Hardware-Software Interfaces
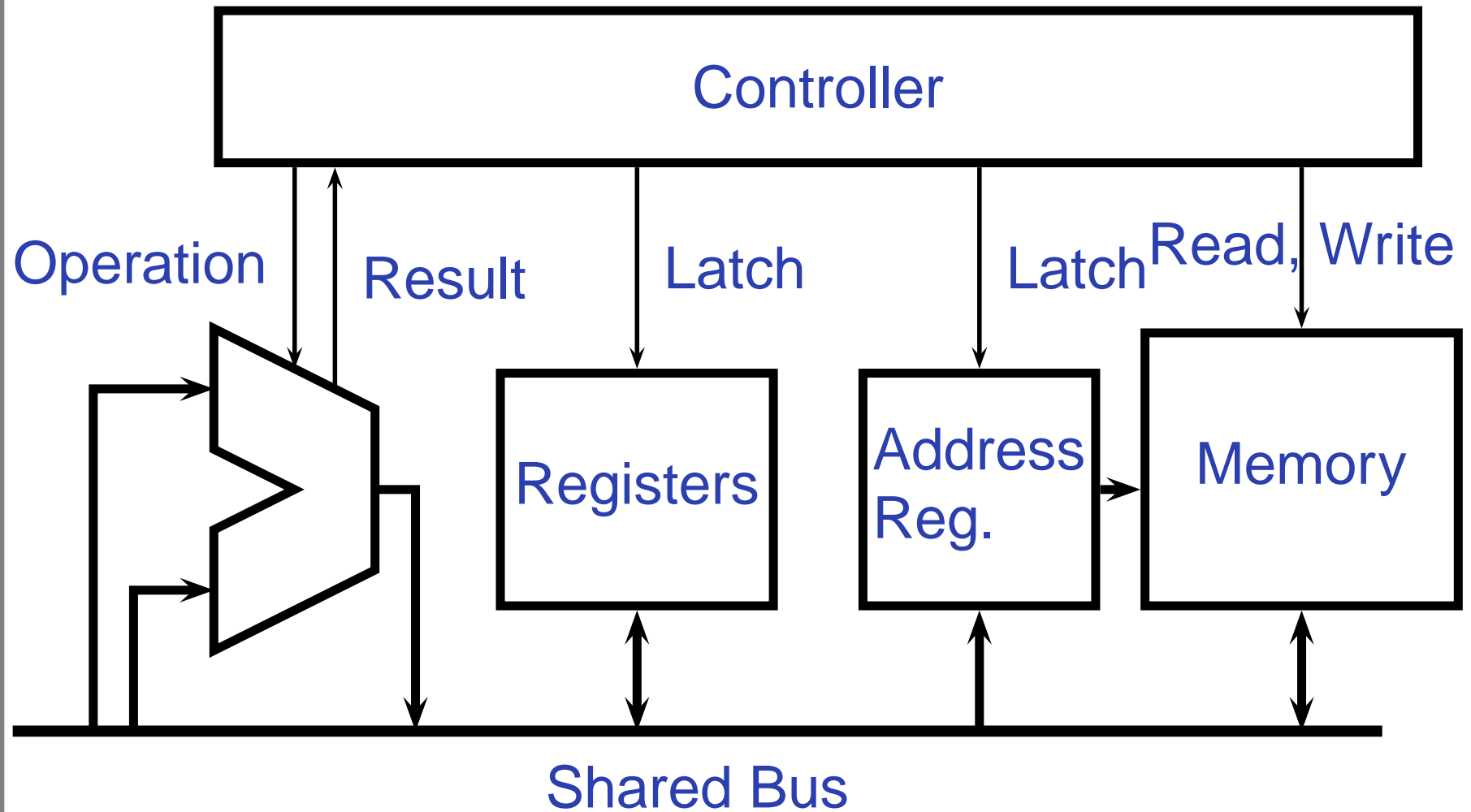## *CSEE W4840*

Prof. Stephen A. Edwards
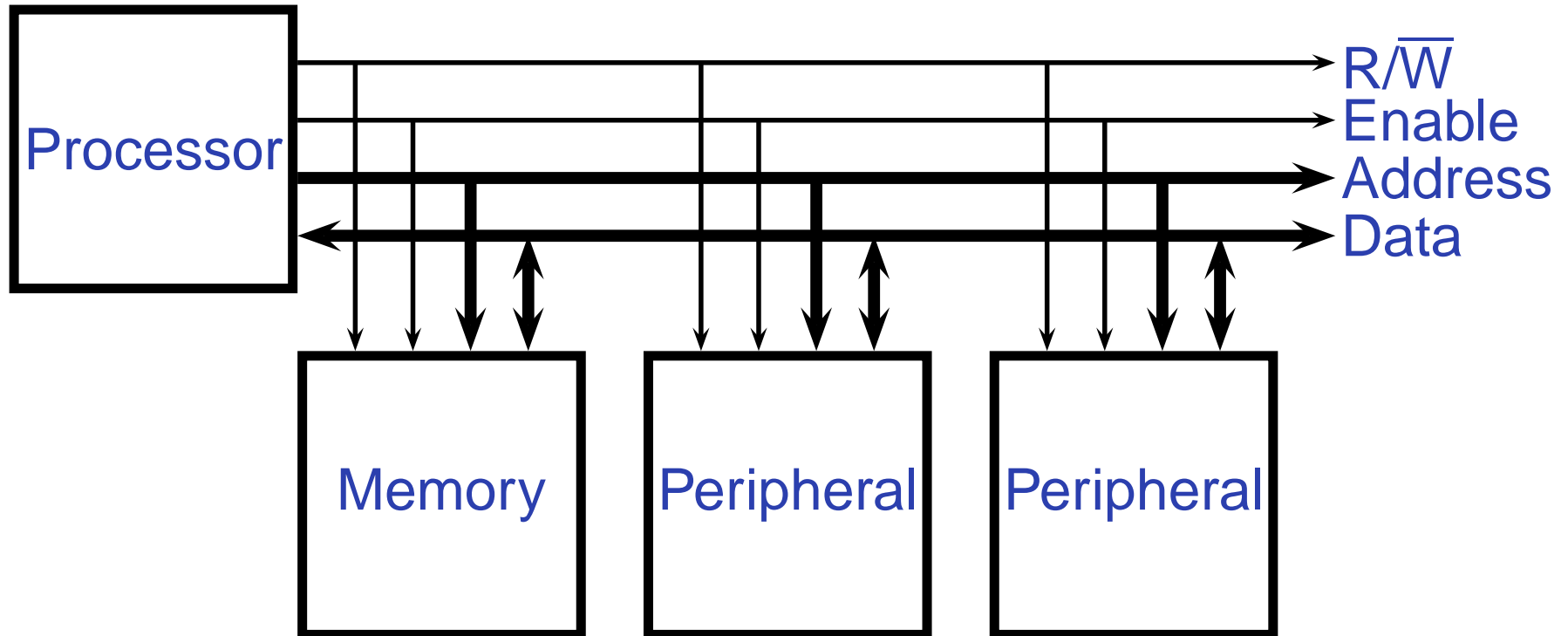
Columbia University

