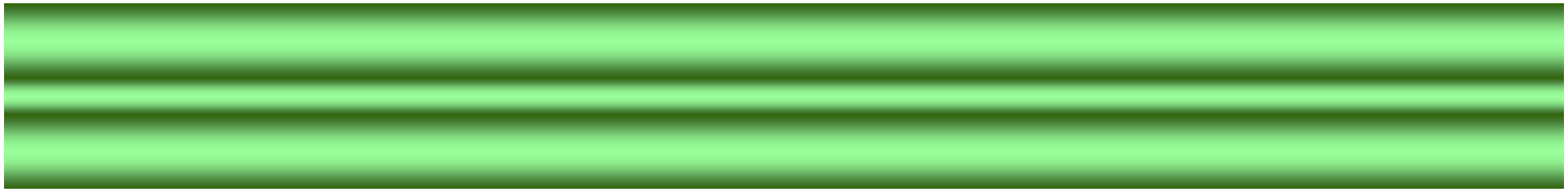
EE414 Embedded Systems

Ch 6. Interfacing

Part 1/4



Byung Kook Kim
School of Electrical Engineering
Korea Advanced Institute of Science and Technology

Overview

- 6.1 Introduction
- 6.2 Communication Basics
- 6.3 Microprocessor interfacing: I/O Addressing

- 6.4 Interrupts
- 6.5 Interrupts in AM3359
- 6.6 Exceptions in ARM
- 6.7 Timers & Counters
- 6.8 Timer & Clock in AM3359
- 6.9 Direct memory access
- 6.10 Arbitration
- 6.11 Multilevel Bus Architectures

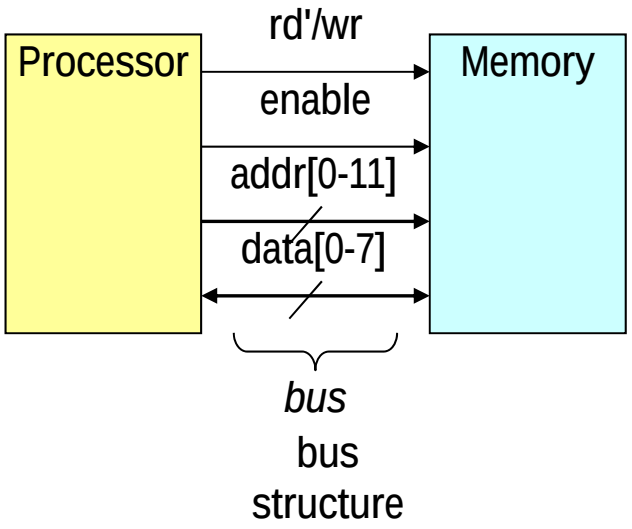
6.1 Introduction

- **Embedded system functionality aspects**
  - **Processing**
    - Transformation of data
    - Implemented using processors
  - **Storage**
    - Retention of data
    - Implemented using memory
  - ***Communication***
    - Transfer of data between processors and memories and peripherals
    - Implemented using ***buses***
    - Called ***interfacing***

6.2 Communication Basics
A. Basic Terminology

A simple bus

- Wires
  - Uni-directional: Address, Control
  - or bi-directional: Data
  - One line may represent multiple wires
- **Bus**
  - *Set of wires with a single function*
    - Address bus, data bus
  - *Or, entire collection of wires*
    - Address, data, and control [& power]
- Associated protocol
  - Rules for communication
  - Low-level protocol
  - Signal level w.r.t. time: Timing diagram

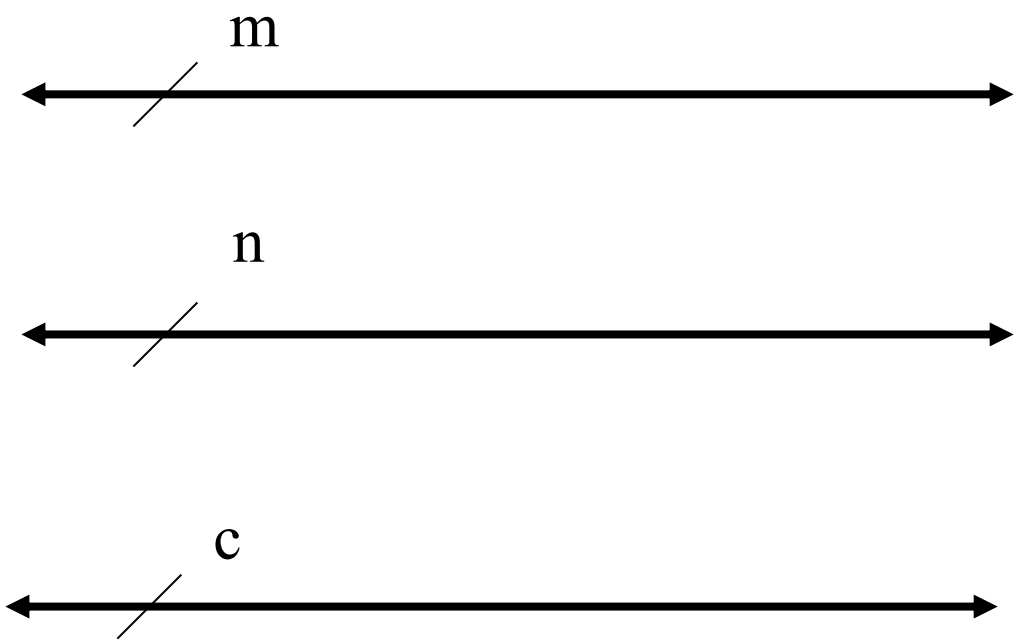


bus structure

Bus

Generic Bus Structure

- Address:
- Data:
- Control:

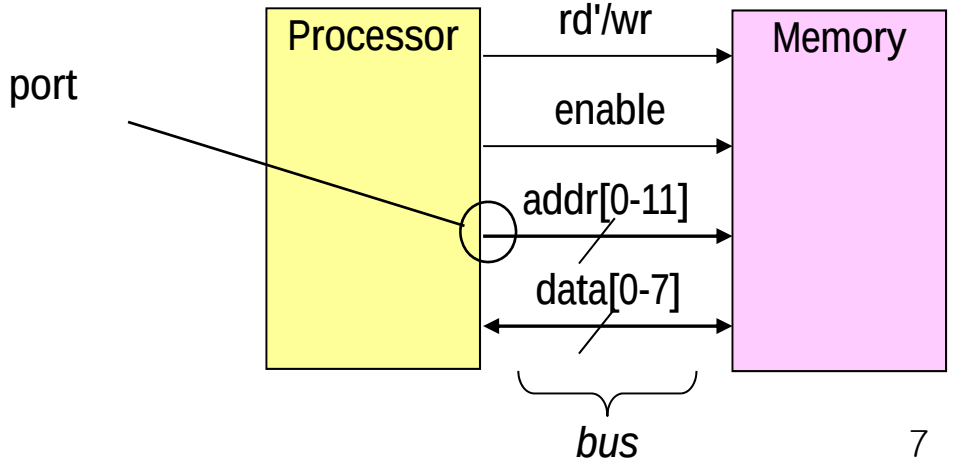


Electrical Bus Design

- Bus signals are usually tri-stated.
- Address and data lines may be multiplexed.
- Every device on the bus must be able to drive the maximum bus load:
  - Bus wires.
  - Other bus devices.
- Bus may include clock signal.
  - Timing is relative to clock.

