COMP 273
Introduction to Computer Systems

COMP 273

I/O & Peripheral Devices

Prof. Joseph Vybihal

Announcements

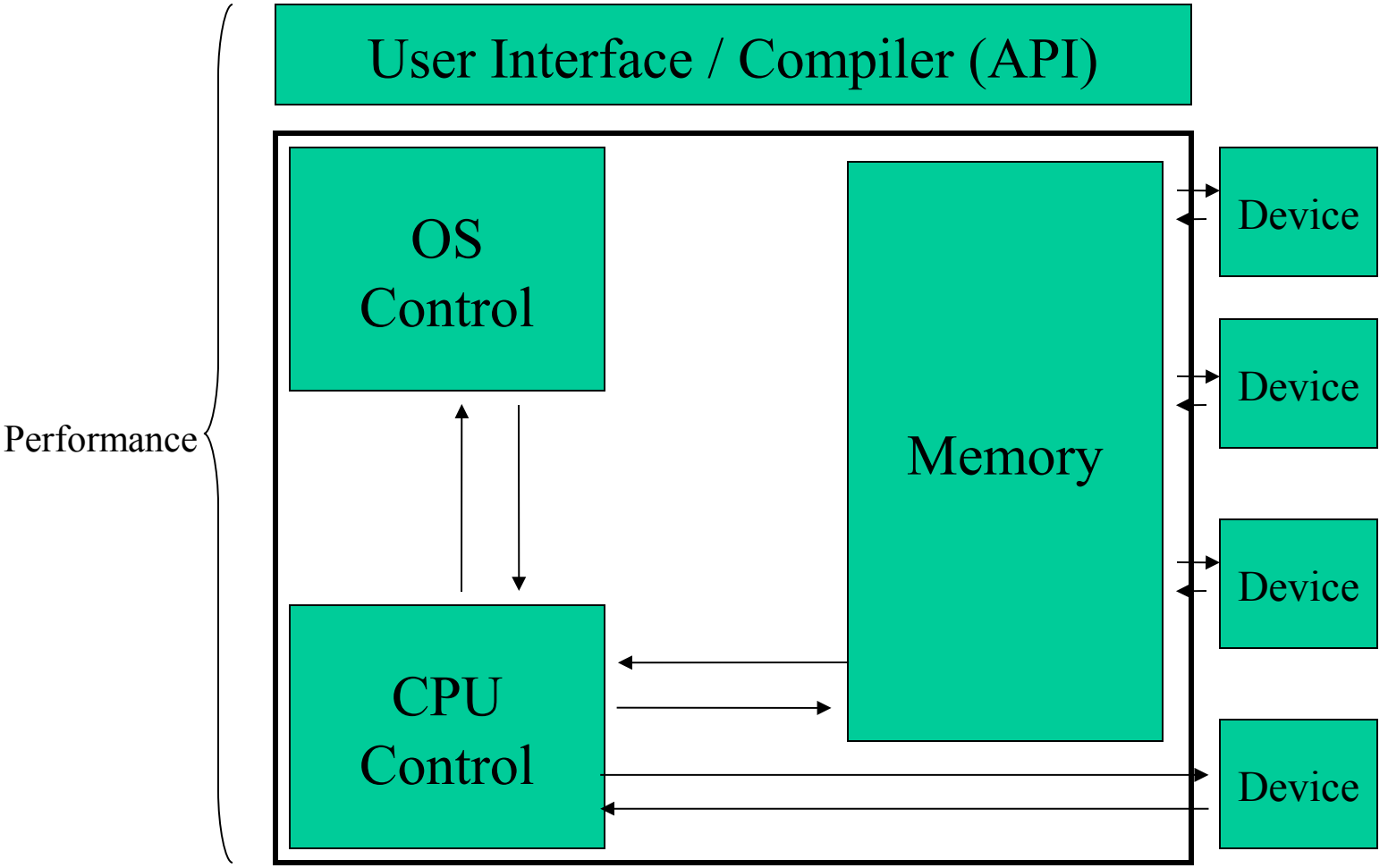
- Course Evaluations

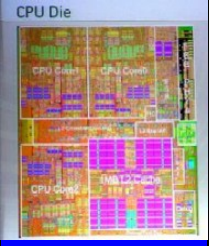
At Home

- Figure out how your printer port functions and write either a C or Assembler program to control it. Google the information.

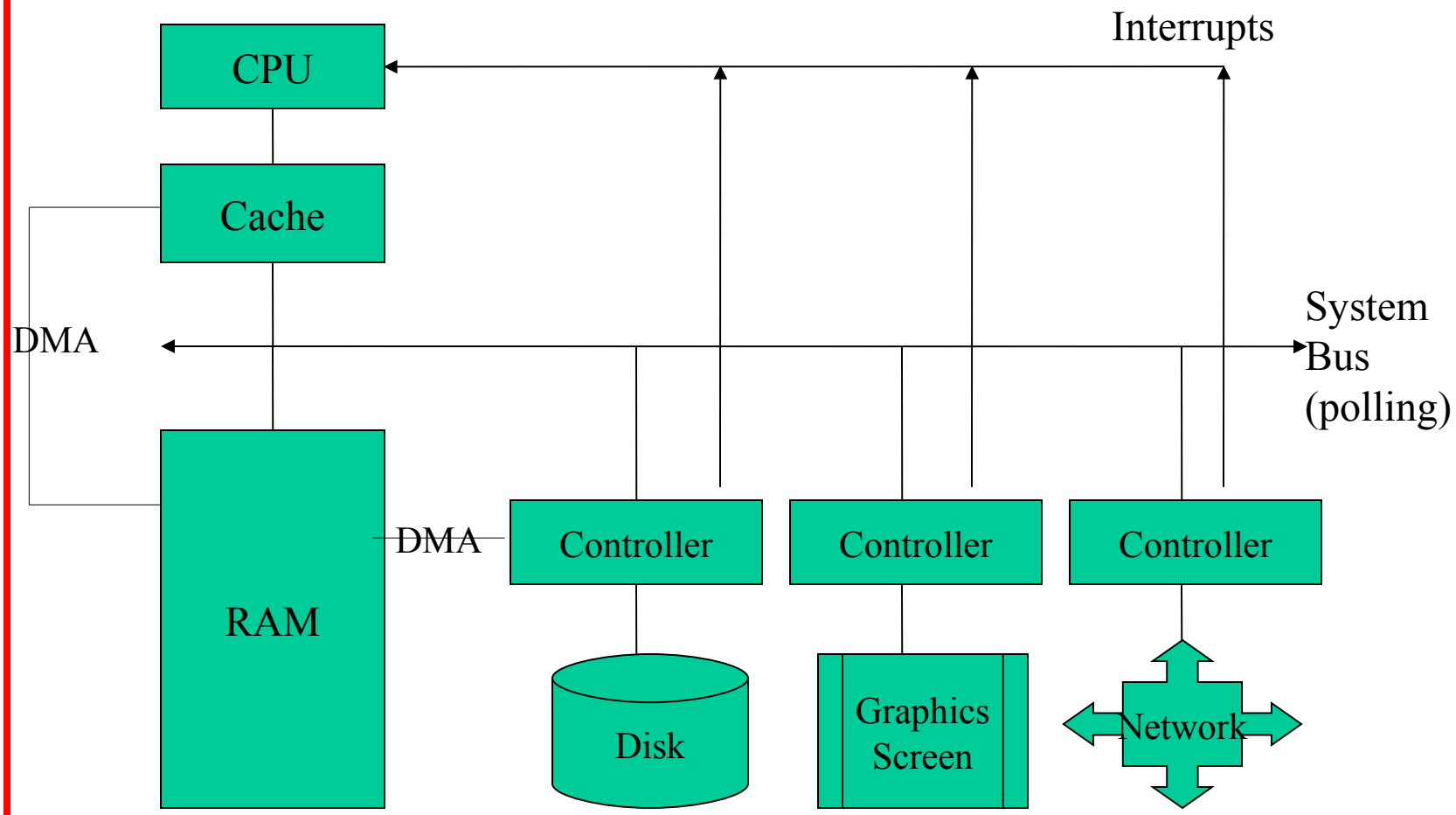
Hardware Views

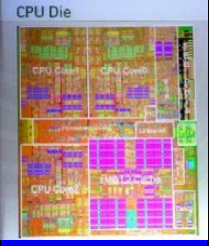
Logical View of Computing





Physical View of Computing





Peripherals

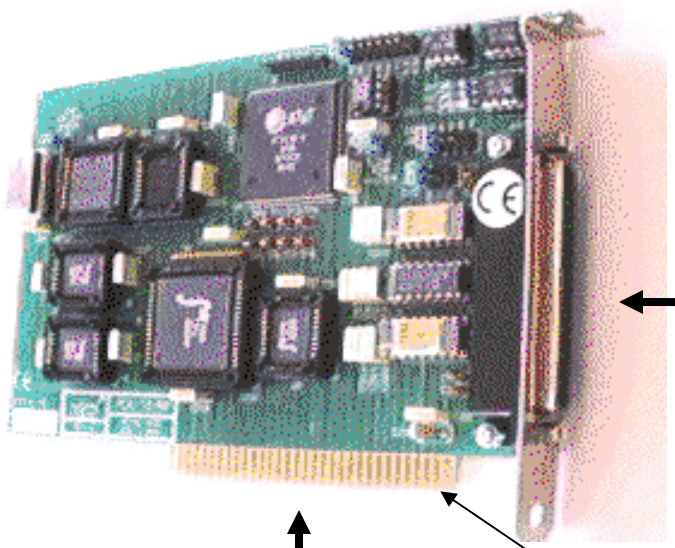
Peripheral

- Definition:
  - A device external to the system board.
- Usage:
  - sending and receiving data to/from device
- Types of controllers:
  - On Board Controllers

    Eg. keyboard, mouse, printer, video
    - The controlling registers are integrated with the motherboard
  - External Controllers

    Eg. video, network, machine controller
    - The controlling registers are part of a card plugged into the motherboard's slot
- In either case they operate similarly.

