Virtualization
Pre-lab Questions — #1

Point total: 7
All homework assignments are weighted equally in the final grade. Point values are unique to each lab assignment.

Note: For all problems which ask you to explain your reasoning or show your work, you do not need to show every step of each calculation, but the answer should include an explanation *written with words* of what you did. Even when work is not required to be shown, it's a good idea to include anyways so that you can earn partial credit.

1. Question [1 point] What does the `cpuid` assembly instruction do in this function?

Solution:

The `cpuid` assembly instruction provides details about the processor. Specifically, when `cpuid` is called with an initial input value of 0 in the EAX register, it returns the highest function number the CPU supports and the vendor ID string. This 12-character ASCII string identifies the manufacturer and is distributed across the EBX, EDX, and ECX registers.

2. Question [1 point] How is the function providing values back to you to use?

Solution:

The `cpuid` function retrieves information directly into the CPU registers EAX, EBX, ECX, and EDX using inline assembly. After the instruction is executed, it utilizes the provided pointers (`eaxp`, `ebxp`, `ecxp`, `edxp`) to relay these values to specific memory locations. This method allows the invoking function, in this instance `vmx_check_support`, to access the results of the `cpuid` instruction by referencing the variables 'eax', 'ebx', 'ecx', and 'edx'.

3. Question [1 point] What are the results of `eax`, `ecx`, and `ebx` values in hexadecimal?

Solution:

The hexadecimal results for the registers are:

- `EAX (rax): 0x8003fffe60`
- `ECX (rcx): 0x444d4163`
- `EBX (rbx): 0x0`

4. Question [1 point] Now examine the values of these variables as strings. Hint: look at the values in hexadecimal and translate them to strings, in the order `ebx`, `edx`, `ecx`. What do you observe? The Wikipedia page for the cpuid instruction may help you interpret this output.

Solution:

1. Given the register values after the `cpuid` call for an AMD processor:
   - `EBX (rbx): 0x68747541` → Translates to ASCII characters ”Auth”
   - `EDX (rdx): 0x69746e65` → Translates to ASCII characters ”entic”
   - `ECX (rcx): 0x444d4163` → Translates to ASCII characters ”cAMD”

   Collating them in the order of EBX, EDX, and ECX, we derive the vendor string: ”AuthenticAMD”. This string is an identifier for AMD processors.

5. Question [1 point] There is a reference in each `Env` struct for another struct called `VmxGuestInfo`. What kind of information does this struct hold?

Solution:

`phys_sz` An integer (64-bit) that possibly represents the physical size of a memory or storage resource.

`vmcs` A pointer to the Virtual Machine Control Structure (VMCS). The VMCS is a data structure in memory that the processor reads and writes to directly. It contains state saved between VMX transitions and controls VM execution.

`exception_bmap` A 32-bit bitmap, likely denoting which exceptions are tracked or handled. When a particular exception occurs, and its corresponding bit is set in this bitmap, control is transferred to the hypervisor.

`io_bmap_a`, `io_bmap_b` Pointers to I/O bitmaps. These bitmaps are used to restrict I/O port access. When a guest tries to access an I/O port that is set in the bitmap, control is passed to the hypervisor. The two different I/O bitmaps might correspond to different sets or ranges of I/O ports.

`msr_count` An integer indicating the count or number of Model-Specific Registers (MSRs) being tracked or managed.

`msr_host_area`, `msr_guest_area` Pointers to areas in memory related to MSRs. These likely represent locations where host and guest MSRs are stored or managed. MSRs are CPU-specific registers that vary in purpose and definition between different types of processors.

`vcpunum` An integer that probably signifies the number or identifier of the virtual CPU (vCPU) associated with this particular guest info.

In essence, the `VmxGuestInfo` struct encompasses information necessary for managing and tracking the state and resources of a virtual machine guest under the VMX environment. It provides mechanisms for exception handling, I/O port access control, and management of MSRs among other tasks.

6. Question [1 point] From this Intel guide, find out what the vmcs pointer in this struct stands for, and what it purpose it serves.

Solution: The VMCS is central to Intel's VMX virtualization. It holds state areas for the host and the guest, ensuring smooth transitions between the Virtual Machine Monitor (VMM) and the virtual machine. It also dictates how the virtual machine behaves during execution. The `vmcs` pointer in `VmxGuestInfo` likely tracks the VMCS location for a specific guest, ensuring proper virtual machine management and control.

7. Question [**1 point**] What assembly instruction initializes the `vmcs` pointer? In other words, how do we change the `vmcs` pointer?

Solution: Your answer here.