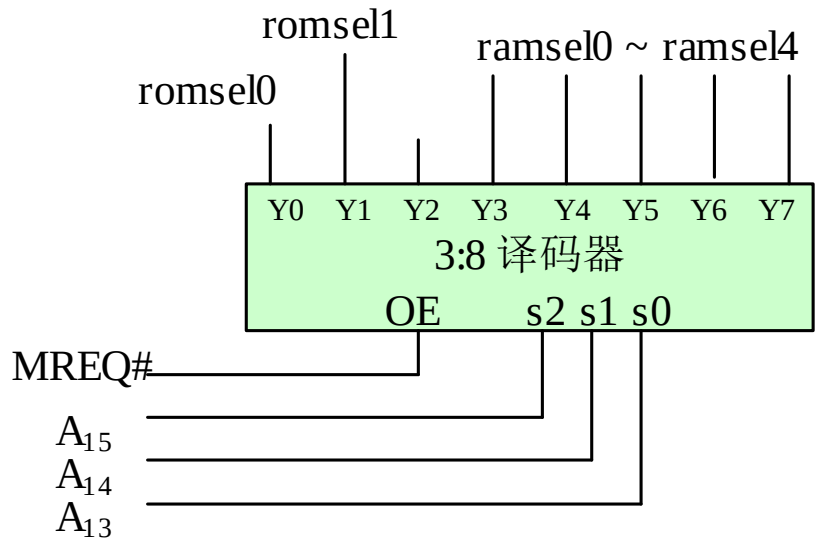
(2) 如果ROM和RAM存储器芯片都采用8K×1的芯片，试画出存储器与CPU的连接图。

(3) 如果ROM存储器芯片采用8K×8的芯片，RAM存储器芯片采用4K×8的芯片，试画出存储器与CPU的连接图。

(4) 如果ROM存储器芯片采用16K×8的芯片，RAM存储器芯片采用8K×8的芯片，试画出存储器与CPU的连接图。

(1) 画出地址译码方案

- 解：(1)



译码器的输出信号逻辑表达式为：

$$romsel0 = \overline{A15} * \overline{A14} * \overline{A13} * \overline{MREQ\#}$$

$$romsel1 = \overline{A15} * \overline{A14} * A13 * \overline{MREQ\#}$$

$$ramsel0 = \overline{A15} * A14 * A13 * \overline{MREQ\#}$$

$$ramsel1 = A15 * \overline{A14} * \overline{A13} * \overline{MREQ\#}$$

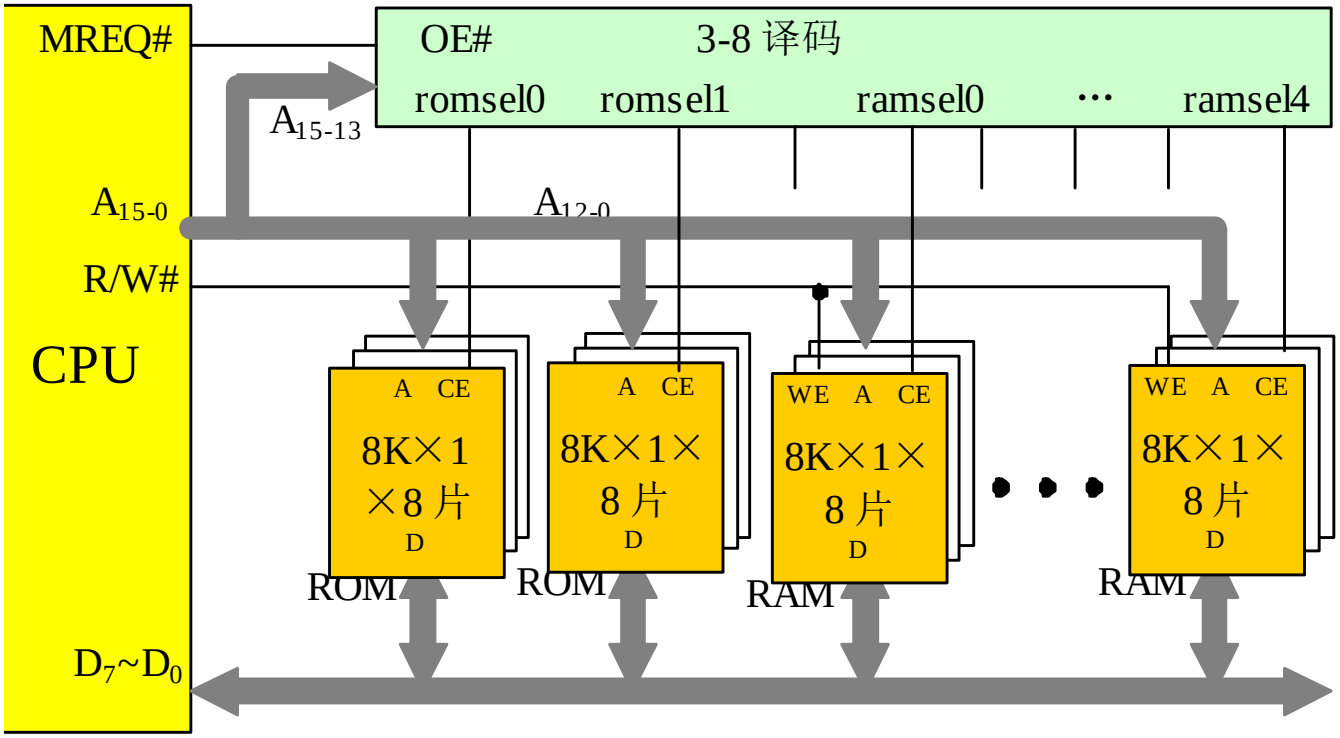
$$ramsel2 = A15 * \overline{A14} * A13 * \overline{MREQ\#}$$

