VMS CPUID feature check

OpenVMS x86-64 requires that the CPU supports certain features that are not present in all x86-64 processors. To help determine whether a given system supports these features or not, and whether therefore the system might be capable of running OpenVMS, a Python script is provided, which can be executed on a Windows, OS X, BSD or Linux system.

Please note that this check is not exhaustive. It only checks those features which can be determined through the use of the CPUID instruction. There are other requirements which can not be checked in this way, so a successful execution of these checks does not guarantee the suitability of the system to run OpenVMS.

Prerequisites

The system to run the checks on must have Python version 2.7 or 3 installed. If you plan to run OpenVMS as a virtual machine guest, the test should be executed on the host system, not on a virtual machine guest.

Running the script

To perform the checks, type the following at a command line, from the directory where the vmscheck.py script is located:

```
python vmscheck.py
```

This should produce output similar to the following:

```
camielvanderhoeven$ python vmscheck.py

OpenVMS 9.x compatibility quick-check

Vendor ID : GenuineIntel
CPU name  : Intel(R) Core(TM) i5-7360U CPU @ 2.30GHz

Necessary for OpenVMS 9.x:
Intel CPU : Yes
SSE 4.1   : Yes
XSAVE     : Yes
TSC       : Yes
APIC      : Yes
MTRR      : Yes

Optional for OpenVMS 9.x:
PCID      : Yes
X2APIC    : Yes
FSGSBASE  : Yes
MD_CLEAR  : Yes
XSAVEOPT  : Yes
```

```
Possible future optionals:
TSX-HLE    : Yes
TSX-RTM    : Yes
UMIP       : --
```

Interpreting the results

```
Vendor ID : GenuineIntel
CPU name  : Intel(R) Core(TM) i5-7360U CPU @ 2.30GHz
```

This first part identifies the processor manufacturer and model name.

```
Necessary for OpenVMS 9.x:
Intel CPU : Yes
SSE 4.1    : Yes
XSAVE      : Yes
TSC        : Yes
APIC       : Yes
MTRR       : Yes
```

This part indicates whether the CPU supports those features that are absolutely necessary for OpenVMS to function (Streaming SIMD Extensions 4.1, XSAVE instructions, Time Stamp Counter, Advanced Programmable Interrupt Controller, and Memory Type Range Registers). If any of these do not say “Yes”, OpenVMS will **not** boot on this machine.

```
Optional for OpenVMS 9.x:
PCID       : Yes
X2APIC     : Yes
FSGSBASE   : Yes
MD_CLEAR   : Yes
XSAVEOPT   : Yes
```

This part checks for optional processor features that OpenVMS 9.x can use. OpenVMS will offer better performance if running on a processor that supports these features, but they are not strictly necessary. These features are Proces Context Identifiers, x2APIC, FS and GS base instructions, invulnerability to MDS, and the XSAVEOPT instruction.

```
Possible future optionals:
TSX-HLE    : Yes
TSX-RTM    : Yes
UMIP       : --
```

The last part identifies support for CPU features that current versions of OpenVMS do not use, but which future versions of OpenVMS might use for better performance. These are two variants of Transactional Synchronization Extensions, and User-Mode Instruction Prevention.

Known issues

OSError: Failed to set RWX

The Python script uses a technique to execute code from the heap. On certain systems, e.g. SELinux, there may be policies to prevent this. Executing the script on these systems may result in an error message similar to:

```
OSError: Failed to set RWX
```

If this occurs, you can try executing the script as root or an administrator, or temporarily disable heap execution prevention, e.g. on SELinux:

```
$ setsebool selinuxuser_execheap on
```

Virtual Machines

Running the script on a virtual machine may provide unreliable results. Some virtual machine hypervisors are aware of which guest OS runs on the virtual machine, and may disable CPU features which that OS does not need.