Cloud Operating Systems

PIC and PIT

Andreas Kogler, Fabian Rauscher, Daniel Gruss

2022-03-29

Outlook

Outlook

Outlook

Outlook

- Classic hardware interactions

Outlook

- Classic hardware interactions
- What is the PIC?

Outlook

- Classic hardware interactions
- What is the PIC?
- What is the PIT?

Outlook

- Classic hardware interactions
- What is the PIC?
- What is the PIT?
- Why should *your* **HV** care?

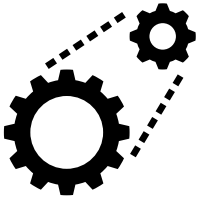
Outlook

- Classic hardware interactions
- What is the PIC?
- What is the PIT?
- Why should *your* **HV** care?
- How to virtualize the PIC and PIT

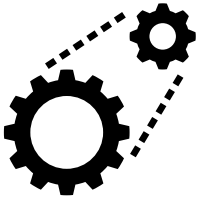
Hardware Interactions?

Hardware Interactions?

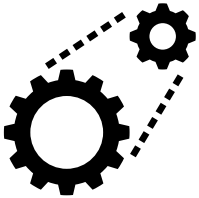
Hardware Interactions?



Hardware Interactions?

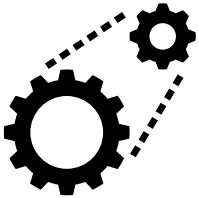


Hardware Interactions?



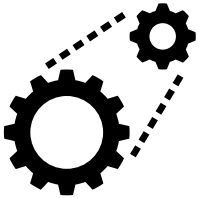
- draw a character on the console?

Hardware Interactions?



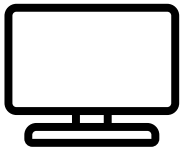
- draw a character on the console?
- receive a keyboard press?

Hardware Interactions?

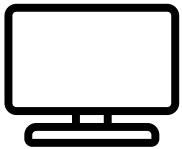


- draw a character on the console?
- receive a keyboard press?
- receive a regular *heartbeat*?

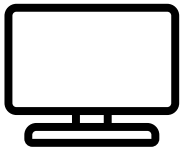
Console Output



Console Output

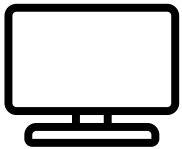


Console Output

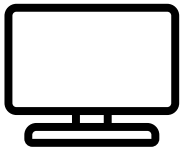


- **frame buffer** at physical address `0xB8000`

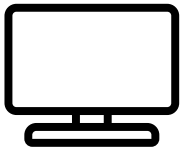
Console Output



- **frame buffer** at physical address `0xB8000`
- how to pass the buffer to the guest?

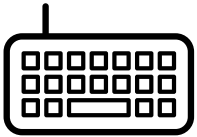


- **frame buffer** at physical address `0xB8000`
- how to pass the buffer to the guest?
  - share it via the EPT?

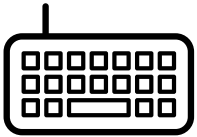


- **frame buffer** at physical address `0xB8000`
- how to pass the buffer to the guest?
  - share it via the EPT?
  - copy it, but when?

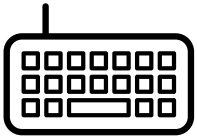
Keyboard



Keyboard

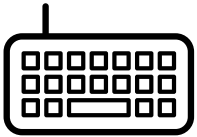


Keyboard



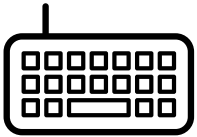
- implemented via interrupts and ports

Keyboard



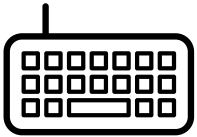
- implemented via interrupts and ports
- check if data is available on port `0x64`

Keyboard



- implemented via interrupts and ports
- check if data is available on port `0x64`
- get the current *scancode* key value from port `0x60`

Keyboard



- implemented via interrupts and ports
- check if data is available on port `0x64`
- get the current *scancode* key value from port `0x60`
- but how to share with a guest?