$$ramsel3 = A15 * A14 * \overline{A13} * \overline{MREQ\#}$$

$$ramsel4 = A15 * A14 * A13 * \overline{MREQ\#}$$

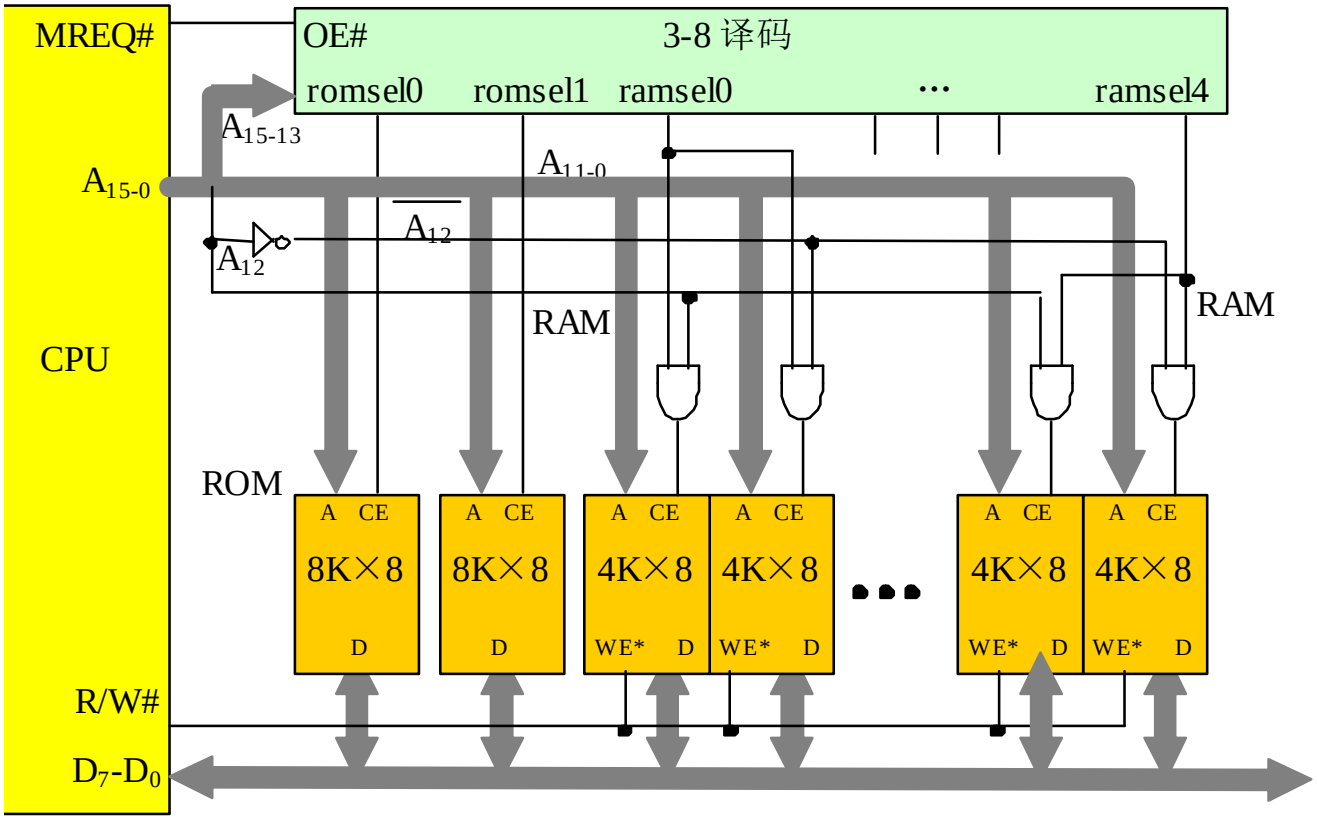
(2) 如果ROM和RAM存储器芯片都采用8K×1的芯片，试画出存储器与CPU的连接图。

解：**(2) 8KB的存储区域可以用8片存储器芯片构成一组实现。8K×1的存储器芯片的地址线需要13条，即A12~0。**



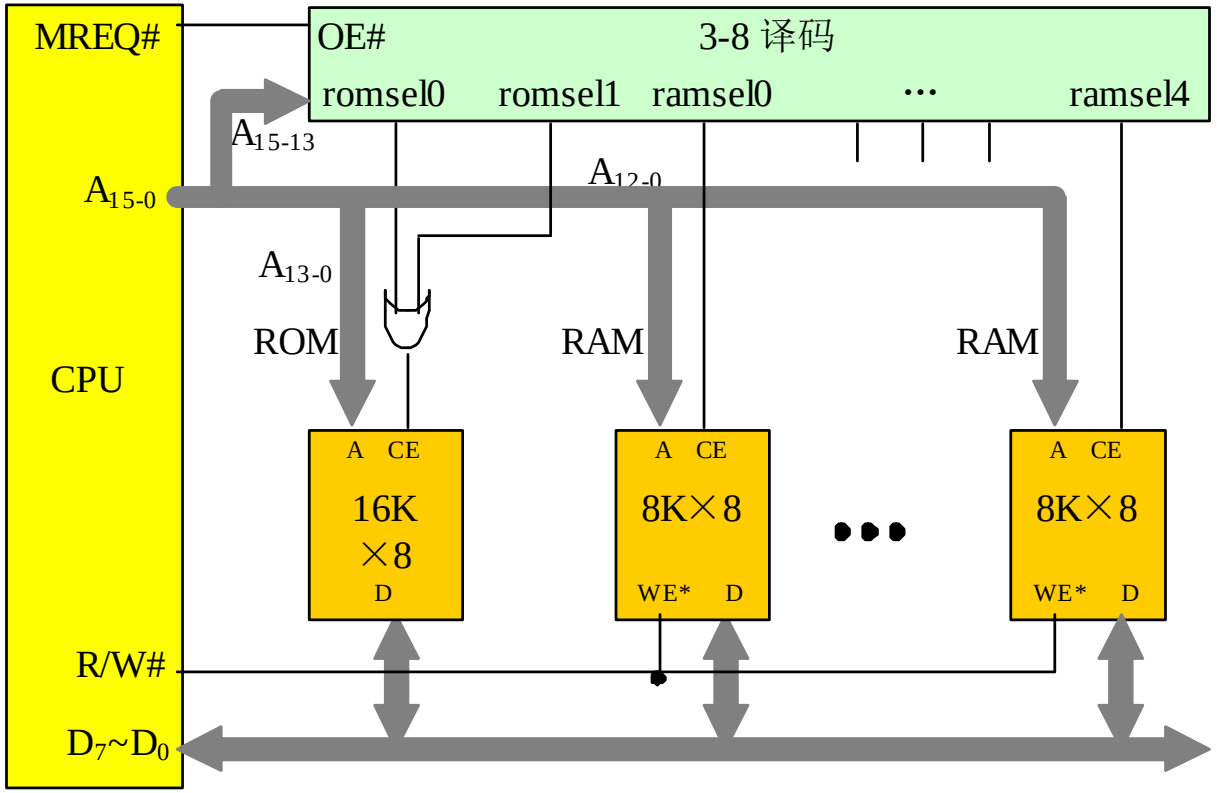
(3) 如果ROM存储器芯片采用8K×8的芯片，RAM存储器芯片采用4K×8的芯片，试画出存储器与CPU的连接图。