Ports

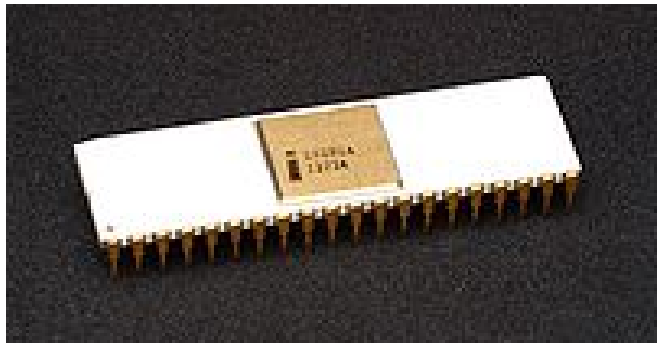
- Conducting device on periphery
  - Connects bus to processor or memory
- Often referred to as a *pin[s]*
  - Actual pins on periphery of IC package that plug into socket on printed-circuit board
  - Sometimes metallic balls instead of pins
  - Today, metal "pads" connecting processors and memories within single IC
- Single wire or set of wires with single function
  - E.g., 12-wire address port



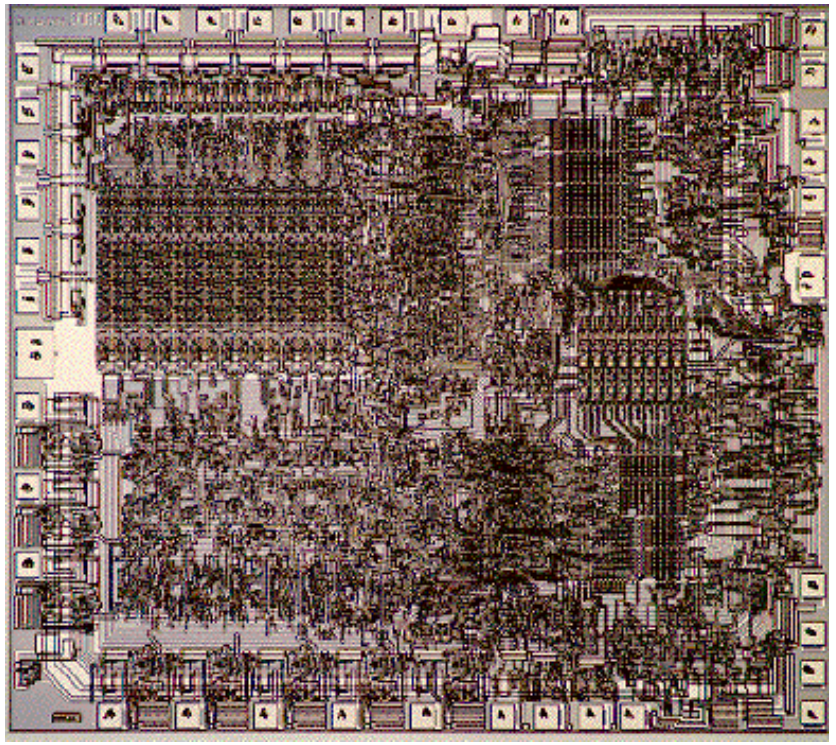
Example: Intel 8080 uP

- **Overview**
  - Produced mid 1974
  - Common manufacturer(s) Intel
  - Max. CPU clock rate 2 MHz
  - Instruction set pre x86
  - Package(s) 40-pin DIP
- **Ports & Bus**
  - Data D0 – D7
  - Address A0 – A15
  - Control: Reset, RD, WR, Rdy, Wait, Int, IntAck, DMA, DMAAck, S
  - Clock: CLC1, CLC2
  - Power: 12V, 5V, GND, -5V

1.



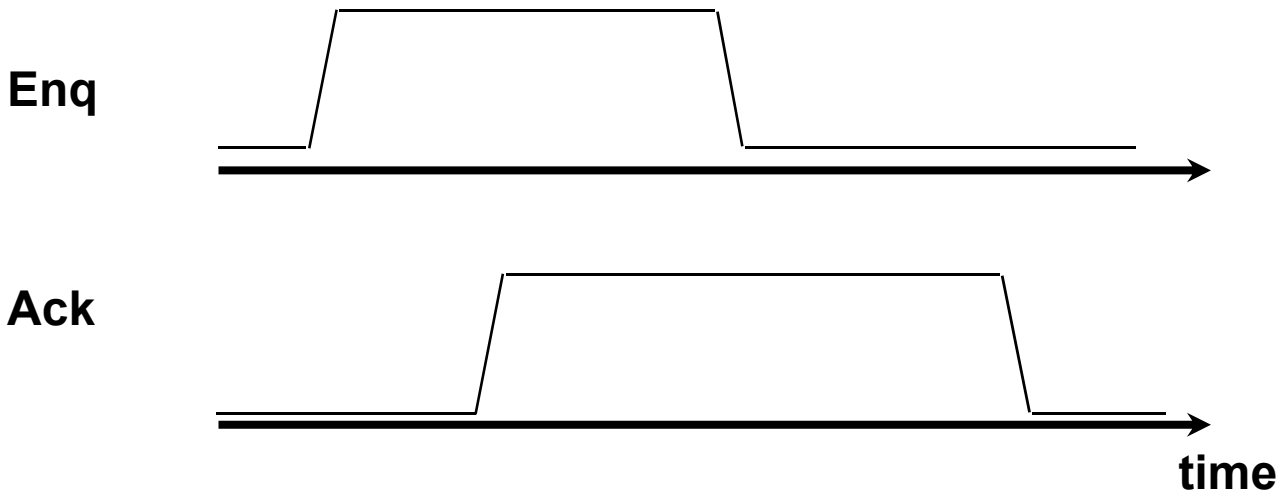
2.



1. (C) http://en.wikipedia.org/wiki/Intel_8080
2. (C) http://www.cs.washington.edu/homes/lazowska/faculty.lecture/chips/8080.html

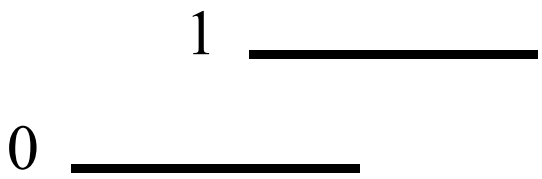
Timing Diagrams

- **Timing diagram**
  - Most common method for describing a communication protocol
  - Shows traces (signal levels) w.r.t. time through the operation of a system.
  - Generally used for asynchronous machines with timing constraints.



