# Embedded System Application

## 4190.303C

## 2010 Spring Semester

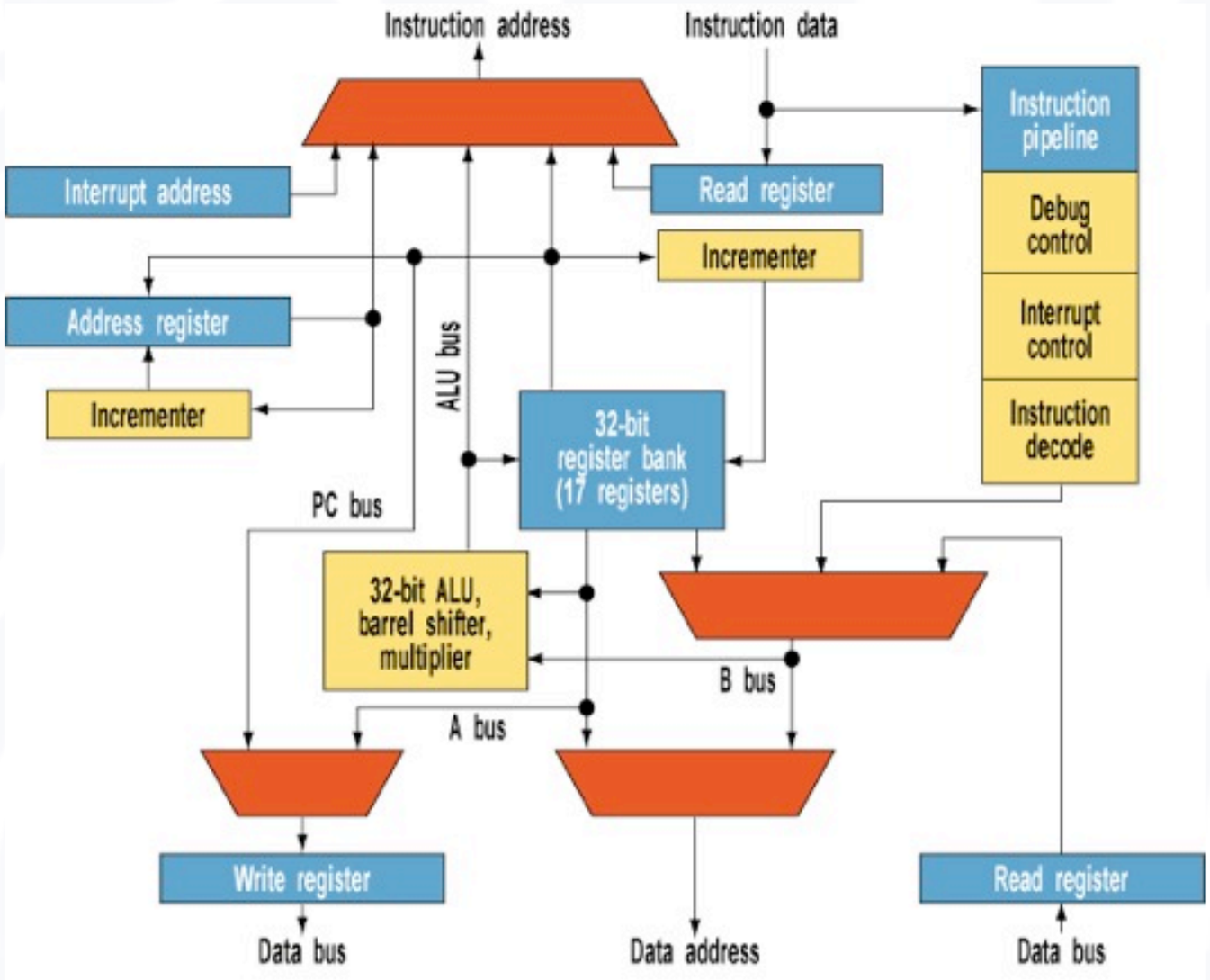
# Introduction to microprocessor interface

**Naehyuck Chang**
**Dept. of EECS/CSE**
**Seoul National University**
**naehyuck@snu.ac.kr**

# Harvard Architecture Microprocessor

- Instruction memory
  - Input: address from PC
  - Output: instruction (read only)
- Data memory
  - Input: memory address
    - Addressing mode
  - Input/output: read/write data
    - Read or write operand

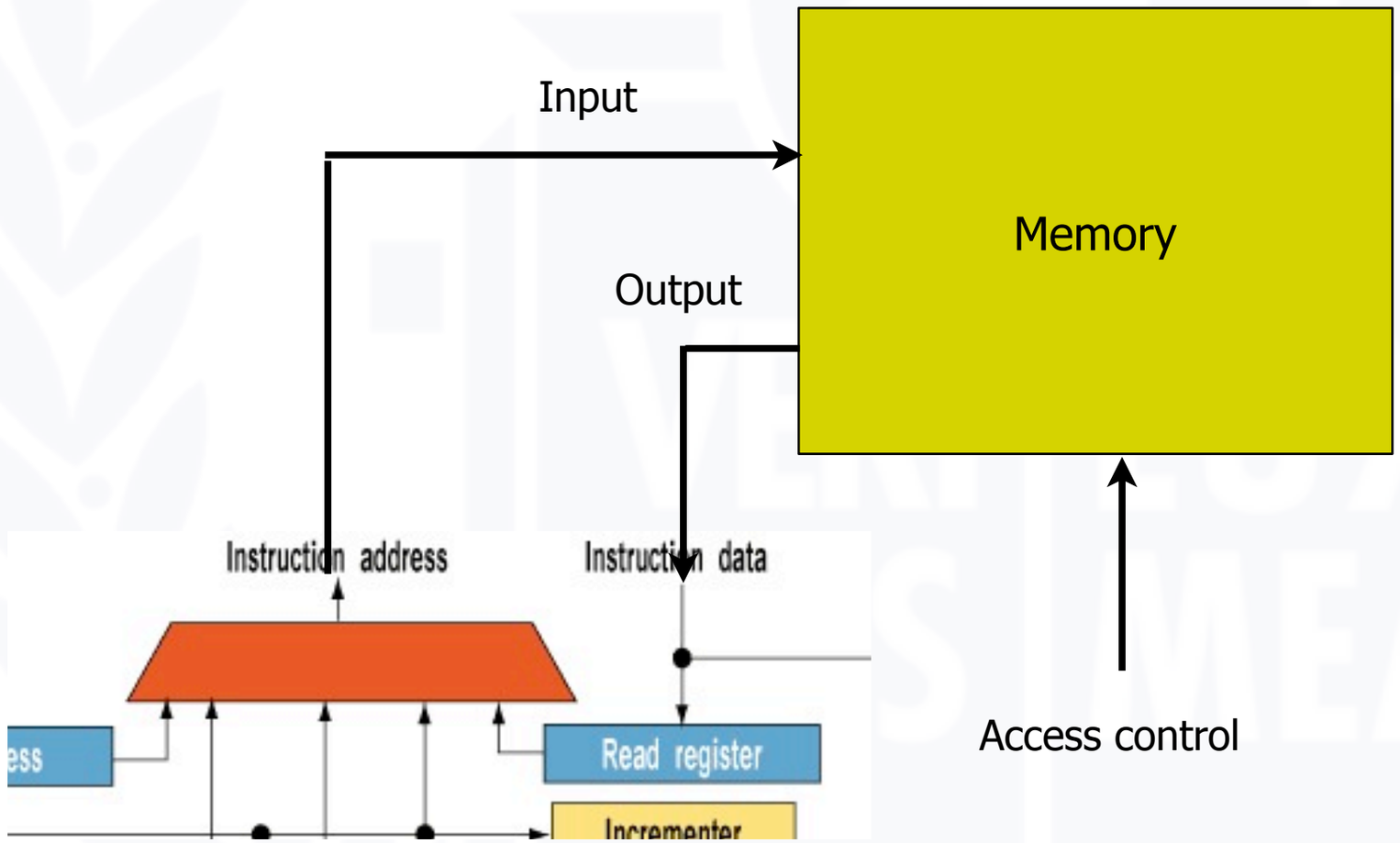
ARM Cortex M3 architecture




The Cortex M3's Thumbnail architecture looks like a conventional Arm processor. The differences are found in the Harvard architecture and the instruction decode that handles only Thumb and Thumb 2 instructions.

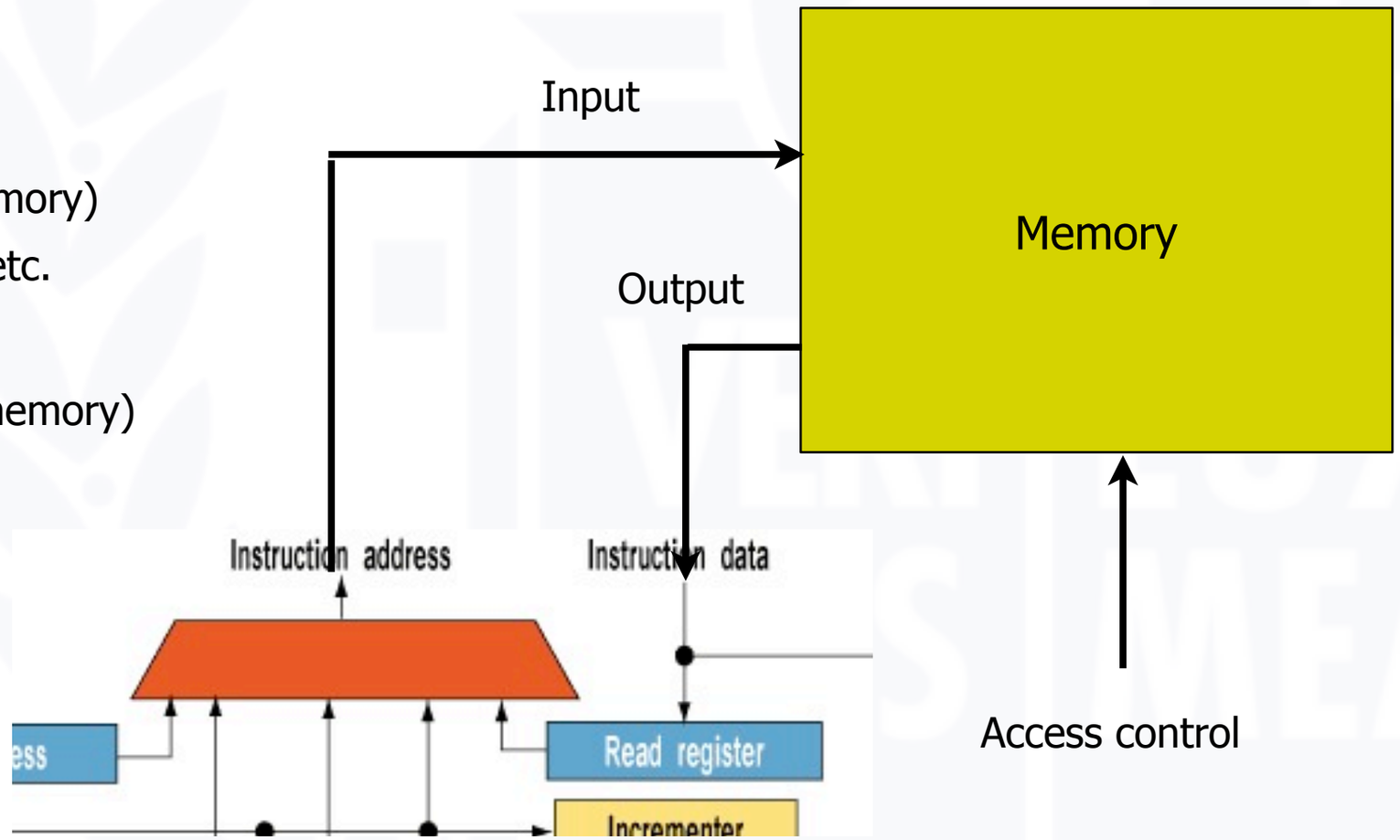
# Memory Interface

- Interface
  - Address bus
  - Data bus
  - Control signals (synchronous and asynchronous)
- Fully static read operation



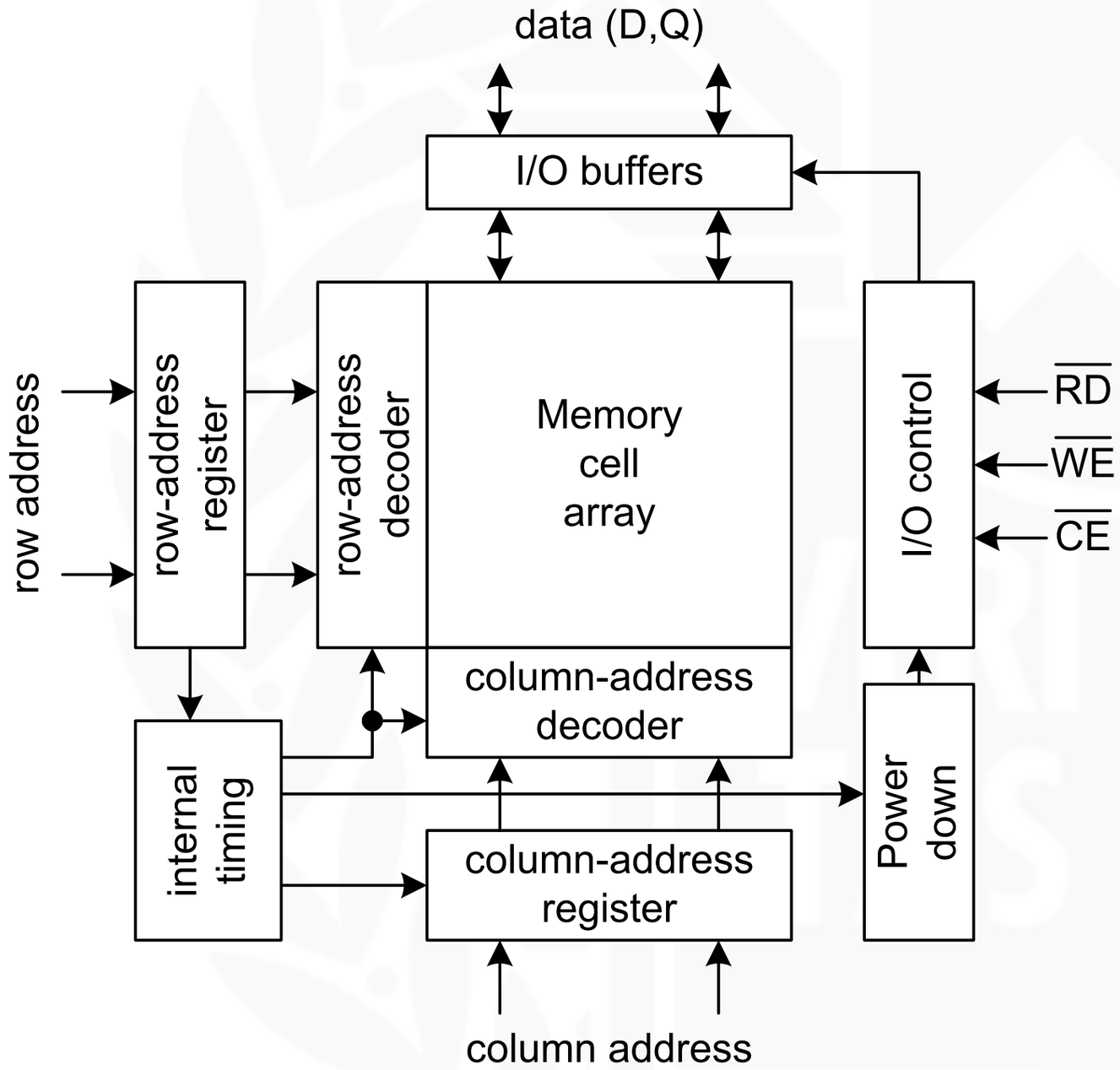
# Memory Interface

- Memory device
  - Collection of memory cells: 1M cells, 1G cells, etc.
  - Memory cells preserve stored data
    - Volatile and non-volatile
    - Dynamic and static
- How access memory?
  - Addressing
    - Normally address of the cell
      (cf. content addressable memory)
    - Random, sequential, page, etc.
  - Exclusive cell access
    - One by one (cf. multi-port memory)
  - Operations
    - Read, write, refresh, etc.
    - RD, WR, CS, OE, etc.