- How and when to preempt threads?

Heartbeat

- How and when to preempt threads?
- Timer waking up the scheduler

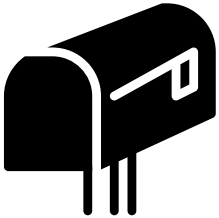
Heartbeat

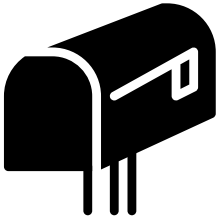
- How and when to preempt threads?
- Timer waking up the scheduler
- The scheduler decides what to do

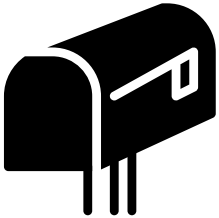
Heartbeat

- How and when to preempt threads?
- Timer waking up the scheduler
- The scheduler decides what to do
- But how to pass this heartbeat to the guest?

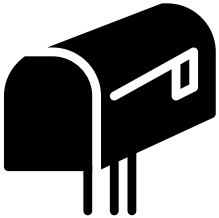
Programmable Interrupt Controller



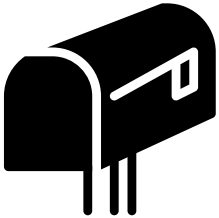




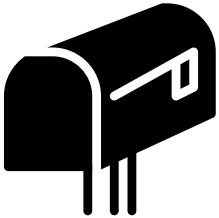
- The PIC makes x86 interrupt driven



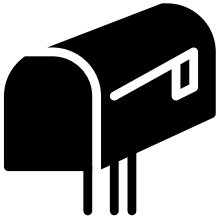
- The PIC makes x86 interrupt driven
- Manages *hardware* interrupts



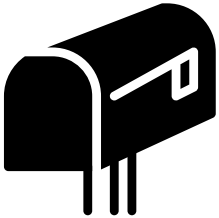
- The PIC makes x86 interrupt driven
- Manages *hardware* interrupts
- Wires *hardware* interrupts to *system* interrupts



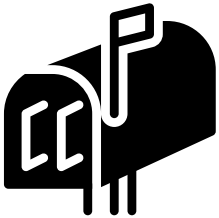
- The PIC makes x86 interrupt driven
- Manages *hardware* interrupts
- Wires *hardware* interrupts to *system* interrupts
- More modern systems use the successor: **APIC**

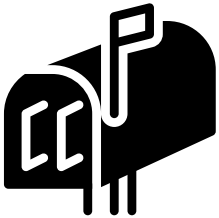


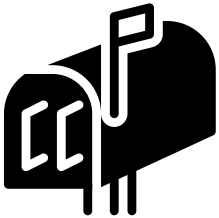
- The PIC makes x86 interrupt driven
- Manages *hardware* interrupts
- Wires *hardware* interrupts to *system* interrupts
- More modern systems use the successor: **APIC**
- SWEB uses the PIC



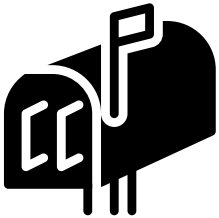
- The PIC makes x86 interrupt driven
- Manages *hardware* interrupts
- Wires *hardware* interrupts to *system* interrupts
- More modern systems use the successor: **APIC**
- SWEB uses the PIC
- Checkout: `https://wiki.osdev.org/8259_PIC`





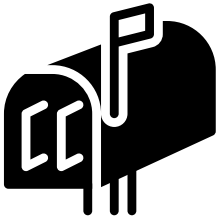


- Features 16 interrupts

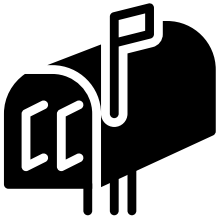


- Features 16 interrupts
- Notation: **Port[P] = D**

PIC Interactions

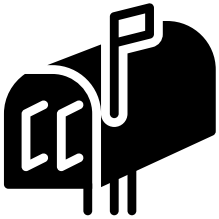


- Features 16 interrupts
- Notation: **Port[P] = D**
- distributed to *Parent* (**P=0x20**) and *Child* (**P=0xA0**) PIC