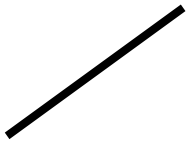
Timing Diagram Syntax

- Constant value:



- Stable:

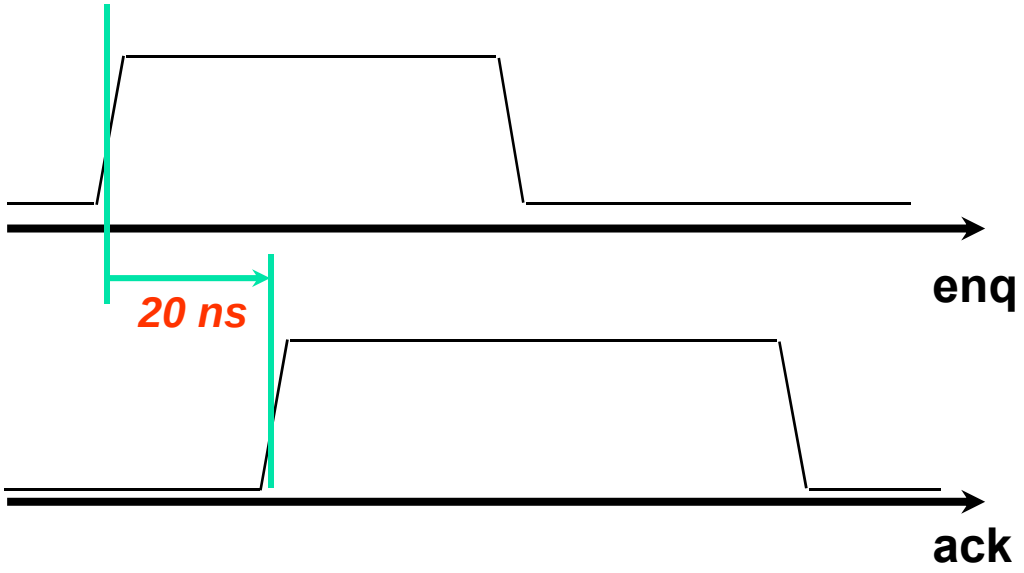
- Changing:



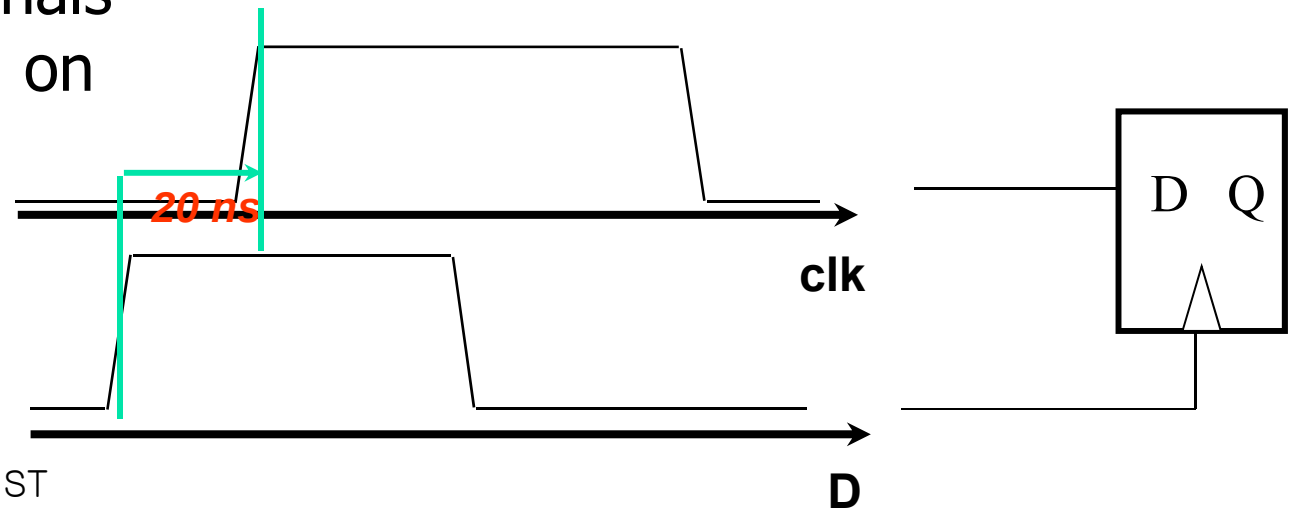
- Unknown:

Timing Constraints

- Minimum time between two events:

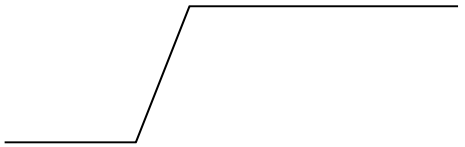


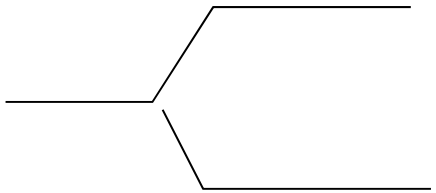
- Control signals are passed on the bus:



Timing Diagrams

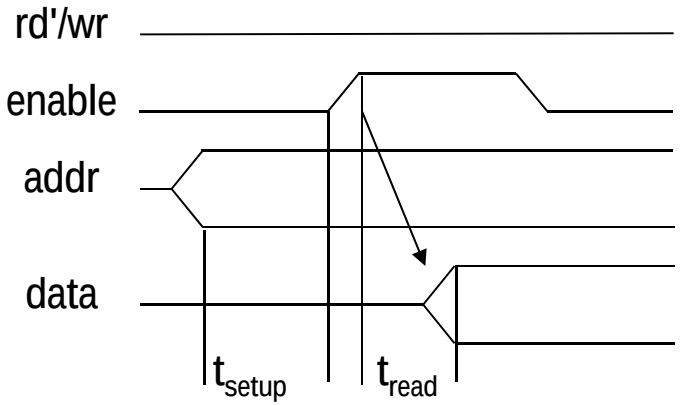
- Time proceeds to the right on x-axis
  - Control signal: low or high
    - May be active low (e.g., go', /go, or go_L)
    - Use terms *assert* (active) and *deassert*
      - Asserting go' means go=0
  - Data signal: not valid or valid
  - Protocol may have subprotocols
    - Called bus cycle, e.g., read and write
    - Each may be several clock cycles





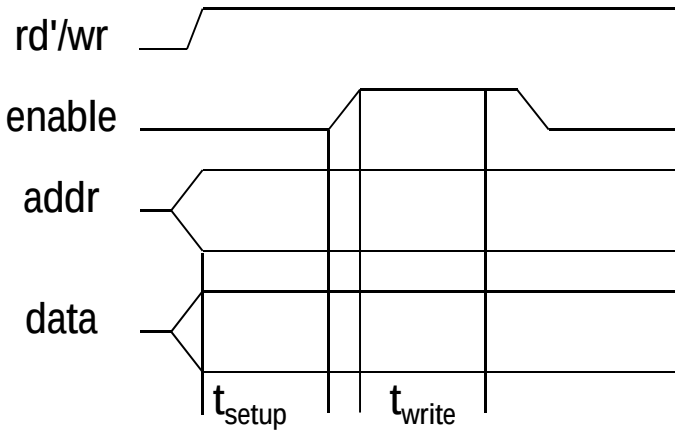
Timing Diagrams - Memory

- **Read example →**
  - *rd'/wr* set low,
  - address placed on *addr* for at least t_{setup} time before *enable* asserted,
  - *enable* triggers memory to place data on *data* wires by time t_{read}