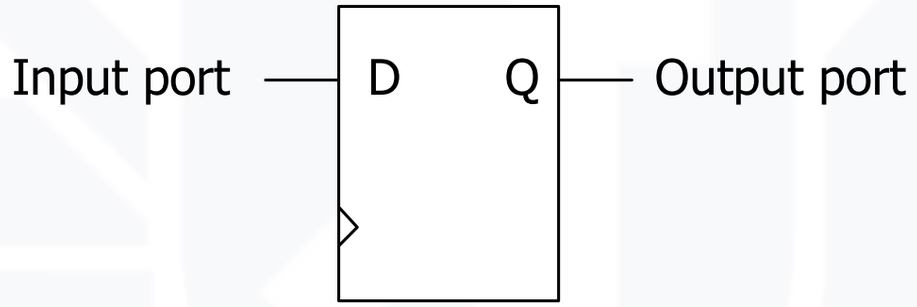
# Memory inside

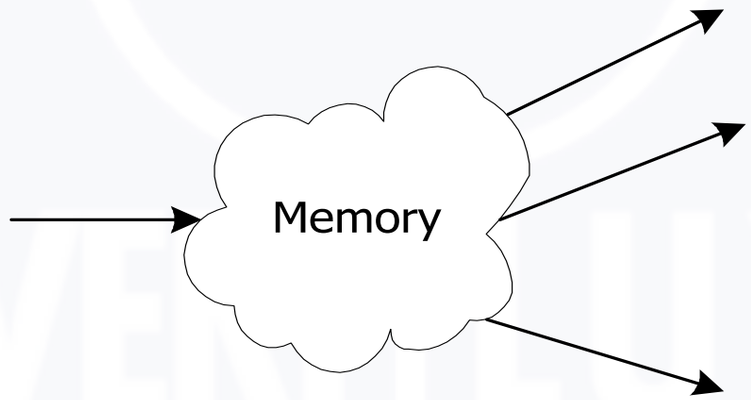
- SRAM structure

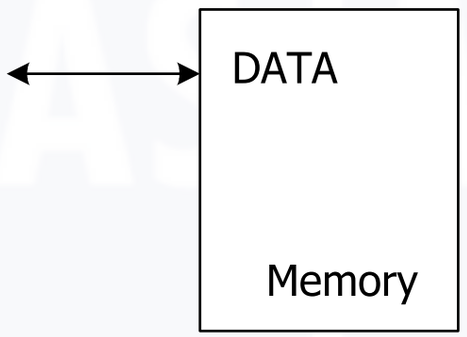


# Memory inside

- Ports
  - Recall D-FF
  - 1 input port and one output port for one cell
- Ports of memory devices
  - Large number of cells
  - One write port for consistency
  - More than one output ports allow simultaneous accesses of multiple cells for read
  - Register file usually has multiple read ports such as 1W 3R
  - Memory devices usually has one read and one write port or shares one port both for read and write



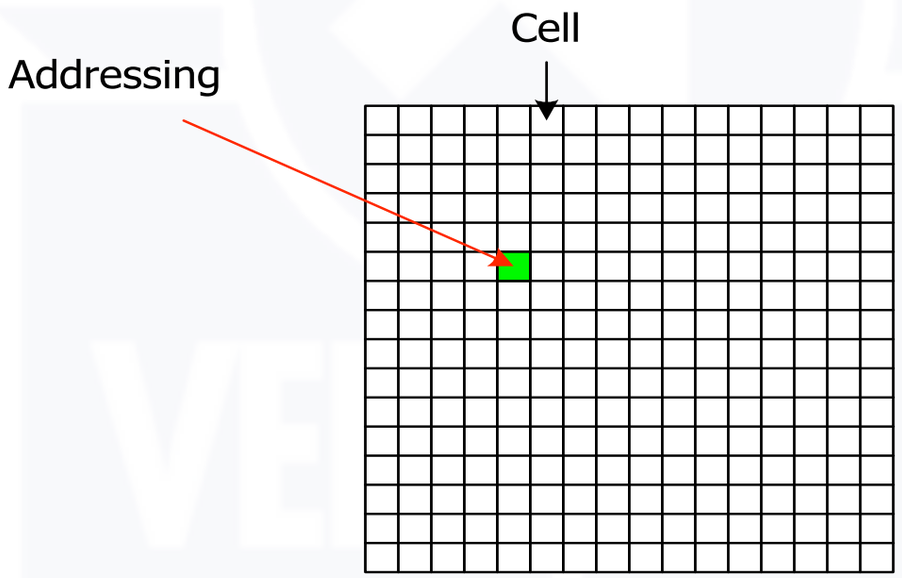



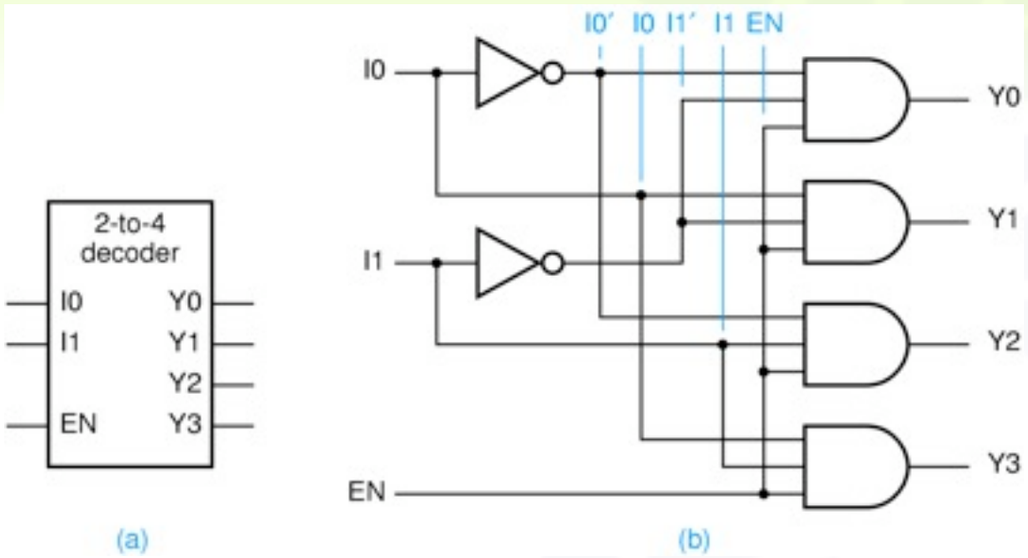
# Memory inside

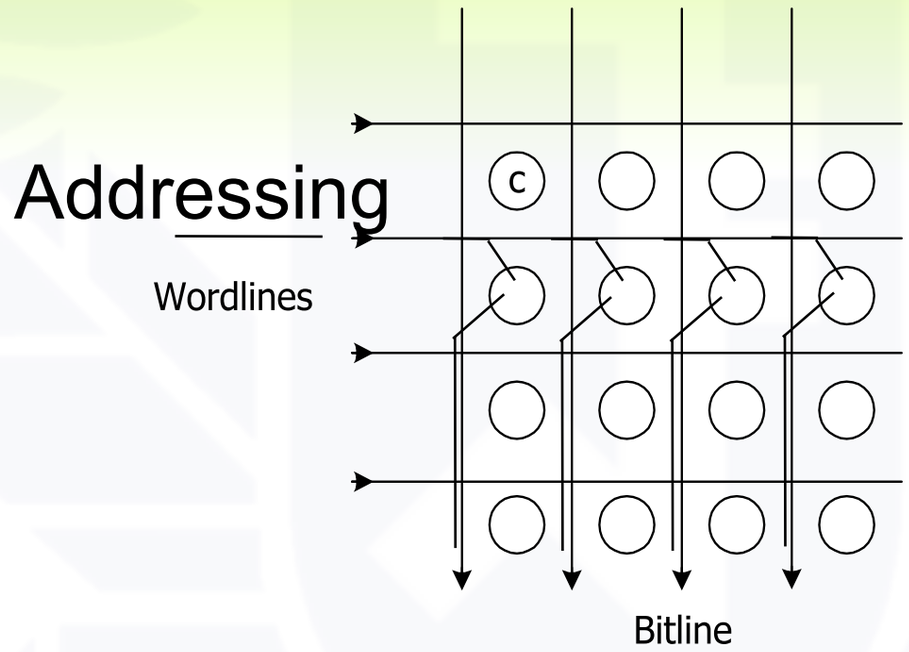
- Address
  - Single port memory allows one cell access at a time
    - Exclusive addressing
  - Why single port?
    - Cost, cost, cost…

- Super computers
  - Use multiple read port main memory
    - Y=A+B where A=($1000), B=($2000) and Y=($4000)



# Memory inside

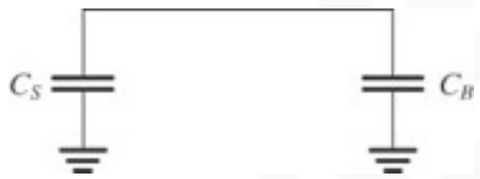


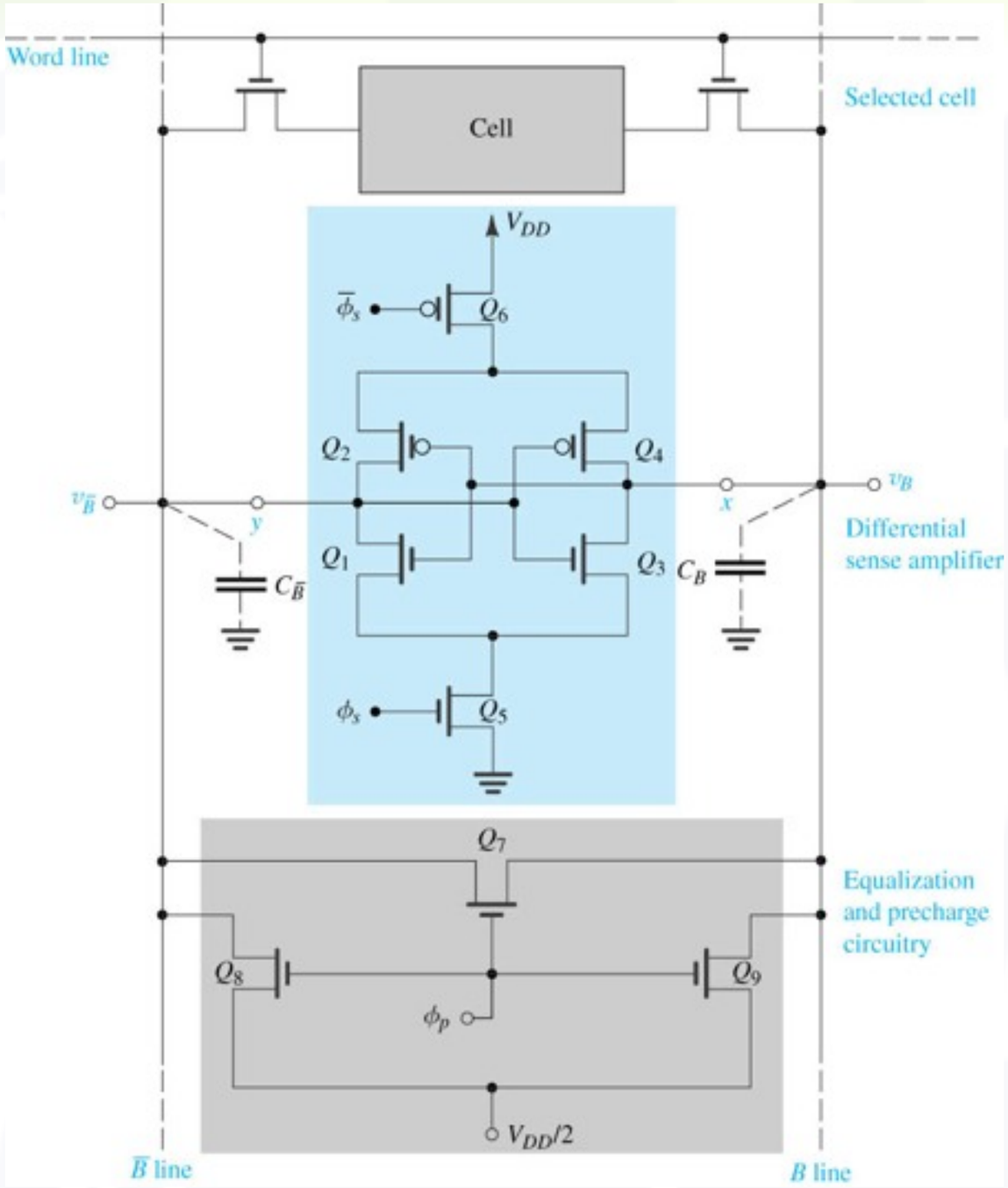

- Wordline
  - Selection (addressing) of cells connected to the same wordline
  - Exclusively selected by a decoder
- Bitline
  - Cell values are transferred via the bitlines
  - Parallel architecture
- Address encoding
  - Non-redundant Address value encoding using binary numbers
- Address decoding
  - Large-size binary decoder: 28-to-268435456 binary decoder for 256Mb memory

# Memory inside

- We need an amplifier
  - Memory cell is a small transistor (capacitor)