Spring 2006

Basic Processor Architecture
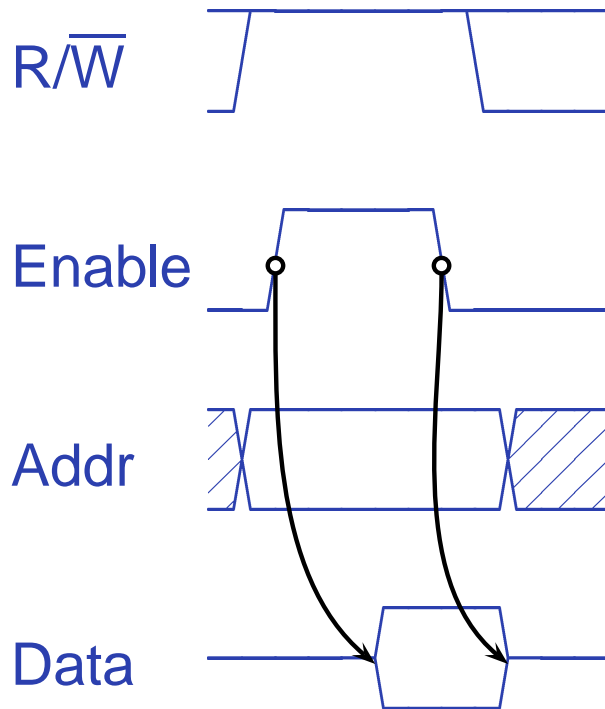
Controller

Operation | Result | Latch | Latch | Read, Write

Registers

Address Reg.