解：（3）



(4) 如果ROM存储器芯片采用16K×8的芯片，RAM存储器芯片采用8K×8的芯片，试画出存储器与CPU的连接图。

解：（4）



例2 某计算机系统的主存采用32位字节地址空间和64位数据线访问存储器，若使用64M位的DRAM芯片组成该机所允许的最大主存空间，并采用内存条的形式，问：

(1) 若每个内存条为64M×32位，共需多少内存条？

(2) 每个内存条内共有多少片DRAM芯片？

(3) 主存共需多少DRAM芯片？

(4) CPU如何有选择地访问各内存条？

解：(1) 主存最大空间为2^{32}=4GB，每个内存条的容量为64×4B = 256MB，主存需要的内存条数量为4GB/256MB=16条。

(2) 每个芯片的容量为8MB，内存条需要的芯片数量为256MB/8MB = 32片。

(3) 整个主存需要的内存芯片数量是16×32=512片。

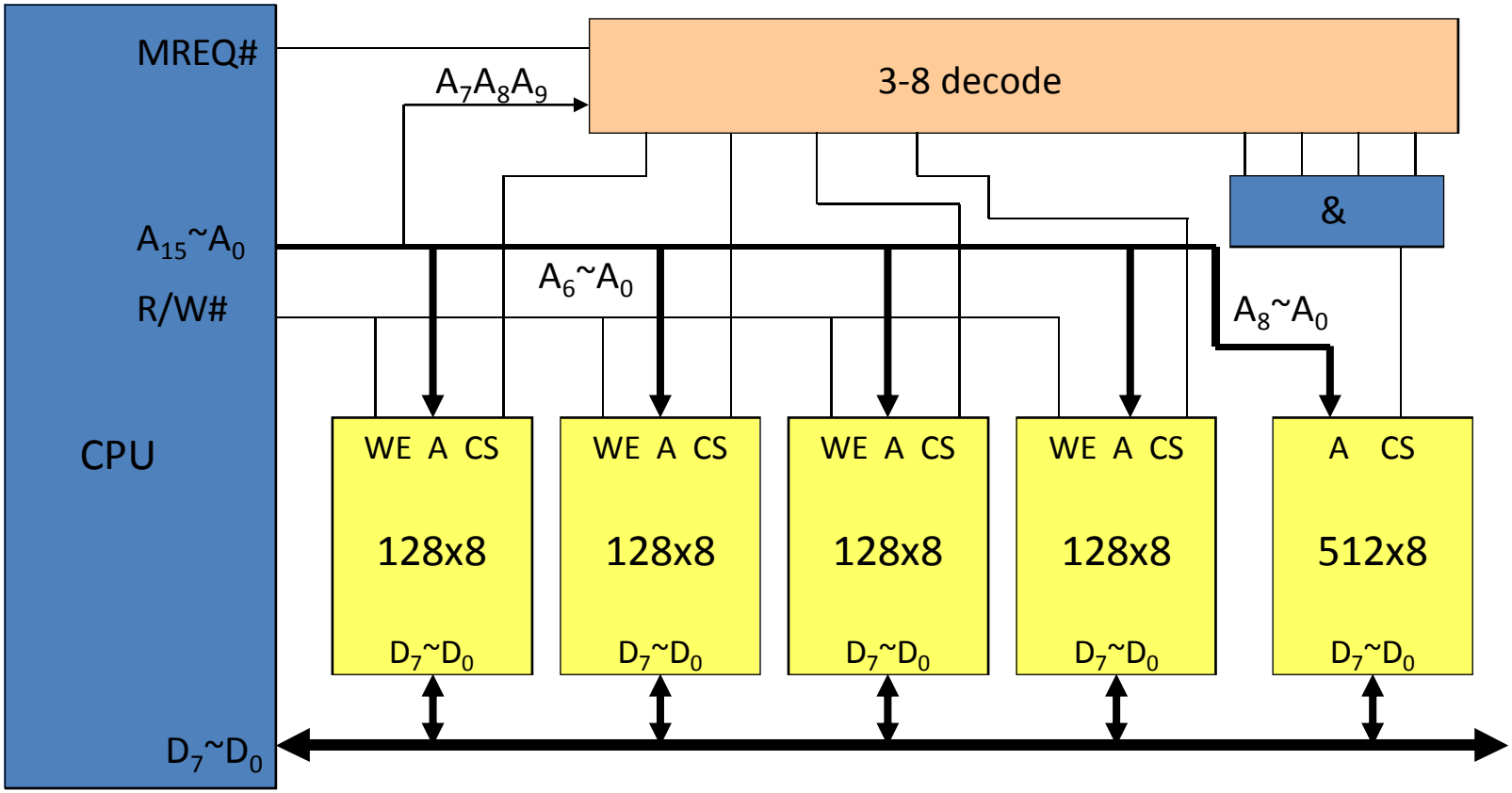
(4) 由于CPU字长为64位，内存条需要进行位扩展，即2个32位的内存条构成一组64位的存储单元组，16个内存条构成8组，为选择这8组内存条，CPU地址中需要用最高3位地址作为产生选择信号的地址码。

例3 假定计算机系统需要512字节RAM和512字节ROM容量。使用的RAM芯片是128字×8位，ROM芯片为512字×8位。RAM芯片有CS*及WE*控制端，ROM芯片有CS*控制端，CPU有地址线A15~A0、数据线D7~D0、读写控制线RW*等，试确定各存储器芯片的地址区间，指出存储器以及各存储器芯片需要的地址线数量，并画出存储器与CPU的连接图。

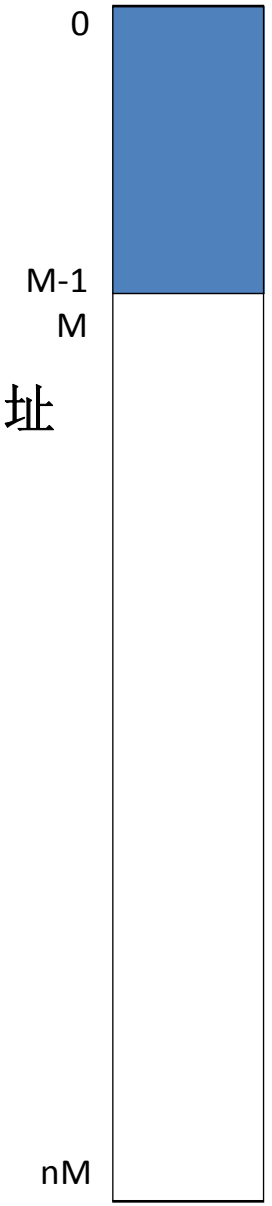
解：各存储器芯片的地址区间：

元件	16 进制地址范围	二进制地址值
RAM1	0000~007F	0 0 0 x x x x x x x
RAM2	0080~00FF	0 0 1 x x x x x x x
RAM3	0100~017F	0 1 0 x x x x x x x
RAM4	0180~01FF	0 1 1 x x x x x x x
ROM	0200~03FF	1 x x x x x x x x x

- 存储器的总容量为1KB，需要10条地址线。
- RAM芯片需要7条信号线(2^7=128)，ROM芯片需要9条地址线(2^9=512)。
- 存储器与CPU的连接图



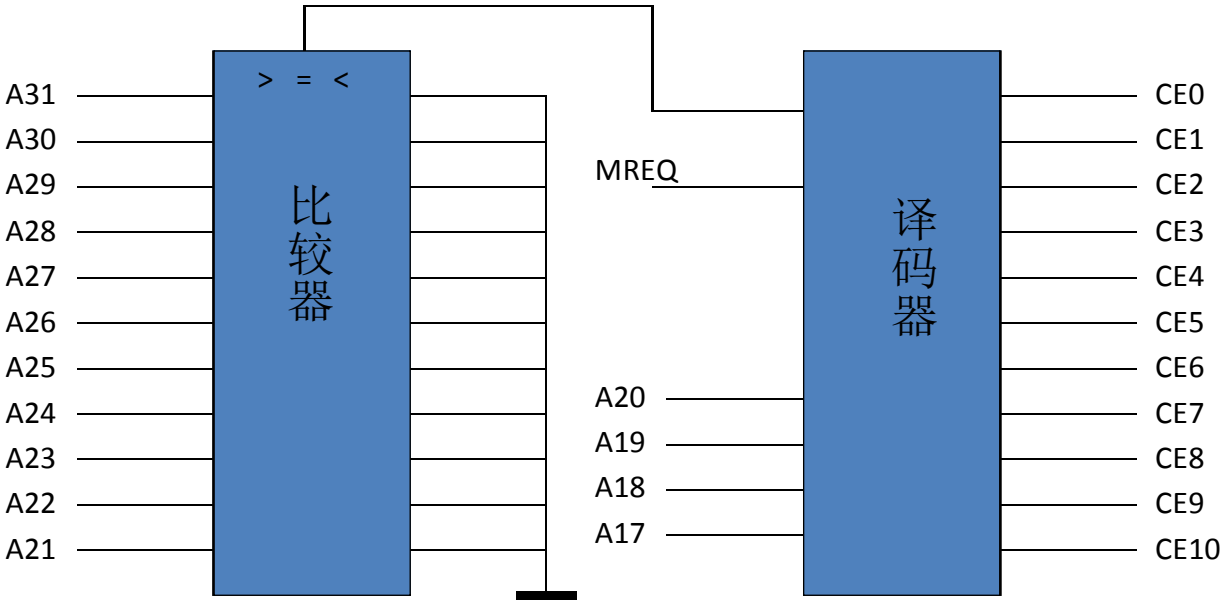
全译码与部分译码



- 全译码
  - 所有CPU高位地址线均参与对存储单元的译码寻址
  - 低位地址线对芯片内各存储单元的译码寻址
    - 片内译码
  - 高位地址线对存储芯片的译码寻址
    - 片选译码
  - 每个存储单元的地址都是唯一的
    - 不存在地址重复
- 部分译码
  - 部分高位地址线参与对存储单元的译码寻址
  - 存在地址段内容重叠
    - 每个单元有多个地址