External Connector



Printer Port

One PIN of data (bit)

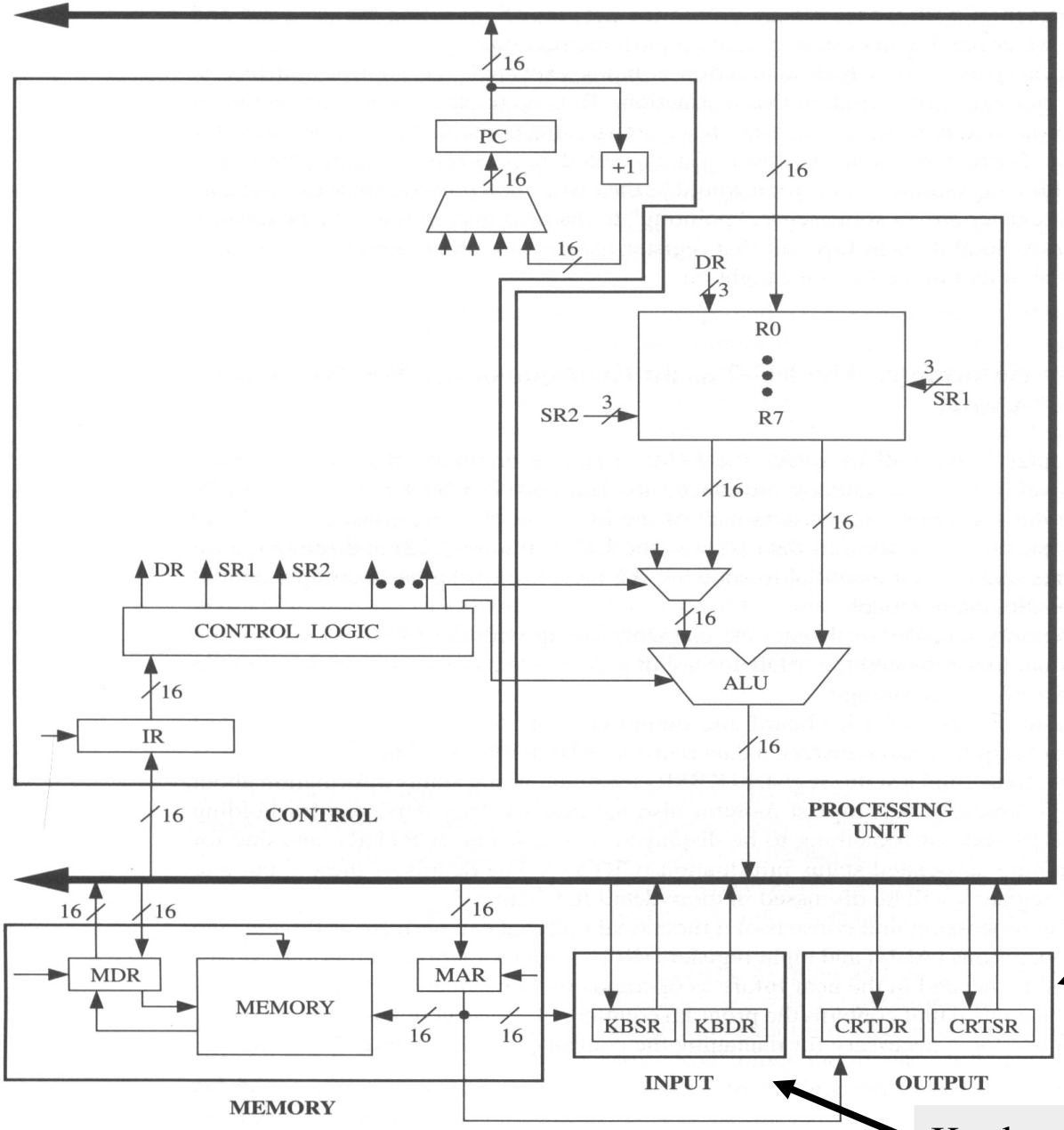
Interconnected

To Bus

A Slot

COMP 273
Introduction to Computer Systems

On-Board Interface



- Status Registers
- Data Registers

Monitor

Mono-directional

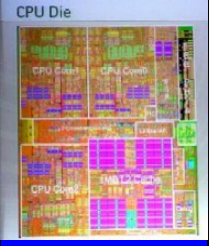
Keyboard

Figure 4.3 The LC-2 as an example of the Von Neumann model

Generic Keyboard Registers

- Keyboard Status Register
  - Bit 0: Key is being pressed
  - Bit 1: Keyboard buffer empty
  - Bit 2: Power exists in keyboard
  - Bit 3: Shift key is being pressed
  - Bit 4: Control key is being pressed
  - Bits 5-7: Unused, for future development
- Keyboard Data Register
  - 8 Bits that store the ASCII of key pressed

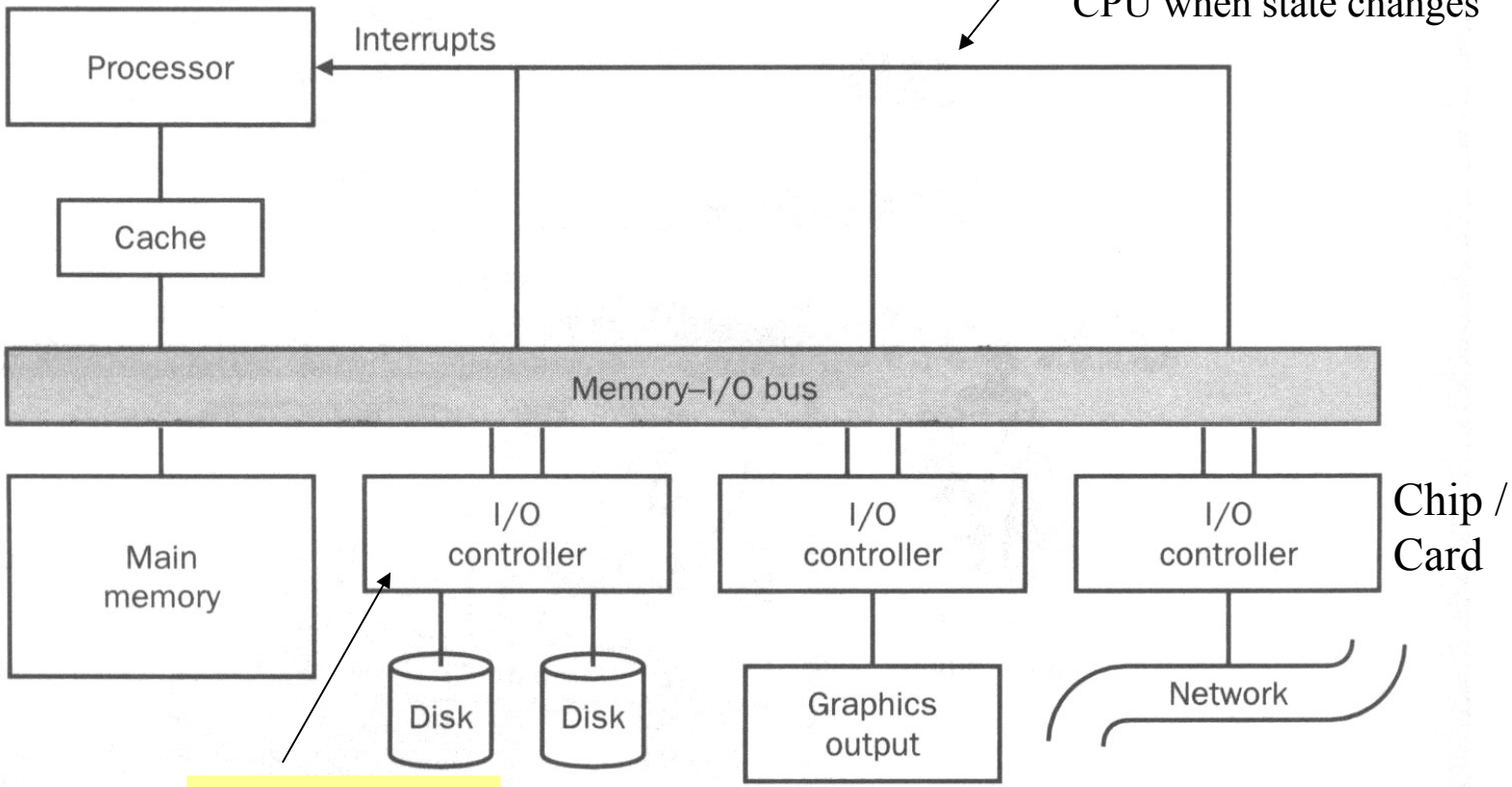
- What happens if we do not look at the registers before the next key press?