Memory

Shared Bus

Typical Processor System

Simple Bus Timing

Read Cycle   **Write Cycle**

$R/\overline{W}$ $R/\overline{W}$

Enable   Enable

Addr   Addr

Data   Data
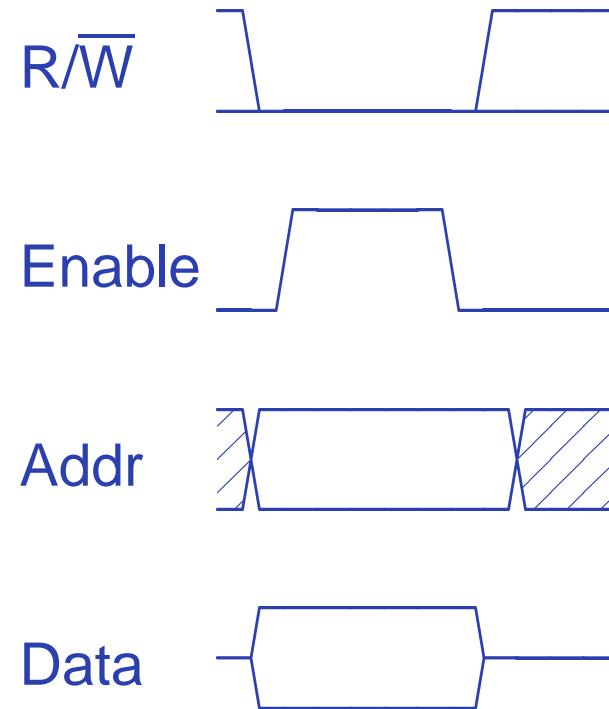
Strobe vs. Handshake

Strobe   **Handshake**

Req   Req

Data   Ack

                              Data
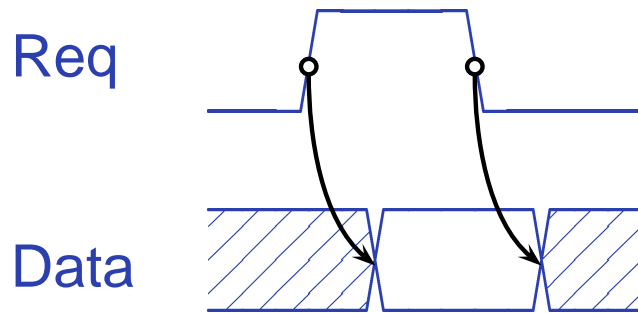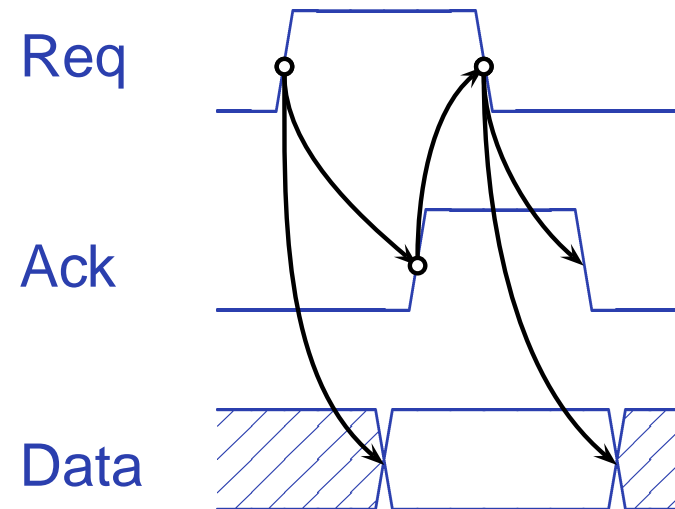
1982: The IBM PC
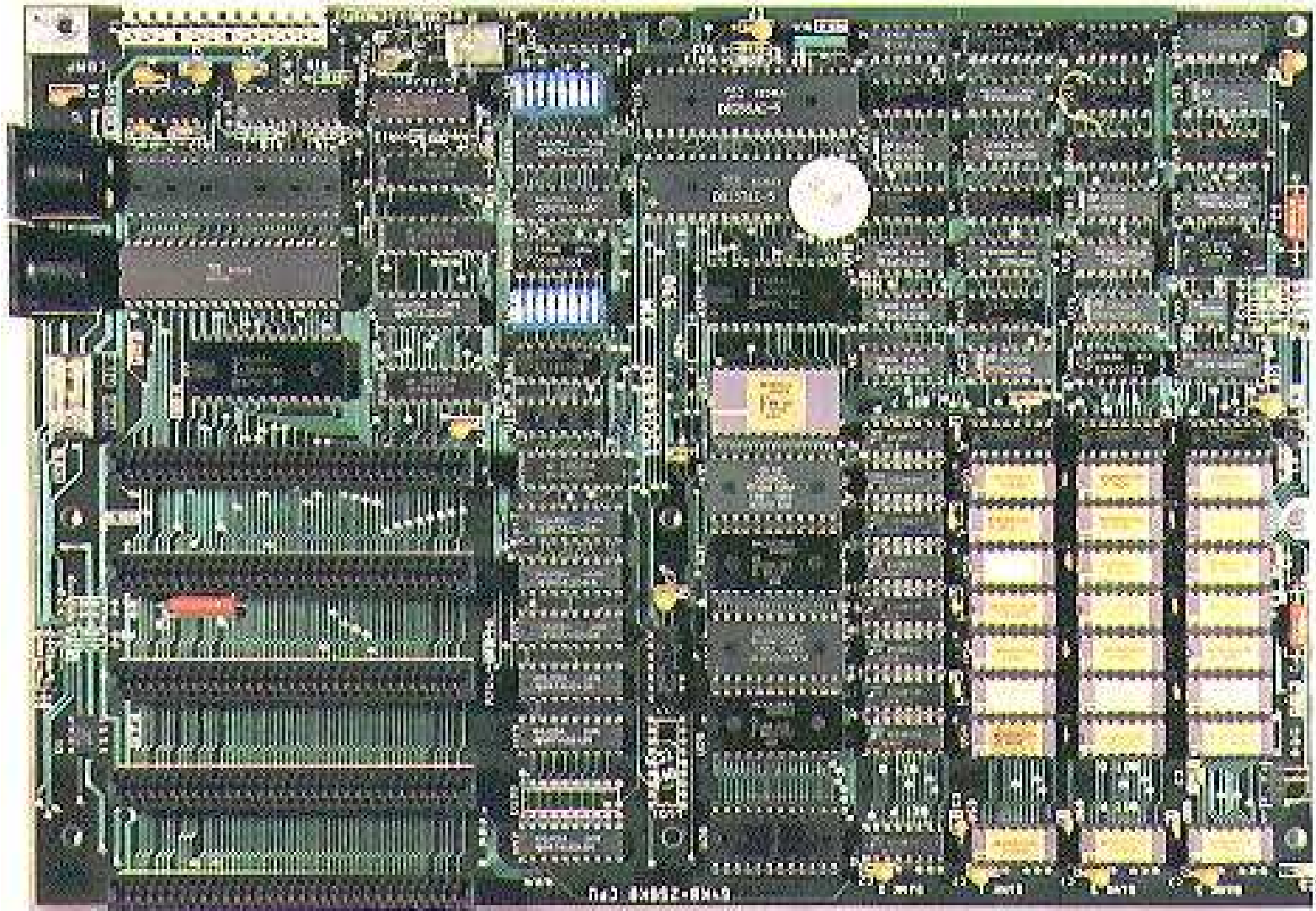
The ISA Bus: Memory Read