全译码的实现

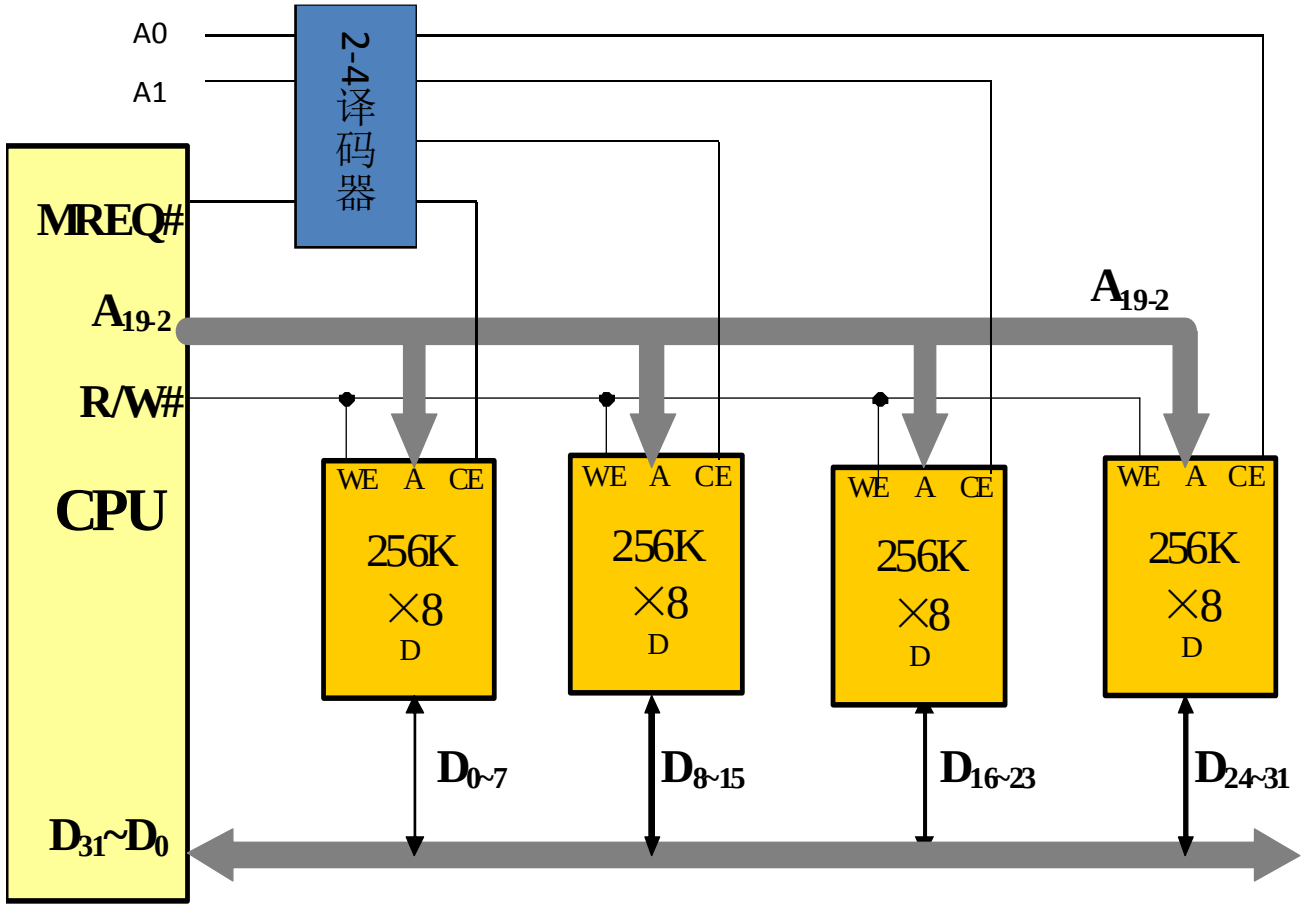
- 与门（常用）
- 地址比较器



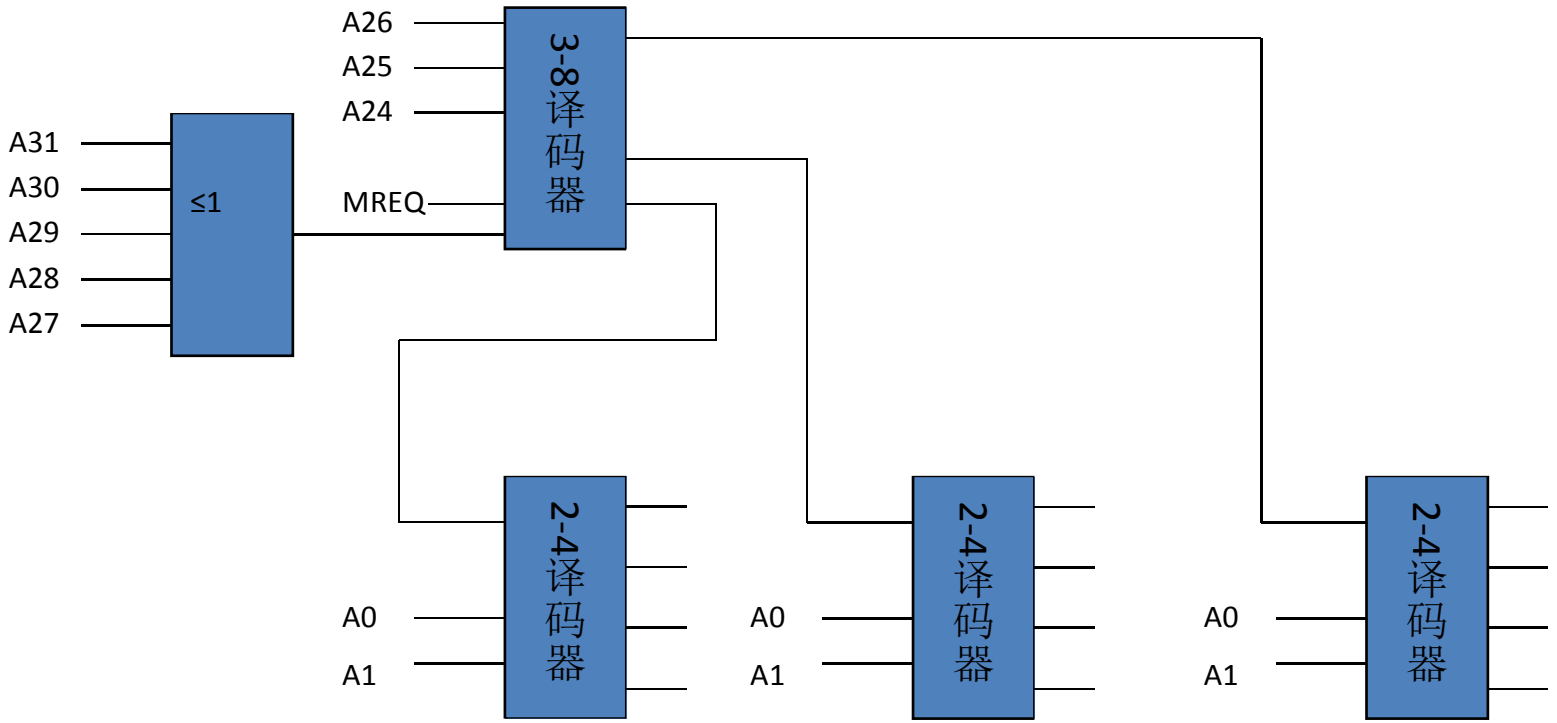
字选择与字节选择

- 字选择
  - 以字为单位访问存储器
  - 位数等于存储器的字长
  - 低位地址不需要
- 字节选择
  - 可以以字节为单位访问存储器
  - 低位地址用于选择字节

字节选择的实现


A0
A1
2-4译码器
MREQ#
A19-2
R/W#
CPU
D31~D0
WE A CE
256K
×8
D
D0~7
D8~15
D16~23
D24~31
A19-2

字位扩展中的字节选择


A26
A25
A24
3-8译码器
A31
A30
A29
A28
A27
≤1
MREQ
2-4译码器
A0
A1
2-4译码器
A0
A1
2-4译码器
A0
A1

Computer Organization and Architecture

Input/Output

Input/Output Problems

- Wide variety of peripherals
  - Delivering different amounts of data
  - At different speeds
  - In different formats
- All slower than CPU and RAM
- Need I/O modules

