

Hardware-Software Interfaces

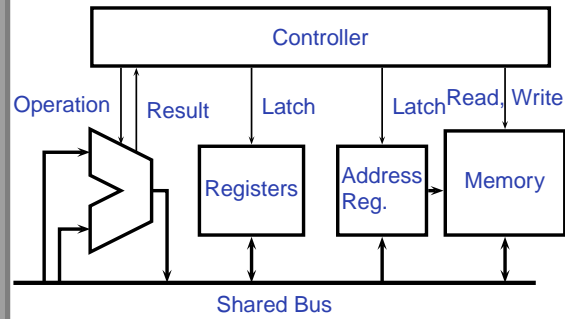
CSEE W4840

Prof. Stephen A. Edwards

Columbia University

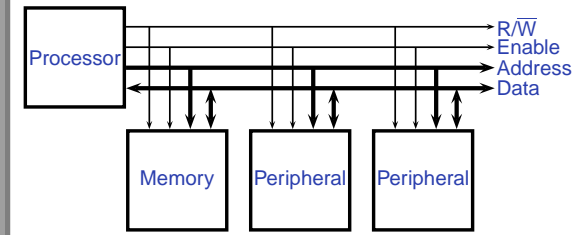
Hardware-Software Interfaces - p. 17

Basic Processor Architecture



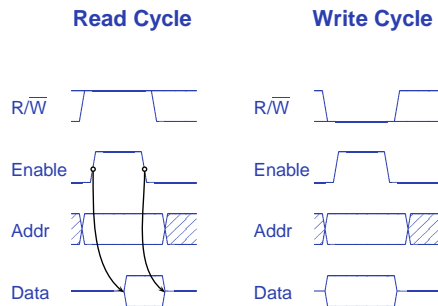
Hardware-Software Interfaces - p. 27

Typical Processor System



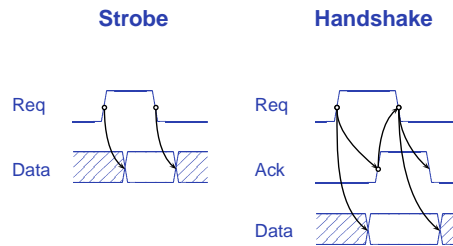
Hardware-Software Interfaces - p. 37

Simple Bus Timing



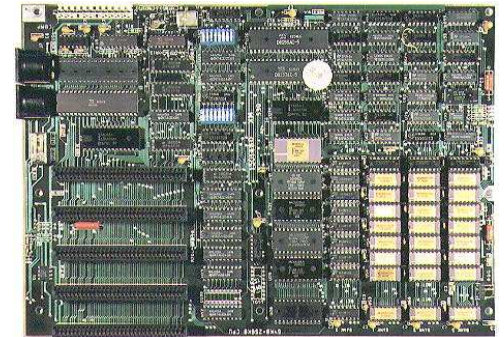
Hardware-Software Interfaces - p. 47

Strobe vs. Handshake



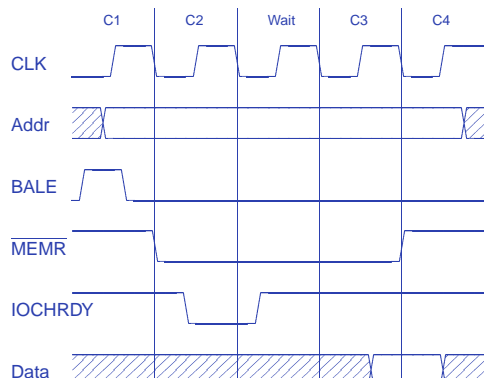
Hardware-Software Interfaces - p. 57

1982: The IBM PC

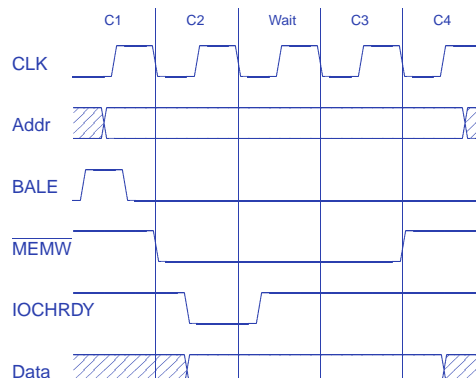


Hardware-Software Interfaces - p. 67

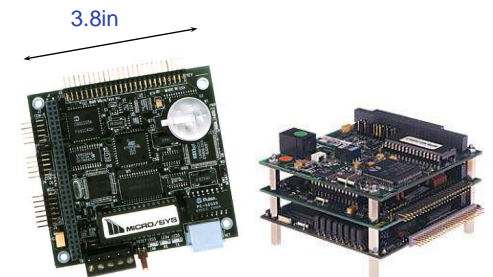
The ISA Bus: Memory Read



The ISA Bus: Memory Write



The PC/104 Form Factor: ISA Lives



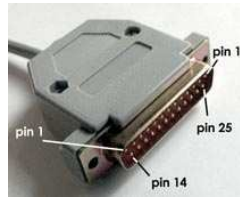
Embedded System Legos. Stack 'em and go.

Memory-Mapped I/O

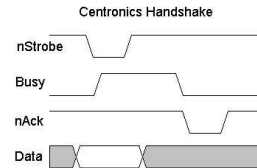
- To a processor, everything is memory.
- Peripherals appear as magical memory locations.
- Status registers: when read, report state of peripheral
- Control registers: when written, change state of peripheral

Hardware-Software Interfaces - p. 107

Typical Peripheral: PC Parallel Port



Signal	Adapter Pin Number
-Strobe	1
+Data Bit 0	2