- Features 16 interrupts
- Notation: **Port[P] = D**
- distributed to *Parent* (**P=0x20**) and *Child* (**P=0xA0**) PIC
- Data (**P+=1**) and command (**P+=0**) **ports**

PIC Interactions



- Features 16 interrupts
- Notation: **Port[P] = D**
- distributed to *Parent* (**P=0x20**) and *Child* (**P=0xA0**) PIC
- Data (**P+=1**) and command (**P+=0**) **ports**
- Most known command: End-Of-Interrupt (EOI) **D=0x20**

PIC Setup

- SWEB only uses the PIC in *one* configuration

- SWEB only uses the PIC in *one* configuration
- Initial write of the config ports with **D=0x11**

- SWEB only uses the PIC in *one* configuration
- Initial write of the config ports with **D=0x11**
- Followed by three writes to the data *port*

- SWEB only uses the PIC in *one* configuration
- Initial write of the config ports with **D=0x11**
- Followed by three writes to the data *port*
  - First, the offset to translate hw to sw IRQ numbers

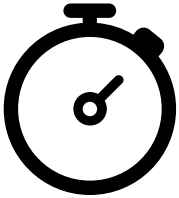
- SWEB only uses the PIC in *one* configuration
- Initial write of the config ports with **D=0x11**
- Followed by three writes to the data *port*
  - First, the offset to translate hw to sw IRQ numbers
- Further writes only change the interrupt **mask**

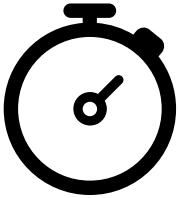
- SWEB only uses the PIC in *one* configuration
- Initial write of the config ports with **D=0x11**
- Followed by three writes to the data *port*
  - First, the offset to translate hw to sw IRQ numbers
- Further writes only change the interrupt **mask**
- Important hardware IRQs:

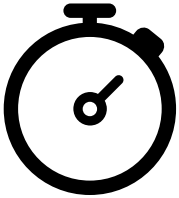
- SWEB only uses the PIC in *one* configuration
- Initial write of the config ports with **D=0x11**
- Followed by three writes to the data *port*
  - First, the offset to translate hw to sw IRQ numbers
- Further writes only change the interrupt **mask**
- Important hardware IRQs:
  - **0**: Timer interrupt

- SWEB only uses the PIC in *one* configuration
- Initial write of the config ports with **D=0x11**
- Followed by three writes to the data *port*
  - First, the offset to translate hw to sw IRQ numbers
- Further writes only change the interrupt **mask**
- Important hardware IRQs:
  - **0**: Timer interrupt
  - **1**: Keyboard interrupt

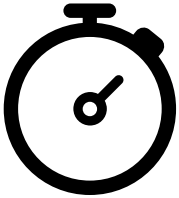
Programmable Interval Timer



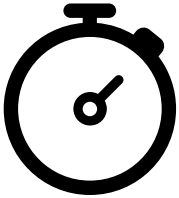




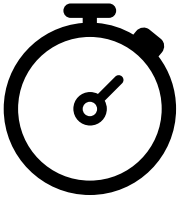
- Basically a programmable oscillator, with selectable:



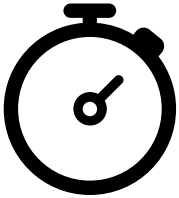
- Basically a programmable oscillator, with selectable:
  - frequency



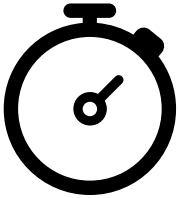
- Basically a programmable oscillator, with selectable:
  - frequency
  - divisor



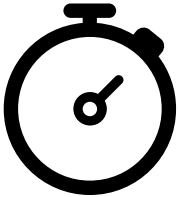
- Basically a programmable oscillator, with selectable:
  - frequency
  - divisor
  - modes