CLK

Addr

BALE

$\overline{\text{MEMR}}$

IOCHRDY

Data

C1   C2   Wait   C3   C4
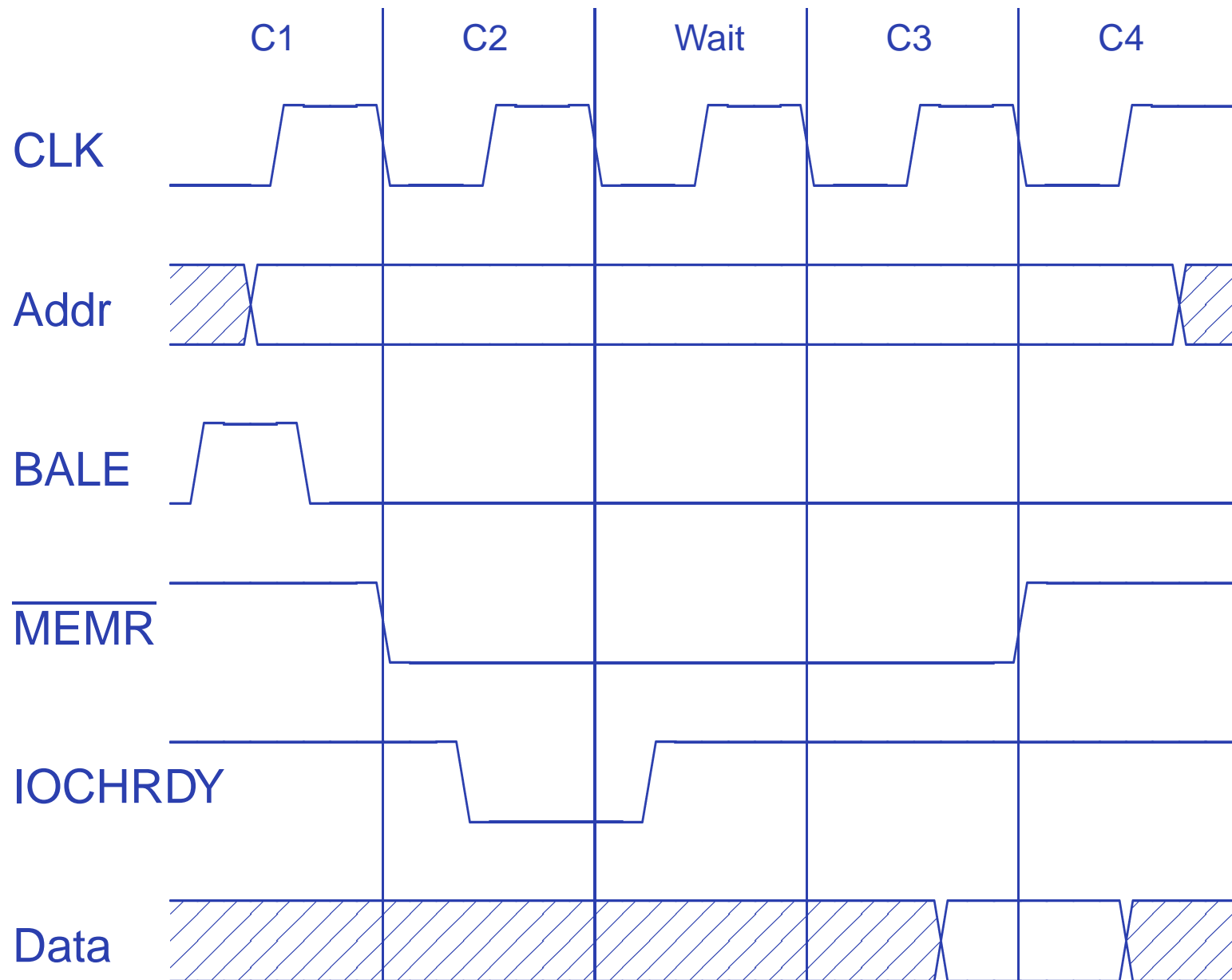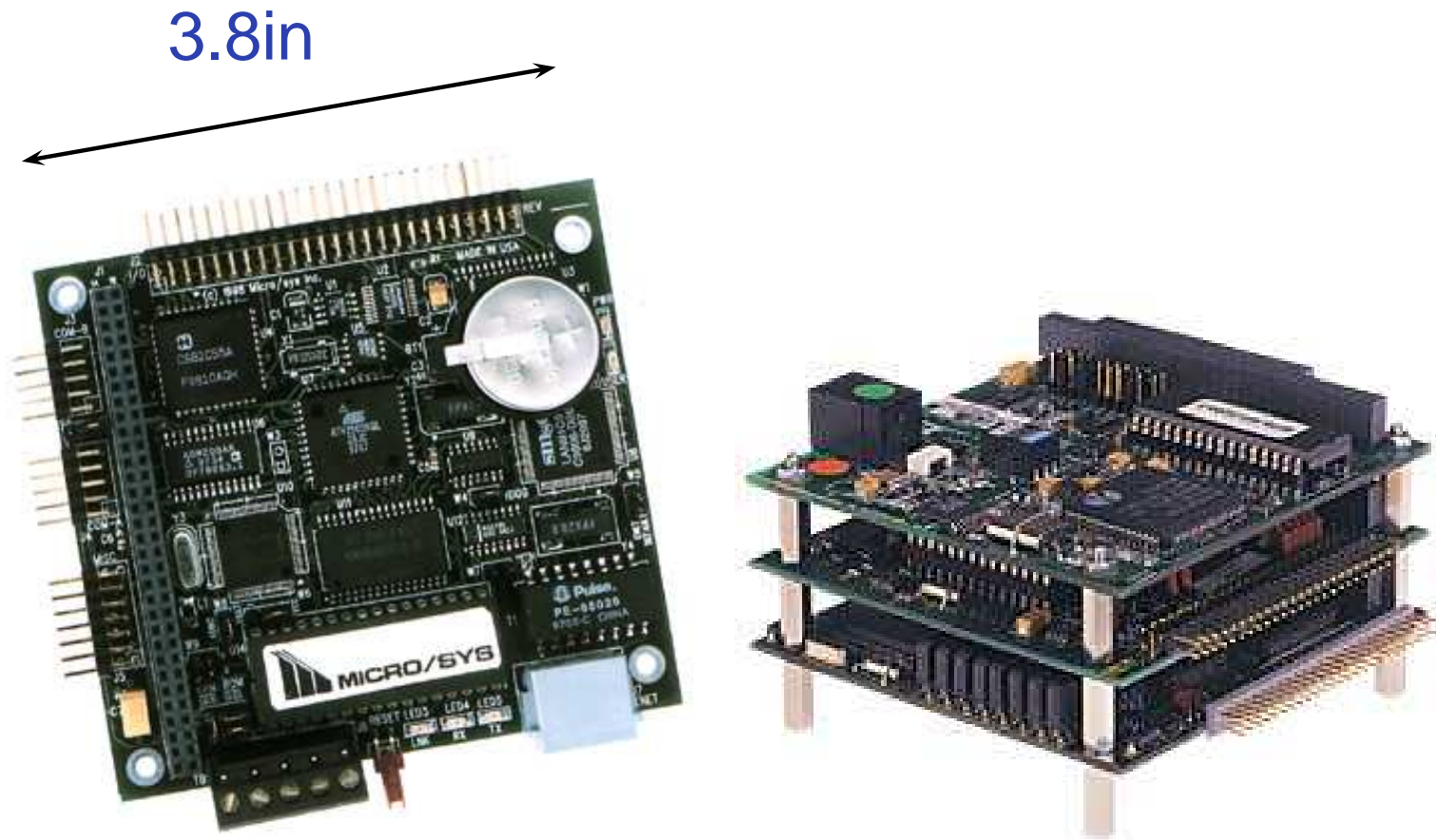