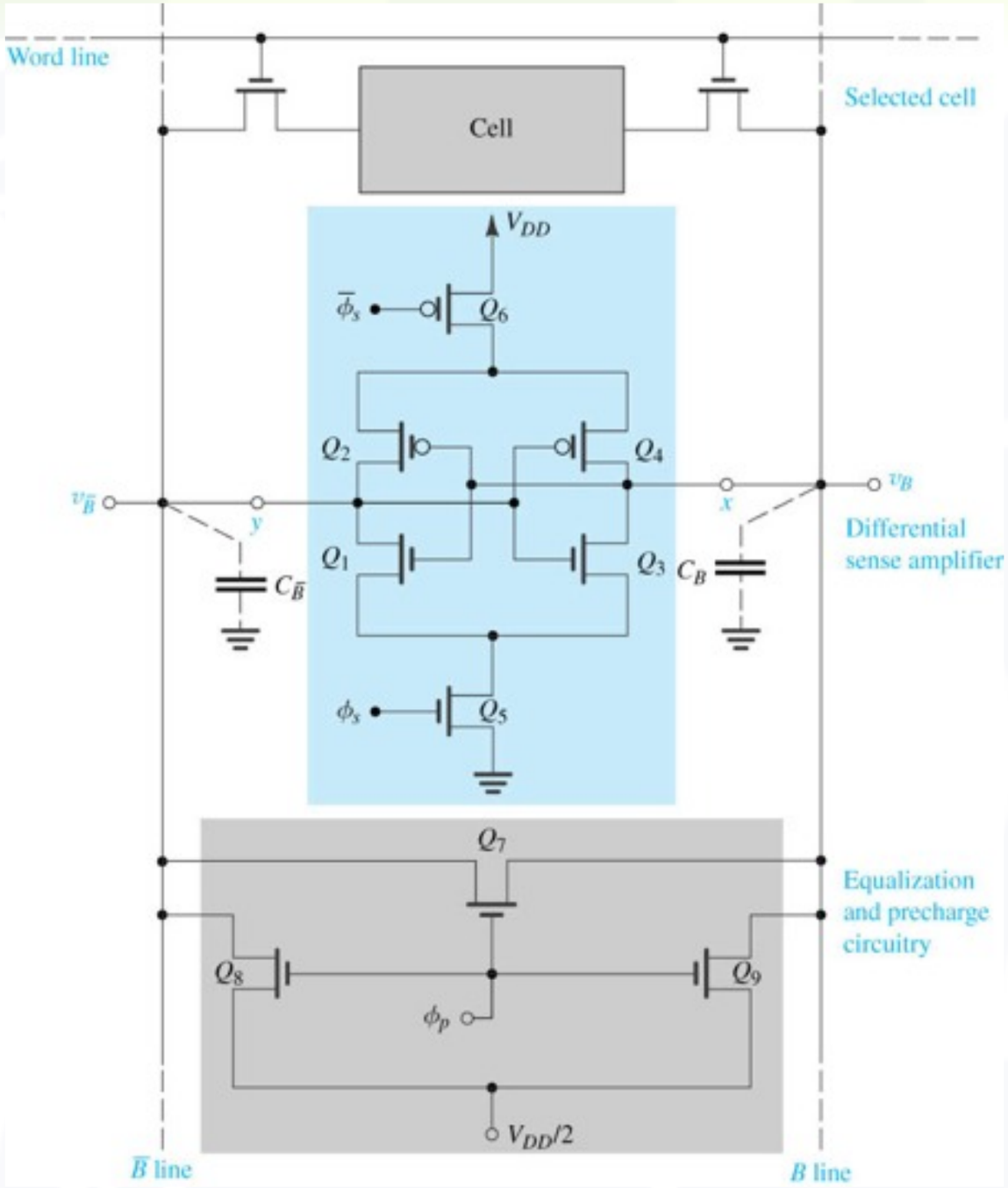


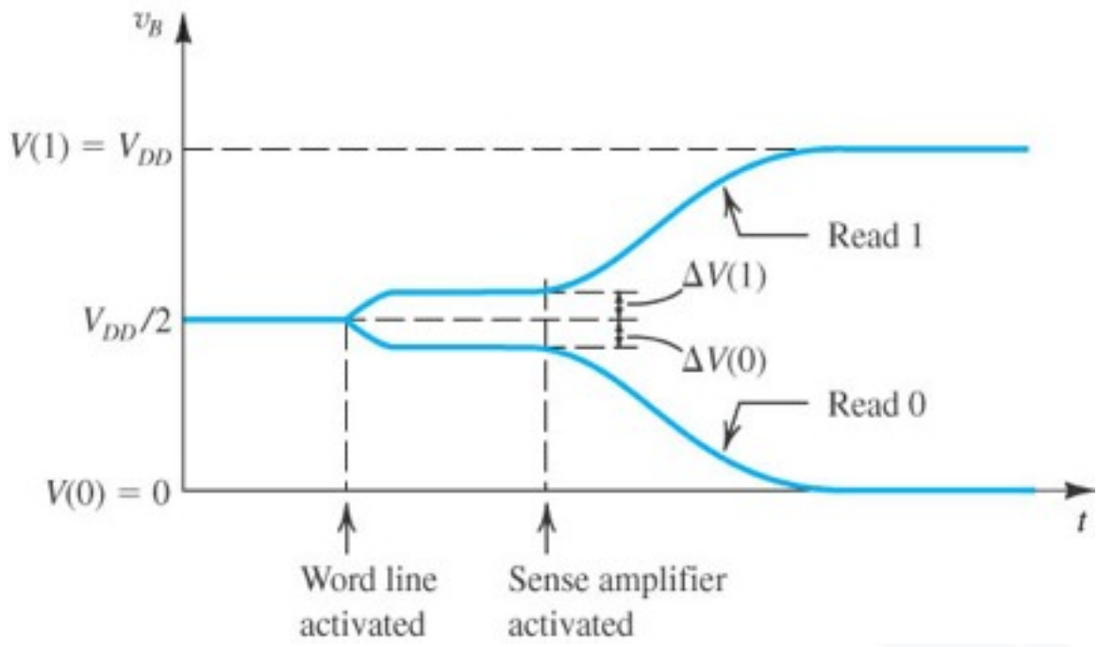

  - Bitline is a big capacitor
  - Connecting a memory cell to the bitline cause only small amount of voltage change
- Differential sense amplifier



# Memory inside

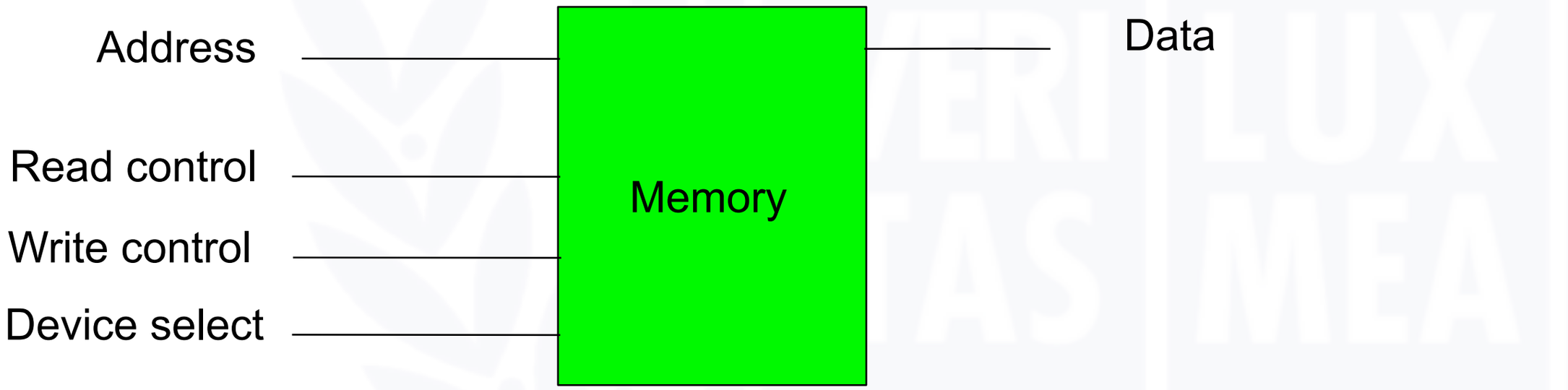
- Amplifiers whose input and output are connected together
  - Positive feedback



# Memory interface

- Microprocessor-compatible interface
  - Most memory devices
  - Binary-encoded cell address
  - One, four, eight, 16, 32 or 64 bit data, mostly bidirectional (common input and output)
  - Device select
  - Read/write control



# Memory interface

- Some alternatives
  - Multiplexed address
    - Higher address and lower address use the same input using time-division multiplexing for less pin count
    - Address strobes
  - Encoded read/write control
    - For example, H means read and L means write
  - No device select
  - Clock input for synchronous memory
  - Separate output enable control
  - Power management control

# Memory interface

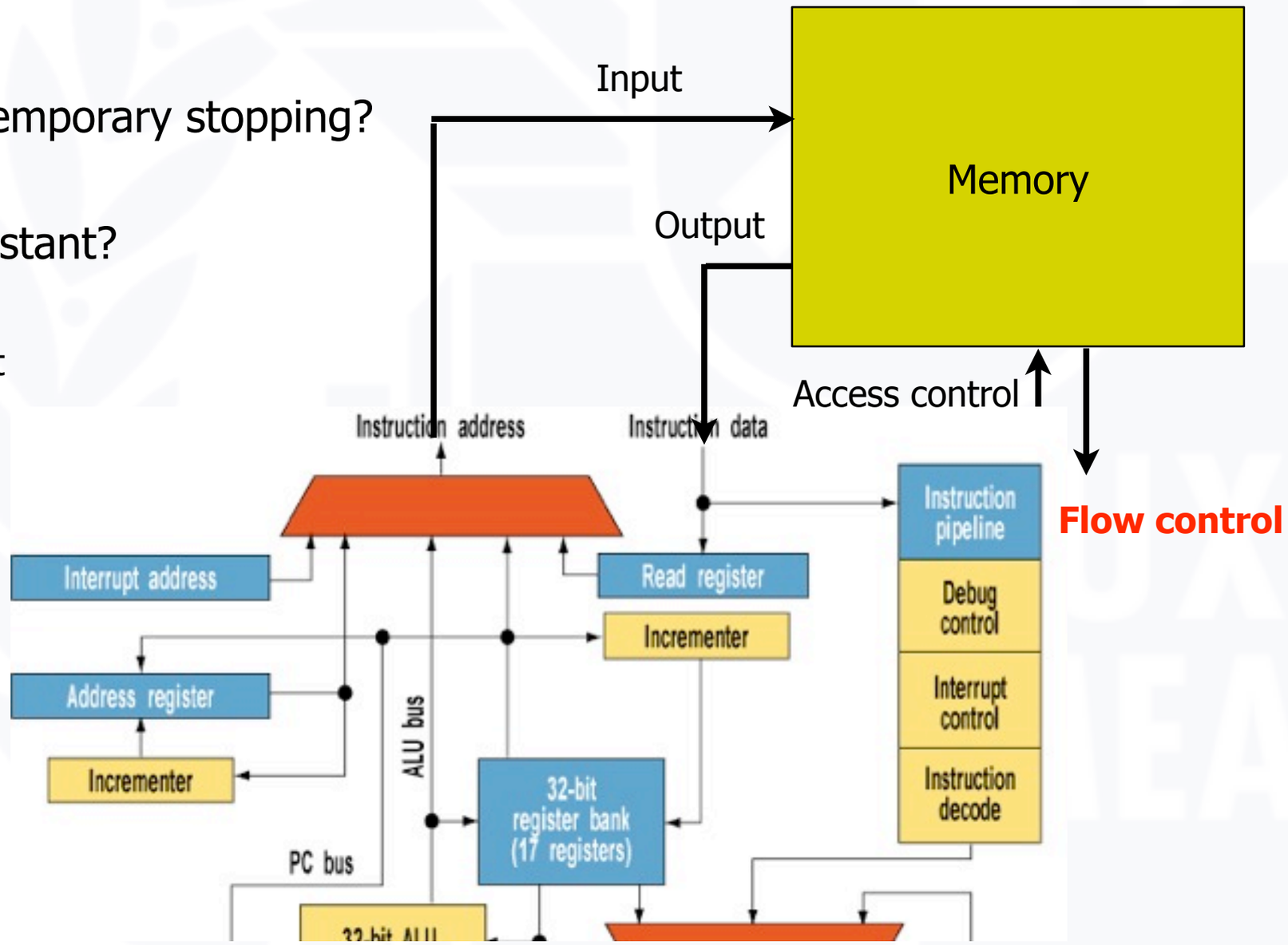
- Synchronous and asynchronous
  - Synchronous
    - Memory device operation is synchronous the bus clock
    - Fast operation
    - High throughput (no idle period between transfers)
    - Interface design is complicated
  - Asynchronous
    - No bus clock
    - Slow operation
    - Low throughput (idle period indicates termination of an access)
    - Interface is simple
  - SRAM (asynchronous), SSRAM (synchronous SRAM), DRAM (asynchronous) and SDRAM (synchronous DRAM)
- Detailed explanation later in timing section

# Memory interface

- Synchronous Bus:
  - Includes a clock in the control lines
  - A fixed protocol for communication that is relative to the clock
  - Advantage: involves very little logic and can run very fast
    - Each transmission takes 1 clock cycle
  - Disadvantages:
    - Every device on the bus must run at the same clock rate
    - To avoid clock skew, they cannot be long if they are fast
- Asynchronous Bus:
  - It is not clocked
  - It can accommodate a wide range of devices
  - It can be lengthened without worrying about clock skew
  - It requires a handshaking protocol
  - Handshake requires one or more clock cycles