+Data Bit 1	3
+Data Bit 2	4
+Data Bit 3	5
+Data Bit 4	6
+Data Bit 5	7
+Data Bit 6	8
+Data Bit 7	9
-Acknowledge	10
+Busy	11
+Paper End	12
+Select	13
+Auto Feed	14
-Error	15
-Initialize	16
-Select Input	17
-Ground	18-25

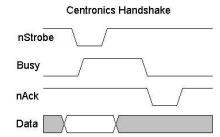


Hardware-Software Interfaces - p. 117

Parallel Port Registers

D7	D6	D5	D4	D3	D2	D1	D0	0x378
Busy	Ack	Paper	Sel	Err				0x379
				Sel	Init	Auto	Strobe	0x37A

- Write Data
- Assert Strobe
- Wait for Busy to clear
- Wait for Acknowledge



Hardware-Software Interfaces - p. 127

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 */
}
    
```

Hardware-Software Interfaces - p. 137

Interrupts and Polling

Two ways to get data from a peripheral:

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

Hardware-Software Interfaces - p. 147

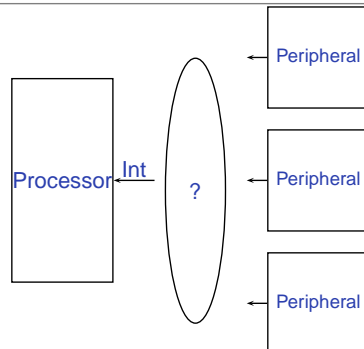
Interrupts

Basic idea:

- Peripheral asserts a processor's interrupt input
- Processor temporarily transfers control to interrupt service routine
- ISR gathers data from peripheral and acknowledges interrupt
- ISR returns control to previously-executing program

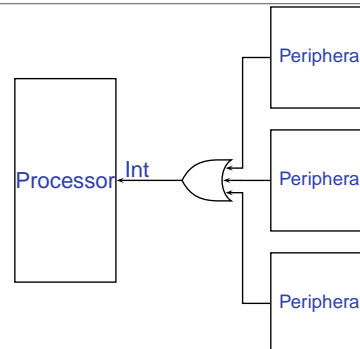
Hardware-Software Interfaces - p. 157

Many Different Interrupts



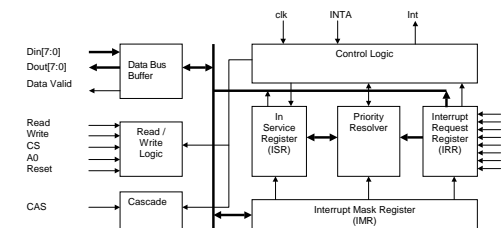
What's a processor to do?

Interrupt Polling



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