The ISA Bus: Memory Write
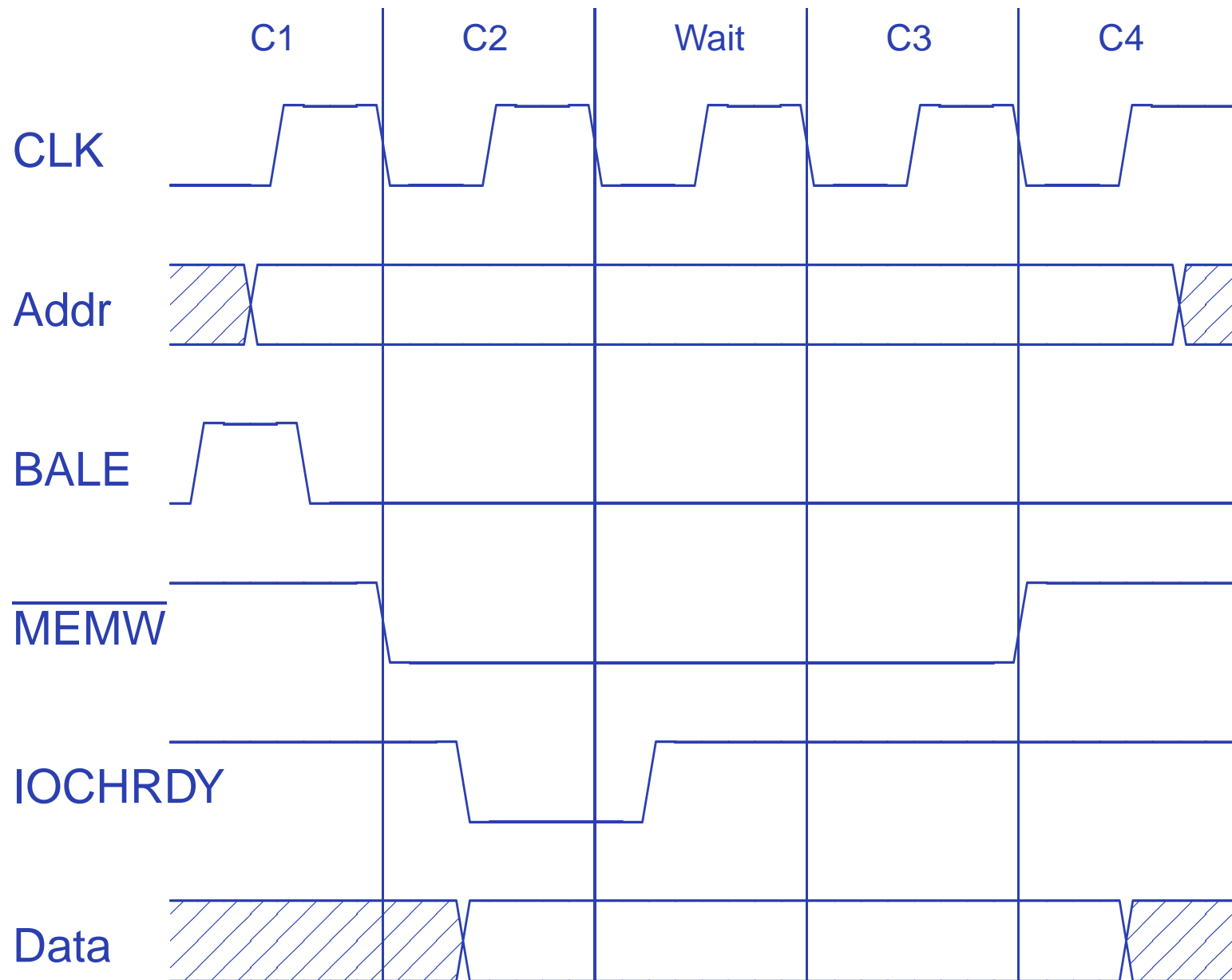
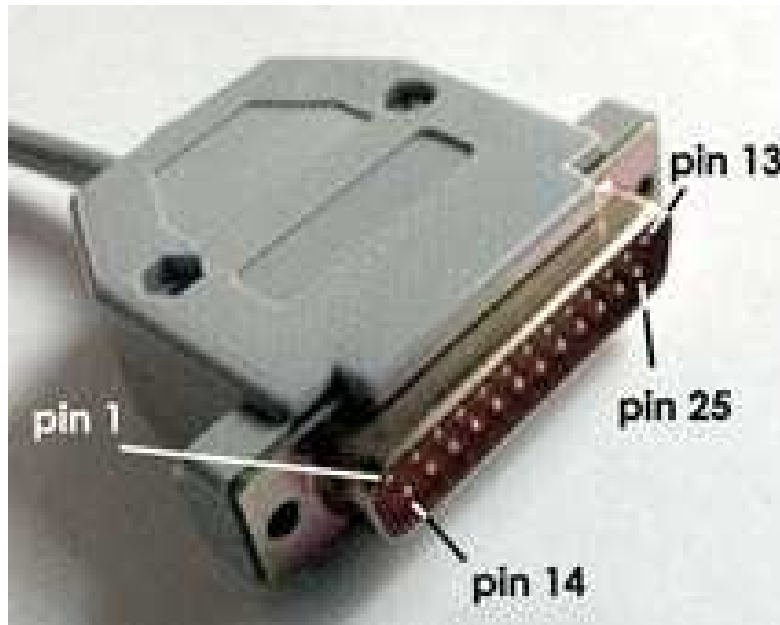# The PC/104 Form Factor: ISA Lives