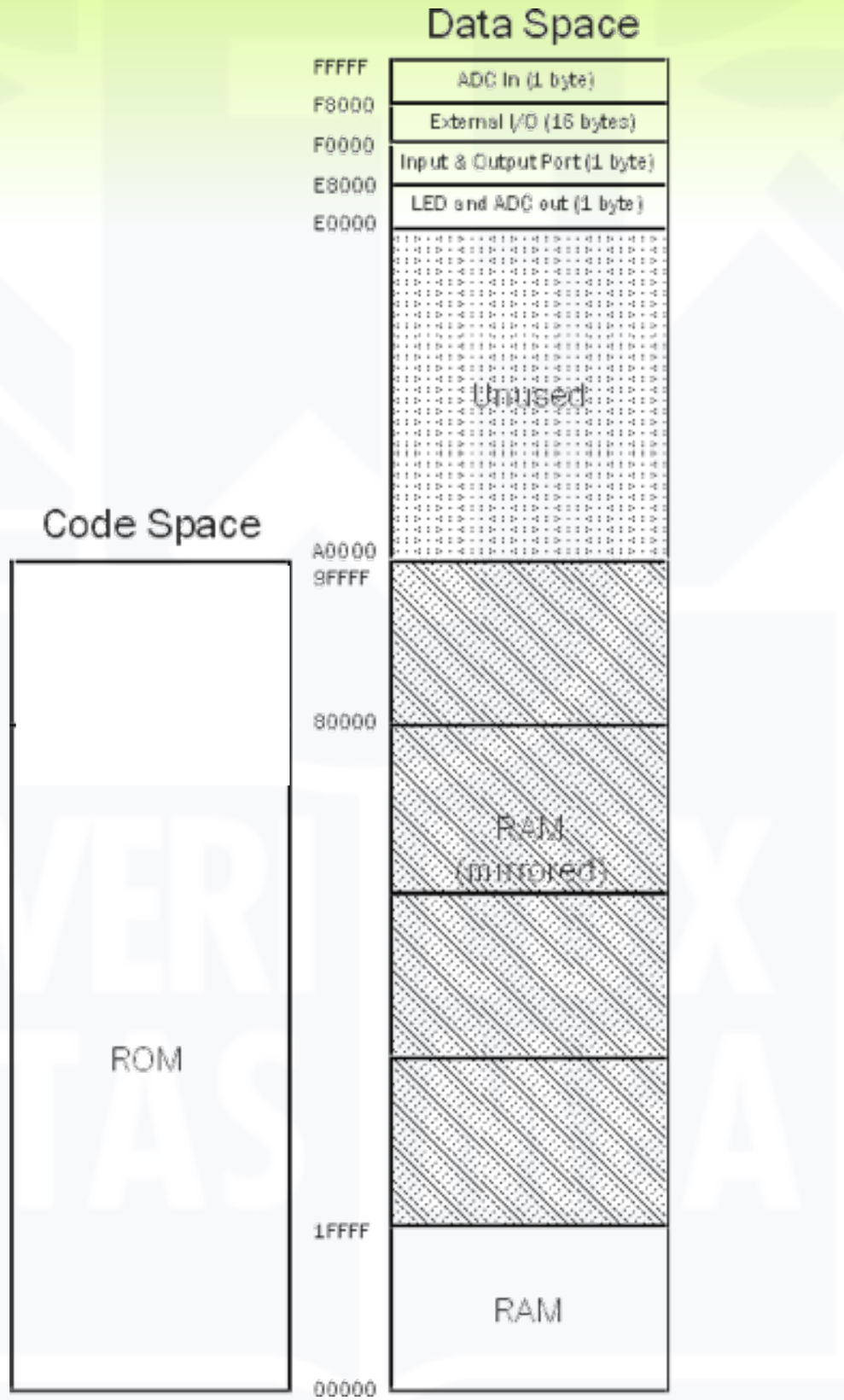
# Flow control

- Memory is a passive device
  - Access control is done by the CPU
    - Logical and timing control
- Need a flow control?
  - Is the CPU running w/o temporary stopping?
    - CPU is a master device
  - Is the memory speed constant?
    - SRAM and DRAM
    - Single port and multiple port
- Wait signal for CPU stall



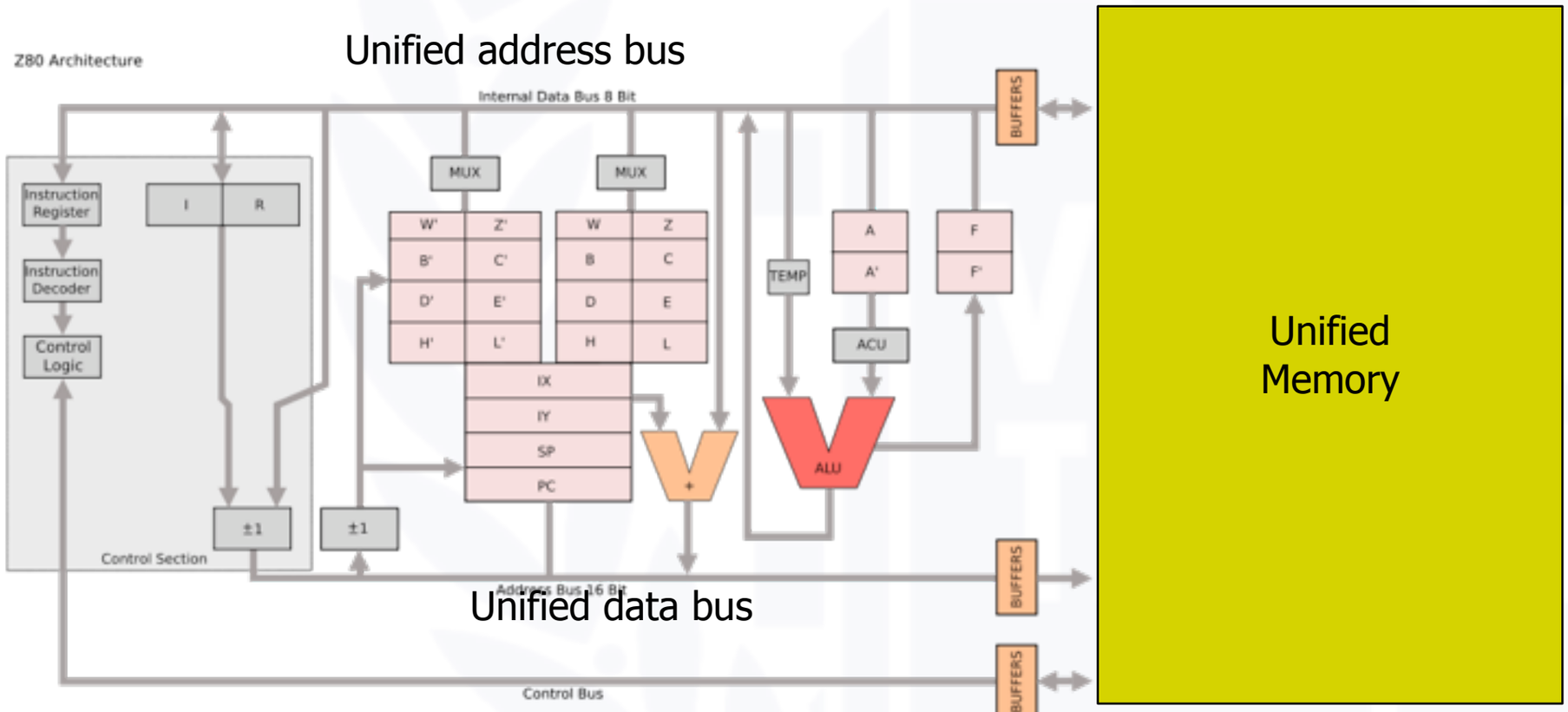
# Memory Maps

- Memory maps for harvard architecture processors
  - Separate instruction (code) and data memory
  - Two memory maps



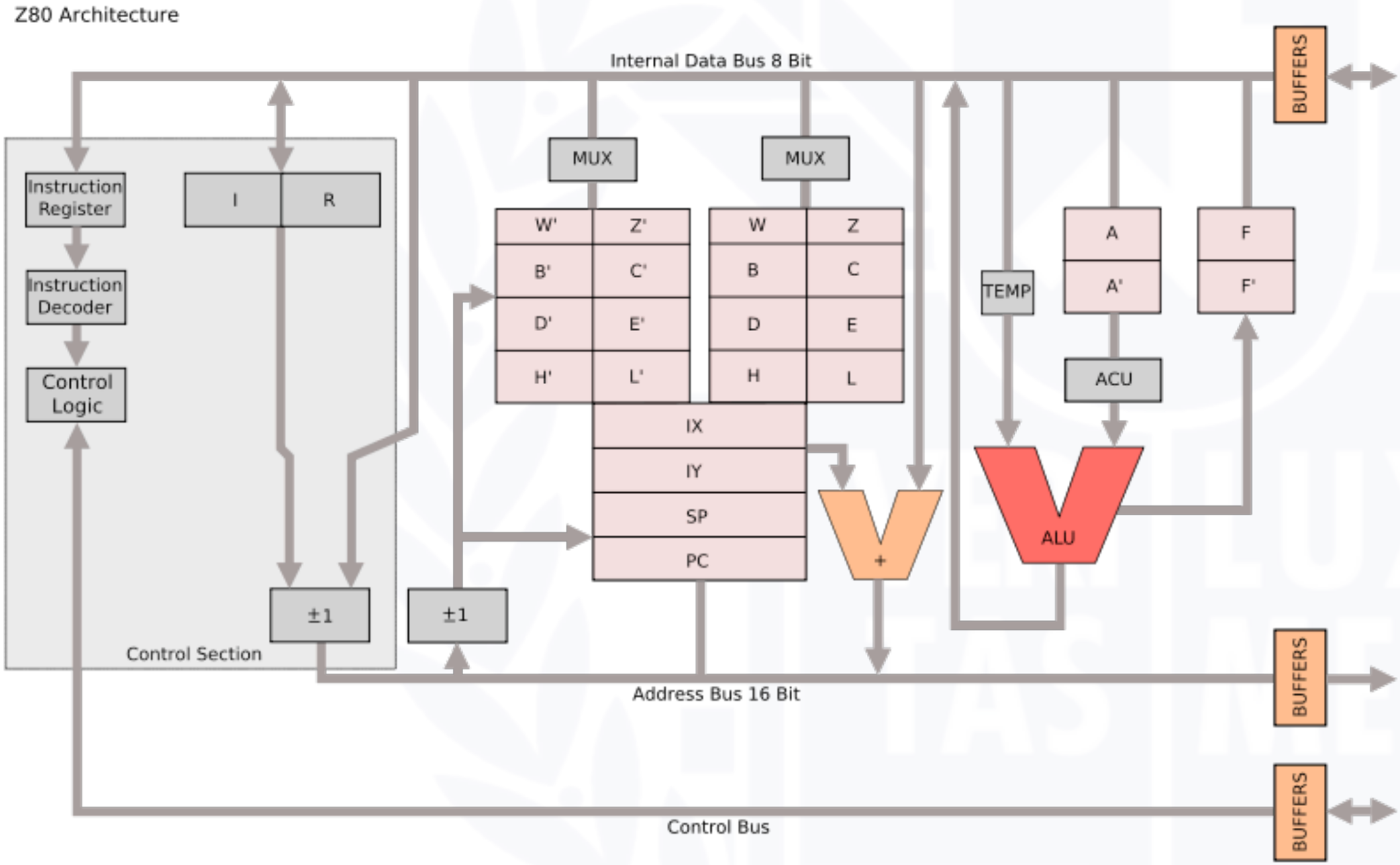
# Unified Memory Architecture

- Harvard architecture is fast but expensive
  - Not 100% memory utilization of non-pipelined microprocessors
  - Too many I/O pins
- Unified memory architecture
  - Sharing instructions and data with a unified memory
  - Need more complicated flow control
    - Instruction and data