Input/Output Module

- Interface to CPU and Memory
- Interface to one or more peripherals

External Devices

- Human readable
  - Screen, printer, keyboard
- Machine readable
  - Monitoring and control
- Communication
  - Modem
  - Network Interface Card (NIC)

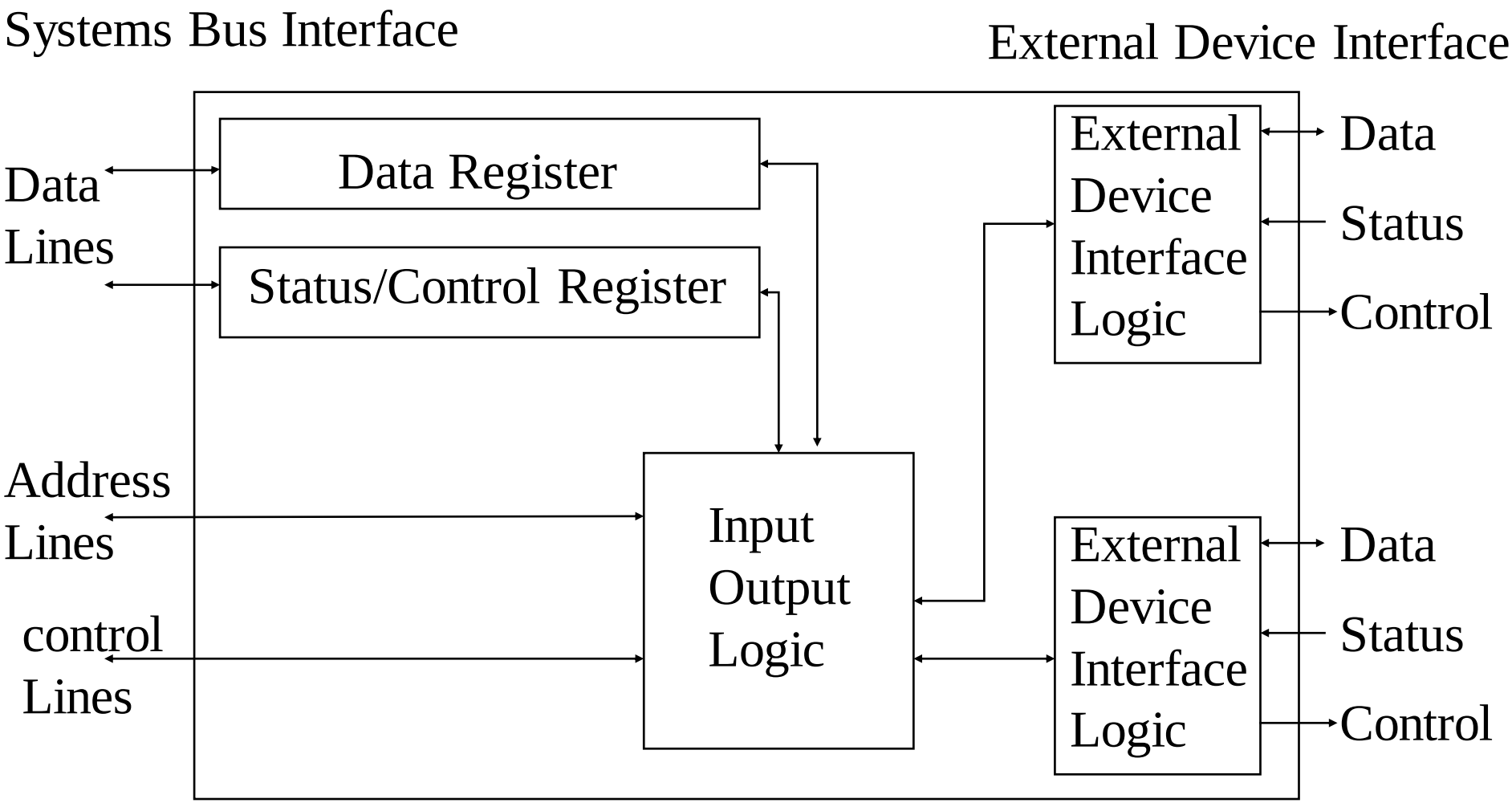
I/O Module Function

- Control & Timing
- CPU Communication
- Device Communication
- Data Buffering
- Error Detection

I/O Steps

- CPU checks I/O module device status
- I/O module returns status
- If ready, CPU requests data transfer
- I/O module gets data from device
- I/O module transfers data to CPU
- Variations for output, DMA, etc.

