NAME

cpu_id_t − This contains the recognized CPU features/info.

SYNOPSIS

#include <libcpuid.h>

Data Fields

cpu_architecture_t architecture
cpu_feature_level_t feature_level
char **vendor_str** [VENDOR_STR_MAX]
char **brand_str** [BRAND_STR_MAX]
cpu_vendor_t vendor
uint8_t **flags** [CPU_FLAGS_MAX]
int32_t **family**
int32_t **model**
int32_t **stepping**
int32_t **ext_family**
int32_t **ext_model**
union {
  struct **x86_id_t x86**
  struct **arm_id_t arm**
};
int32_t **num_cores**
int32_t **num_logical_cpus**
int32_t **total_logical_cpus**
int32_t **l1_data_cache**
int32_t **l1_instruction_cache**
int32_t **l2_cache**
int32_t **l3_cache**
int32_t **l4_cache**
int32_t **l1_assoc**
int32_t **l1_data_assoc**
int32_t **l1_instruction_assoc**
int32_t **l2_assoc**
int32_t **l3_assoc**
int32_t **l4_assoc**
int32_t **l1_cacheline**
int32_t **l1_data_cacheline**
int32_t **l1_instruction_cacheline**
int32_t **l2_cacheline**
int32_t **l3_cacheline**
int32_t **l4_cacheline**
int32_t **l1_data_instances**
int32_t **l1_instruction_instances**
int32_t **l2_instances**
int32_t **l3_instances**
int32_t **l4_instances**
char **cpu_codename** [CODENAME_STR_MAX]
int32_t **sse_size**
uint8_t **detection_hints** [CPU_HINTS_MAX]
struct **cpu_sgx_t sgx**
cpu_affinity_mask_t affinity_mask
cpu_purpose_t purpose
char **technology_node** [TECHNOLOGY_STR_MAX]

Detailed Description

This contains the recognized CPU features/info.

Field Documentation

union { ... } cpu_id_t

contains architecture specific info. Use **cpu_id_t::architecture** to know which member is valid.

cpu_affinity_mask_t cpu_id_t::affinity_mask

bitmask of the affinity ids this processor type is occupying

cpu_architecture_t cpu_id_t::architecture

contains the CPU architecture ID (e.g. ARCHITECTURE_X86)

char cpu_id_t::brand_str[BRAND_STR_MAX]

contains the brand string, e.g. "Intel(R) Xeon(TM) CPU 2.40GHz"

char cpu_id_t::cpu_codename[CODENAME_STR_MAX]

The brief and human-friendly CPU codename, which was recognized.
Examples:

```
+--------+--------+-------+-------+-------+----------------------------------------+-------
| Vendor | Family | Model | Step. | Cache |       Brand String                     | cpu_id_t.cpu_codename |
+--------+--------+-------+-------+-------+----------------------------------------+-------
| AMD    |      6 |     8 |     0 |   256 | (not available − will be ignored)      | "K6−2"                |
| Intel  |     15 |     2 |     5 |   512 | "Intel(R) Xeon(TM) CPU 2.40GHz"        | "Xeon (Prestonia)"    |
| Intel  |      6 |    15 |    11 |  4096 | "Intel(R) Core(TM)2 Duo CPU E6550..." | "Conroe (Core 2 Duo)" |
| AMD    |     15 |    35 |     2 |  1024 | "Dual Core AMD Opteron(tm) Proces..." | "Opteron (Dual Core)" |
+--------+--------+-------+-------+-------+----------------------------------------+-------
```

uint8_t cpu_id_t::detection_hints[CPU_HINTS_MAX]

contain miscellaneous detection information. Used to test about specifics of certain detected features. See **CPU_HINT_*** macros below.

See also

Hints

int32_t cpu_id_t::ext_family

CPU display ("true") family (computed as BaseFamily[3:0]+ExtendedFamily[7:0])

Deprecated

replaced by **x86_id_t::ext_family** (prefix member with x86., e.g. id.x86.ext_family)

int32_t cpu_id_t::ext_model

CPU display ("true") model (computed as (ExtendedModel[3:0]<<4) + BaseModel[3:0]) For detailed discussion about what BaseModel / ExtendedModel / Model are, see Github issue #150.

Deprecated

replaced by **x86_id_t::ext_model** (prefix member with x86., e.g. id.x86.ext_model)

int32_t cpu_id_t::family

CPU family (BaseFamily[3:0])

Deprecated

replaced by **x86_id_t::family** (prefix member with x86., e.g. id.x86.family)

cpu_feature_level_t cpu_id_t::feature_level

contains the CPU feature level, also know as microarchitecture levels (x86) and architecture version (ARM)

uint8_t cpu_id_t::flags[CPU_FLAGS_MAX]

contain CPU flags. Used to test for features. See the **CPU_FEATURE_*** macros below.