Communication Techniques

Interrupt driven

Device signals the CPU when state changes



Polling Driven

The CPU, when it wants, looks to the device's status register

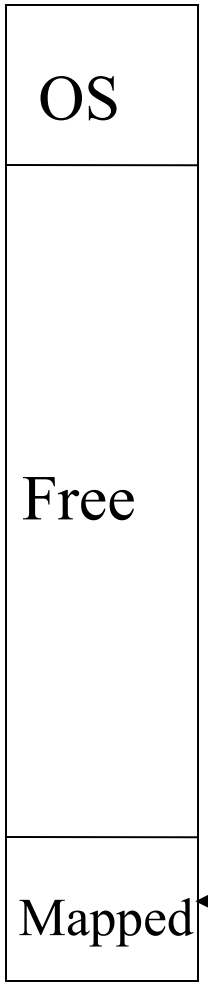
Asynchronous vs. Synchronous

- Synchronous I/O
  - CPU monitors the device and sends/reads 1 byte at a time.
- Asynchronous I/O
  - CPU loads a register with a start address
  - CPU loads another register with a limit
  - CPU gives command to device to download or upload information
  - CPU then does something else until device finished
  - Device sends an interrupt to say “done”

Common Hardware Interface

- Peripheral registers
  - Status: flags indicating the state
  - Data: information to be written or just read
  - Command: codes/switches activate motors
- Accessing the registers
  - Special data paths: special assembler commands
  - General data paths: RAM zero page

I/O Mapping



Memory Map Management:
- Back-plane direct connection
- OS manages using polling

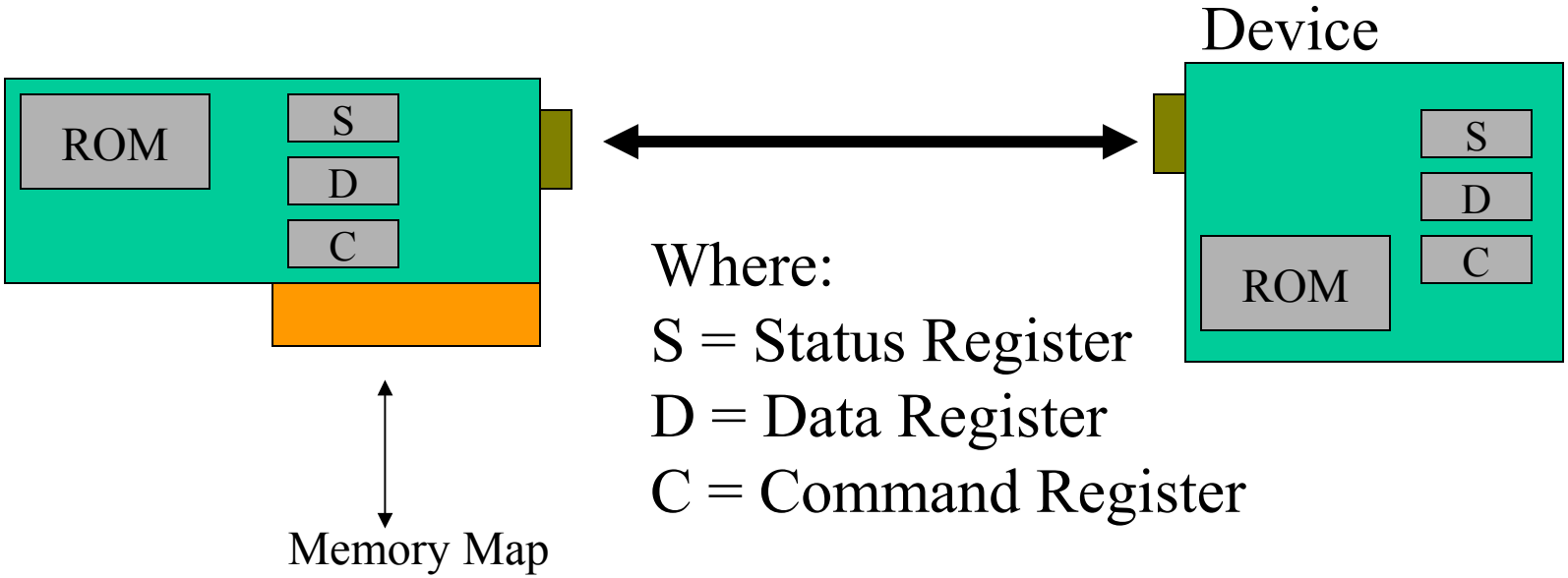
Registers:
- Status (busy, error, change)
- Data (actual I/O)
- Command (code number)

```
sw $t1, ADDRESS
```

Memory Mapped
A portion of RAM's memory is physically connected to the address space of the peripheral registers (on-board and off-board ~ via slots)

Draw

Flow Diagram



Where:
S = Status Register
D = Data Register
C = Command Register

Device Operation:
1. Device updates S and D
2. Waits a small unit of time and then reads C and D (possible data loss)

Polling Method:
1. Check S for change in a busy loop
2. If change then copy S and D to memory

Interrupt Method:
1. If S or D updated signal CPU
2. CPU Task switch to OS

Build Your Own Polling Program

1. Save register on stack
2. Poll the status register (mask the bits)
   - Busy? Go to 2
   - Error? Return error code, pop registers, return
   - New Data? Copy Data register, pop registers, return
   - Not Busy & Want to send data?
     - Load data register
     - Load command register
       - Automatic transmission to device in t micro sec

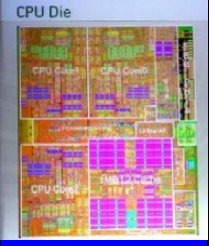
3 If nothing to do

- Pop registers and return

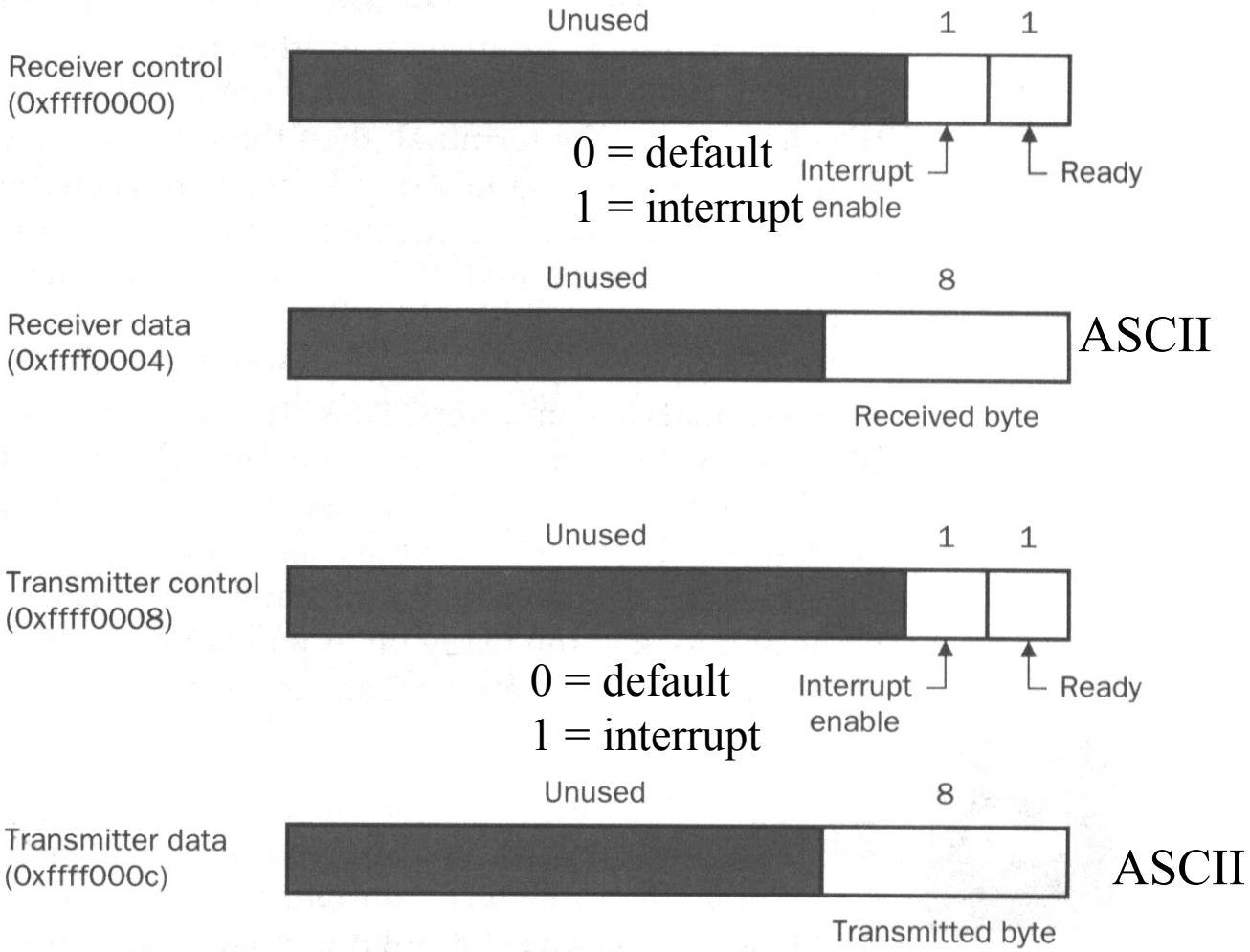
MIPS I/O Communication

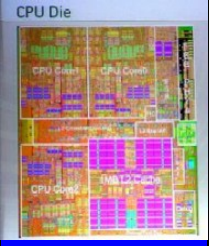
MIPS Limitations

- Only simulates one terminal connection:
  - Text Keyboard
  - Text video

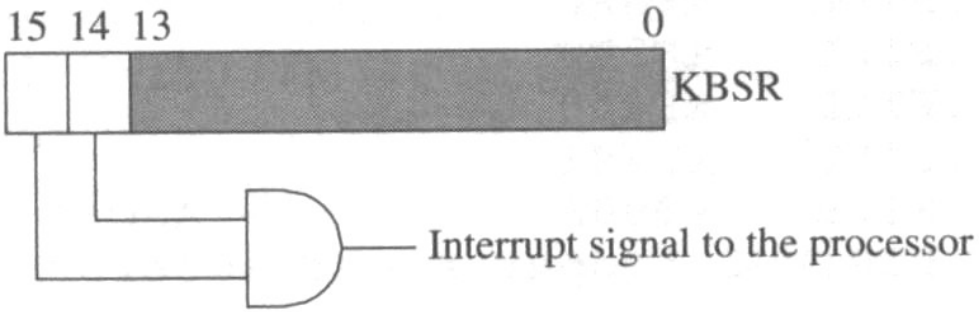


Terminal Connection





MIPS Interrupt Control



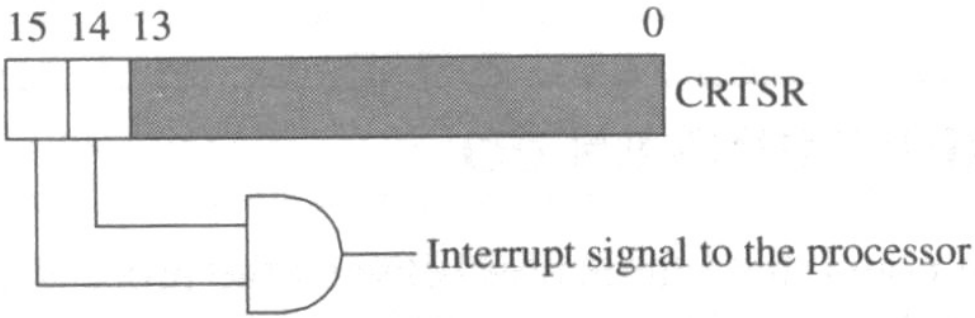
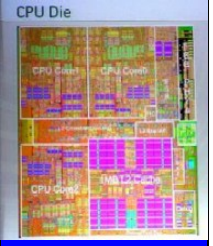