3.8in
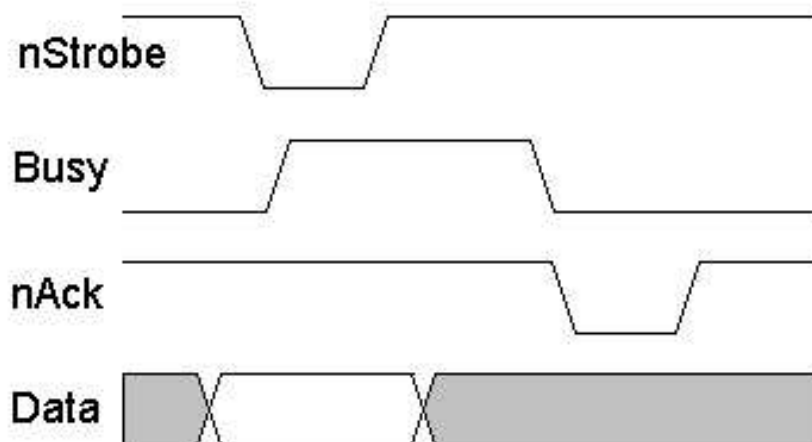
Embedded System Legos. Stack 'em and go.

Memory-Mapped I/O

- To a processor, everything is memory.

- Peripherals appear as magical memory locations.

- Status registers: when read, report state of peripheral

- Control registers: when written, change state of peripheral

```
At Standard TTL Levels
                Signal          Adapter Pin
                 Name            Number
             ┌─────────────────────────────┐
          ←──────-Strobe────────────1──
        E ←──────+Data Bit 0───────2──── P
        X ←──────+Data Bit 1───────3──── A
        T ←──────+Data Bit 2───────4──── R
        E ←──────+Data Bit 3───────5──── A
        R ←──────+Data Bit 4───────6──── L
        N ←──────+Data Bit 5───────7──── L
        A ←──────+Data Bit 6───────8──── E
        L ←──────+Data Bit 7───────9──── L
             ────────-Acknowledge────10─→
        D ────────+Busy────────────11─→ A
        E ────────+Paper End───────12─→ D
        V ────────+Select──────────13─→ A
        I ←──────-Auto Feed────────14── P
        C ────────-Error ──────────15─→ T
        E ←──────-Initialize───────16── E
             ←──────-Select Input───17── R
             ────────Ground─────────18-25──
```

Centronics Handshake

nStrobe

Busy

nAck

Data

Parallel Port Registers

D7	D6	D5	D4	D3	D2	D1	D0	
D7	D6	D5	D4	D3	D2	D1	D0	0x378
$\overline{\text{Busy}}$	Ack	Paper	Sel	Err				0x379
				$\overline{\text{Sel}}$	Init	$\overline{\text{Auto}}$	$\overline{\text{Strobe}}$	0x37A

1. Write Data
2. Assert Strobe
3. Wait for Busy to clear
4. Wait for Acknowledge

Centronics Handshake

nStrobe

Busy
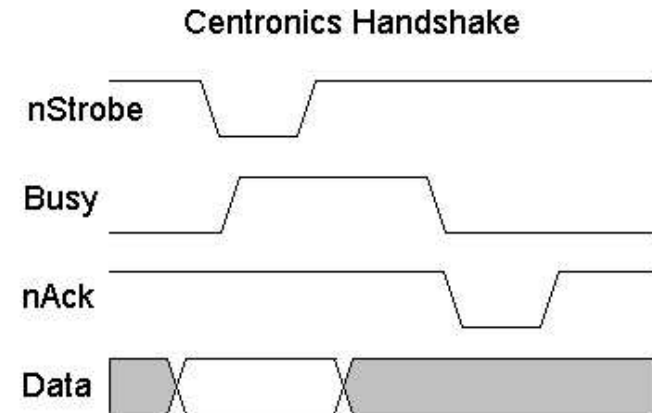
nAck

Data

A Parallel Port Driver

```
#define DATA    0x378
#define STATUS  0x379
#define CONTROL 0x37A

#define NBSY 0x80
#define NACK 0x40
#define OUT  0x20
#define SEL  0x10
#define NERR 0x08
#define STROBE 0x01

#define INVERT  (NBSY | NACK |       SEL | NERR)
#define MASK    (NBSY | NACK | OUT | SEL | NERR)
#define NOT_READY(x) ((inb(x)^INVERT)&MASK)

void write_single_character(char c) {
  while (NOT_READY(STATUS)) ;
  outb(DATA, c);
  outb(CONTROL, control | STROBE); /* Assert STROBE */
  outb(CONTROL, control ); /* Clear STROBE */
}
```

Interrupts and Polling

Two ways to get data from a peripheral:

