How CPU topology info is exported via sysfs

CPU topology info is exported via sysfs. Items (attributes) are similar to /proc/cpuinfo output of some architectures. They reside in /sys/devices/system/cpu/cpuX/topology/. Please refer to the ABI file: Documentation/ABI/stable/sysfs-devices-system-cpu.

Architecture-neutral, drivers/base/topology.c, exports these attributes. However the die, cluster, book, and drawer hierarchy related sysfs files will only be created if an architecture provides the related macros as described below.

For an architecture to support this feature, it must define some of these macros in include/asm-XXX/topology.h:

```
#define topology_physical_package_id(cpu)
#define topology_die_id(cpu)
#define topology_cluster_id(cpu)
#define topology_core_id(cpu)
#define topology_book_id(cpu)
#define topology_drawer_id(cpu)
#define topology_sibling_cpumask(cpu)
#define topology_core_cpumask(cpu)
#define topology_cluster_cpumask(cpu)
#define topology_die_cpumask(cpu)
#define topology_book_cpumask(cpu)
#define topology_drawer_cpumask(cpu)
```

The type of `**_id macros` is int. The type of `**_cpumask macros` is `(const) struct cpumask *`. The latter correspond with appropriate `**_siblings` sysfs attributes (except for topology_sibling_cpumask() which corresponds with thread_siblings).

To be consistent on all architectures, include/linux/topology.h provides default definitions for any of the above macros that are not defined by include/asm-XXX/topology.h:

1. topology_physical_package_id: -1
2. topology_die_id: -1
3. topology_cluster_id: -1
4. topology_core_id: 0
5. topology_book_id: -1
6. topology_drawer_id: -1
7. topology_sibling_cpumask: just the given CPU
8. topology_core_cpumask: just the given CPU
9. topology_cluster_cpumask: just the given CPU
10. topology_die_cpumask: just the given CPU
11. topology_book_cpumask: just the given CPU
12. topology_drawer_cpumask: just the given CPU

Additionally, CPU topology information is provided under /sys/devices/system/cpu and includes these files. The internal source for the output is in brackets ("[]").

kernel_max:	the maximum CPU index allowed by the kernel configuration. [NR_CPUS-1]
offline:	CPUs that are not online because they have been HOTPLUGGED off or exceed the limit of CPUs allowed by the kernel configuration (kernel_max above). [~cpu_online_mask + cpus >= NR_CPUS]
online:	CPUs that are online and being scheduled [cpu_online_mask]
possible:	CPUs that have been allocated resources and can be brought online if they are present. [cpu_possible_mask]
present:	CPUs that have been identified as being present in the system. [cpu_present_mask]

The format for the above output is compatible with cpulist_parse() [see <linux/cpumask.h>]. Some examples follow.

In this example, there are 64 CPUs in the system but cpus 32-63 exceed the kernel max which is limited to 0..31 by the NR_CPUS config option being 32. Note also that CPUs 2 and 4-31 are not online but could be brought online as they are both present and possible:

```
kernel_max: 31
   offline: 2,4-31,32-63
    online: 0-1,3
  possible: 0-31
   present: 0-31
```

In this example, the NR_CPUS config option is 128, but the kernel was started with possible_cpus=144. There are 4 CPUs in the system and cpu2 was manually taken offline (and is the only CPU that can be brought online.):

```
kernel_max: 127
   offline: 2,4-127,128-143
    online: 0-1,3
  possible: 0-127
```

```
present: 0-3
```

See Documentation/core-api/cpu_hotplug.rst for the possible_cpus=NUM kernel start parameter as well as more information on the various cpumasks.