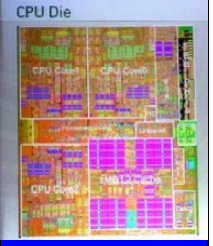
Figure 8.4 Interrupt enable bits and their use

Example Polling Code

```
##############################################
#                    GETCHAR
##############################################

GetChar:
        lui $a3, 0xffff          #base address of memory map
CkReady:
        lw $t1, 0($a3)           #read from receiver control register
        andi $t1, $t1, 0x1       #extract ready bit
        beqz $t1, CkReady        #if 1, then load char, else keep looping
        lw $v0, 4($a3)           #load character from keyboard
        jr $ra                   #return to place in program before function call
```



```
#############################################
#                   PUTCHAR
#############################################

PutChar:
        lui $a3, 0xffff         #base address of memory map
XReady:
        lw $t1, 8($a3)          #read from transmitter control register
        andi $t1, $t1, 0x1      #extract ready bit
        beqz $t1, XReady        #if 1, then store char, else keep looping
        sw $a0, 12($a3)         #send character to display
        jr $ra                  #return to place in program before
                                #       function call
```



COMP 273
Introduction to Computer Systems

Interrupts

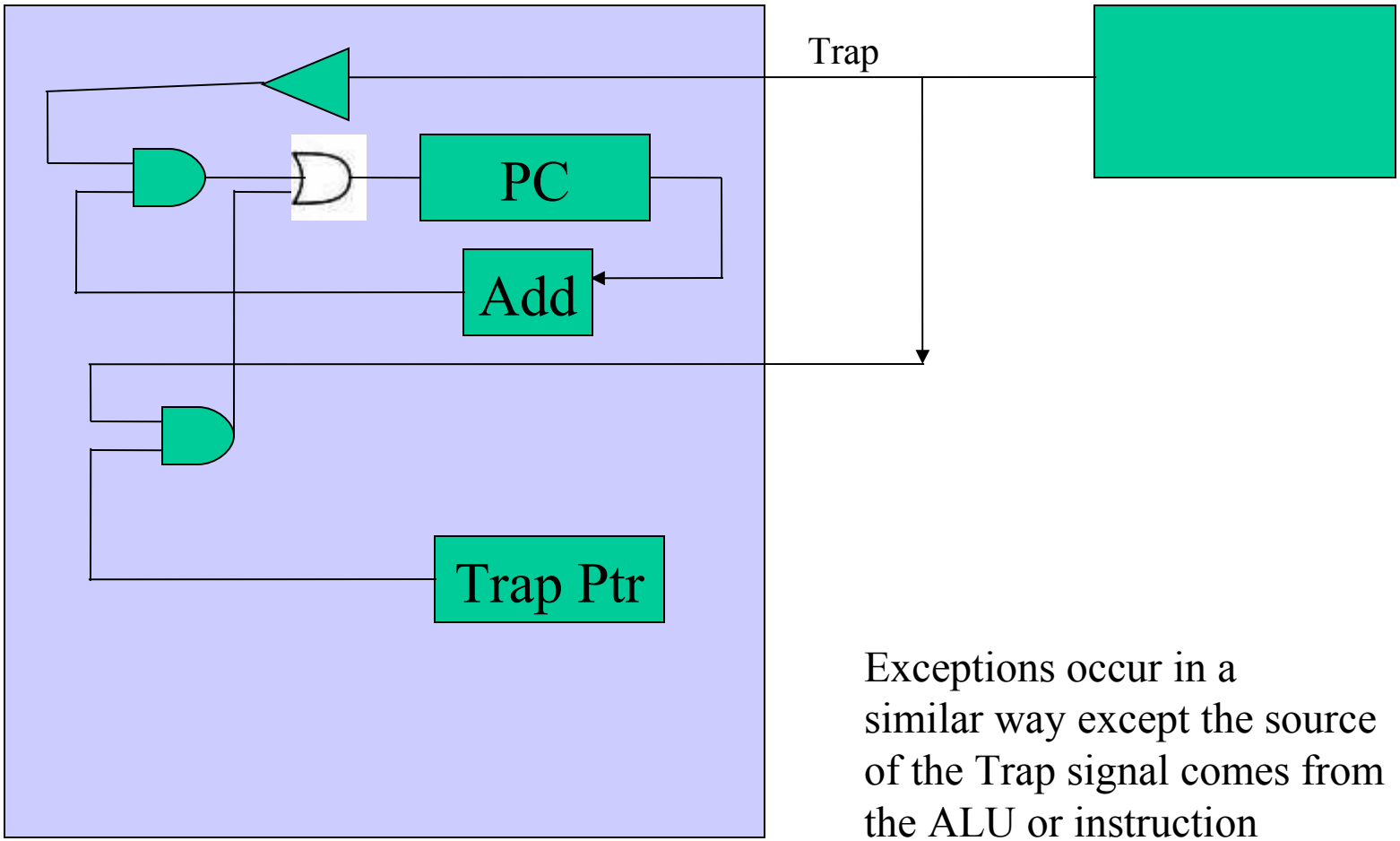
CPU Execution Changes

- Normal operation:
  - Fetch → Decode → Execute → Store → Repeat
- Exceptions
  - Any event that stops the normal flow of the CPU execution cycle. ie. interrupting the currently executing process due to a run-time error (overflow, divide by zero, illegal memory reference) or request.
- Interrupt
  - Signal
    - An event purposely triggered by the current process to re-route the CPU to execute another process or function
  - Trap
    - An event purposely triggered by a device to re-route the CPU to execute another process or function

Types of exceptions

- External events
  - Peripheral wants CPU time
- Memory translation
  - Non-existent memory location
- Run-time errors
  - Divide by zero
  - Overflow, etc.
- Coding Errors
  - Non-existent op-code
  - Privileged instruction
- Data Parity Integrity

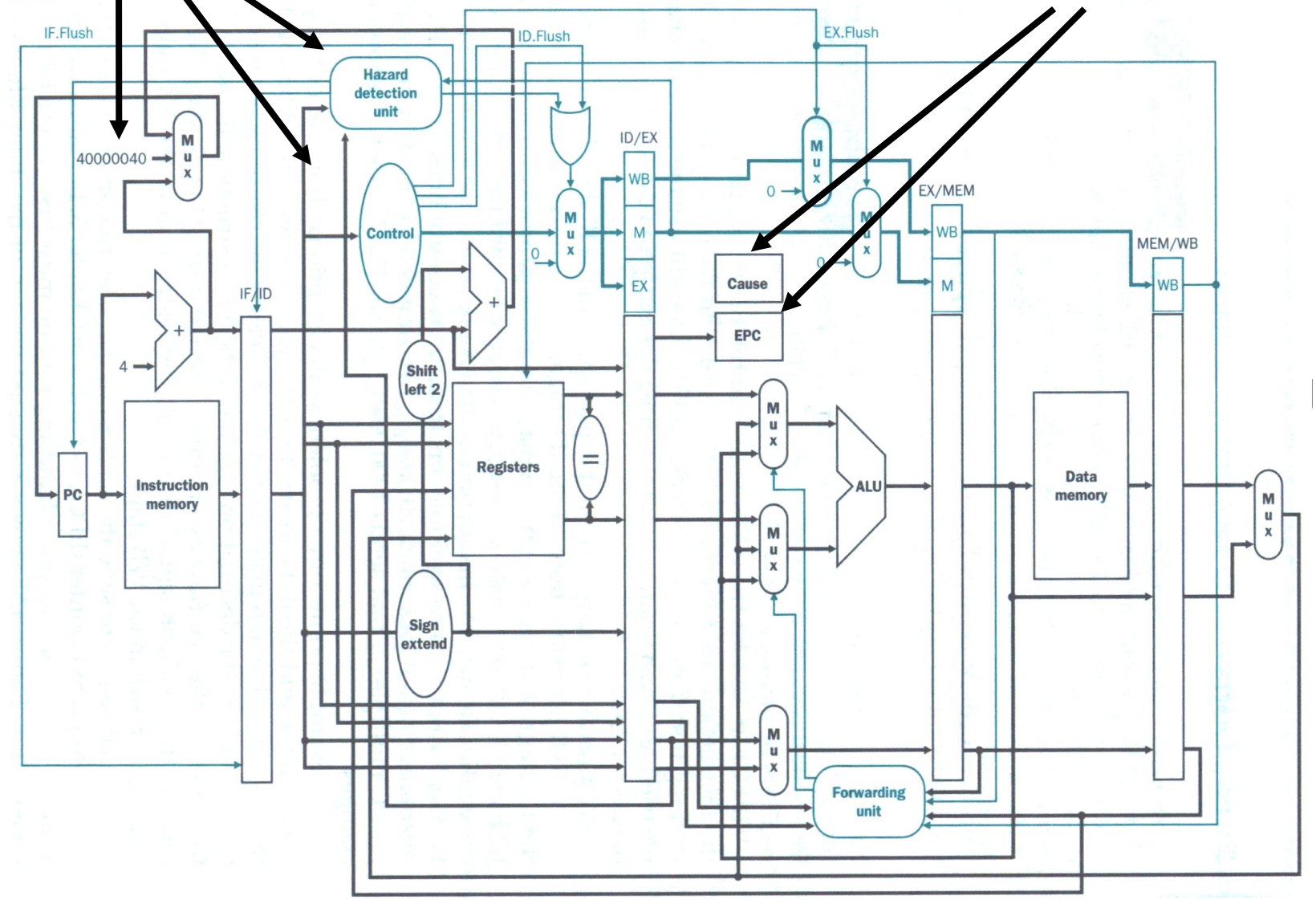
Hardware Implementation



Exceptions occur in a similar way except the source of the Trap signal comes from the ALU or instruction