- Polling: "Are we there yet?"
- Interrupts: Ringing Telephone
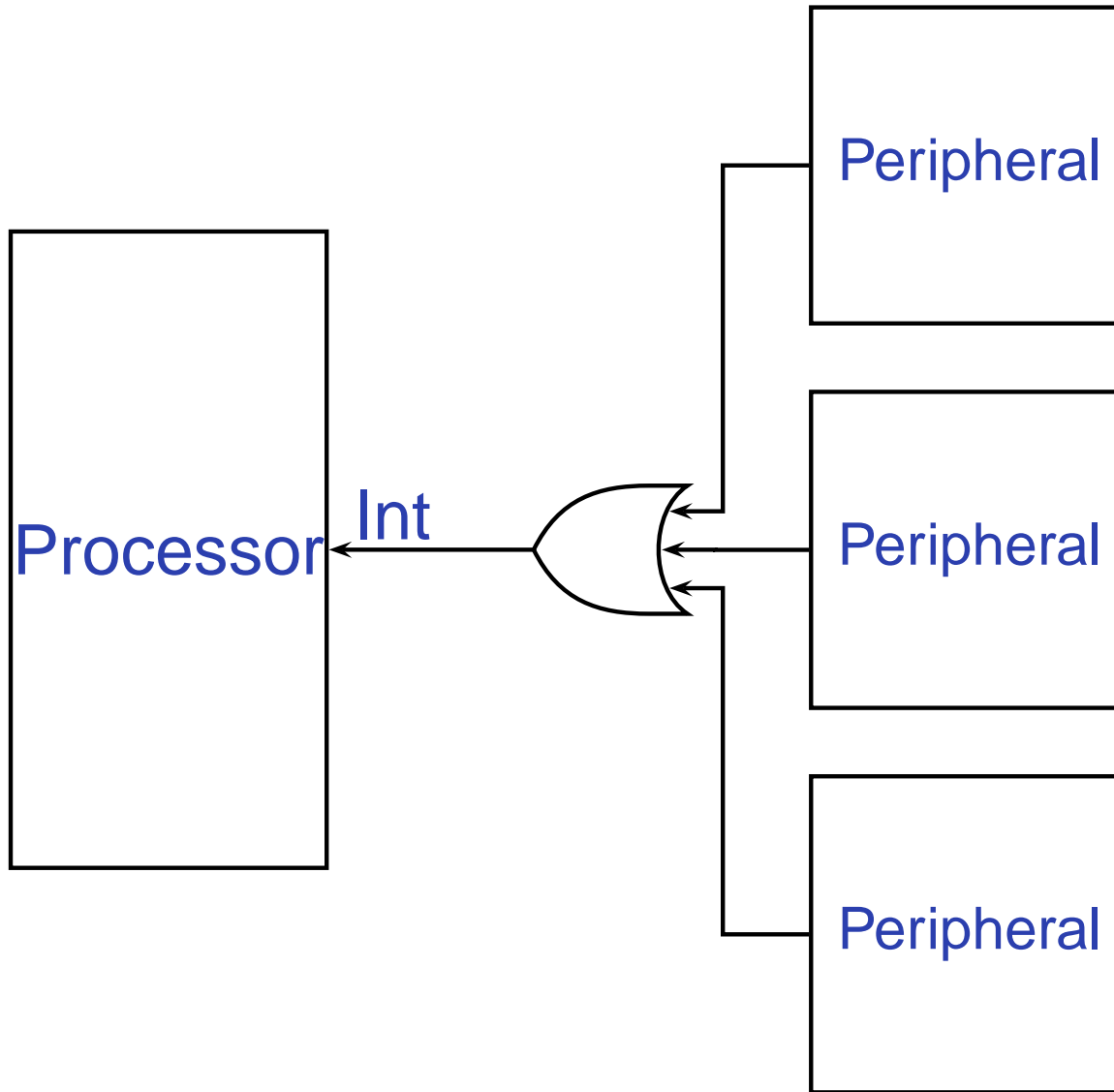
Interrupts

Basic idea:

1. Peripheral asserts a processor's interrupt input

2. Processor temporarily transfers control to interrupt service routine

3. ISR gathers data from peripheral and acknowledges interrupt

4. ISR returns control to previously-executing program
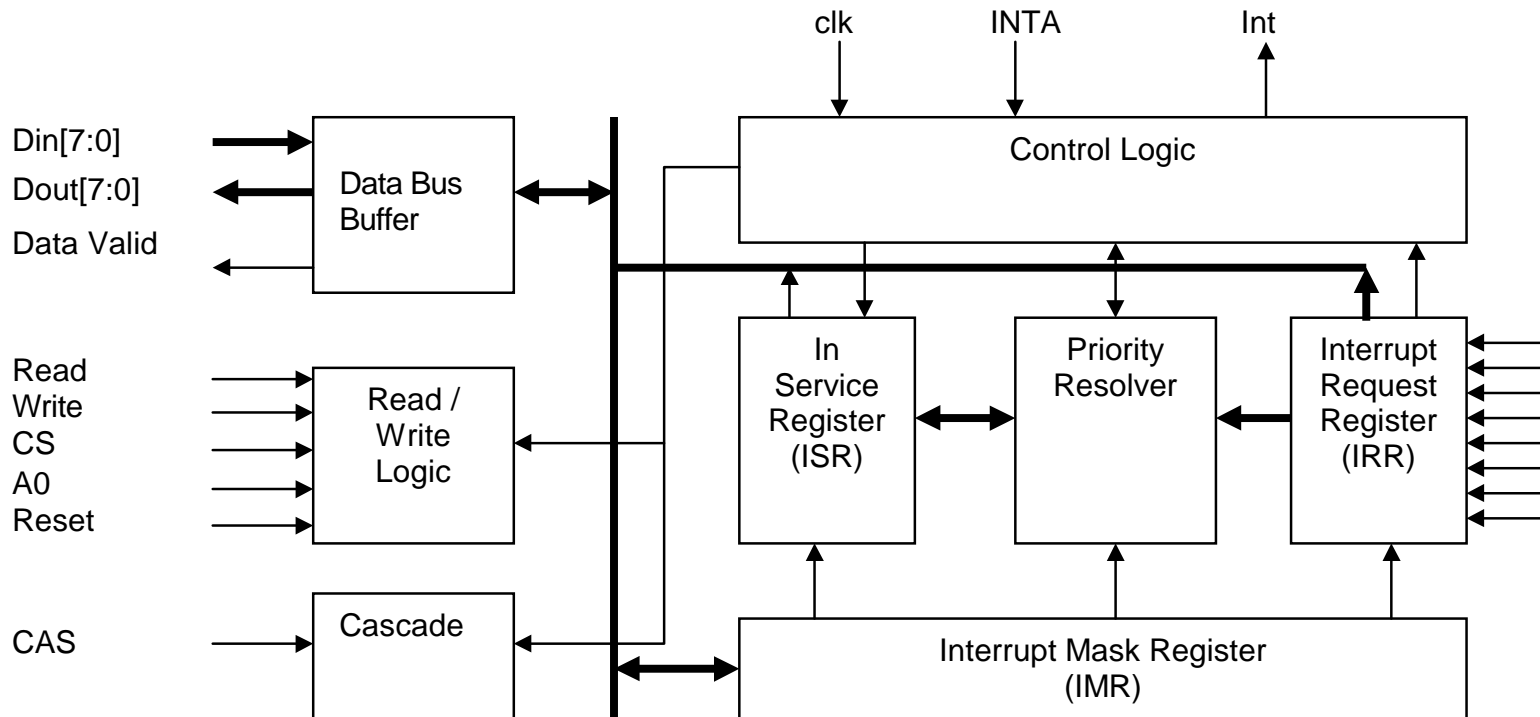
Many Different Interrupts