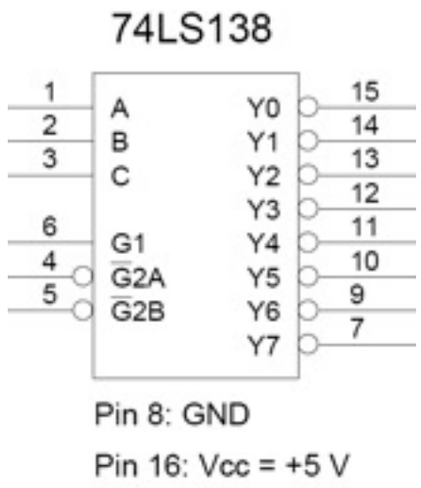
# Unified Memory Architecture

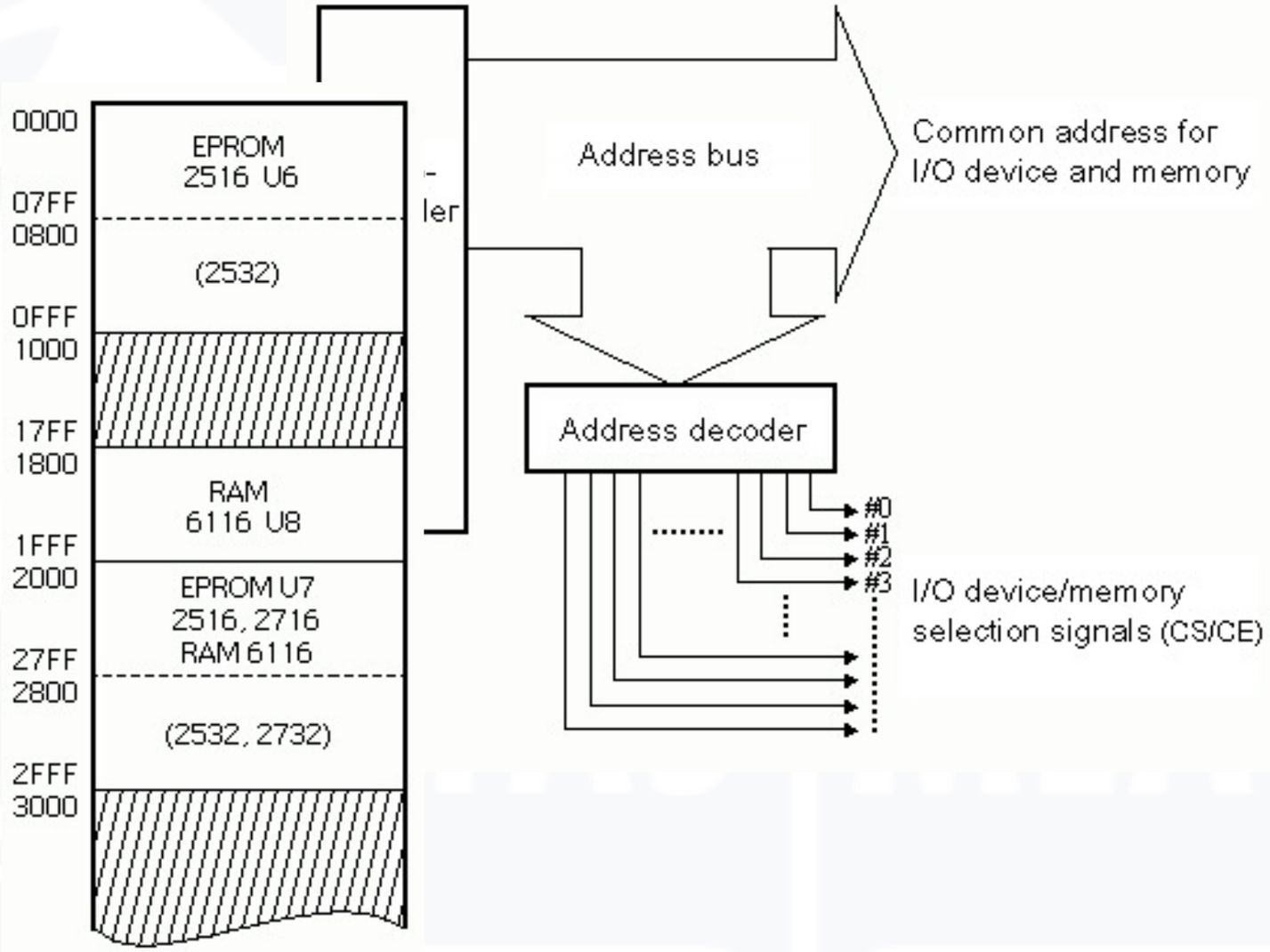
- Z80 microprocessor



# Address Decoding

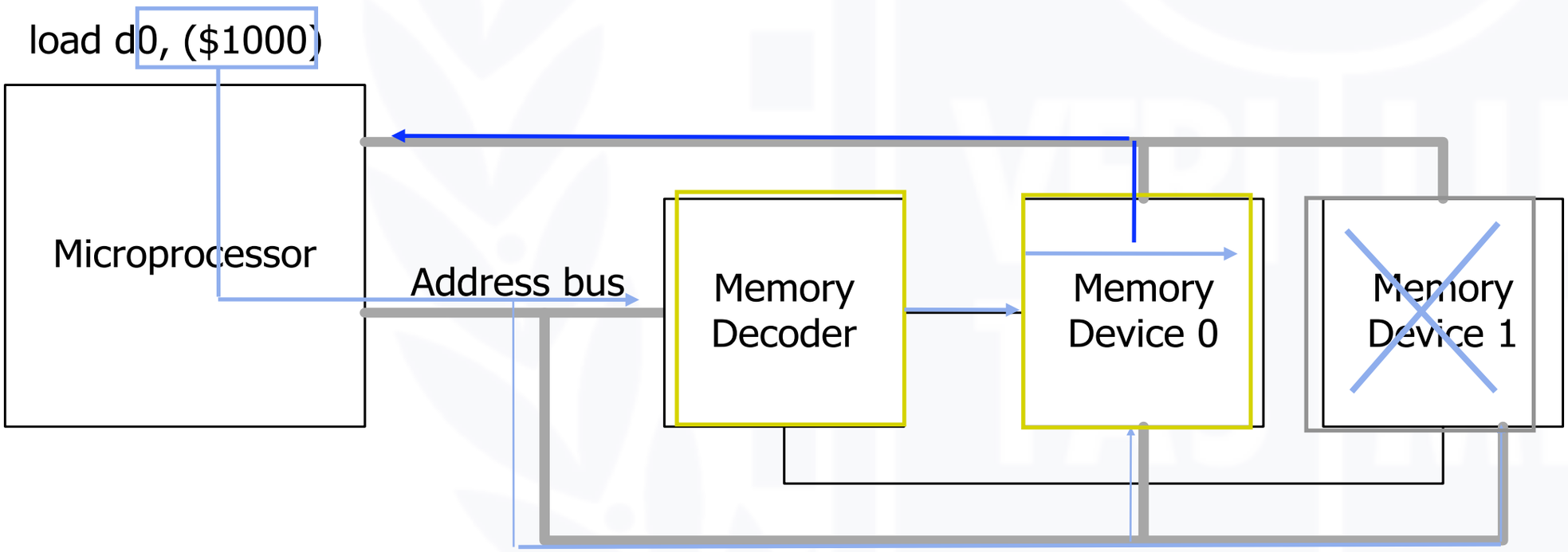
- Design a memory system for a Z80 microprocessor
  - Single unified memory device
    - Memory map
  - More number of memory devices
    - Address decoding
    - Memory maps



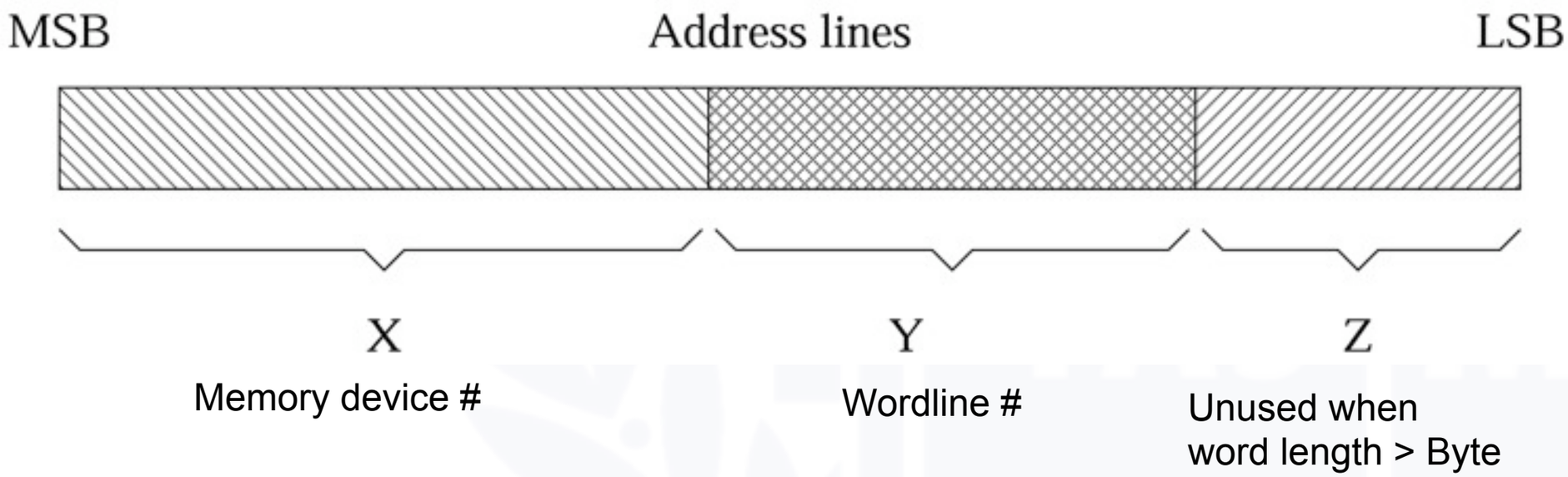
# Memory interface

- Microprocessor generates memory request
  - Instruction fetch and data fetch/write back
  - Microprocessor generates electrical signal changes on the address bus
  - Memory decoder activates and selects a specific memory device
  - Selected memory device is activated and the target wordline is selected
  - Target memory cells are selected and bitline values are updated
  - Bitline values are transferred to the microprocessor via data bus



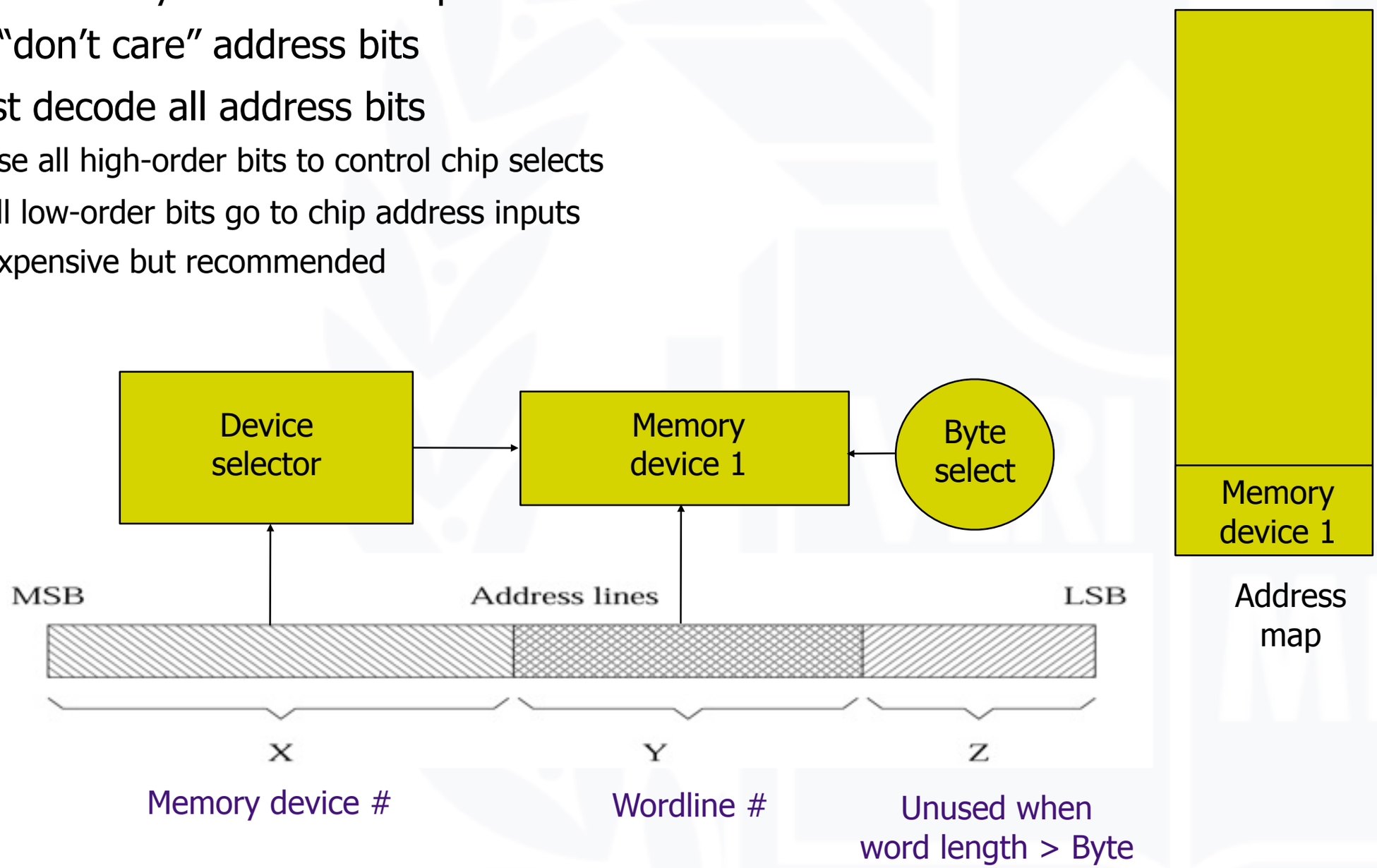
# Memory interface

- Address mapping
  - Design is simplified by partitioning the address lines (M X N memory with D X W memory chips)
    - Z bits are not connected (Z = log2(N/8))
    - Y bits are connected to all chips (Y = log2D)
    - X remaining bits are used to map the memory block
      - Used to generate chip selects



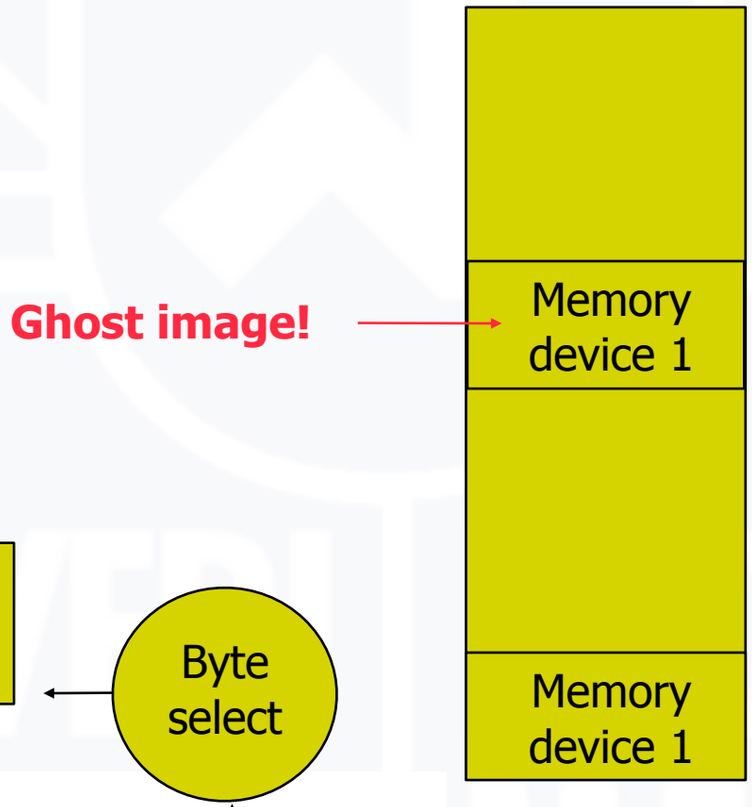
# Memory interface

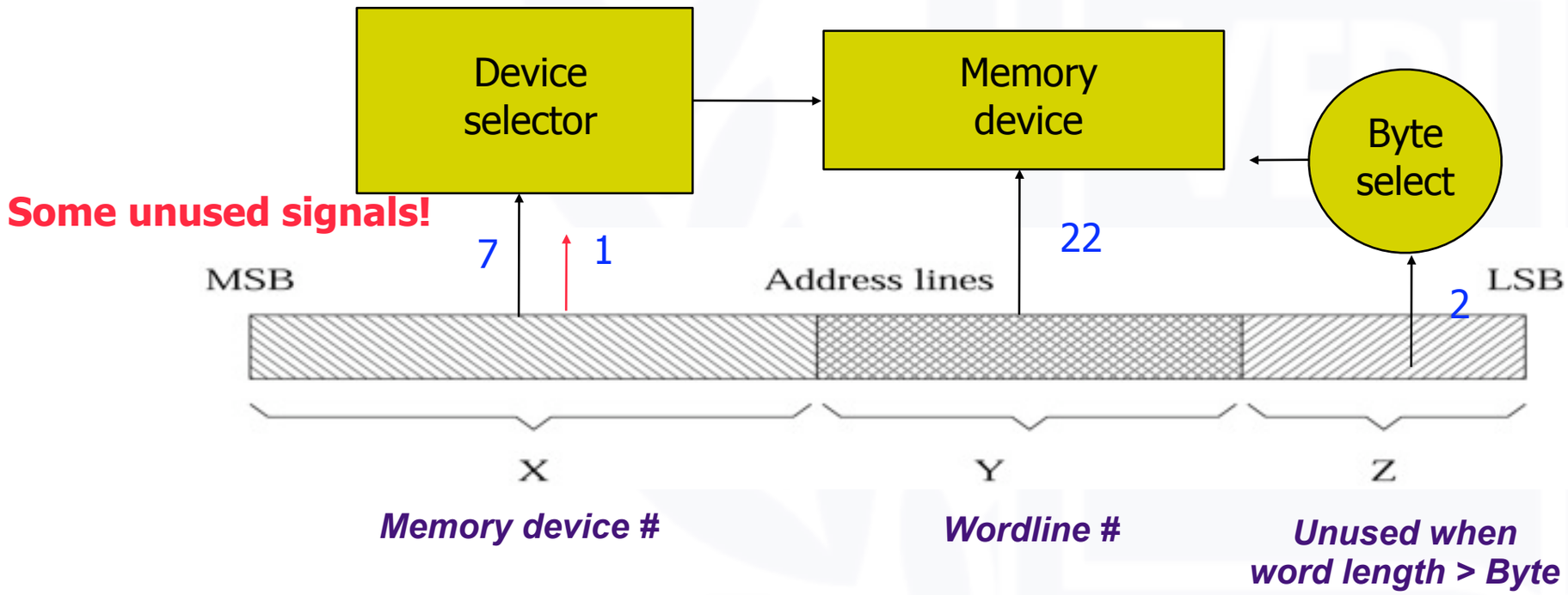
- Full Address Decoding
  - No unused address signals
  - Each memory location corresponds to one address
  - No "don't care" address bits
  - Must decode all address bits
    - use all high-order bits to control chip selects
    - all low-order bits go to chip address inputs
    - Expensive but recommended