COMP 273
Introduction to Computer Systems

MIPS Implementation



The Exception Co-Processor 0

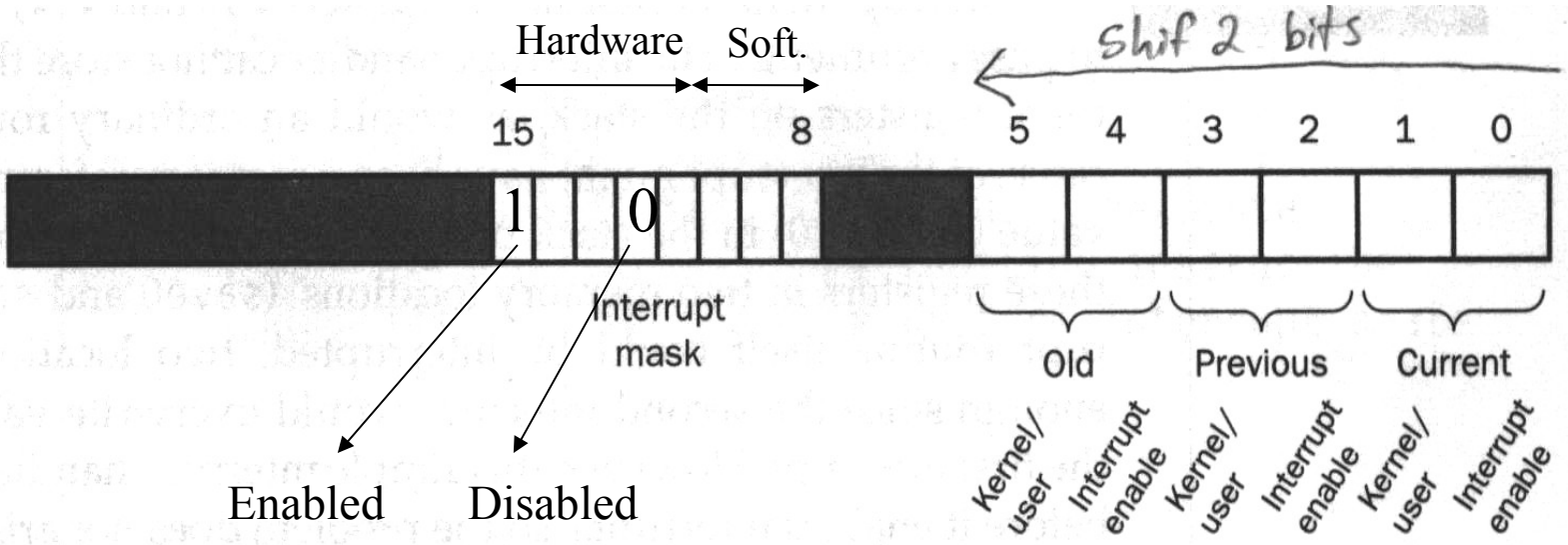
- EPC (Error PC)
  - Register 14
  - Contains the address of the affected instruction (where exception occurred)
- CAUSE
  - Register 13
  - Exception type & pending interrupt bits
- BadVAddr
  - Register 8
  - Contains the memory address at which the bad memory reference occurred
- Status
  - Register 12
  - Interrupt mask and enable bits

Co-Processor Accessibility

- Instructions
  - Lwc0 (load word to c0)
  - Mfc0 (move from c0 to CPU)
  - Mtc0 (move CPU to c0)
  - Swc0 (store word from c0)
- Example
  - Mfc0 $t1, $13 # move Cause to t1

The Status Register

Mini Interrupt Stack



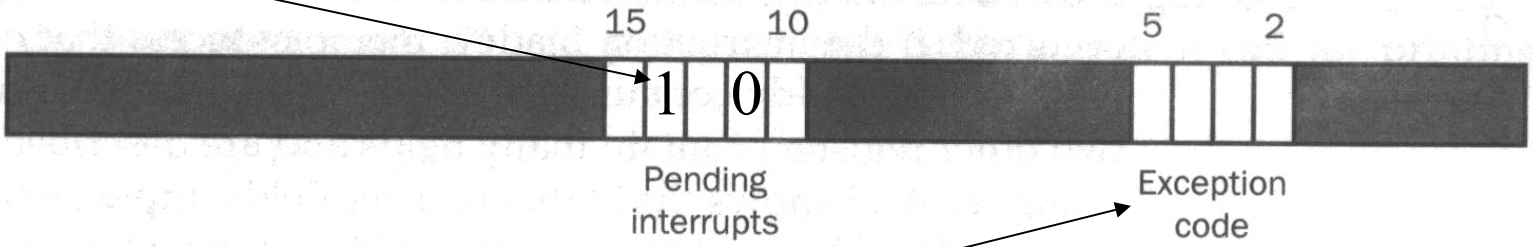
Algorithm:

Interrupt signal occurred
Shift left 2 bits
Default new bits set to zero
Kernel/user= 0 → interrupted when in kernel
Int Enabled= 1 → Enabled

The Cause Register

Interrupt occurred at this level, not yet serviced

Interrupt Levels



Number	Name	Description
0	INT	external interrupt
4	ADDRL	address error exception (load or instruction fetch)
5	ADDRS	address error exception (store)
6	IBUS	bus error on instruction fetch
7	DBUS	bus error on data load or store
8	SYSCALL	syscall exception
9	BKPT	breakpoint exception
10	RI	reserved instruction exception
12	OVF	arithmetic overflow exception

Notes

- All interruptions go to Kernel 80000080 Hex
- The code at that address is called the *interrupt handler*
- Default code is placed at this location by the OS so that interrupts automatically access the OS
  - Program error (run-time error)
  - Program request for OS service (print to screen)
  - External device needs CPU
- Programmer's can put their own code here

TABLE 5.1 Hardwired reset and exception entry points for MIPS CPU

Exception type	*Entry point*			
	SR(BEV)==0		*SR(BEV)==1*	
	Program	*Physical*	*Program*	*Physical*
Reset, NMI			0xBFC0 0000	0x1FC0 0000
TLB refill, 32-bit task	0x8000 0000	0x0	0xBFC0 0200	0x1FC0 0200
XTLB refill, 64-bit task	0x8000 0080	0x80	0xBFC0 0280	0x1FC0 0280
Cache error (R4x00 and later)	0xA000 0100	0x100	0xBFC0 0300	0x1FC0 0300
Interrupt (some QED CPUs only)	0x8000 0200	0x200	0xBFC0 0400	0x1FC0 0400
All other exceptions	0x8000 0180	0x180	0xBFC0 0380	0x1FC0 0380

Basic Interrupt Processing

```
Cause = Get value in cause register
switch(cause)
{
    Case 1: to handle specific issue
    Case 2: ...
    Default:
         return, or print error and return
}
```

Example

```
            .ktext 0x80000080
            sw $a0, save0    # Handler is not re-entrant and can't use
            sw $a1, save1    # stack to save $a0, $a1
                             # Don't need to save $k0/$k1

            mfc0    $k0, $13        # Move Cause into $k0
            mfc0    $k1, $14        # Move EPC into $k1

            sgt     $v0, $k0, 0x44 # Ignore interrupts
            bgtz    $v0, done

            mov     $a0, $k0        # Move Cause into $a0
            mov     $a1, $k1        # Move EPC into $a1
            jal     print_excp      # Print exception error message
    done:
            lw    $a0, save0
            lw    $a1, save1
            addiu $k1, $k1, 4  # Do not reexecute
                               # faulting instruction
            rfe                # Restore interrupt state
            jr    $k1

            .kdata
    save0:  .word 0
    save1:  .word 0
```

Set >

Bigger pending interrupts (1000100)

External interrupt?

(mask & bits in Status)

Miscellaneous Peripherals

The following examples are not part of the course and only for your information or for experimentation.