read protocol

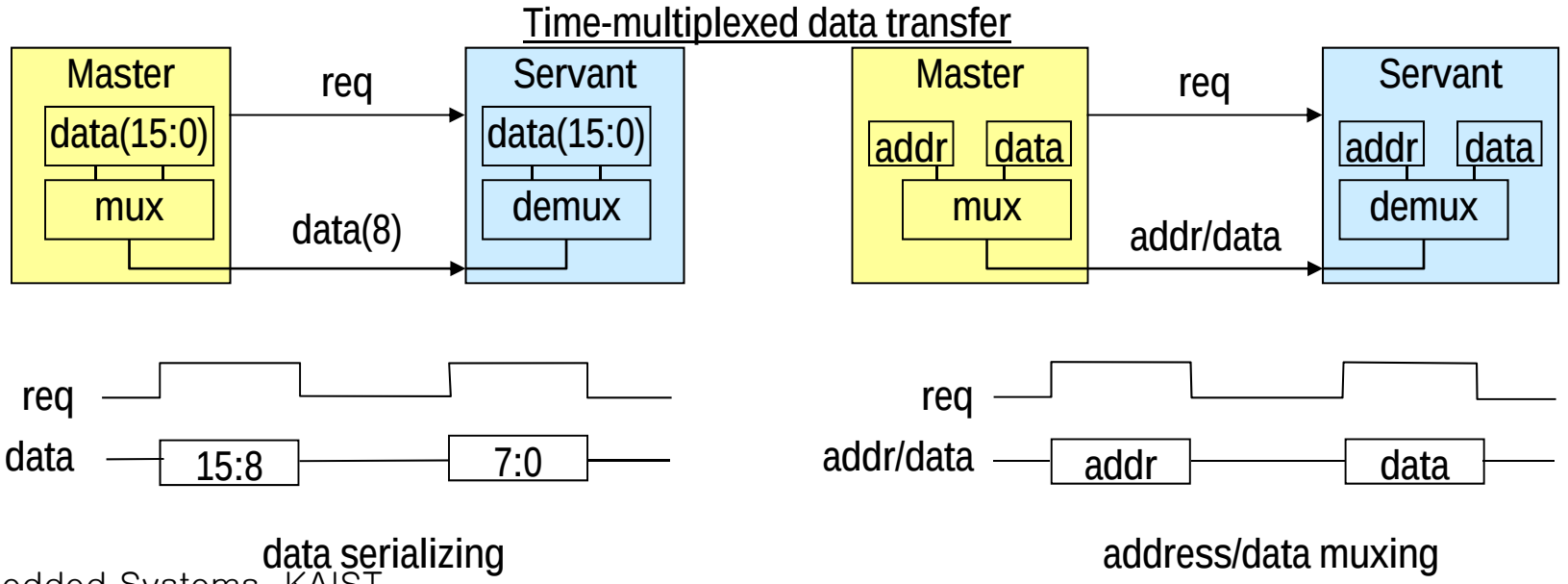
- **Write example →**
  - *rd'/wr* set high,
  - address and data placed on *addr* & *data* for at least t_{setup} time before *enable* asserted,
  - *enable* triggers memory to place data on *data* wires by time t_{write}



write protocol

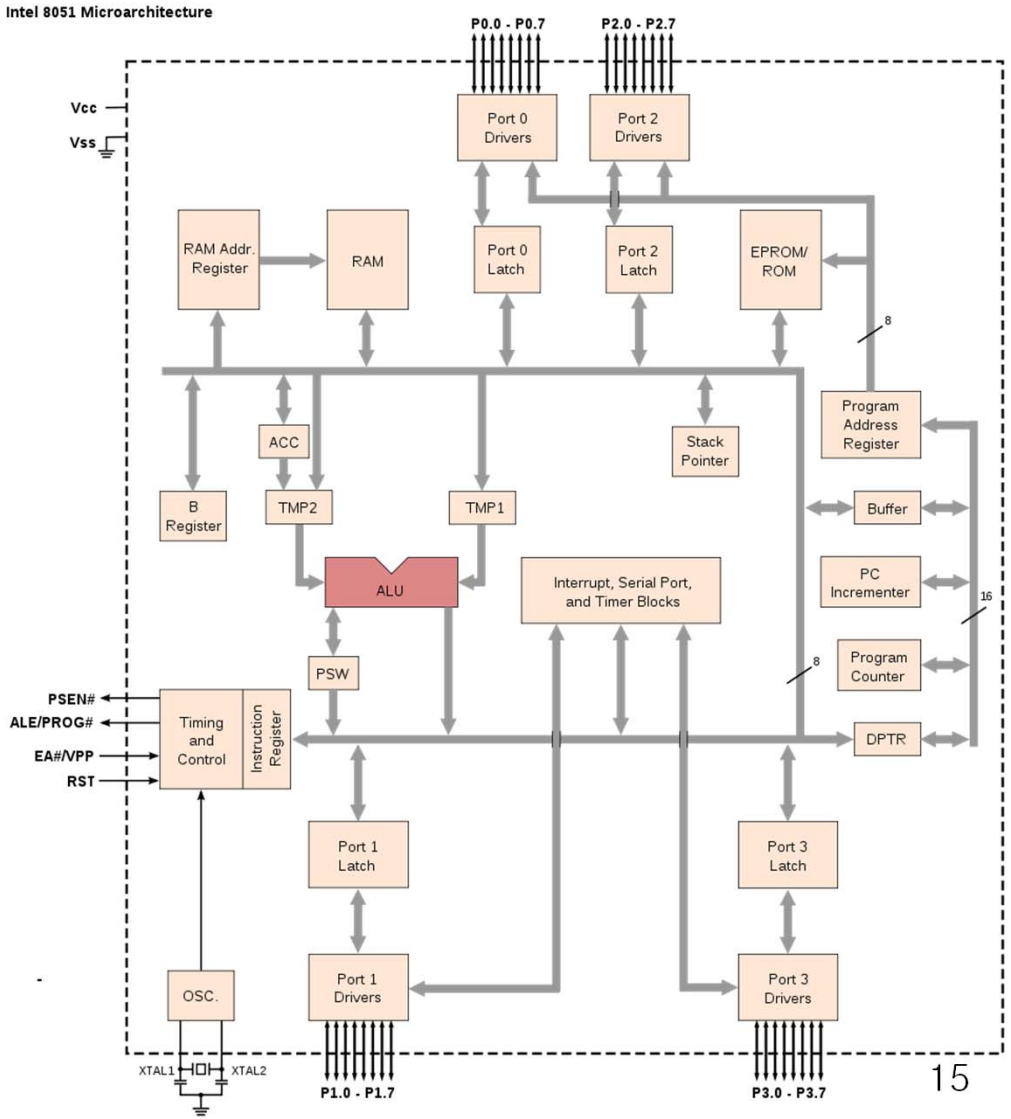
B. Basic Protocol Concepts

- Actor: master initiates, servant (slave) respond
- Data direction: sender, receiver
- Addresses: special kind of data
  - Specifies a location in memory, a peripheral, or a register within a peripheral
- **Time multiplexing** ->
  - Share a single set of wires for multiple pieces of data
  - Saves wires at expense of time