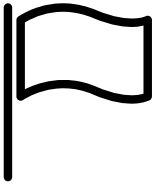
- Basically a programmable oscillator, with selectable:
  - frequency
  - divisor
  - modes
- SWEB uses channel 0 (**P=0x40**)

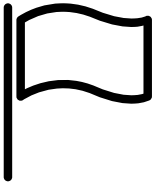


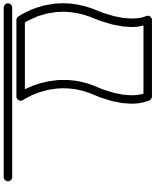
- Basically a programmable oscillator, with selectable:
  - frequency
  - divisor
  - modes
- SWEB uses channel 0 (**P=0x40**)
- Including command port (**P=0x43**) with (**D=0x36**)



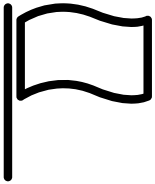
- Basically a programmable oscillator, with selectable:
  - frequency
  - divisor
  - modes
- SWEB uses channel 0 (**P=0x40**)
- Including command port (**P=0x43**) with (**D=0x36**)
- Checkout:
  `https://wiki.osdev.org/Programmable_Interval_Timer`



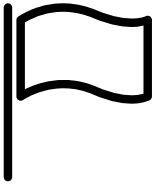




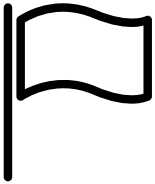
- Needed by the scheduler to change threads



- Needed by the scheduler to change threads
- Channel 0 counts with 18.2065 Hz



- Needed by the scheduler to change threads
- Channel 0 counts with 18.2065 Hz
- Giving an **interrupt** every $\approx$ 54 ms



- Needed by the scheduler to change threads
- Channel 0 counts with 18.2065 Hz
- Giving an **interrupt** every $\approx$ 54 ms
- If enabled in the **PIC**

How to virtualize PIC and PIT?

- Examine VMExits based on IO port operations

Virtualized PIC and PIT

- Examine VMExits based on IO port operations
- Emulate PIC and PIT for their ports

Virtualized PIC and PIT

- Examine VMExits based on IO port operations
- Emulate PIC and PIT for their ports
- You don't need to implement all features

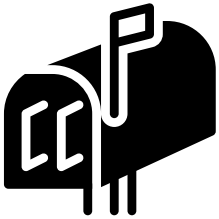
Virtualized PIC and PIT

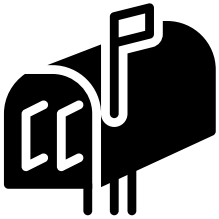
- Examine VMExits based on IO port operations
- Emulate PIC and PIT for their ports
- You don't need to implement all features
  - only what SWEB needs

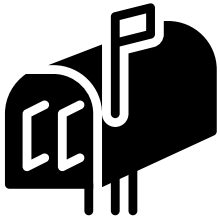
Virtualized PIC and PIT

- Examine VMExits based on IO port operations
- Emulate PIC and PIT for their ports
- You don't need to implement all features
  - only what SWEB needs
  - check SWEB's code if unsure

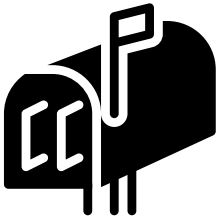
Virtualized PIC



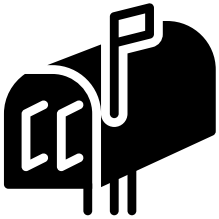




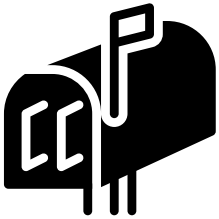
- Emulate the ports for the configuration:



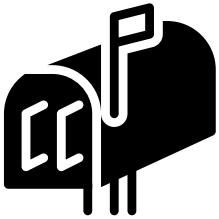
- Emulate the ports for the configuration:
  - the offset mapping HW to SW interrupt vectors



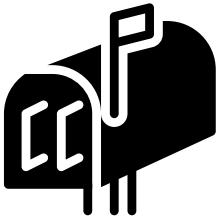
- Emulate the ports for the configuration:
  - the offset mapping HW to SW interrupt vectors
  - the interrupt mask enabling and disabling interrupts



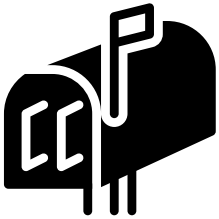
- Emulate the ports for the configuration:
  - the offset mapping HW to SW interrupt vectors
  - the interrupt mask enabling and disabling interrupts
  - ...