IBM Memory Map

7.43. GAME ADAPTER I/O PORT USAGE

Port	*Direction*	*Function*
201H	Write	Fire joysticks four one-shots
	Read	Read joystick position and status

Notes: Resistive inputs are read by first outputting to port 201H, then noting the amount of time they remain high by inputting continuously from port 201H

Source: IBM Technical Reference Options and Adapters Volume 2, pages Game Adapter 3 to 6

See Also: 7.44. Game Adapter AB Joystick Data Byte
7.45. Game Adapter ABCD Paddle Data Byte

7.44. GAME ADAPTER AB JOYSTICK DATA BYTE

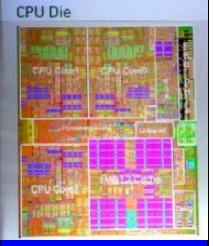
Bit Numbers

7	*6*	*5*	*4*	*3*	*2*	*1*	*0*	*Function*
X								Status of B joystick button 2
	X							Status of B joystick button 1
		X						Status of A joystick button 2
			X					Status of A joystick button 1
				X				B joystick Y coordinate*
					X			B joystick X coordinate*
						X		A joystick Y coordinate*
							X	A joystick X coordinate*

Notes: *Coordinates are determined by the length of time the bit is held high

Source: IBM Technical Reference Options and Adapters Volume 2, page Game Adapter 6

See Also: 7.43. Game Adapter I/O Port Usage
7.45. Game Adapter ABCD Paddle Data Byte



IBM Memory Map

7.41. MODEM CONTROL REGISTER

Bit Number

7	6	5	4	3	2	1	0	*Function*	*State on Reset*	*Allowable Values*
X	X	X						Always zero	000	No function
			X					Loopback test mode	0	0=disabled; 1=enabled
				X				-OUT2 signal	0	0=-OUT2 forced high; 1=-OUT2 forced low
					X			-OUT1 signal	0	0=-OUT1 forced high; 1=-OUT1 forced low
						X		-RTS output	0	0=-RTS forced high; 1=-RTS forced low
							X	-DTR output	0	0=-DTR forced high; 1=-DTR forced low

Source: IBM Technical Reference Options and Adapters Volume 2, pages Async 15 to 16

See Also: 7.35. Asynchronous Adapter I/O Port Usage

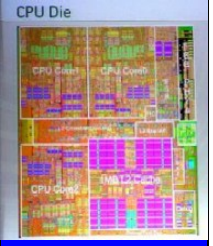
7.42. MODEM STATUS REGISTER

Bit Number

7	6	5	4	3	2	1	0	*Function*	*State on Reset*	*Allowable Values*
X								-RLSD complement	Input signal	
	X							-RI complement	Input signal	
		X						-DSR complement	Input signal	
			X					-CTS complement	Input signal	
				X				Delta RLSD	0	0=no change; 1=-RLSD has changed state
					X			Trailing edge ring detector	0	0=no TE RI; 1=-RI has changed to OFF
						X		Delta DSR indicator	0	0=no change; 1=-DSR has changed state
							X	Delta CTS indicator	0	0=no change; 1=-CTS has changed state

Source: IBM Technical Reference Options and Adapters Volume 2, pages Async 16 to 18

See Also: 7.35. Asynchronous Adapter I/O Port Usage



IBM Memory Map

7.09. PC & XT TYPEAHEAD BUFFER LAYOUT

Offset	*Length*	*Name*	*Description*
0 (0)	word	Buffer_Head	Points to next character in buffer
2 (2)	word	Buffer_Tail	Points to next blank space in buffer
4 (4)	32 bytes	Buffer_Area	Area used to store keystroke data

Notes:
- If Buffer_Head = Buffer_Tail, the buffer is empty
- Two bytes are necessary to store each keystroke, since the IBM extended keys (F1-F10, for example) consist of two byte codes. If the first byte for a keystroke is non-zero, then it represents the ASCII key and the second byte will be zero. If the first byte is zero, then it represents an extended key, and the second byte indicates the actual key pressed.
- Two low-memory words store the location of the buffer start (at 0040:0080) and one byte past its end (at 0040:0082)
- On a standard PC, the keyboard buffer is usually located at 0040:001A

Source: IBM PC/XT Technical Reference, BIOS Listing, page A-3 (original manuals only)
IBM PS/2 and PC BIOS Interface Technical Reference, Page 3-5

See Also: 4.002. BIOS Memory Usage Summary

7.10. AT KEYBOARD STATUS REGISTER

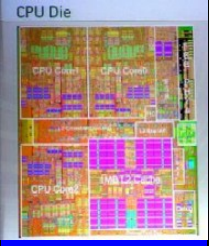
Bit Numbers

7	*6*	*5*	*4*	*3*	*2*	*1*	*0*	*Name*	*Allowable Values*
X								Parity error	0=odd parity (no error), 1=even parity
	X							Receive time out	0=no error, 1=keyboard did not finish
		X						Transmit time out	0=no error, 1=keyboard did not finish
			X					Inhibit switch	0=keyboard inhibited, 1=not inhibited
				X				Command/data	0=addressed as port 60H, 1=port 64H
					X			System flag	0=reset by power ON, 1=self test OK
						X		Input buffer full	0=empty, 1=full
							X	Output buffer full	0=empty, 1=full

Notes: The status register is at I/O address 64H

Source: IBM PC/AT Technical Reference, pages 1-49 to 1-50

See Also: 7.11. AT Keyboard I/O Command Summary
7.12. AT Keyboard Input Port Bit Definitions
7.13. AT Keyboard Output Port Bit Definitions



7.11. AT KEYBOARD I/O COMMAND SUMMARY

Command Value	Command Name	Comments	Bit Numbers 7	6	5	4	3	2	1	0
20H	Read keyboard controller									
60H	Write keyboard controller	Writes command byte -- see bitmap at right								
		RESERVED--always 0	0							
		IBM PC compatibility mode		X						
		IBM PC mode			X					
		Disable keyboard				X				
		Inhibit override					X			
		System flag						X		
		RESERVED--always 0							0	
		Enable output-buffer-full interrupt								X
AAH	Self test	55H placed in output buffer if successful								
ABH	Interface test	Returns code in output buffer as follows:								
		No error detected	0	0	0	0	0	0	0	0
		Keyboard clock line is stuck low	0	0	0	0	0	0	0	1
		Keyboard clock line is stuck high	0	0	0	0	0	0	1	0
		Keyboard data line is stuck low	0	0	0	0	0	0	1	1
		Keyboard data line is stuck high	0	0	0	0	0	1	0	0
ACH	Diagnostic dump	Sends 16 bytes of controller's RAM								
ADH	Disable keyboard feature	Sets bit 4 of controller's command byte								
AEH	Enable keyboard interface	Clears bit 4 of controller's command byte								
C0H	Read input port	Reads input port, data put in output buffer								
D0H	Read output port	Reads output port, data put in output buffer								
D1H	Write output port	Next byte placed in controller's output port								
E0H	Read test inputs	T0 and T1 inputs placed in output buffer								
F0-FFH	Pulse output port	Bits 0-3 of command determine bits to pulse								

Source: IBM PC/AT Technical Reference, pages 1-51 to 1-54

7.12. AT KEYBOARD INPUT PORT BIT DEFINITIONS

Bit Numbers 7	6	5	4	3	2	1	0	Function	Allowable Values
X								Keyboard inhibit switch	0=inhibited, 1=not inhibited
	X							Display switch	0=CGA, 1=MDA
		X						Manufacturing jumper status	0=jumper installed, 1=not installed
			X					System RAM	0=512K, 1=256K
				X	X	X	X	RESERVED	

Source: IBM PC/AT Technical Reference, page 1-55

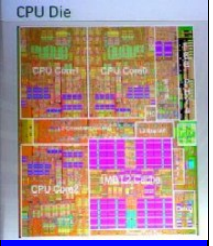
See Also: 7.11. AT Keyboard I/O Command Summary

7.13. AT KEYBOARD OUTPUT PORT BIT DEFINITIONS

Bit Numbers 7	6	5	4	3	2	1	0	Function	Allowable Values
X								Keyboard data output	
	X							Keyboard clock output	
		X						Input buffer empty	0=buffer full, 1=buffer empty
			X					Output buffer full	0=buffer empty, 1=buffer full
				X	X			RESERVED	
						X		Gate A20	
							X	System reset	

Source: IBM PC/AT Technical Reference, page 1-55

See Also: 7.11. AT Keyboard I/O Command Summary



PCMCIA

Table 1. PCMCIA Expansion- Slot Pinout Details

16-Bit PCMCIA Expansion Slot Memory-Only Interface
(68-Pin Dual-In-Line Connector)

Pin	I/O	Signal	Description
1	—	GND	Ground
2	I/O	D3	Data Bit 3
3	I/O	D4	Data Bit 4
4	I/O	D5	Data Bit 5
5	I/O	D6	Data Bit 6
6	I/O	D7	Data Bit 7
7	O	CE1	Card Enable 1
8	I/O	A10	Address Bit 10
9	O	OE	Output Enable
10	I/O	A10	Address Bit 11
11	I/O	A9	Address Bit 9
12	I/O	A8	Address Bit 8
13	I/O	A13	Address Bit 13
14	I/O	A14	Address Bit 14
15	I/O	WE/PGM	Write Enable/Program
16	I/O	RDY/BSY	Card Ready
17	O	Vcc	Power Supply
18	—	Vpp1	Progamming Supply Voltage 1
19	I/O	A16	Address Bit 16
20	I/O	A15	Address Bit 15
21	I/O	A12	Address Bit 12
22	I/O	A7	Address Bit 7
23	I/O	A6	Address Bit 6
24	I/O	A5	Address Bit 5
25	I/O	A4	Address Bit 4
26	I/O	A3	Address Bit 3
27	I/O	A2	Address Bit 2
28	I/O	A1	Address Bit 1
29	I/O	A0	Address Bit 0
30	I/O	D0	Data Bit 0
31	I/O	D1	Data Bit 1
32	I/O	D2	Data Bit 2
33	O	WP	Write Protect
34	—	GND	Ground
35	—	GND	Ground
36	I	D1	Card Detect 1
37	I/O	D11	Data Bit 11
38	I/O	D12	Data Bit 12
39	I/O	D13	Data Bit 13
40	I/O	D14	Data Bit 14
41	I/O	D15	Data Bit 15
42	O	CE2	Card Enable 2
43	O	RFSH	Refresh
44	—	NA	Reserved
45	—	NA	Reserved
46	I/O	A17	Address Bit 17
47	I/O	A18	Address Bit 18
48	I/O	A19	Address Bit 19
49	I/O	A20	Address Bit 20
50	I/O	A21	Address Bit 21
51	O	Vcc	Power Supply
52	O	Vpp2	Progamming Supply Voltage 2
53	I/O	A22	Address Bit 22
54	I/O	A23	Address Bit 23
55	I/O	A24	Address Bit 24
56	I/O	A25	Address Bit 25
57	—	NA	Reserved
58	I	RESET	Reset
59	I	WAIT	Wait
60	—	NA	Reserved
61	O	REG	Register Select
62	I	BVD2	Battery Voltage Detect 2
63	—	BVD1	Battery Voltage Detecr 1
64	I/O	D8	Data Bit 8
65	I/O	D9	Data Bit 9
66	I/O	D10	Data Bit 10
67	I	CD2	Card Detect 2
68	—	GND	Ground