Processor

Int

?

Peripheral

Peripheral

Peripheral
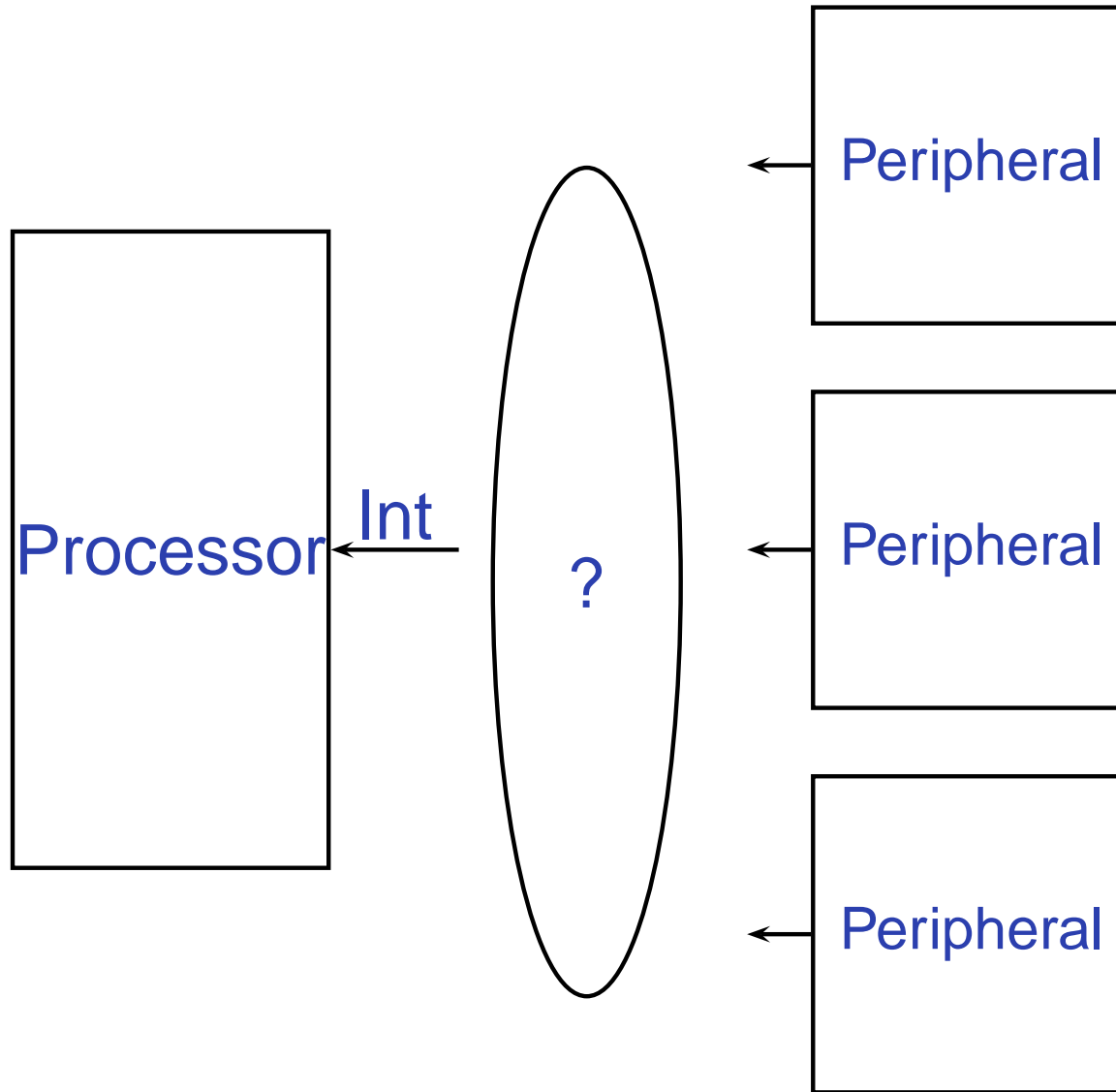
What's a processor to do?

Interrupt Polling

Processor

Int

Peripheral

Peripheral

Peripheral

Processor receives interrupt
ISR polls all potential interrupt sources

Intel 8259 PIC

Prioritizes incoming requests & notifies processor

ISR reads 8-bit interrupt vector number of winner

IBM PC/AT: two 8259s; became standard