2009-12-22 Qemu monitor command that triggers IRQ

Plan

- See how PC platform is initialized
- Drag 8259 and IOAPIC interfaces to the reachability of monitor
- Add monitor commands to trigger IRQ through 8259 or through IOAPIC
- See how it works

Worklog

- look how 'nmi' command is wired into monitor interface (do_inject_nmi), make similar do_inject_irq:

```
diff -burp qemu-0.10.5-orig/monitor.c qemu-0.10.5/monitor.c
--- qemu-0.10.5-orig/monitor.c  2009-05-21 00:46:59.000000000 +0400
+++ qemu-0.10.5/monitor.c   2009-12-24 00:59:20.000000000 +0300
@@ -1550,6 +1562,8 @@ static const term_cmd_t term_cmds[] = {
 #if defined(TARGET_I386)
     { "nmi", "i", do_inject_nmi,
       "cpu", "inject an NMI on the given CPU", },
+    { "irq", "ii", do_inject_irq,
+      "cpu irq", "inject given IRQ on the given CPU", },
 #endif
     { "migrate", "-ds", do_migrate,
       "[-d] uri", "migrate to URI (using -d to not wait for completion)" },
```

- see that there's no simple way to issue APIC interrupt directly (LAPIC and IOAPIC structs are private)
- see how 8259 is initialized in hw/pc.c
- see there how IRQs are dispatched: through qemu_irq structures
- see how generic IRQ is invoked: hw/irq.h (qemu_irq_*)
- put i8259 field into CPUX86State:

```
diff -burp qemu-0.10.5-orig/hw/pc.c qemu-0.10.5/hw/pc.c
--- qemu-0.10.5-orig/hw/pc.c    2009-05-21 00:46:59.000000000 +0400
+++ qemu-0.10.5/hw/pc.c 2009-12-24 00:45:41.000000000 +0300
@@ -963,6 +963,7 @@ vga_bios_error:

     cpu_irq = qemu_allocate_irqs(pic_irq_request, NULL, 1);
     i8259 = i8259_init(cpu_irq[0]);
+    env->i8259 = i8259;
     ferr_irq = i8259[13];

     if (pci_enabled) {
```

```
diff -burp qemu-0.10.5-orig/target-i386/cpu.h qemu-0.10.5/target-i386/cpu.h
--- qemu-0.10.5-orig/target-i386/cpu.h  2009-05-21 00:47:00.000000000 +0400
+++ qemu-0.10.5/target-i386/cpu.h   2009-12-24 00:55:32.000000000 +0300
@@ -670,6 +670,7 @@ typedef struct CPUX86State {
     /* in order to simplify APIC support, we leave this pointer to the
        user */
     struct APICState *apic_state;
+    void *i8259;
 } CPUX86State;

 CPUX86State *cpu_x86_init(const char *cpu_model);
```

- use it in handler:

```
diff -burp qemu-0.10.5-orig/monitor.c qemu-0.10.5/monitor.c
--- qemu-0.10.5-orig/monitor.c  2009-05-21 00:46:59.000000000 +0400
+++ qemu-0.10.5/monitor.c   2009-12-24 00:59:20.000000000 +0300
@@ -1441,6 +1441,18 @@ static void do_inject_nmi(int cpu_index)
             break;
         }
 }
+
+static void do_inject_irq(int cpu_index, int irq)
+{
+    CPUState *env;
+
+    for (env = first_cpu; env != NULL; env = env->next_cpu)
+        if (env->cpu_index == cpu_index) {
+            if (irq >= 0 && irq < 16)
+                qemu_irq_pulse(((qemu_irq*)env->i8259)[irq]);
+            break;
+        }
+}
 #endif

 static void do_info_status(void)
```

- load 2009-12-21 bzImage into this qemu (qemu -kernel bzImage -monitor stdio), insmod test.ko test_irq_no=5, type in monitor “irq 0 5”, see it!

Conclusion

- qemu pc platform have self-contained design, hard to expose internal details to some generic debug monitor;
- not clear, how e.g. PCI interrupts are routed to CPUs;
- would be nice to implement PCI device emulation, with full configuration/IO/IRQ features;