16-Bit PCMCIA Expansion Slot I/O and Memory Interface
(68-Pin Dual-In-Line Connector)

Pin	I/O	Signal	Description
1	—	GND	Ground
2	I/O	D3	Data Bit 3
3	I/O	D4	Data Bit 4
4	I/O	D5	Data Bit 5
5	I/O	D6	Data Bit 6
6	I/O	D7	Data Bit 7
7	O	CE1	Card Enable 1
8	I/O	A10	Address Bit 10
9	O	OE	Output Enable
10	I/O	A10	Address Bit 11
11	I/O	A9	Address Bit 9
12	I/O	A8	Address Bit 8
13	I/O	A13	Address Bit 13
14	I/O	A14	Address Bit 14
15	I/O	WE/PGM	Write Enable/Progam
16	O	IREQ	Interrupt Request
17	O	Vcc	Power Supply
18	O	Vpp1	Programming and Peripheral Supply 1
19	I/O	A16	Address Bit 16
20	I/O	A15	Address Bit 15
21	I/O	A12	Address Bit 12
22	I/O	A7	Address Bit 7
23	I/O	A6	Address Bit 6
24	I/O	A5	Address Bit 5
25	I/O	A4	Address Bit 4
26	I/O	A3	Address Bit 3
27	I/O	A2	Address Bit 2
28	I/O	A1	Address Bit 1
29	I/O	A0	Address Bit 0
30	I/O	D0	Data Bit 0
31	I/O	D1	Data Bit 1
32	I/O	D2	Data Bit 2
33	O	IOIS16	I/O Port is 16 Bits
34	—	GND	Ground
35	—	GND	Ground
36	I	CD1	Card Detect 1
37	I/O	D11	Data Bit 11
38	I/O	D12	Data Bit 12
39	I/O	D13	Data Bit 13
40	I/O	D14	Data Bit 14
41	I/O	D15	Data Bit 15
42	O	CE2	Card Enable 2
43	O	RFSH	Refresh
44	I/O	IORD	I/O Read
45	I/O	IOWR	I/O Write
46	I/O	A17	Address Bit 17
47	I/O	A18	Address Bit 18
48	I/O	A19	Address Bit 19
49	I/O	A20	Address Bit 20
50	I/O	A21	Address Bit 21
51	O	Vcc	Power Supply
52	O	Vpp2	Programming and Peripheral Supply 2
53	I/O	A22	Address Bit 22
54	I/O	A23	Address Bit 23
55	I/O	A24	Address Bit 24
56	I/O	A25	Address Bit 25
57	—	NA	Reserved
58	I	RESET	Reset
59	I	WAIT	Wait
60	O	INPACK	Input Port Acknowlede
61	I/O	REG	Register Select & I/O Eenable
62	O	SPKR	Audio Digital Waveform
63	I	STSCHG	Card Status Changed
64	I/O	D8	Data Bit 8
65	I/O	D9	Data Bit 9
66	I/O	D10	Data Bit 10
67	I	CD2	Card Detect 2
68	—	GND	Ground

PCMCIA Interface

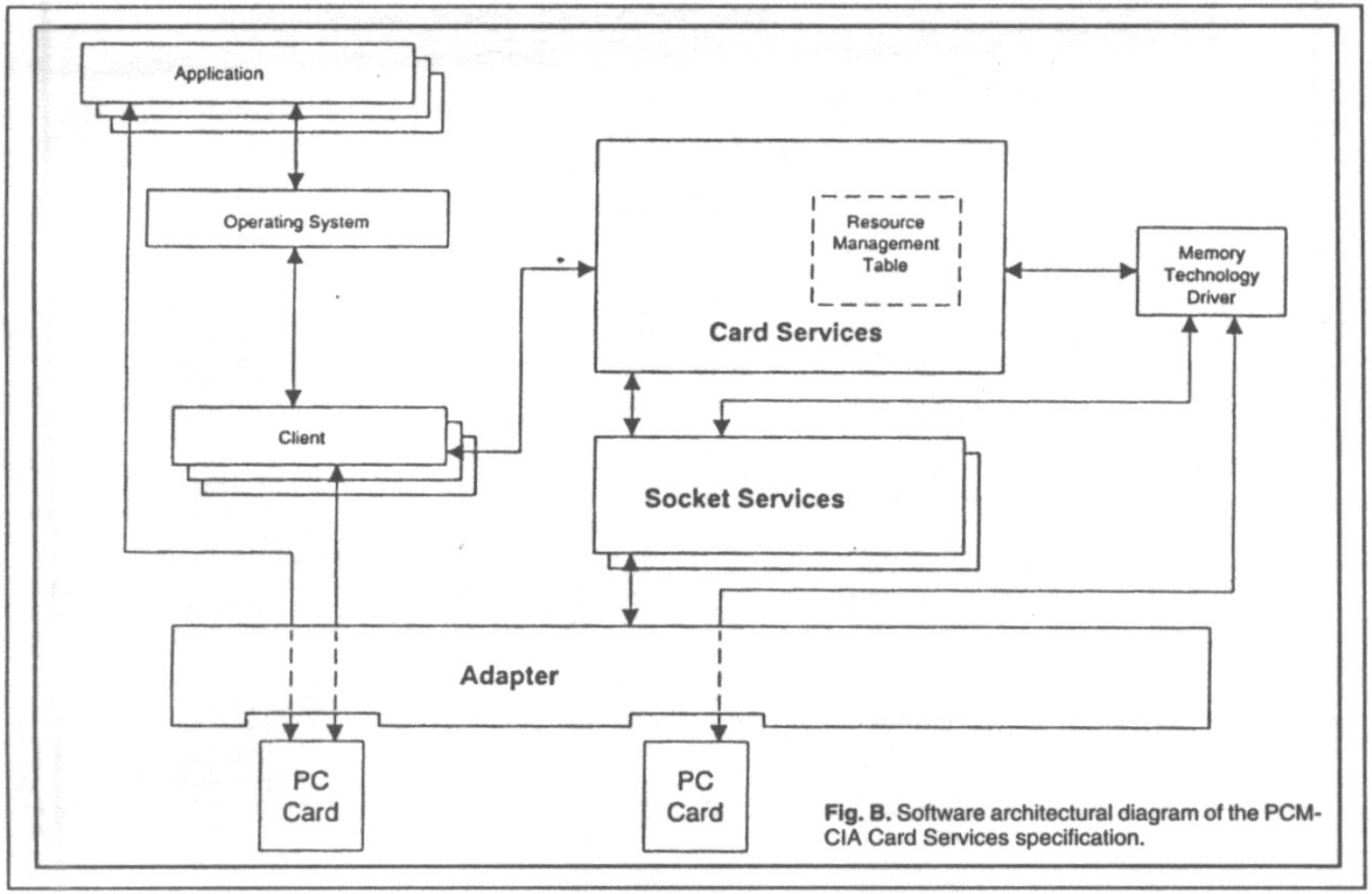


Fig. B. Software architectural diagram of the PCM-CIA Card Services specification.

