i386 Interupt Handling

Interupts and the IDT

The i386 processor treats hardware interrupts in the same manner as Exceptions and Software interupts. Interrupt service routines are registerd with the IDT and then are vectored accordingly base on the 8259A Programmable Interrupt Controller.

The 8259A Programmable Interrupt Controller

The 8259A is had 8 inputs, 1 interupt line, and a way for the proccessor to query it. Each of the 8 lines are hooked up to a specific IRQ (0-7). When that IRQ fires the 8259A remembers which line it came in on and reaises it's interupt line which is connected to the proccessor. Then, by *magic*, the processor queries the 8259A, figures out which IRQ fired and vecorts the correct spot in the IDT.

Some time ago, after the IBM XT, a brilliant engineer decided that 8 IRQs was not enough. Because at that time cards could not share interupts well and they needed to maintain compatablilty with the XT, it was decided to just chain annother 8259A off of the original one. These days PCs have both a master and a slave ineterrupt controller.

Programming the 8259A

Each 8259A has two io ports assigned to each of them. For simplicity we'll call them port a and port b. The are assigned as follows:

	port a	port b
Master 8259A	0x20	0x21
Slave 8259A	0xA0	0xA1

The 8259A is configured throught a series of 4 control words. To configure it send the first word (ICW1) to port a and the rest are sent to port b.

ICW1

7	6	5	4	3	2	1	0
0	0	0	1	Trig	0	M/S	ICW4

Trig	0 = Edge triggered	1 = Level Triggered
M/S	0 = Master/Slave configuration	1 = Master only
ICW4	0 = No ICW4	1 = ICW4 will be send

Sane master value: 00010001b (0x11)
Sane slave value: 00010001b (0x11)

ICW2

7	6	5	4	3	2	1	0
Off 7	Off 6	Off 5	Off 4	Off 3	0	0	0

Off 7 .. Off 3 Offeset into the IDT for interupt service routines
The last 3 bits are always 0 because it needs to be 3 bit alligned

Sane master value: 00100000b (0x20)
Sane slave value: 00101000b (0x28)

ICW3 (master)

7	6	5	4	3	2	1	0
S7	S6	S5	S4	S3	S2	S1	S0

S7..S0 0 = IR Line is connected to peripheral device
1 = IR Line is connected to Slave 8259A

Sane master value: 00000100b (0x04)

ICW3 (slave)

7	6	5	4	3	2	1	0
0	0	0	0	0	ID2	ID1	ID0

ID2..ID0 IRQ on master this slave is connected to

Sane slave value: 00000010b (0x02)

ICW4 (optional)

7	6	5	4	3	2	1	0
0	0	0	SFNM	BUF	M/S	AEOI	Mode

SFNUM	0 = No Special Fully Nested Mode	1 = Special Fully Nested Mode
BUF	0 = No Buffered Mode	1 = Buffered Mode
M/S	0 = Slave PIC	1 = Master PIC
AEOI	0 = Manual EOI	1 = Automatic EOI
Mode	0 = MCS-80/85 Mode	1 = 8086/88 Mode

Sane master value: 00000101b (0x05)
Sane slave value: 00000001b (0x01)

Initalization

To initalize the 8259A just send the 4 ICWs to both the slave and master like so:

```
/* ICW1 */
outb( 0x11, 0x20 ); /* Master port A */
outb( 0x11, 0xA0 ); /* Slave port A */
```

```
/* ICW2 */
outb( 0x20, 0x21 ); /* Master offset of 0x20 in the IDT */
outb( 0x28, 0xA1 ); /* Master offset of 0x28 in the IDT */

/* ICW3 */
outb( 0x04, 0x21 ); /* Slaves attached to IR line 2 */
outb( 0x02, 0xA1 ); /* This slave in IR line 2 of master */

/* ICW4 */
outb( 0x05, 0x21 ); /* Set as master */
outb( 0x01, 0xA1 ); /* Set as slave */
```

Interrupt Mask

The 8259A has the abiltiy to mask off interrupts you tell it to. This is done by writting an 8 bit value corresponding to the mask to port b while the PIC is in normal opperation mode. The PIC will only listen to interrupt which have a 0 in the corresponding bit in its mask.

example:

```
/* Only listen to irqs 0, 1, and 2 */
outb( 0xf8, 0x21 ); /* master PIC */
outb( 0xff, 0xA1 ); /* slave PIC */
```

EOI

When the 8259A is in Manual EOI mode (ICW4[1]==0) it will wait for an EOI (End Of Interrput) acknowlegment to be written to it before it will send any more. To send the EOI simply write 0x20 to prot a of the PIC.

example:

```
/* Send EOI to both master and slave */
outb( 0x20, 0x20 ); /* master PIC */
outb( 0x20, 0xA0 ); /* slave PIC */
```