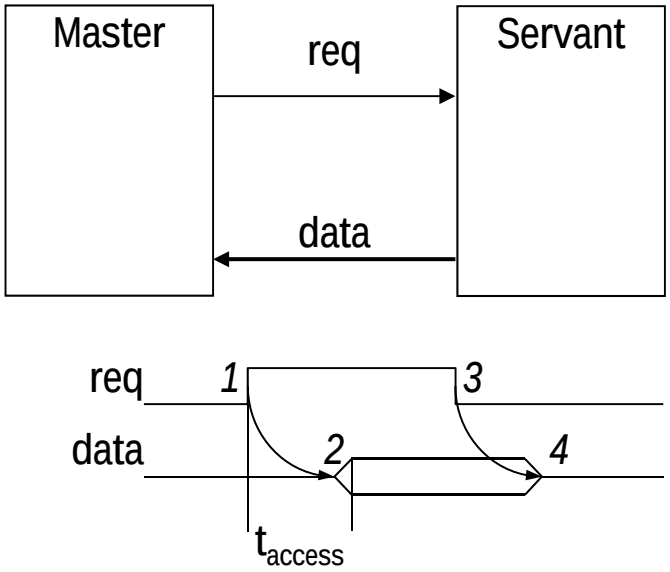
Ex. Intel 8051 uC

- **Features** [1] →
  - Intel in 1980
  - Provides many functions (CPU, RAM, ROM, I/O, interrupt logic, timer, etc.) in a single [40-pin] package
  - 8-bit ALU, Accumulator and 8-bit Registers
  - 8-bit data bus
  - 16-bit address bus
  - On-chip RAM – 128 bytes (data memory)
  - On-chip ROM – 4 K byte (program memory)
  - Four byte bi-directional input/output port
  - UART (serial port)
  - Two 16-bit Counter/timers
  - Two-level interrupt priority
  - Power saving mode (on some derivatives)
- **Architecture** [2] →



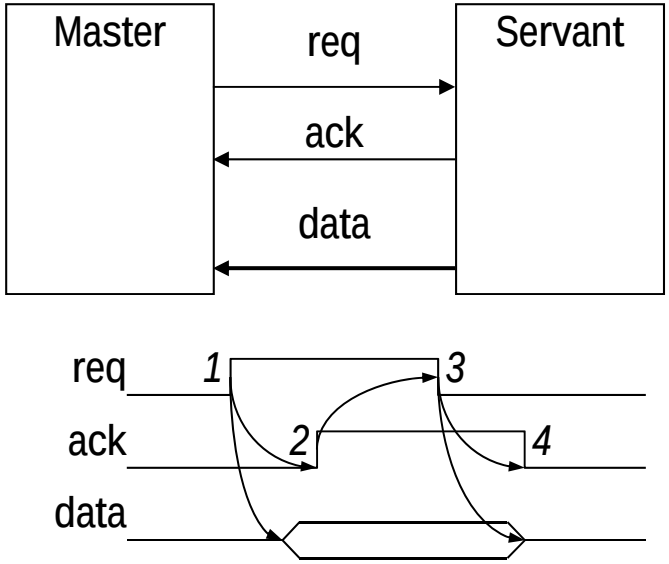
1. (C) http://en.wikipedia.org/wiki/Intel_MCS-51
2. (C) http://upload.wikimedia.org/wikipedia/commons/thumb/c/cd/Intel_8051_arch.svg/1000px-Intel_8051_arch.svg.png

Basic Protocol Concepts: Control Methods



1. Master asserts *req* to receive data
2. Servant puts data on bus within time t_{access}
3. Master receives data and deasserts *req*
4. Servant ready for next request

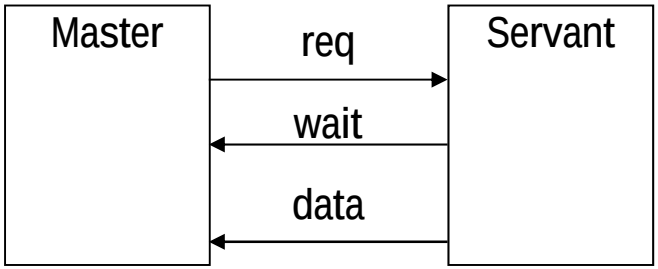
Strobe protocol

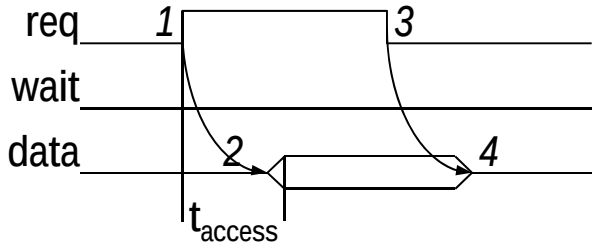


1. Master asserts *req* to receive data
2. Servant puts data on bus and asserts *ack*
3. Master receives data and deasserts *req*
4. Servant ready for next request

Handshake protocol

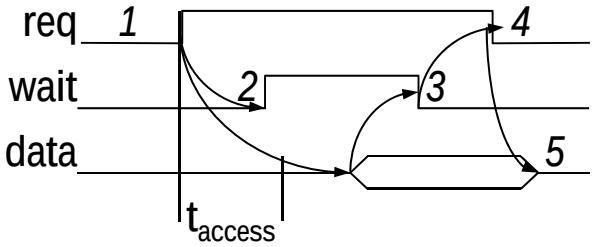
A Strobe/Handshake Compromise





1. Master asserts *req* to receive data
2. Servant puts data on bus within time t_{access} (wait line is unused)
3. Master receives data and deasserts *req*
4. Servant ready for next request

Fast-response case



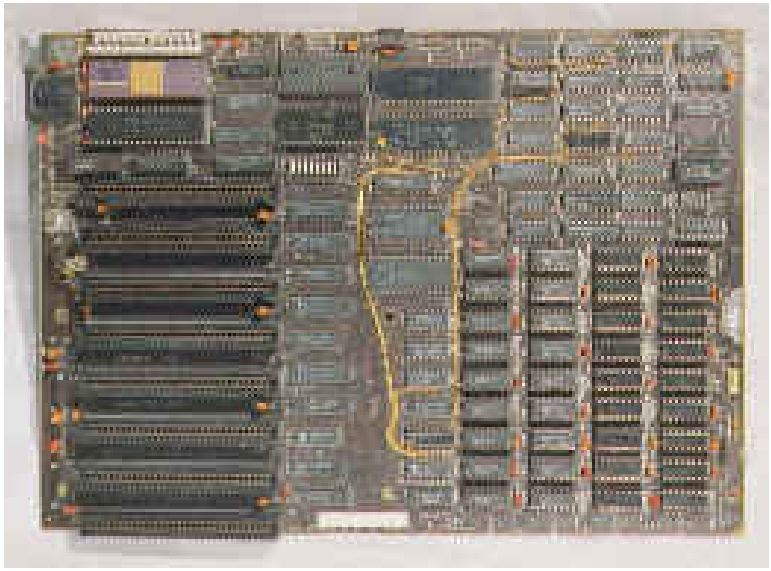
1. Master asserts *req* to receive data
2. Servant can't put data within t_{access}, asserts *wait* ack
3. Servant puts data on bus and deasserts *wait*
4. Master receives data and deasserts *req*
5. Servant ready for next request