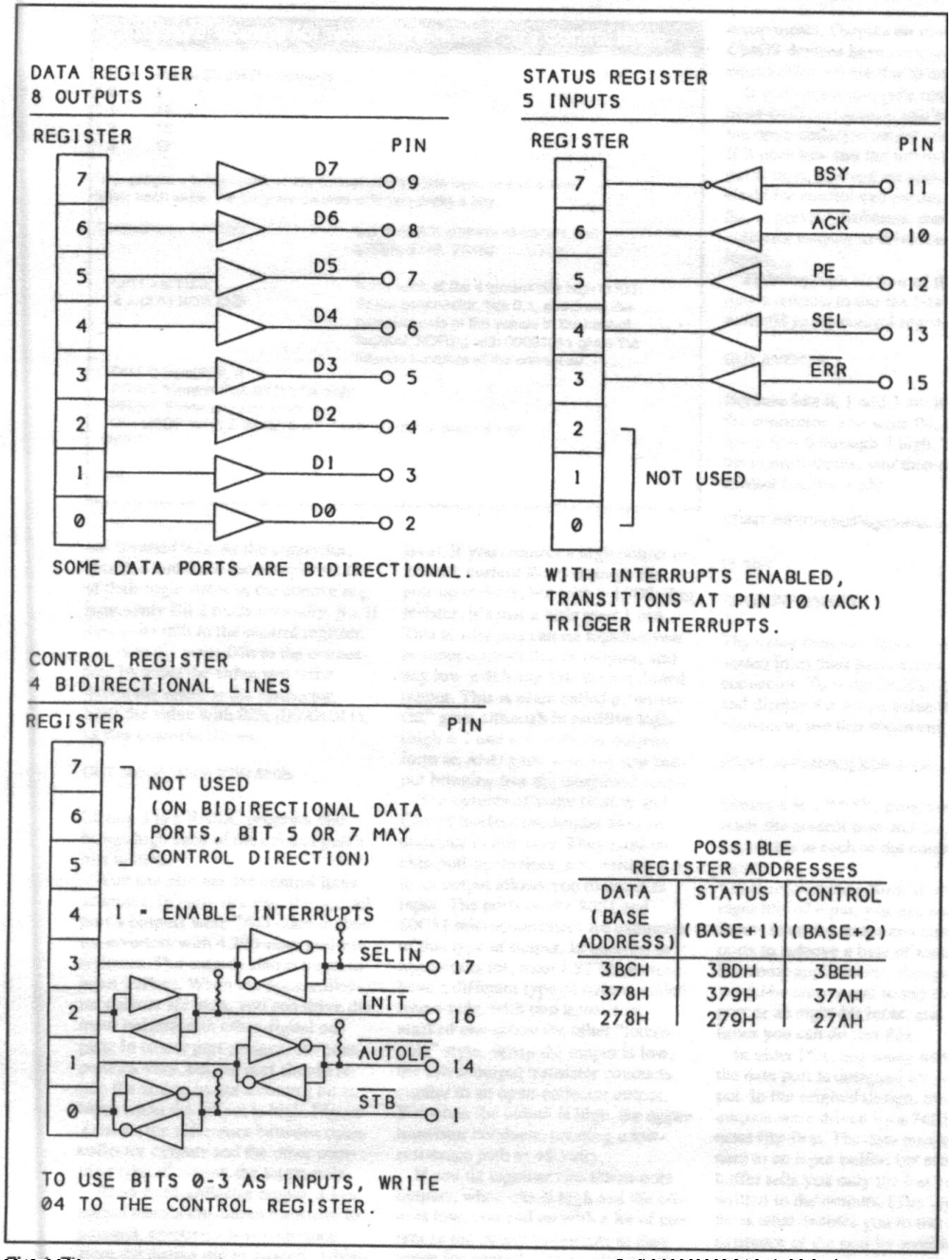
PC's Parallel Port

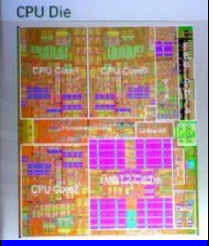
Table 1. Signals and Their Functions on PC Parallel (Printer) Port

D-Shell Connector Pin	Signal	Function	I/O at D-Shell Connector	Register	Register Bit	Inverted at Register	Centronics Connector Pin
1	–STB	Strobe D0 Thru D7	I/O	Control	0	Y	1
2	D0	Data Bit 0	O*	Data	0	N	2
3	D1	Data Bit 1	O*	Data	1	N	3
4	D2	Data Bit 2	O*	Data	2	N	4
5	D3	Data Bit 3	O*	Data	3	N	5
6	D4	Data Bit 4	O*	Data	4	N	6
7	D5	Data Bit 5	O*	Data	5	N	7
8	D6	Data Bit 6	O*	Data	6	N	8
9	D7	Data Bit 7	O*	Data	7	N	9
10	–ACK	Acknowledge (Triggers interrupt)	I	Status	6	N	10
11	BSY	Printer Busy	I	Status	7	Y	11
12	PE	Paper End (Out of Paper)	I	Status	5	N	12
13	SEL	Printer Selected (On-Line)	I	Status	4	N	13
14	–AUTOLF	Automatic Line Feed After CR	I/O	Control	1	Y	14
15	–ERR	Error	I	Status	3	N	32
16	–INIT	Initialize Printer	I/O	Control	2	N	31
17	–SELIN	Select Printer (Plate On-Line)	I/O	Control	3	Y	36
18-25	GND	Ground	I	—	—	N	16, 19-30,33
—	NC	No Connection at PC	—	—	—	—	15,17,18,34,35

*Some data ports are bidirectional (I/O).

Parallel Port Circuit Diagram





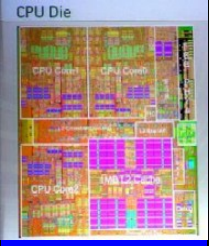
Parallel Port Algorithms

Listing 1. Routine For Writing to Parallel Port's Data Lines

```
' Bit #    Pin on 25-pin D-connector
' 0        2
' 1        3
' 2        4
' 3        5
' 4        6
' 5        7
' 6        8
' 7        9

'The program brings each of the data lines high, one at a time.
'After each write, the program pauses until you press a key.

DataPort = &H3BC:                    'set to match address of current port
                                     '(3BCh, 378h, 278h)

 FOR I = 0 TO 7                      'bring each of the 8 data bits high in turn
  OUT DataPort, 2 ^ I
  PRINT "Data Port Bit "; I; " is high"
  PRINT "Press any key to continue..."
  DO: LOOP WHILE INKEY$ = ""         'wait for key press
 NEXT I

END
```

Listing 2. Routine For Reading Parallel Port's Status Lines

```
' Bit #    Pin on 25-pin D-connector
' 3        15
' 4        13
' 5        12
' 6        10
' 7        11

'The program reads the status register and displays the result.

StatusPort = &H3BD                   'set to match address of current port
                                     '(3BDh, 379h, 279h)

 A = INP(StatusPort):                'read status register
 A = A XOR &H80:                     'register contains complement of bit 7
                                     'XORing with 10000000 gives the value
                                     'at the connector
A = A AND &HF8:                      'ignore bits 0-2

 PRINT "Status Port = "; HEX$(A)
                                     'display hex value of bits 3-7 at connector
 FOR I = 3 TO 7                      'display each bit individually

   PRINT "Status Port Bit "; I; " = "; (A AND 2 ^ I) / 2 ^ I
NEXT I

END
```



COMP 273
Introduction to Computer Systems

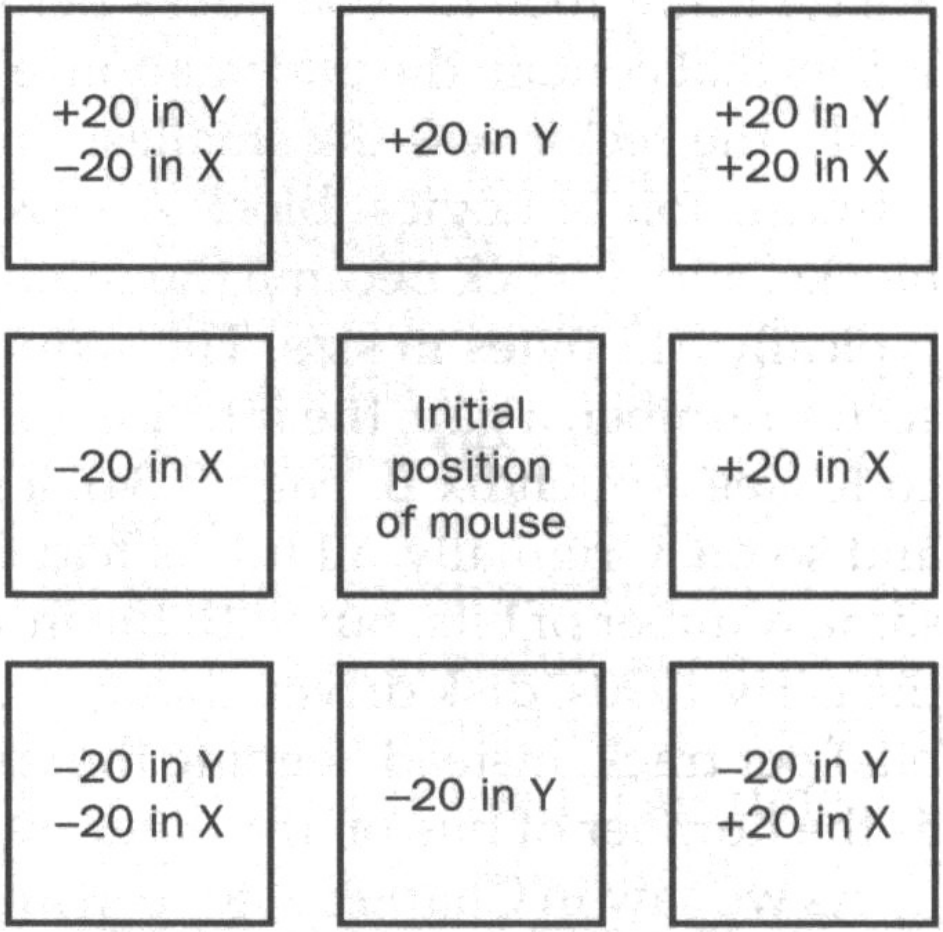
Listing 3. Routine For Writing to Parallel Port's Control Lines

```
' Bit #    Pin on 25-pin D-connector
' 0        1
' 1        14
' 2        16
' 3        17

'The program brings each of the control port's 4 bits high, one at a time.
'After each write, the program pauses until you press a key.
'
ControlPort = &H3BE:                    'set to match address of current port
                                        '(3BEh, 37Ah, 27Ah)

 FOR I = 0 TO 3                         'bring each of the 4 control bits high in turn
   A = (2 ^ I XOR &HB                   'At the connnector, bits 0,1, and 3 are the
                                        'complements of the values in the control
                                        'register. XORing with 00001011 gives the
                                        'intended outputs at the connector.
   OUT ControlPort, A
   PRINT "Control Port Bit "; I; "is high"
   PRINT "Press any key to continue..."
   DO: LOOP WHILE INKEY$ = ""  'wait for user to press a key
 NEXT I

END
```

Listing 4. Routine For Reading Parallel Port's Control Lines

```
' Bit #    Pin on 25-pin D-connector
' 0        1
' 1        14
' 2        16
' 3        17

'The program reads the control register and displays the result.

ControlPort = &H3BE:                    'set to match address of current port
                                        '(3BEh, 37Ah, 27Ah)

OUT ControlPort, 4 AND &HF:             'bring bits 0-3 high to allow use as inputs

A = INP(ControlPort):                   'read control register
A = A XOR &HB:                          'register contains complements of bits 0,1,3
                                        'exclusive ORing with 00001011 gives
                                        'the values at the connector
 A = A AND &HF:                         'ignore bits 4-7
 PRINT "Control Port = "; HEX$(A)

                                        'display hex value of bits 0-3 at connector
 FOR I = 0 TO 3                         'display each bit individually
   PRINT "Bit ", I, " = ", (A AND 2 ^ I) / 2 ^ I
 NEXT I

END
```

The Mouse



- X register
- Y register
- Wheel reg.
- Status
  - Buttons
  - Wheel

Programming:
- polling intensive
- graphical output intensive

Improvable?
Interrupt driven? DMA?