See also

Features

int32_t cpu_id_t::l1_assoc
Cache associativity for the L1 data cache. -1 if undetermined

Deprecated
replaced by **cpu_id_t::l1_data_assoc**

int32_t cpu_id_t::l1_cacheline
Cache-line size for L1 data cache. -1 if undetermined

Deprecated
replaced by **cpu_id_t::l1_data_cacheline**

int32_t cpu_id_t::l1_data_assoc
Cache associativity for the L1 data cache. -1 if undetermined

int32_t cpu_id_t::l1_data_cache
L1 data cache size in KB. Could be zero, if the CPU lacks cache. If the size cannot be determined, it will be -1.

int32_t cpu_id_t::l1_data_cacheline
Cache-line size for L1 data cache. -1 if undetermined

int32_t cpu_id_t::l1_data_instances
Number of L1 data cache instances. -1 if undetermined

int32_t cpu_id_t::l1_instruction_assoc
Cache associativity for the L1 instruction cache. -1 if undetermined

int32_t cpu_id_t::l1_instruction_cache
L1 instruction cache size in KB. Could be zero, if the CPU lacks cache. If the size cannot be determined, it will be -1.

Note
On some Intel CPUs, whose instruction cache is in fact a trace cache, the size will be expressed in K uOps.

int32_t cpu_id_t::l1_instruction_cacheline
Cache-line size for L1 instruction cache. -1 if undetermined

int32_t cpu_id_t::l1_instruction_instances
Number of L1 instruction cache instances. -1 if undetermined

int32_t cpu_id_t::l2_assoc
Cache associativity for the L2 cache. -1 if undetermined

int32_t cpu_id_t::l2_cache
L2 cache size in KB. Could be zero, if the CPU lacks L2 cache. If the size of the cache could not be determined, it will be -1

int32_t cpu_id_t::l2_cacheline
Cache-line size for L2 cache. -1 if undetermined

int32_t cpu_id_t::l2_instances
Number of L2 cache instances. -1 if undetermined

int32_t cpu_id_t::l3_assoc
Cache associativity for the L3 cache. -1 if undetermined

int32_t cpu_id_t::l3_cache
L3 cache size in KB. Zero on most systems

int32_t cpu_id_t::l3_cacheline
Cache-line size for L3 cache. -1 if undetermined

int32_t cpu_id_t::l3_instances
Number of L3 cache instances. -1 if undetermined

int32_t cpu_id_t::l4_assoc
Cache associativity for the L4 cache. -1 if undetermined

int32_t cpu_id_t::l4_cache
L4 cache size in KB. Zero on most systems

int32_t cpu_id_t::l4_cacheline
Cache-line size for L4 cache. -1 if undetermined

int32_t cpu_id_t::l4_instances
Number of L4 cache instances. -1 if undetermined

int32_t cpu_id_t::model
CPU model (BaseModel[3:0])

Deprecated
replaced by **x86_id_t::model** (prefix member with x86., e.g. id.x86.model)

int32_t cpu_id_t::num_cores
Number of CPU cores on the current processor

int32_t cpu_id_t::num_logical_cpus
Number of logical processors on the current processor. Could be more than the number of physical cores, e.g. when the processor has HyperThreading.

cpu_purpose_t cpu_id_t::purpose
processor type purpose, relevant in case of hybrid CPU (e.g. PURPOSE_PERFORMANCE)

struct cpu_sgx_t cpu_id_t::sgx
contains information about SGX features if the processor, if present

Deprecated
replaced by **x86_id_t::sgx** (prefix member with x86., e.g. id.x86.sgx)

int32_t cpu_id_t::sse_size
SSE execution unit size (64 or 128; -1 if N/A)

Deprecated
replaced by **x86_id_t::sse_size** (prefix member with x86., e.g. id.x86.sse_size)

int32_t cpu_id_t::stepping
CPU stepping

Deprecated
replaced by **x86_id_t::stepping** (prefix member with x86., e.g. id.x86.stepping)

char cpu_id_t::technology_node[TECHNOLOGY_STR_MAX]
contains the technology node string, e.g. "32 nm"

int32_t cpu_id_t::total_logical_cpus
The total number of logical processors. The same value is available through **cpuid_get_total_cpus**.

This is num_logical_cpus * {total physical processors in the system} (but only on a real system, under a VM this number may be lower).

If you’re writing a multithreaded program and you want to run it on all CPUs, this is the number of threads you need.

Note
in a VM, this will exactly match the number of CPUs set in the VM’s configuration.

cpu_vendor_t cpu_id_t::vendor
contains the recognized CPU vendor

char cpu_id_t::vendor_str[VENDOR_STR_MAX]
contains the CPU vendor string, e.g. "GenuineIntel"

Author

Generated automatically by Doxygen for libcpuid from the source code.