I/O Module Diagram



I/O Module Decisions

- Hide or reveal device properties to CPU
- Support multiple or single device
- Control device functions or leave for CPU

Input Output Techniques

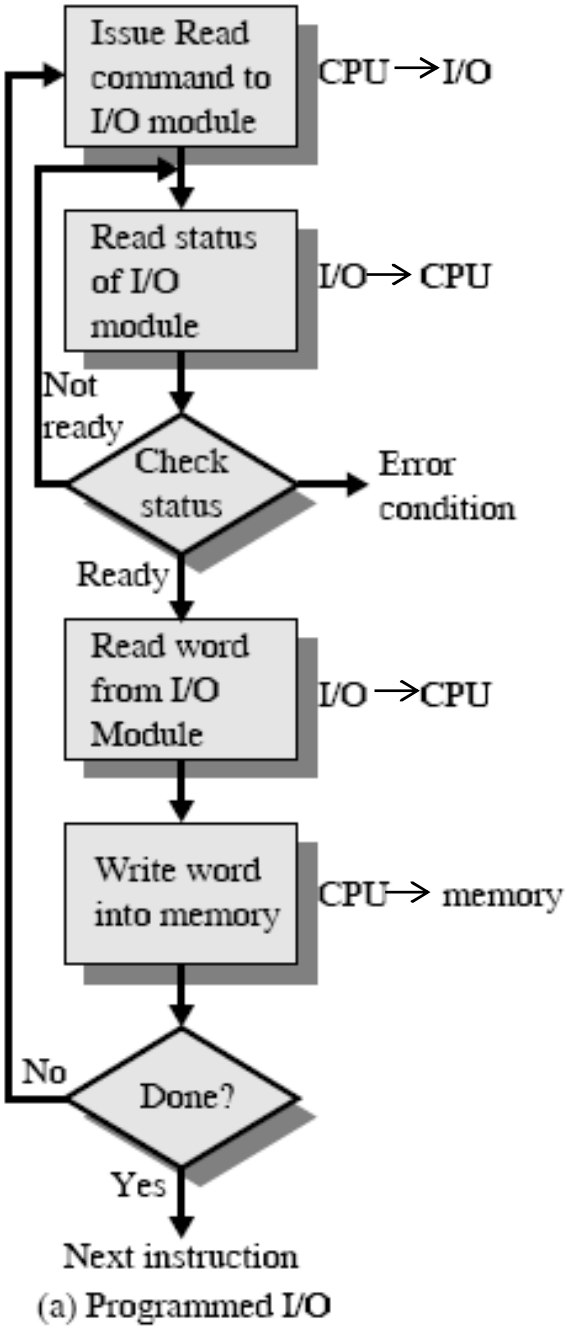
- Programmed
- Interrupt driven
- Direct Memory Access (DMA)

Programmed I/O

- CPU has direct control over I/O
  - Sensing status
  - Read/write commands
  - Transferring data
- CPU waits for I/O module to complete operation
- Wastes CPU time

Programmed I/O - deta

- CPU requests I/O operation
- I/O module performs operation
- I/O module sets status bits
- CPU checks status bits periodica
- I/O module does not inform CPU
- I/O module does not interrupt C
- CPU may wait or come back late



(a) Programmed I/O

I/O Commands

- CPU issues address
  - Identifies module (& device if >1 per module)
- CPU issues command
  - Control - telling module what to do
    - e.g. spin up disk
  - Test - check status
    - e.g. power? Error?
  - Read/Write
    - Module transfers data via buffer from/to device

Addressing I/O Devices

- Under programmed I/O data transfer is very like memory access (CPU viewpoint)
- Each device given unique identifier
- CPU commands contain identifier (address)