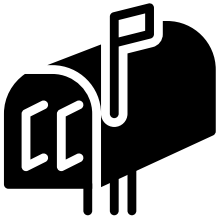
- Emulate the ports for the configuration:
  - the offset mapping HW to SW interrupt vectors
  - the interrupt mask enabling and disabling interrupts
  - ...
- Forward only enabled interrupts to the guest



- Emulate the ports for the configuration:
  - the offset mapping HW to SW interrupt vectors
  - the interrupt mask enabling and disabling interrupts
  - ...
- Forward only enabled interrupts to the guest
- Manage pending interrupts in the HV

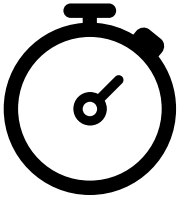


- Emulate the ports for the configuration:
  - the offset mapping HW to SW interrupt vectors
  - the interrupt mask enabling and disabling interrupts
  - ...
- Forward only enabled interrupts to the guest
- Manage pending interrupts in the HV
- Think about interrupts during interrupts (hint: EOI)

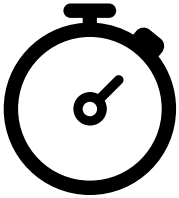


- Emulate the ports for the configuration:
  - the offset mapping HW to SW interrupt vectors
  - the interrupt mask enabling and disabling interrupts
  - ...
- Forward only enabled interrupts to the guest
- Manage pending interrupts in the HV
- Think about interrupts during interrupts (hint: EOI)
- Goal: Be able to forward keyboard scancodes to the guest

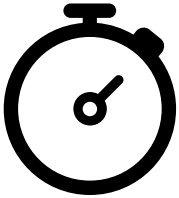
Virtualized PIT



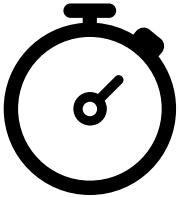
Virtualized PIT



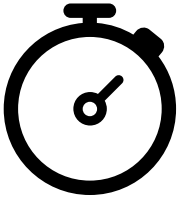
Virtualized PIT



- Emulate the ports similar to the PIC



- Emulate the ports similar to the PIC
- Generate timer interrupts if enabled in the PIC



- Emulate the ports similar to the PIC
- Generate timer interrupts if enabled in the PIC
- Goal: Generate timer interrupts for the guest's scheduler

- Register: CPU_BASED_VM_EXEC_CONTROL

- Register: CPU_BASED_VM_EXEC_CONTROL
  - Bit: CPU_BASED_VIRTUAL_INTR_PENDING

- Register: CPU_BASED_VM_EXEC_CONTROL
  - Bit: CPU_BASED_VIRTUAL_INTR_PENDING
- Register: VM_ENTRY_INTR_INFO_FIELD

Virtualized PIC and PIT - VMX

- Register: CPU_BASED_VM_EXEC_CONTROL
  - Bit: CPU_BASED_VIRTUAL_INTR_PENDING
- Register: VM_ENTRY_INTR_INFO_FIELD
- Register: VMX_PREEMPTION_TIMER_VALUE

- Register: CPU_BASED_VM_EXEC_CONTROL
  - Bit: CPU_BASED_VIRTUAL_INTR_PENDING
- Register: VM_ENTRY_INTR_INFO_FIELD
- Register: VMX_PREEMPTION_TIMER_VALUE
- ExitReasons: INT_WINDOW, IO_INST and NMI

Questions?