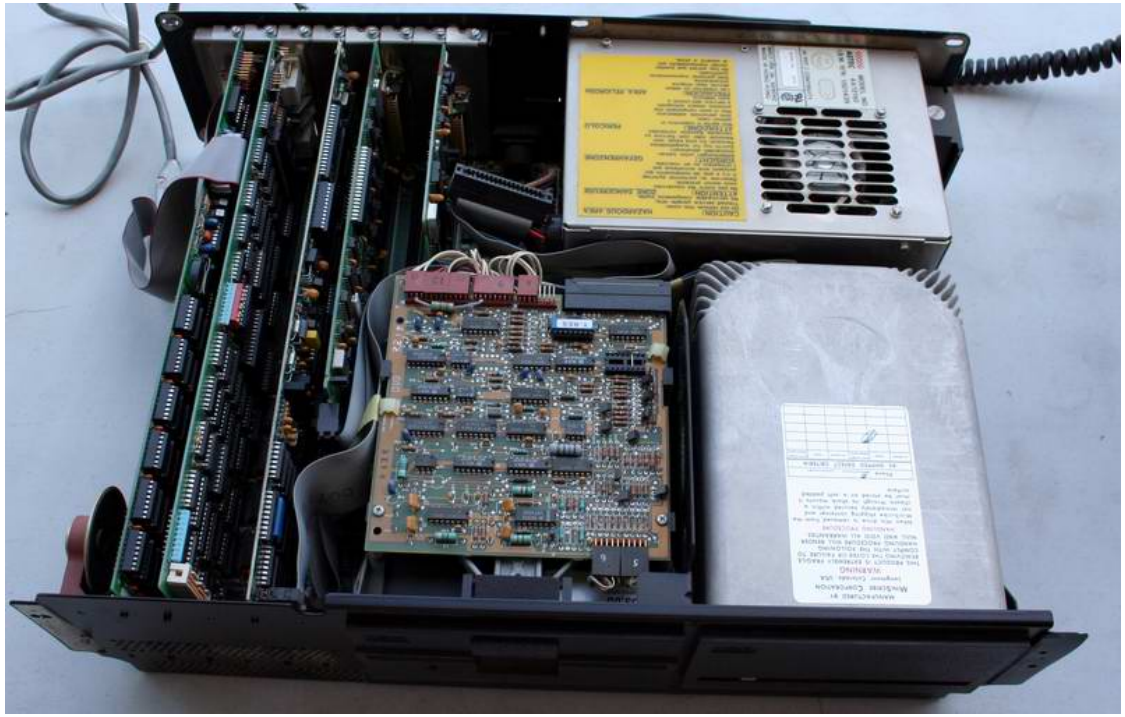
Slow-response case

Ex. IBM PC XT

- Type Personal computer
- Release date March 8, 1983
- Discontinued April 1987
- Operating system IBM BASIC / PC-DOS 2.0-3.20 / SCO Xenix
- CPU Intel 8088 @ 4.77 MHz
- Memory 128kB ~ 640kB

3. 2. 1.

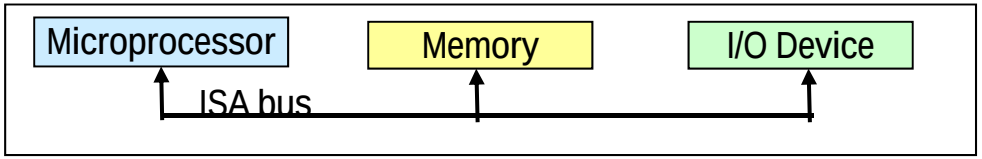




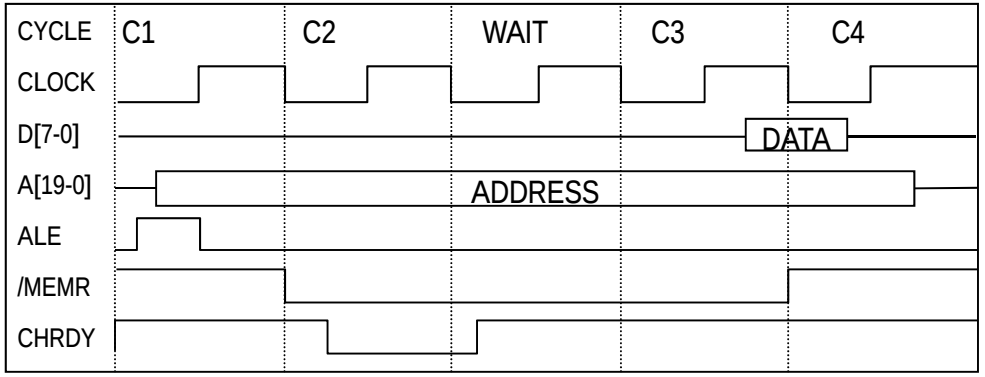
1. (C) http://wecandobiz.wordpress.com/2009/08/18/wecando-biz-server-upgrade-complete-residual-issues/
2. (C) www.vintage-computer.com/ibmpcxt.shtml
3. (C) www.oldcomputers.arcula.co.uk/intl1.htm

Ex. ISA Bus Protocol – Memory Access

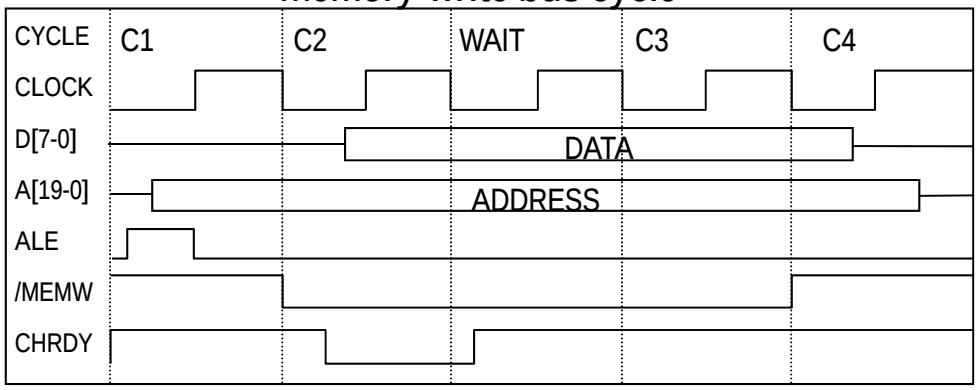
- **ISA: Industry Standard Architecture**
  - Common in 80x86's
- Features
  - 20-bit address
  - Compromise strobe/handshake control
    - 4 cycles default
    - Unless CHRDY deasserted – resulting in additional wait cycles (up to 6)



memory-read bus cycle

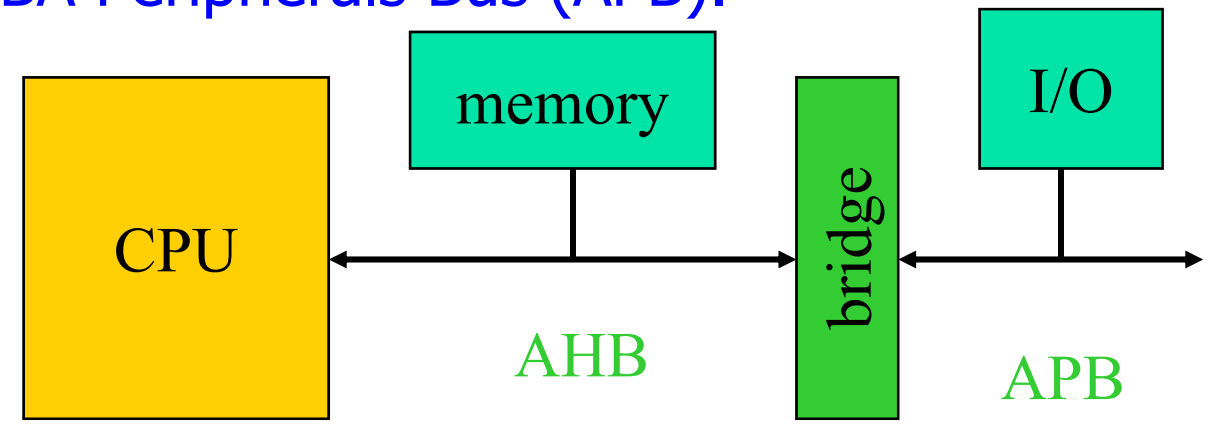


memory-write bus cycle



Ex. ARM Busses

- AMBA:
  - Advanced Microcontroller Bus Architecture
  - Open standard by ARM.
  - Many external devices.
- Two varieties:
  - AMBA High-Performance Bus (AHB).
  - AMBA Peripherals Bus (APB).