# Memory interface

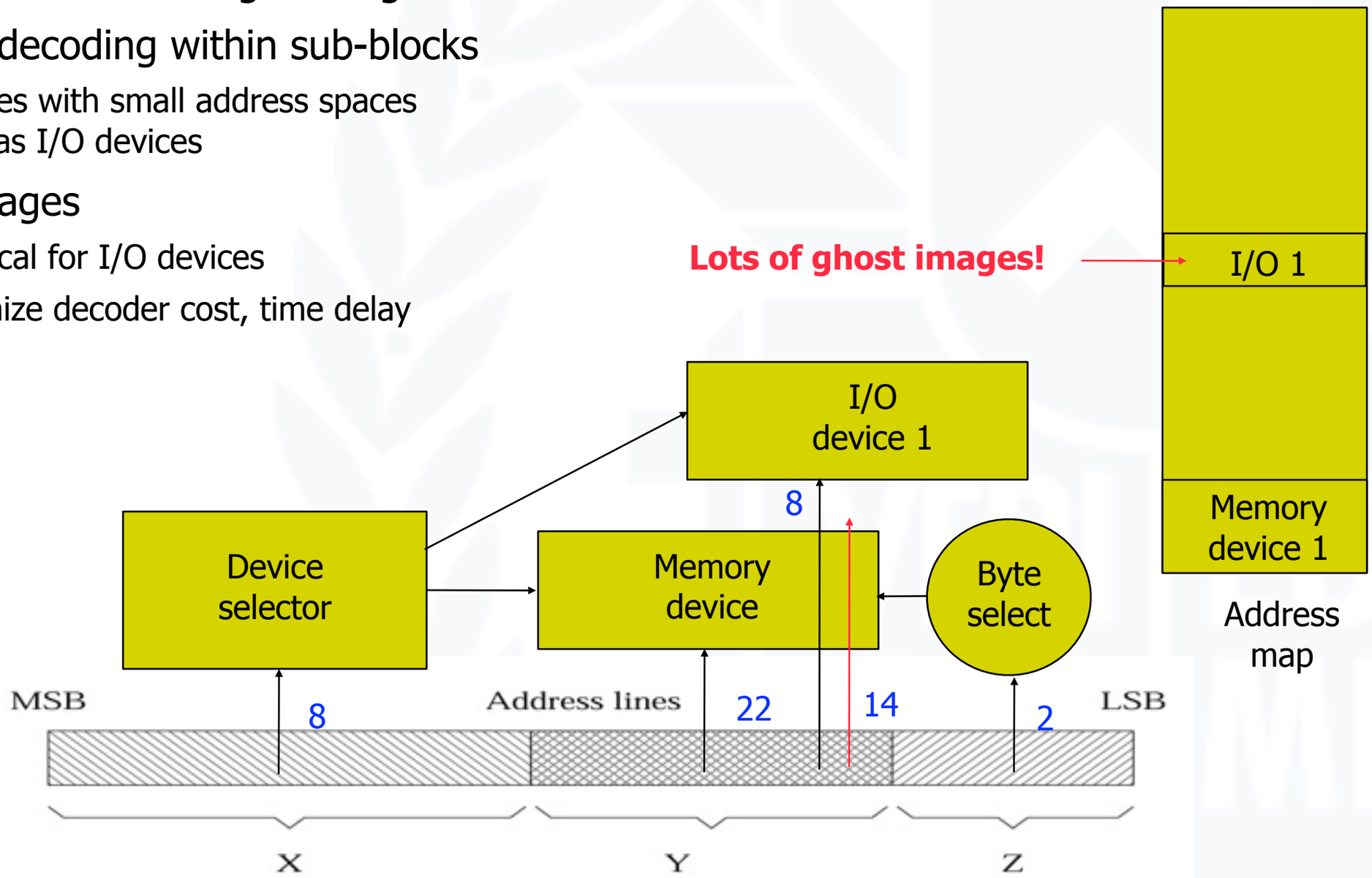
- Partial Address Decoding
  - Some address lines not decoded
  - There exist "don't care" values
  - Multiple addresses have same decoding
  - Advantage
    - Cheap
  - Disadvantage
    - Multiple mapping
    - Error prone - accidental illegal memory access



# Memory interface

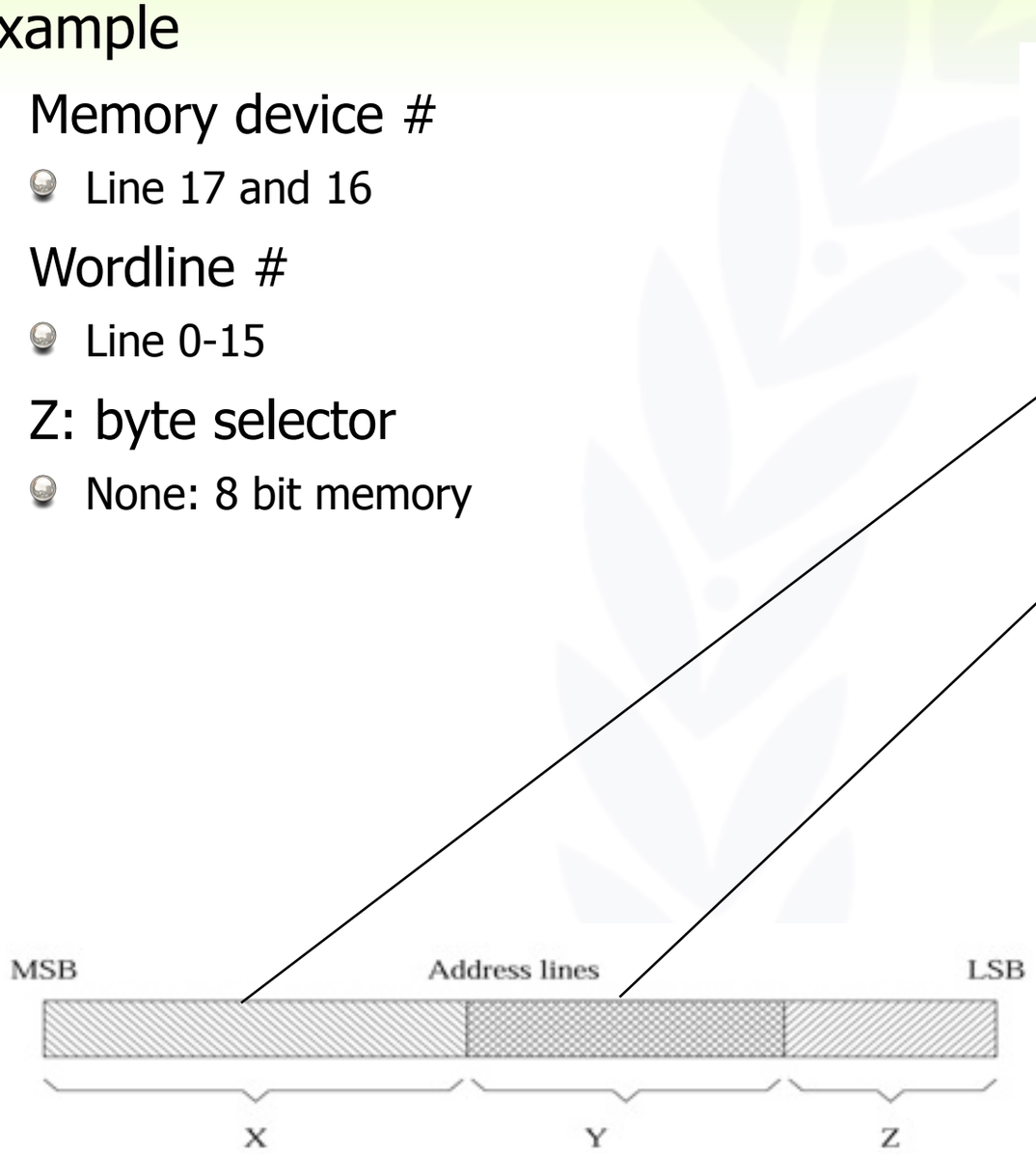
- Mixed Decoding Schemes
  - Full address decoding for large blocks
  - Partial decoding within sub-blocks
    - Devices with small address spaces such as I/O devices
  - Advantages
    - Practical for I/O devices
    - Minimize decoder cost, time delay



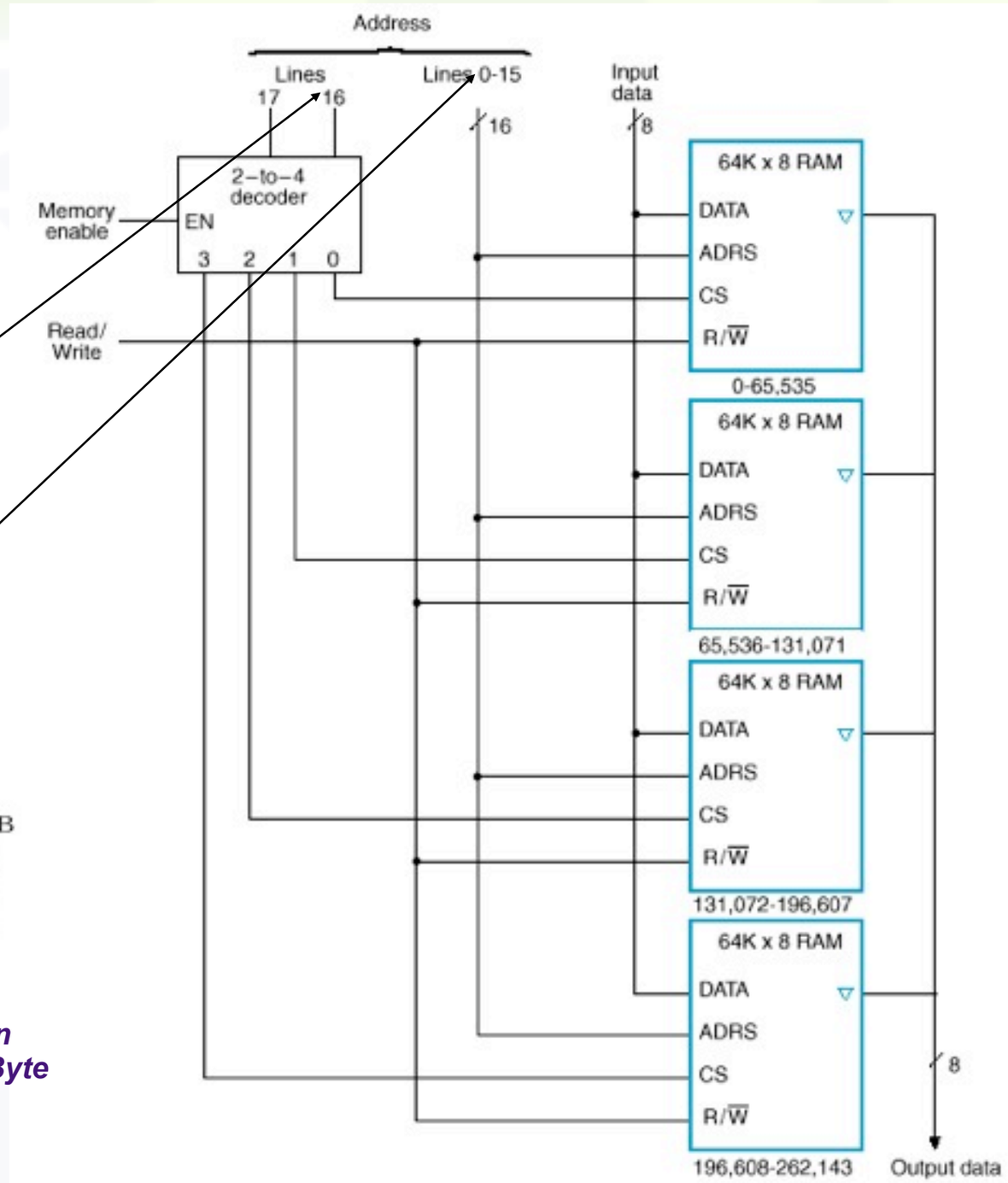

**Some unused signals for I/O device 1!**

# Memory interface

- Example
  - Memory device #
    - Line 17 and 16
  - Wordline #
    - Line 0-15
  - Z: byte selector
    - None: 8 bit memory




*Memory device #* *Wordline #* *Unused when word length > Byte*



# Memory interface

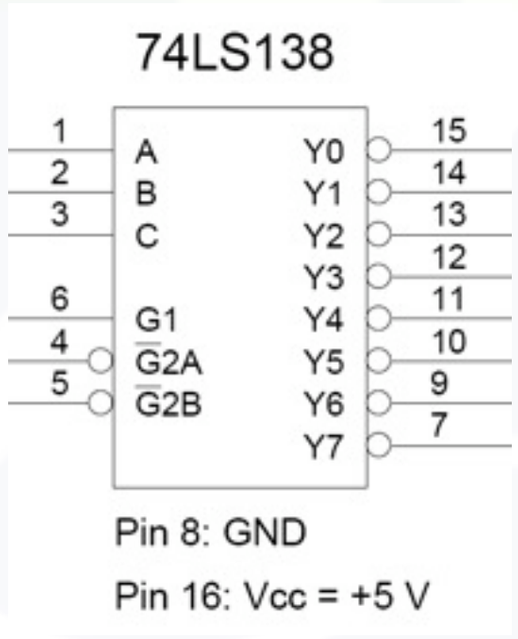
- Address Decoder Design
  - Approaches
    - Random logic
    - m-to-n-line decoder
    - PROM
    - Programmable array logic (PAL)
    - Field programmable gate array (FPGA)
  - Issues
    - Cost
    - Chip count
    - Speed
    - Design change

# Memory interface

- Address Decoder Design
  - Random Logic
    - Implement decoding in SSI gates
      - Boolean function to recognize address
    - Pros and cons
      - Fast (below 1ns delay per gate)
      - Low power and small foot print
      - Could take many chips when the decoder size is large
        - Best suited for very simple partial decode
      - Circuit board redesign for modification
      - Mostly used in old systems
        - Complementary usage for modern systems
        - Single gates

