NAME

OPENSSL_ia32cap - the x86[_64] processor capabilities vector

SYNOPSIS

```
env OPENSSL_ia32cap=... <application>
```

DESCRIPTION

OpenSSL supports a range of x86[_64] instruction set extensions. These extensions are denoted by individual bits in capability vector returned by processor in EDX:ECX register pair after executing CPUID instruction with EAX=1 input value (see Intel Application Note #241618). This vector is copied to memory upon toolkit initialization and used to choose between different code paths to provide optimal performance across wide range of processors. For the moment of this writing following bits are significant:

bit #4 denoting presence of Time-Stamp Counter.
bit #19 denoting availability of CLFLUSH instruction;
bit #20, reserved by Intel, is used to choose among RC4 code paths;
bit #23 denoting MMX support;
bit #24, FXSR bit, denoting availability of XMM registers;
bit #25 denoting SSE support;
bit #26 denoting SSE2 support;
bit #28 denoting Hyperthreading, which is used to distinguish cores with shared cache;
bit #30, reserved by Intel, denotes specifically Intel CPUs;
bit #33 denoting availability of PCLMULQDQ instruction;
bit #41 denoting SSSE3, Supplemental SSE3, support;
bit #43 denoting AMD XOP support (forced to zero on non-AMD CPUs);
bit #54 denoting availability of MOVBE instruction;
bit #57 denoting AES-NI instruction set extension;
bit #58, XSAVE bit, lack of which in combination with MOVBE is used to identify Atom Silvermont core;
bit #59, OSXSAVE bit, denoting availability of YMM registers;
bit #60 denoting AVX extension;
bit #62 denoting availability of RDRAND instruction;

For example, in 32-bit application context clearing bit #26 at run-time disables high-performance SSE2 code present in the crypto library, while clearing bit #24 disables SSE2 code operating on 128-bit XMM register bank. You might have to do the latter if target OpenSSL application is executed on SSE2 capable CPU, but under control of OS that does not enable XMM registers. Historically address of the capability vector copy was exposed to application through *OPENSSL_ia32cap_loc()*, but not anymore. Now the only way to affect the capability detection is to set OPENSSL_ia32cap environment variable prior target application start. To give a specific example, on Intel P4 processor 'env OPENSSL_ia32cap=0x16980010 apps/openssl', or better yet 'env OPENSSL_ia32cap=~0x1000000 apps/openssl' would achieve the desired effect. Alternatively you can reconfigure the toolkit with no-sse2 option and recompile.

Less intuitive is clearing bit #28, or ~0x10000000 in the ''environment variable'' terms. The truth is that it's not copied from CPUID output verbatim, but is adjusted to reflect whether or not the data cache is actually shared between logical cores. This in turn affects the decision on whether or not expensive countermeasures against cache-timing attacks are applied, most notably in AES assembler module.

The capability vector is further extended with EBX value returned by CPUID with EAX=7 and ECX=0 as input. Following bits are significant:

bit #64+3 denoting availability of BMI1 instructions, e.g. ANDN;
bit #64+5 denoting availability of AVX2 instructions;
bit #64+8 denoting availability of BMI2 instructions, e.g. MULX and RORX;
bit #64+16 denoting availability of AVX512F extension;
bit #64+18 denoting availability of RDSEED instruction;
bit #64+19 denoting availability of ADCX and ADOX instructions;
bit #64+29 denoting availability of SHA extension;

bit #64+30 denoting availability of AVX512BW extension;
bit #64+31 denoting availability of AVX512VL extension;

To control this extended capability word use ':' as delimiter when setting up OPENSSL_ia32cap environment variable. For example assigning ':~0x20' would disable AVX2 code paths, and ':0' - all post-AVX extensions.

It should be noted that whether or not some of the most ''fancy'' extension code paths are actually assembled depends on current assembler version. Base minimum of AES-NI/PCLMULQDQ, SSSE3 and SHA extension code paths are always assembled. Besides that, minimum assembler version requirements are summarized in below table:

```
Extension | GNU as | nasm | llvm
------------+--------+--------+--------
AVX | 2.19 | 2.09 | 3.0
AVX2 | 2.22 | 2.10 | 3.1
AVX512 | 2.25 | 2.11.8 | 3.6
```

OPENSSL_ia32cap is a macro returning the first word of the vector.

COPYRIGHT