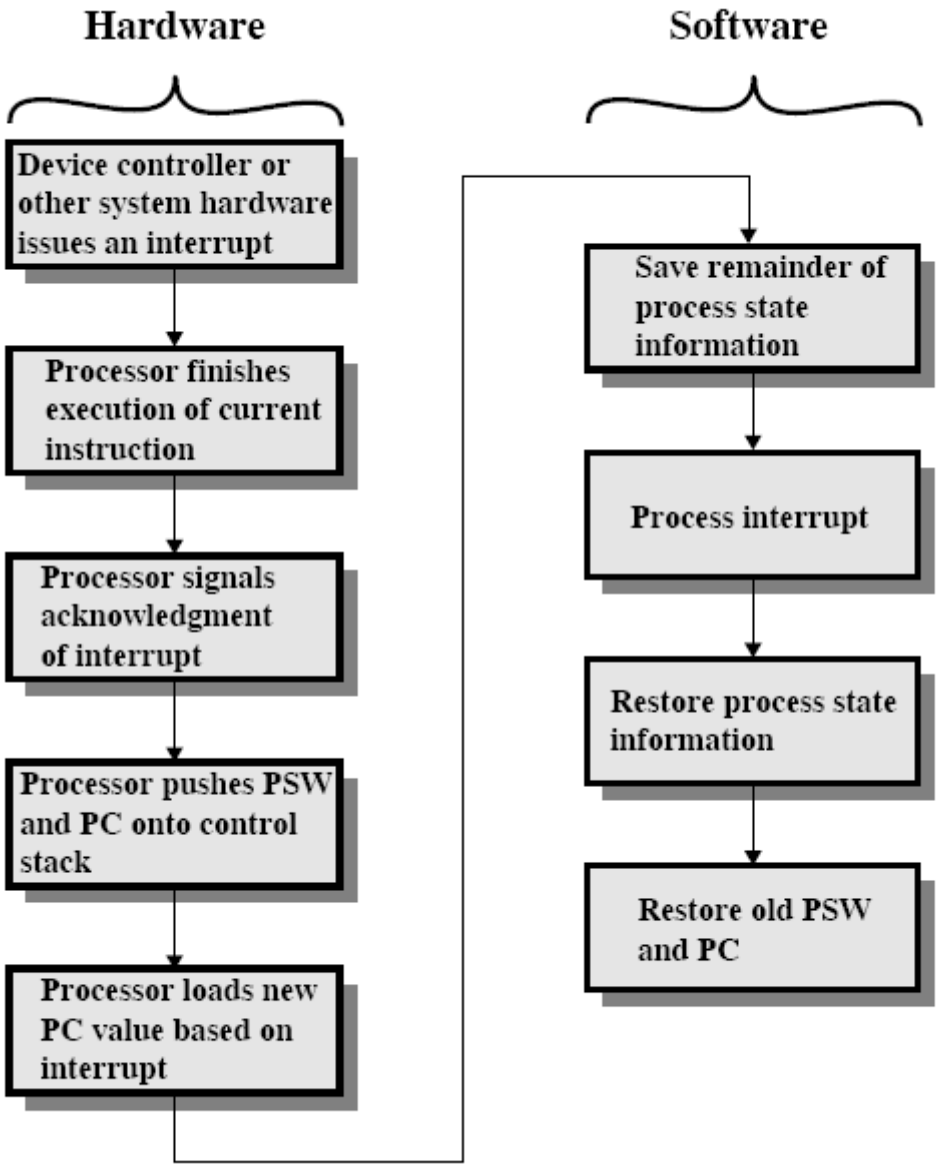
I/O Mapping

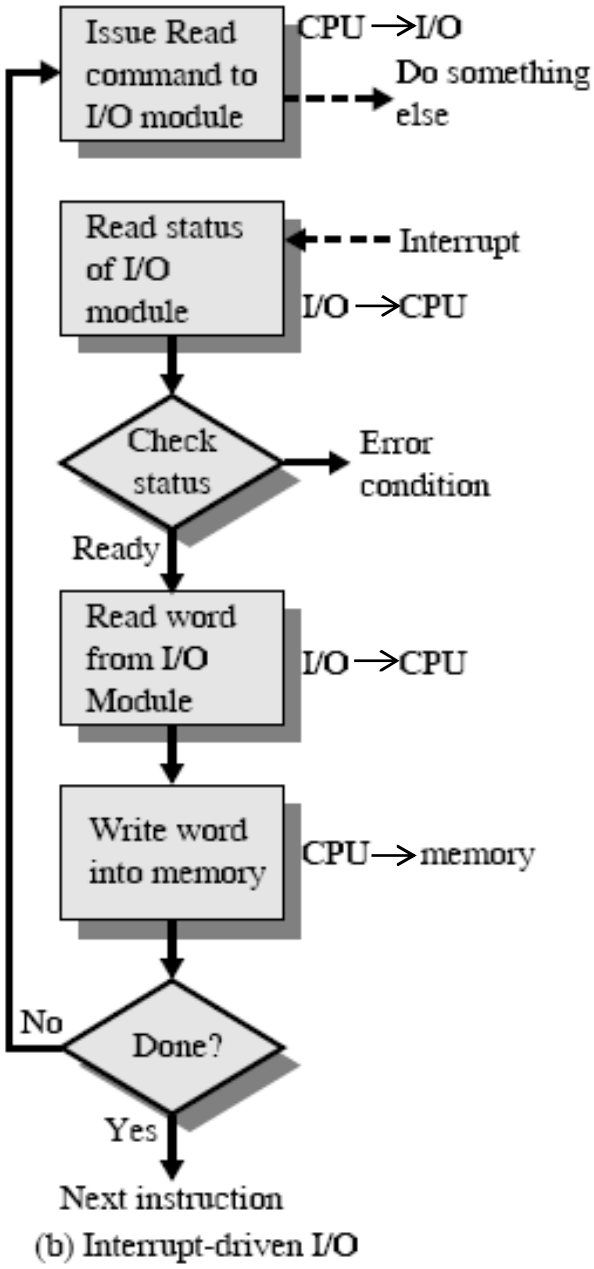
- Memory mapped I/O
  - Devices and memory share an address space
  - I/O looks just like memory read/write
  - No special commands for I/O
    - Large selection of memory access commands available
- Isolated I/O
  - Separate address spaces
  - Need I/O or memory select lines
  - Special commands for I/O
    - Limited set

Interrupt Driven I/O

- Overcomes CPU waiting
- No repeated CPU checking of device
- I/O module interrupts when ready

Interrupt Driven I/O





(b) Interrupt-driven I/O

CPU Viewpoint

- Issue read command
- Do other work
- Check for interrupt at end of each instruction cycle
- If interrupted:-
  - Save context (registers)
  - Process interrupt
    - Fetch data & store

Design Issues

- How do you identify the module issuing the interrupt?
- How do you deal with multiple interrupts?
  - i.e. an interrupt handler being interrupted

Identifying Interrupting Module (1)

- Different line for each module
  - Limits number of devices
- Software poll
  - CPU asks each module in turn
  - Slow

Identifying Interrupting Module (2)

- Daisy Chain or Hardware poll
  - Interrupt Acknowledge sent down a chain
  - Module responsible places vector on bus
  - CPU uses vector to identify handler routine
- Bus Master
  - Module must claim the bus before it can raise interrupt
  - e.g. PCI & SCSI

Multiple Interrupts

- Each interrupt line has a priority
- Higher priority lines can interrupt lower priority lines
- If bus mastering only current master can interrupt

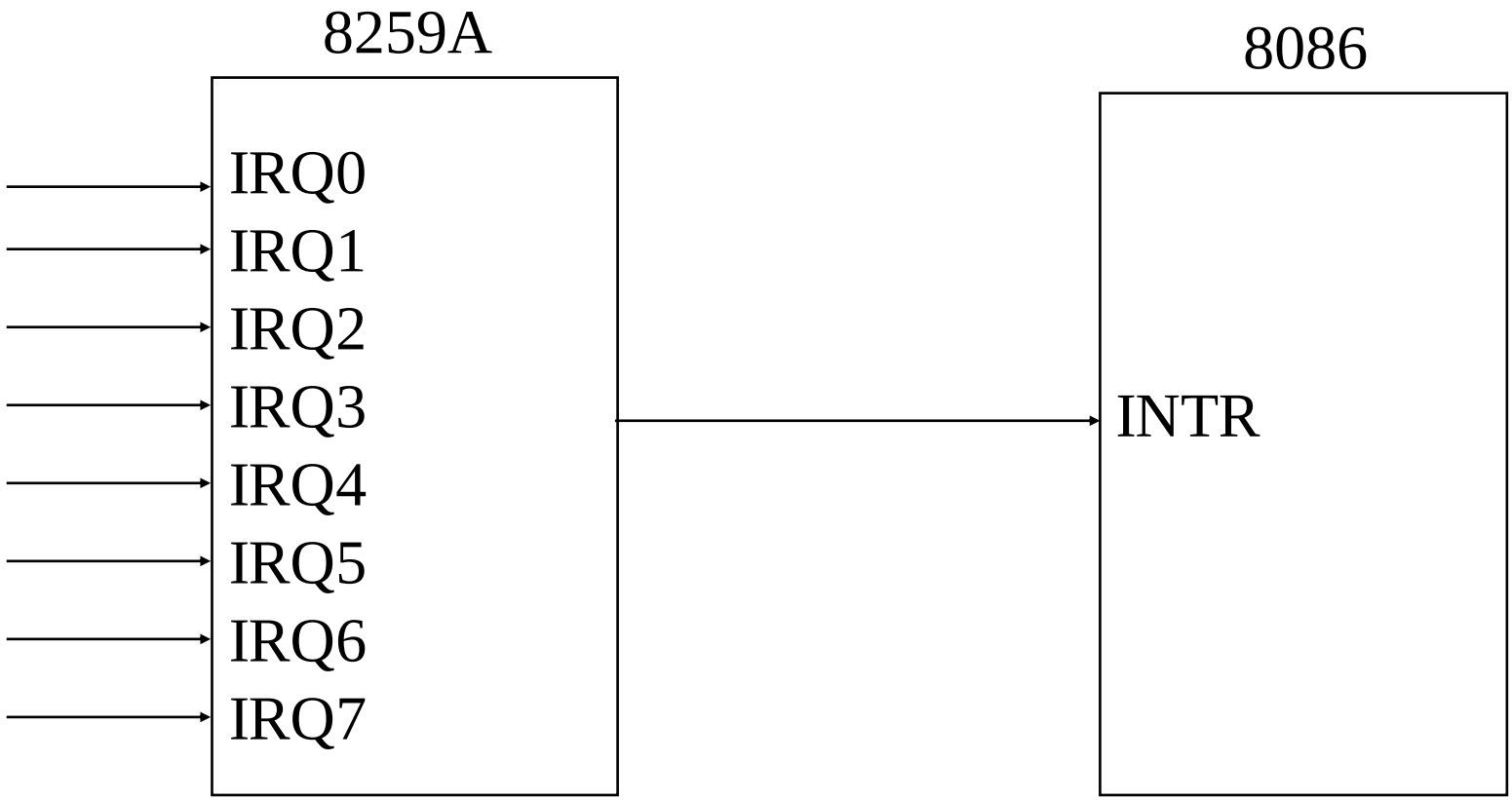
Example - PC Bus

- 80x86 has one interrupt line
- 8086 based systems use one 8259A interrupt controller
- 8259A has 8 interrupt lines

Sequence of Events

- 8259A accepts interrupts
- 8259A determines priority
- 8259A signals 8086 (raises INTR line)
- CPU Acknowledges
- 8259A puts correct vector on data bus
- CPU processes interrupt

PC Interrupt Layout



Direct Memory Access

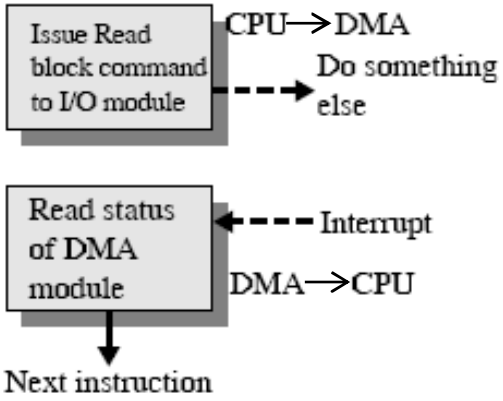
- Interrupt driven and programmed I/O require active CPU intervention
  - Transfer rate is limited
  - CPU is tied up
- DMA is the answer