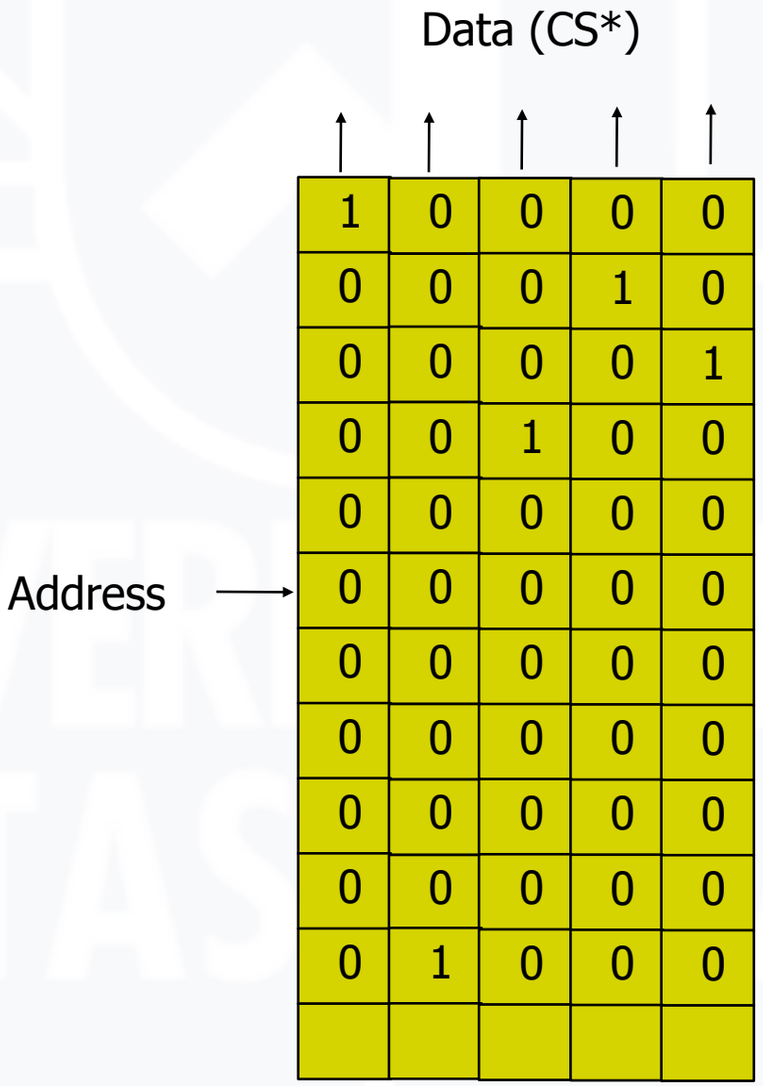
# Memory interface

- Address Decoder Design
  - M-To-N-Line Decoder
    - Convert m-bit binary code to one of n outputs
      - n = 2m
    - Famous decoder families
      - 74LS154 - 4-to-16-line decoder
      - 74LS138 - 3-to-8-line decoder
      - 74LS139 - dual 2-to-4-line decoder
    - Additional enable inputs permits decoders to be cascaded
    - Pros and cons
      - Relatively fast and low power
      - Could take many chips when the decoder size is large
        - Best suited for very simple partial decode
      - Circuit board redesign for modification
      - Mostly used in old systems
        - Complementary usage for modern systems



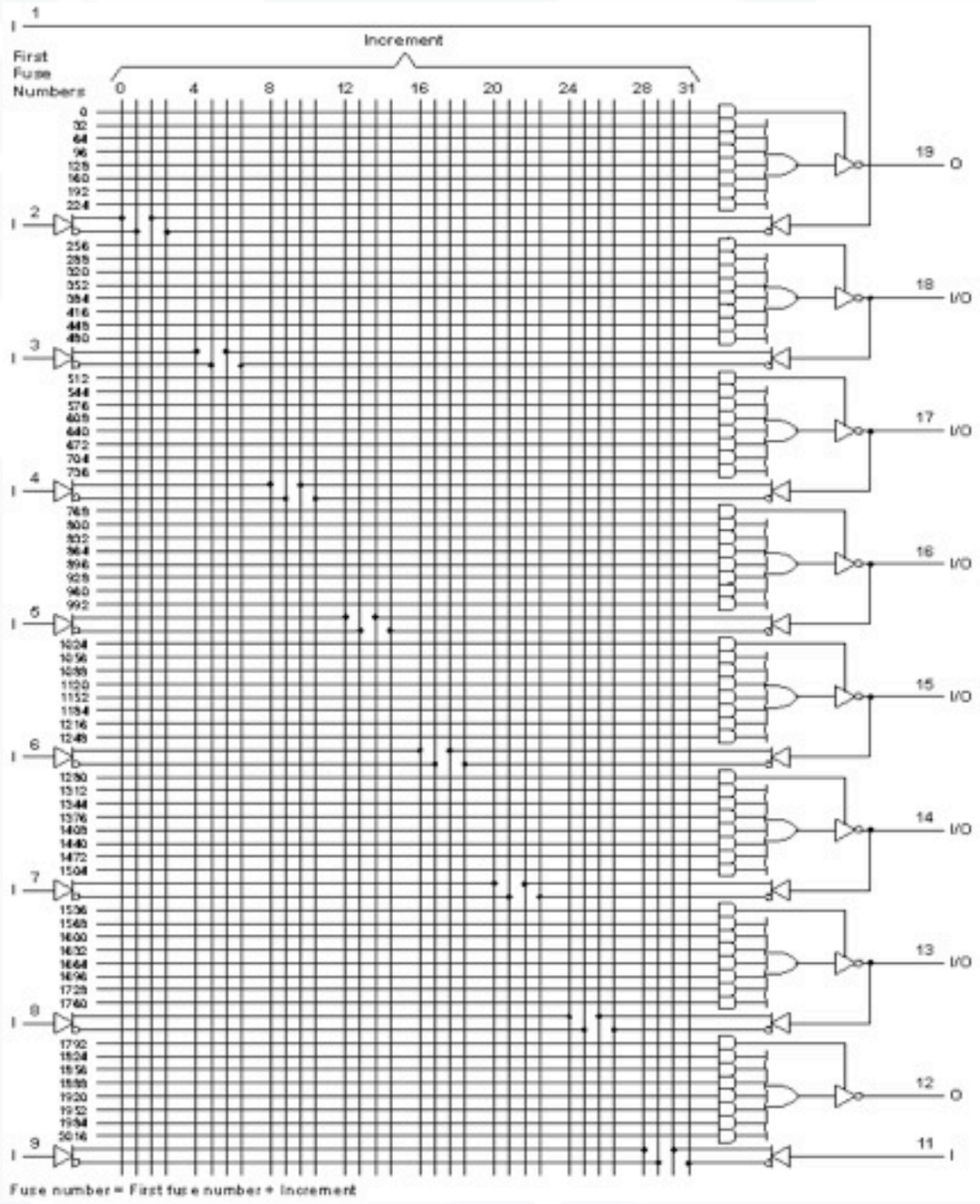
# Memory interface

- Address Decoder Design
  - PROM Decoding
    - Lookup table
    - Put 1 of 2m p-bit words on output when selected
      - m address inputs
      - p data outputs
      - Chip select input
    - Implement decoding truth table directly
    - Pros and cons
      - Very inefficient gate usage for large size decoder
        - n inputs, 2^n products and m outputs
        - Decoder does not require OR plane
      - Slow and high power
      - Flexible
      - Old fashioned



# Memory interface

- Address Decoder Design
  - PLA Decoding
  - Reduced OR plane from PROM
    - PROM
      - n inputs, 2^n products and m outputs
    - PAL
      - n inputs, k products and m outputs
    - Decoder does not require OR plane
      → still inefficient but far better than PROM
  - Pros and cons
    - High speed
    - High power but some low power versions are available
    - Expensive
    - Flexible
- CPLD → array of PALs

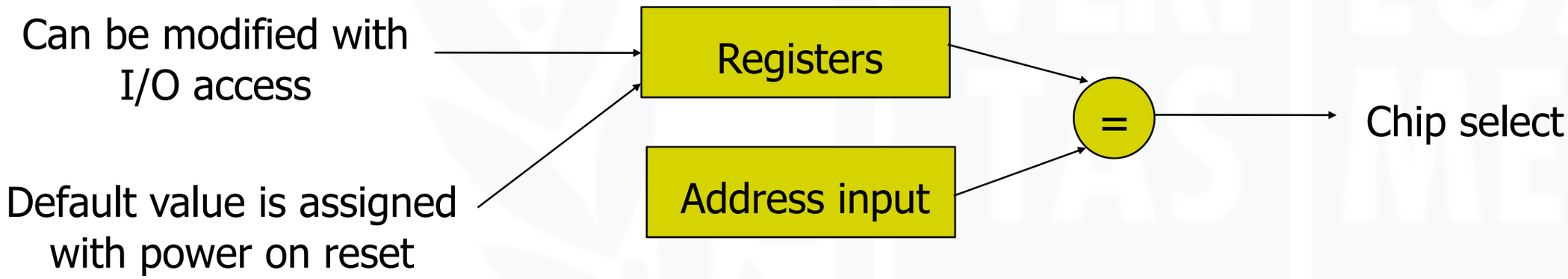


# Memory interface

- Address Decoder Design
  - FPGA Decoding
    - Look-up table logic architecture with 4 to 5 variables
      - Extremely inefficient for large size decoders
      - Very slow for large-size decoders
      - Internal special logics for large-size decoders
    - Not solely used for decoders
      - FPGA is a big, expensive device

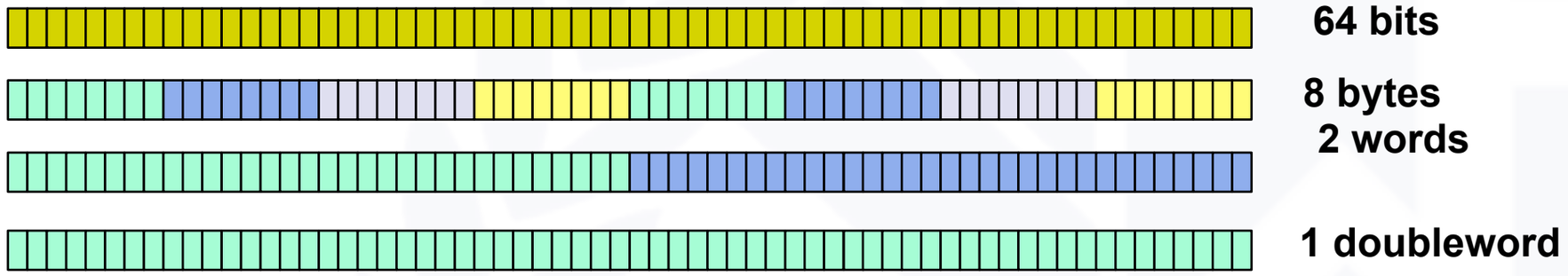
# Memory interface

- Reconfigurable address decoder
  - Modern embedded processors have built-in programmable address decoders
  - All the embedded peripherals have built-in decoders
  - Several built-in decoders for external peripherals
  - Additional decoders can be implemented by
    - Single random gates
    - SSI decoders
    - PALs or FPGAs

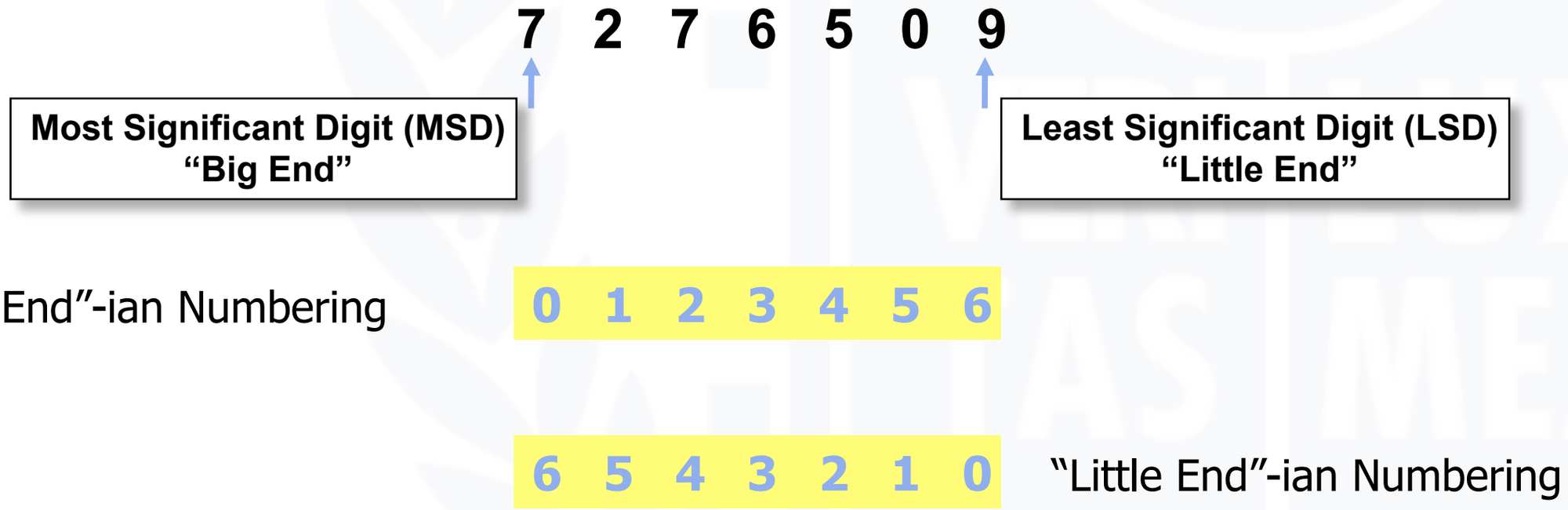


# Memory interface

- Notations for Information Representation



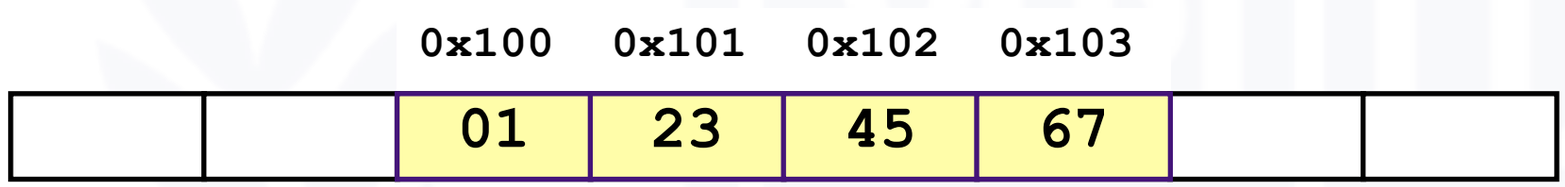

- How do we number these various units of information in a consistent manner



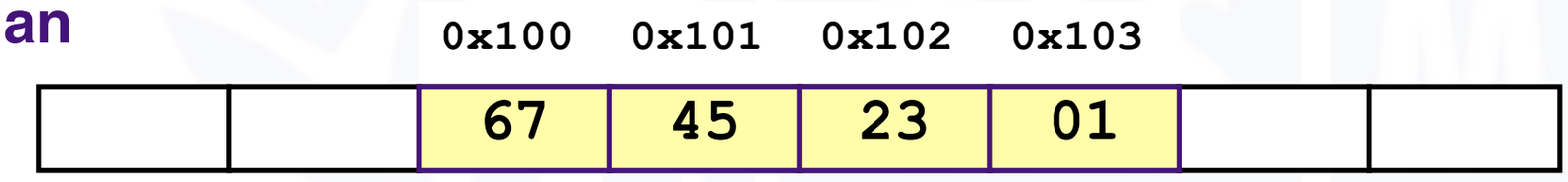
# Memory interface

- Big Endian
  - Least significant byte has highest address
- Little Endian
  - Least significant byte has lowest address
- Example
  - Variable x has 4-byte representation 0x01234567
  - Address given by &x is 0x100