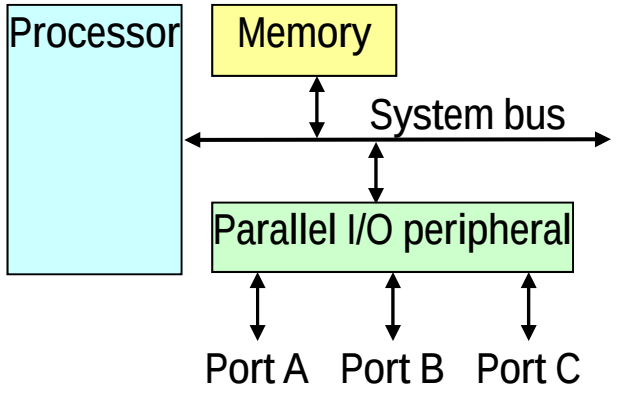
6.3 Microprocessor Interfacing: I/O addressing

A. Port and Bus-Based I/O

- A microprocessor communicates with other devices using some of its pins
  - **Port-based I/O (parallel I/O)**
    - Processor has one or more N-bit ports
    - Processor's software reads and writes a *port* just like a *register*
      - E.g., P0 = 0xFF; v = P1.2; -- P0 and P1 are 8-bit ports
    - Associated configuration register
      - E.g., CP0=0x0F; Upper 4 bit input, lower 4 bit output)
  - **Bus-based I/O [Memory Mapped I/O]**
    - Processor has address, data, and control ports that form a single bus
    - Communication protocol is built into the processor
    - A single instruction carries out the read or write protocol on the bus to access *memory* as well as *peripherals*

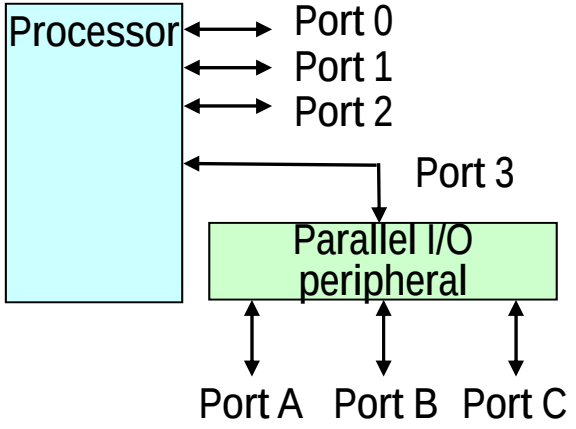
Compromises/Extensions

- **Parallel I/O peripheral**
  - When processor only supports bus-based I/O but parallel I/O needed →
  - Each port on peripheral connected to a register within peripheral that is read/written by the processor



Adding parallel I/O to a bus-based I/O processor

- **Extended parallel I/O**
  - When processor supports port-based I/O but more ports needed
  - One or more processor ports interface with parallel I/O peripheral extending total number of ports available for I/O
  - e.g., extending 4 ports to 6 ports in figure →



Extended parallel I/O

B. Types of Bus-Based I/O: Memory-Mapped I/O and Standard I/O

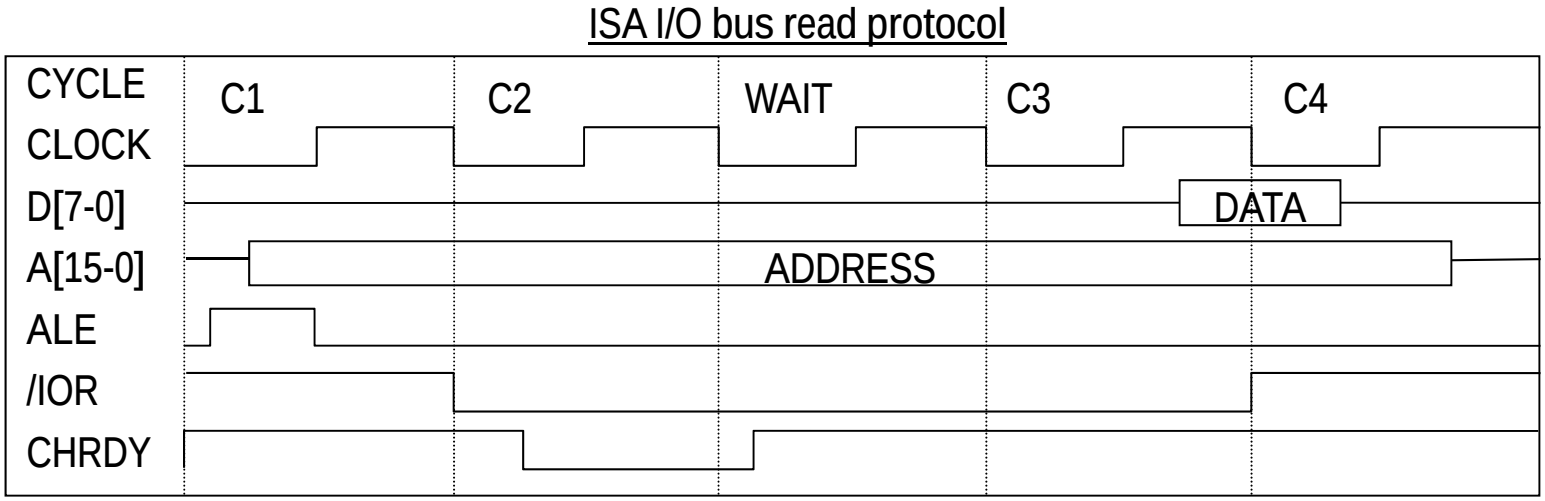
- Processor talks to both memory and peripherals using same bus – two ways to talk to peripherals
  - **Memory-mapped I/O**
    - Peripheral registers occupy addresses in same address space as memory
    - e.g., Bus has 16-bit address
      - lower 48K addresses may correspond to memory
      - upper 16K addresses may correspond to peripherals
  - **Standard I/O (I/O-mapped I/O)**
    - Additional pin (*M/IO*) on bus indicates whether a memory or peripheral access
    - e.g., Bus has 16-bit address
      - all 64K addresses correspond to memory when *M/IO* set to 0
      - all 64K (or 4K) addresses correspond to peripherals when *M/IO* set to 1