DMA Function

- Additional Module (hardware) on bus
- DMA controller takes over from CPU for I/O

DMA Operation

- CPU tells DMA controller:-
  - Read/Write
  - Device address
  - Starting address of memory block for
  - Amount of data to be transferred
- CPU carries on with other work
- DMA controller deals with transfer
- DMA controller sends interrupt when finished

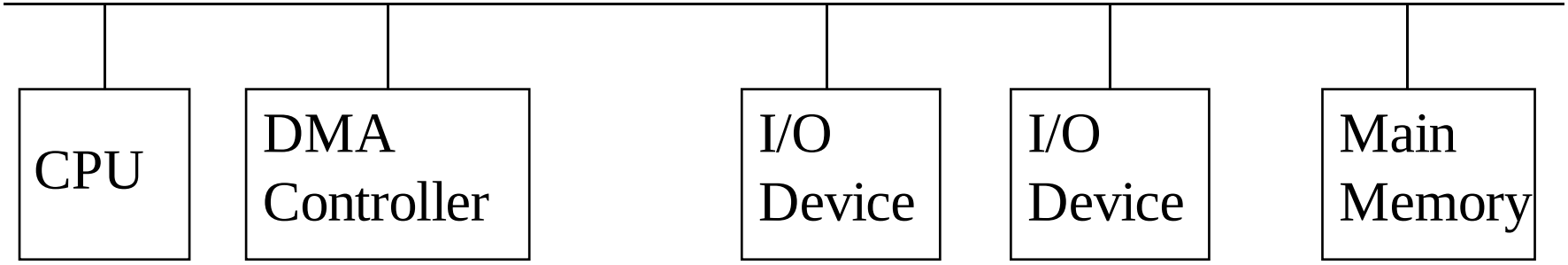


(c) Direct memory access

DMA Transfer
Cycle Stealing

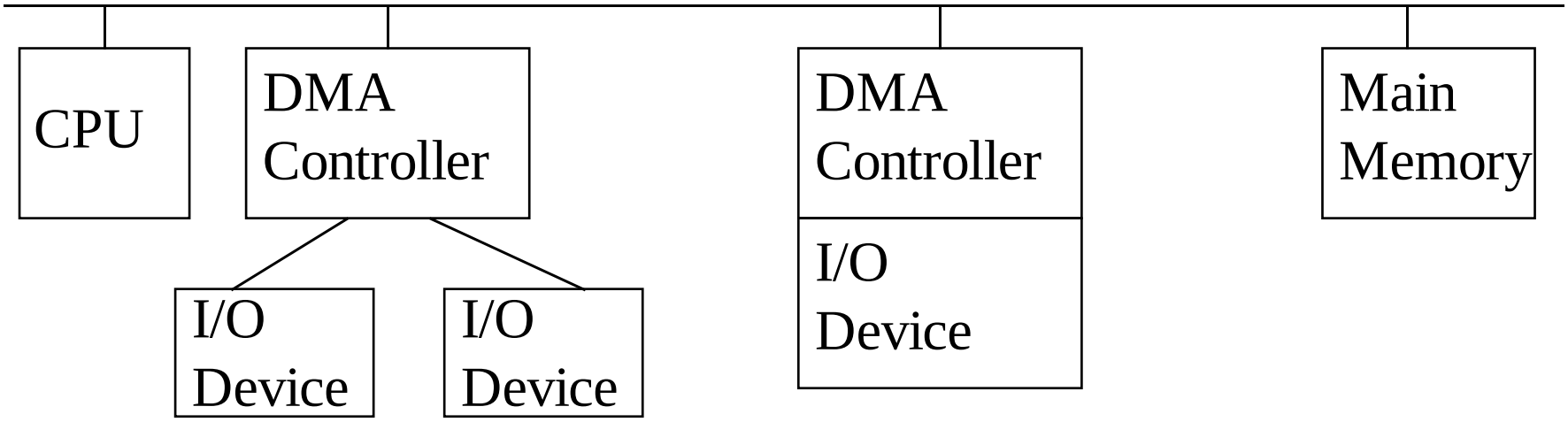
- DMA controller takes over bus for a cycle
- Transfer of one word of data
- Not an interrupt
  - CPU does not switch context
- CPU suspended just before it accesses bus
  - i.e. before an operand or data fetch or a data write
- Slows down CPU but not as much as CPU doing transfer

DMA Configurations (1)



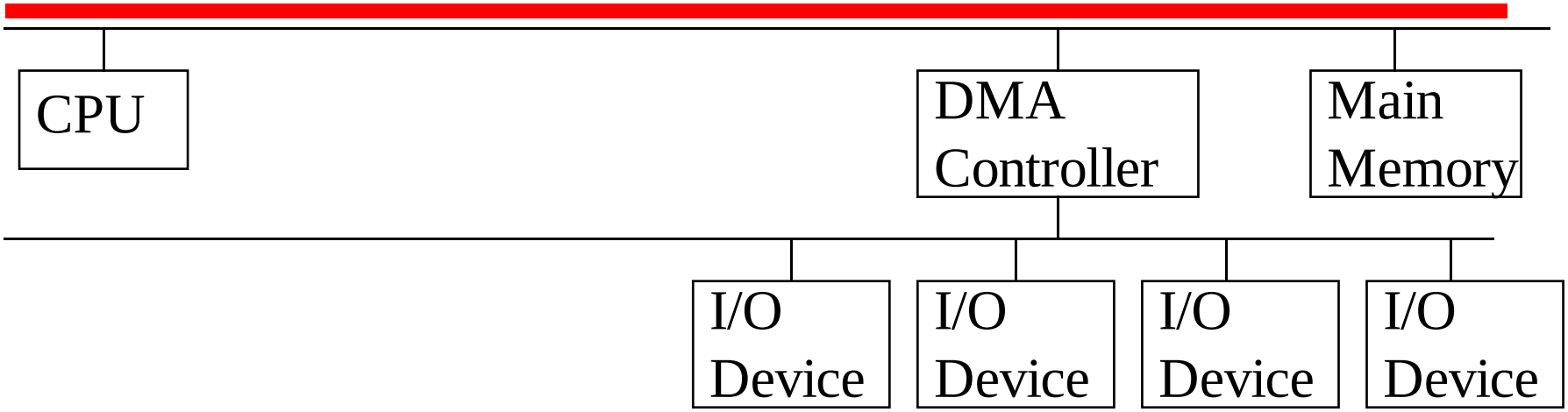
- Single Bus, Detached DMA controller
- Each transfer uses bus twice
  - I/O to DMA then DMA to memory
- CPU is suspended twice

DMA Configurations (2)



- Single Bus, Integrated DMA controller
- Controller may support >1 device
- Each transfer uses bus once
  - DMA to memory
- CPU is suspended once

DMA Configurations (3)



- Separate I/O Bus
- Bus supports all DMA enabled devices
- Each transfer uses bus once
  - DMA to memory
- CPU is suspended once

I/O Channels

- I/O devices getting more sophisticated
  - e.g. 3D graphics cards
- CPU instructs I/O controller to do transfer
- I/O controller does entire transfer
- Improves speed
  - Takes load off CPU
  - Dedicated processor is faster