**Big Endian**




**Little Endian**

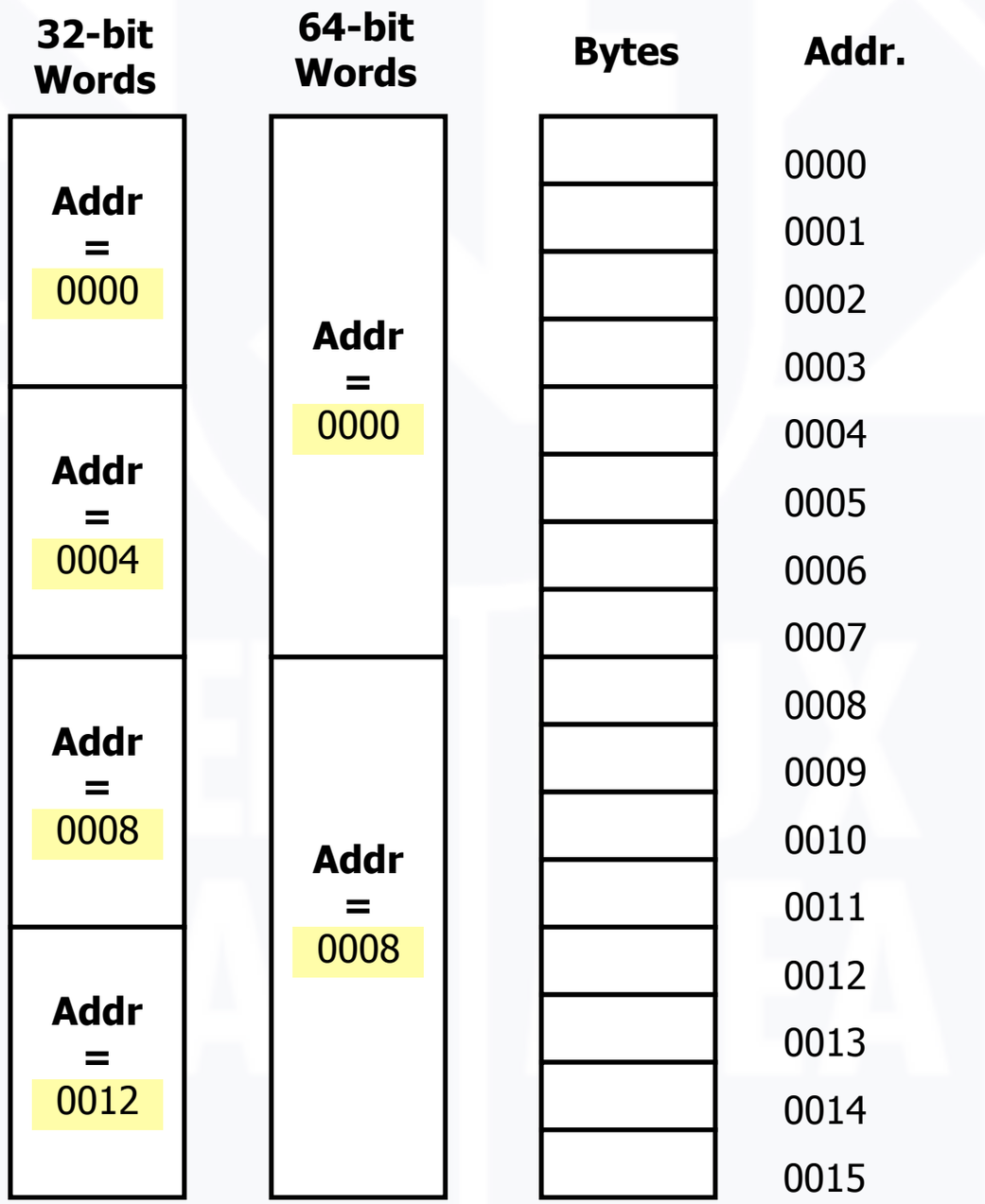


# Memory interface

- Bit ordering
  - Usually little endian
  - Mathematically compatible
  - B0 is LSB
- Byte Ordering
  - How should bytes within multi-byte word be ordered in memory?
  - Conventions
    - Motorola CISC and many RISCs use “Big Endian”
      - Least significant byte has highest address
    - Intel CISC and PC’s use “Little Endian”
      - Least significant byte has lowest address

# Memory interface

- Addresses Specify Byte Locations
  - Address of first byte in word
  - Addresses of successive words differ by 4 (32-bit) or 8 (64-bit)

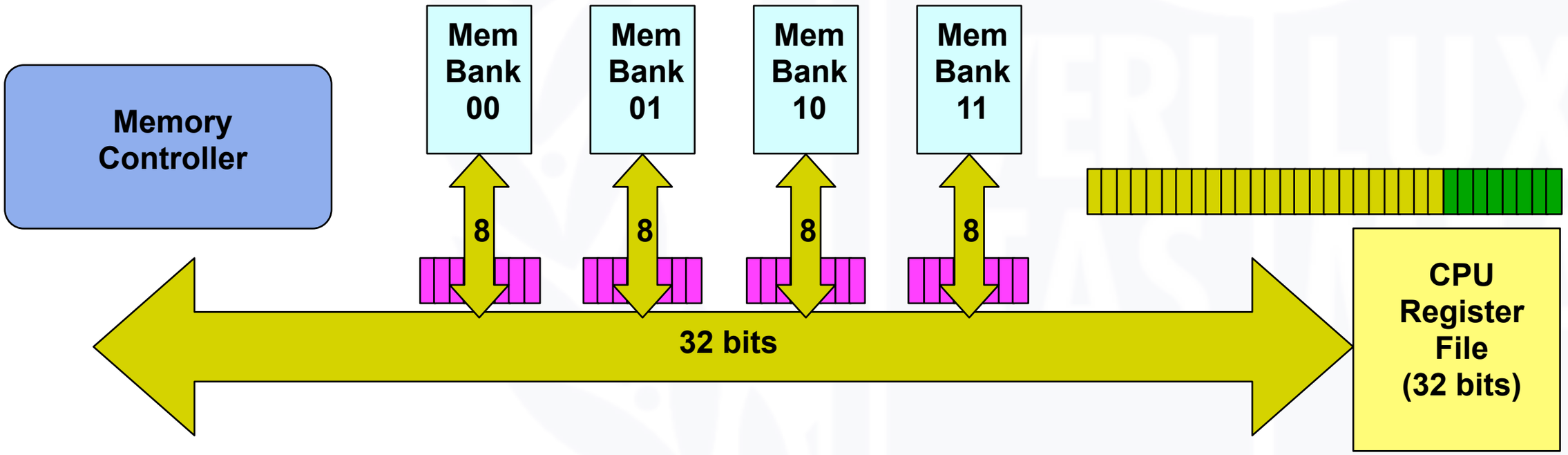


# Memory interface

- Sub-word access
  - Word length is greater than 8bit
    - 16bit
    - 32bit
    - 64bit
  - Sub-word access is necessary
    - 8bit, 16bit and 32bit access in 64bit systems
  - Valid data appear on specific data bus bits
  - Enabling specific memory devices both for full-word and sub-word accesses

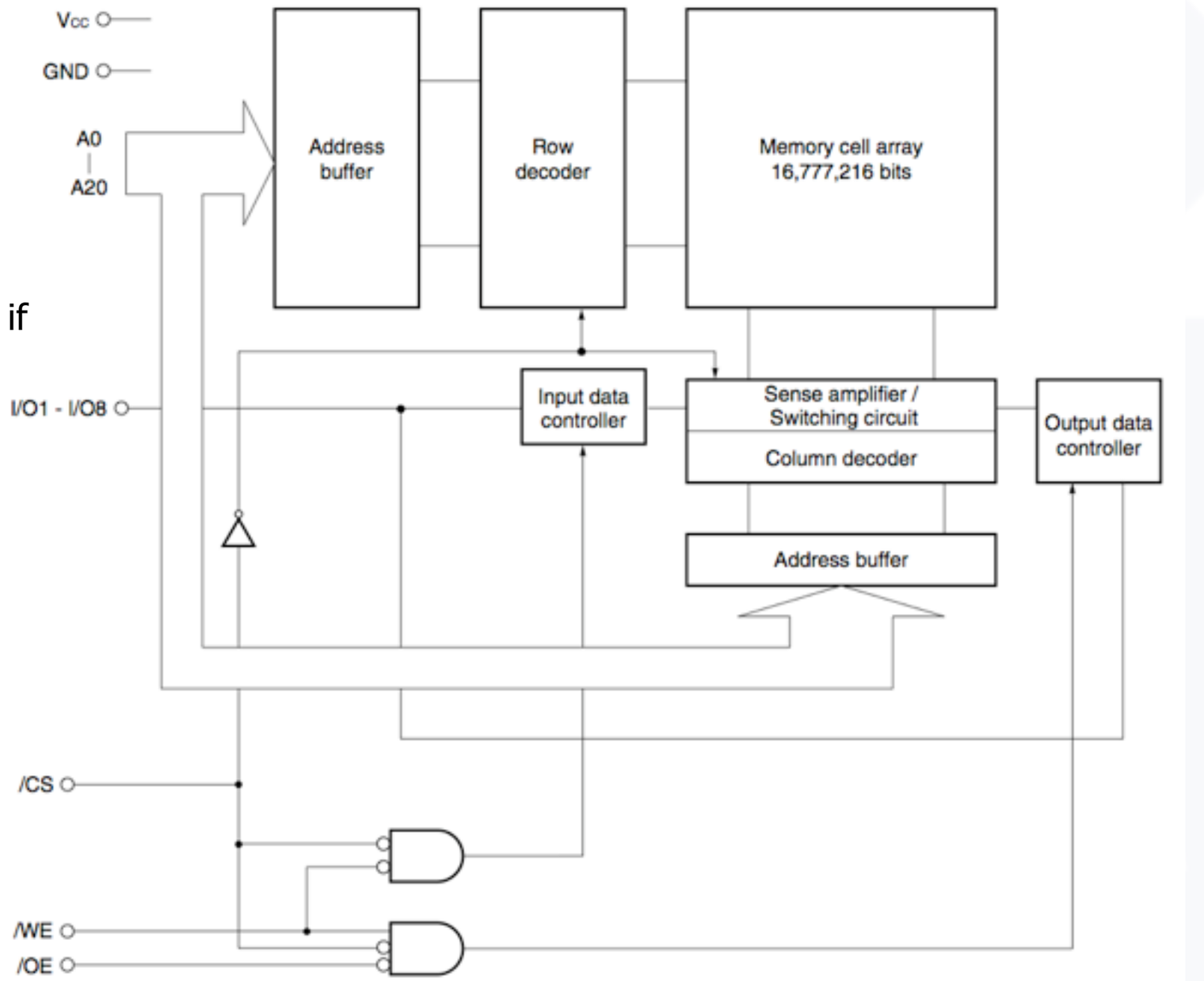
# Memory interface

- Word access
  - 32 bit access for 32-bit machines
  - Active all the memory banks at the same time
- Sub-word access
  - Allow 16 bit or 8 bit access
  - Active associated banks only



# Control Signals

- Asynchronous SRAM
  - CS
    - Master enable
    - Some devices go to a power down mode if disabled
  - OE
    - Enable the output buffer
    - Access is generally performed even if disabled
  - WE
    - Internal control
    - Data is destructed if improperly controlled



# Electrical Consideration

- Buffering
  - Address buffers
    - Mostly microprocessor's address bus buffers are not that strong (eg. 8 mA)
    - Large fanout results in longer delay and/or bad signal integrity
    - Bipolar or BiCMOS address buffers are used
      - 74LVT16244 • 74LVTH16244 Low Voltage16-Bit Buffer/Line Driver with 3-STATE Outputs

| | | | | |
|---|---|---|---|---|
| $I_{IK}$ | DC Input Diode Current | −50 | $V_I$ < GND | mA |
| $I_{OK}$ | DC Output Diode Current | −50 | $V_O$ < GND | mA |
| $I_O$ | DC Output Current | 64 | $V_O$ > $V_{CC}$ Output at HIGH State | mA |
| | | 128 | $V_O$ > $V_{CC}$ Output at LOW State | |

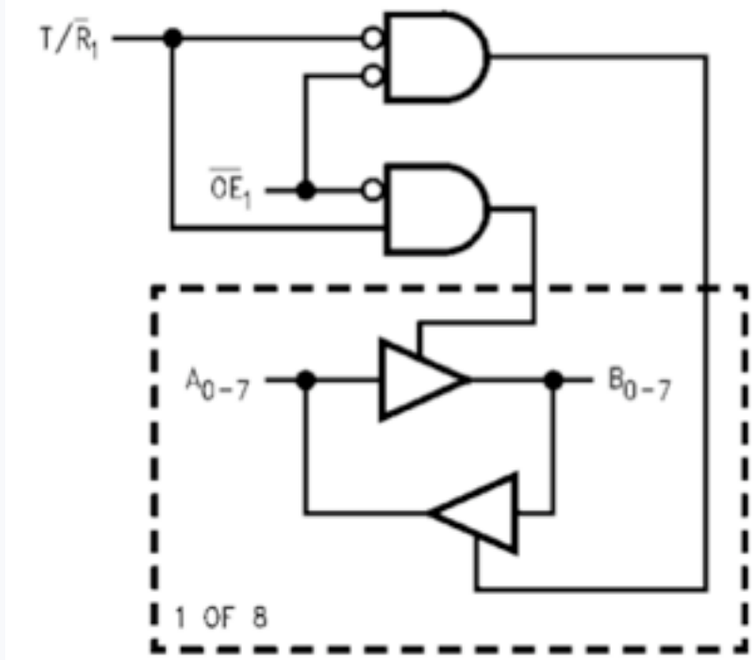
| Symbol | Parameter | $T_A$ = −40°C to +85°C, $C_L$ = 50 pF, $R_L$ = 500Ω | | | | Units |
|---|---|---|---|---|---|---|
| | | $V_{CC}$ = 3.3V ± 0.3V | | $V_{CC}$ = 2.7V | | |
| | | Min | Max | Min | Max | |
| $t_{PLH}$ | Propagation Delay Data to Output | 1.2 | 3.5 | 1.2 | 3.9 | ns |
| $t_{PHL}$ | | 1.2 | 3.5 | 1.2 | 3.9 | |
| $t_{PZH}$ | Output Enable Time | 1.2 | 4.0 | 1.2 | 5.0 | ns |
| $t_{PZL}$ | | 1.2 | 5.0 | 1.2 | 6.5 | |
| $t_{PHZ}$ | Output Disable Time | 2.0 | 4.7 | 2.0 | 5.2 | ns |
| $t_{PLZ}$ | | 1.5 | 4.2 | 1.5 | 4.4 | |
| $t_{OSHL}$ | Output to Output Skew | | 1.0 | | 1.0 | ns |
| $t_{OSLH}$ | (Note 11) | | | | | |

# Buffering

- Data buffers
  - Same to the address buffer
  - Direction control is additionally required
  - Bus switches (CBT) are used for DRAMs
    - 74LVT16245 • 74LVTH16245 Low Voltage 16-Bit Transceiver with 3-STATE Outputs




CBT bus switch




BiCMOS bus buffer

# Buffering

- Using a CBT to switch Dual Inline Memory Modules (DIMMs) to a common bus can reduce a memory controller's loading by 50 percent
  - CBT16292 is connecting DIMMs 2 and 4 to the memory controller and DIMMs 1 and 3 through a 500-Ω resistor to ground