Memory-Mapped I/O vs. Standard I/O

- Memory-mapped I/O
  - Adv: Requires no special instructions
    - Assembly instructions involving memory like MOV and ADD work with peripherals as well.
    - Standard I/O requires special instructions (e.g., IN, OUT) to move data between peripheral registers and memory.
- Standard (I/O mapped) I/O
  - Adv: No loss of memory addresses to peripherals
  - Simpler address decoding logic in peripherals possible
    - When number of peripherals much smaller than address space then high-order address bits can be ignored
      - smaller and/or faster comparators

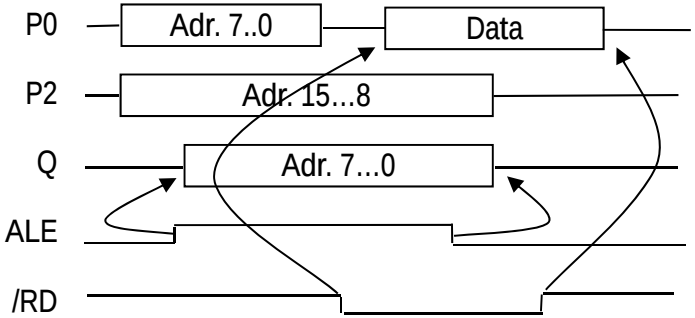
Ex. ISA Bus – Standard I/O

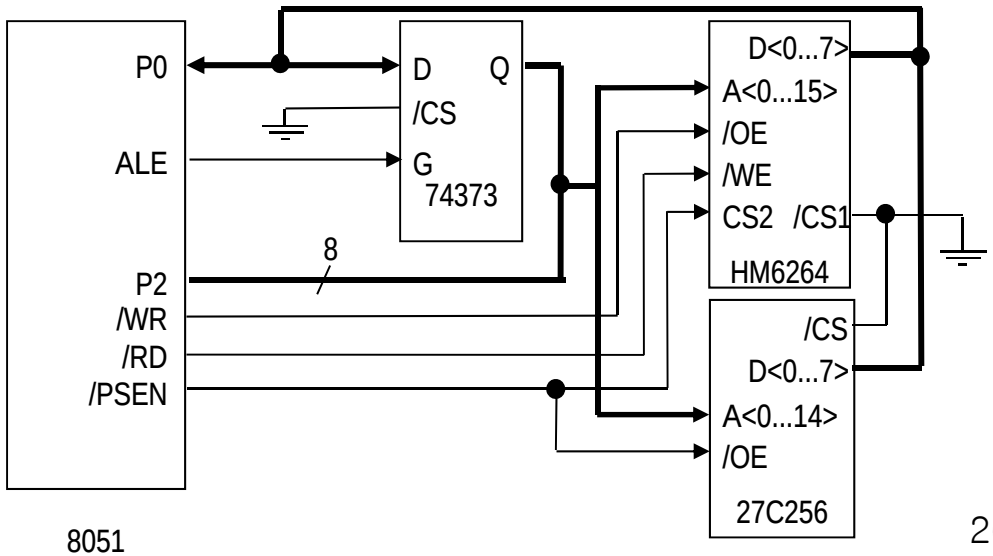
- ISA supports standard I/O
  - /IOR distinct from /MEMR for peripheral read
    - /IOW used for writes
  - 16-bit address space for I/O vs. 20-bit address space for memory
  - Otherwise very similar to memory protocol.



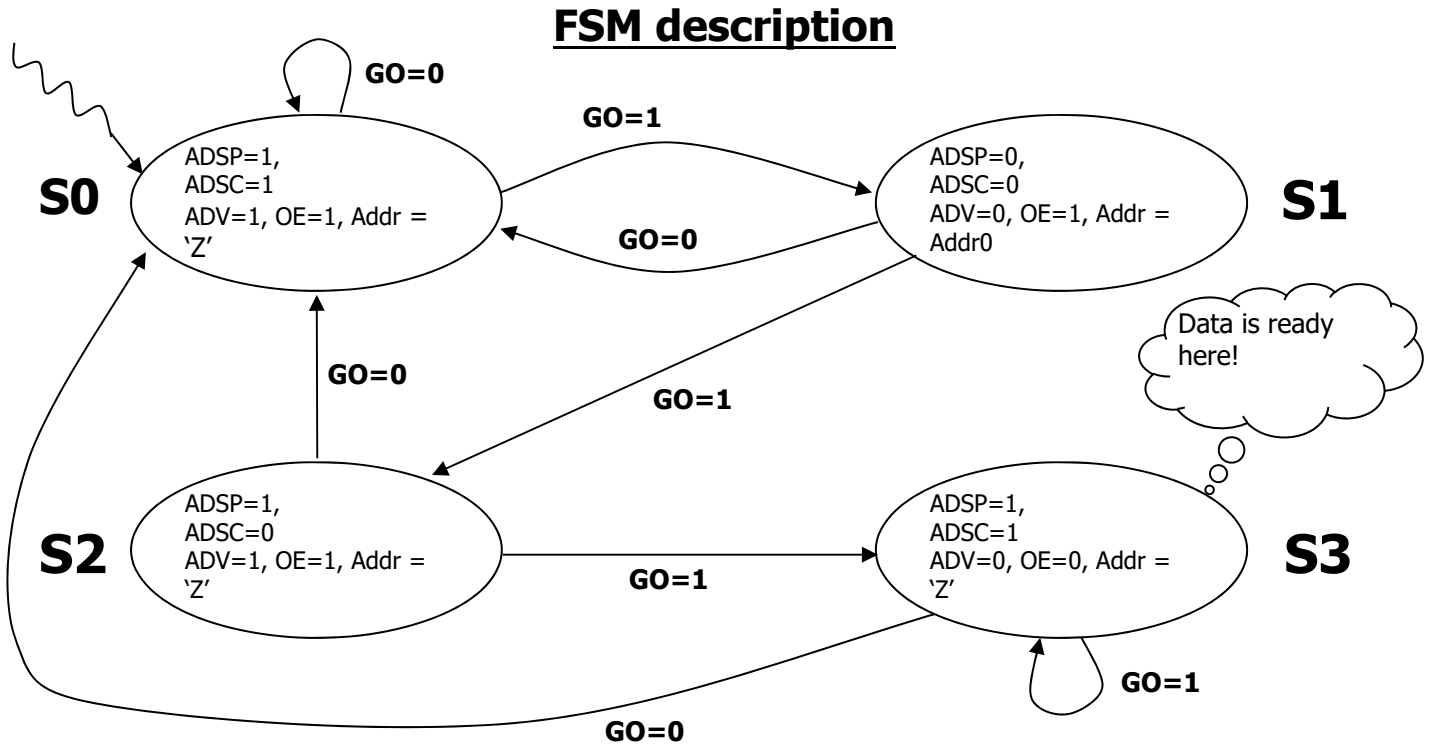
Ex. A Basic Memory Protocol

- Interfacing an 8051 to external memory
  - Ports P0 and P2 support port-based I/O when 8051 internal memory being used
  - Those ports serve as data/address buses when external memory is being used
  - 16-bit address and 8-bit data are *time multiplexed*; low 8-bits of address must therefore be latched with aid of *ALE signal*
  - *Timing diagram & Interface schematic ->*





Ex. A more complex memory protocol



- Generates control signals to drive the TC55V2325FF memory chip in burst mode
  - *Addr0* is the starting address input to device
  - *GO* is enable/disable input to device

References

- [1] Frank Vahid, “Embedded system design: A unified hardware/software introduction”, John Wiley & Sons, 2002.