Padlock Instruction Set Reference

Shanghai Zhaoxin Semiconductor Co., Ltd.

CPUID flag

Issue CPUID with EAX = 0xC0000001 will return extended feature flags in EDX, of which:

EDX bit[2] – XSTORE, REP XSTORE and REP XRNG2 instructions are present

EDX bit[3] – XSTORE, REP XSTORE and REP XRNG2 instructions are enabled

EDX bit[6] - REP XCRYPT instructions are present

EDX bit[7] - REP XCRYPT instructions are enabled

EDX bit[10] - REP XSHA1, REP XSHA256, REP XSHA384 and REP XSHA512 instructions are present

EDX bit[11] - REP XSHA1, REP XSHA256, REP XSHA384 and REP XSHA512 instructions are enabled

1 RANDOM NUMBER GENERATION

1.1 XSTORE

Encoding: 0x0F 0xA7 0xC0

Modes: REAL, VIRTUAL 8086, COMPAT, PROTECT, LONG

Description: Store random bytes within 16-byte with specific control word settings.

- **Input Registers**
  - **EDX** Control word. Bits[1:0]==0 means output raw data generated by hardware random number generators; Bits[1:0]~=0 means output random number which raw data after postprocessing.
  - **EDI** Pointer to the memory for the random number of bytes to be stored. Assumes ES segment. The memory must be writable
- **Output Registers**
  - **EAX** The number of bytes for the random number to be stored.
  - **EDX** Bits[1:0] are unchanged, other bits are zero.
  - **EDI** Incremented by bytes for the random number to be stored. The memory will contain the random number.

1.2 REP XSTORE

Encoding: 0xF3 0x0F 0xA7 0xC0

Modes: REAL, VIRTUAL 8086, COMPAT, PROTECT, LONG

Description: Store multiple count random bytes with specific control word settings.

- **Input Registers**
  - ECX The number of bytes for the random number to be stored.
  - EDX Control word. Bits[1:0]==0 means output raw data generated by hardware random number generators; Bits[1:0]~=0 means output random number which raw data after

postprocessing.

EDI Pointer to the memory for the random number of bytes to be stored. Assumes ES segment. The memory must be writable.

- **Output Registers**

  ECX 0.

  EDX Bits[1:0] are unchanged, other bits are zero.

  EDI Incremented by bytes for the random number to be stored. The memory will contain the random number.

1.3 REP_XRNG2

Encoding: 0xF3 0x0F 0xA7 0xF8

Modes: REAL, VIRTUAL 8086, COMPAT, PROTECT, LONG

Description: Store multiple count random bytes with specific control word settings.

- **Input Registers**

  EAX Random number generators hardware control. Each CPU core include two random number generators, bits[10:9]==0 means to select output raw bits from two random number generators; bits[10:9]==1 means to select raw bits from one random number generator.

  ECX The number of bytes for the random number to be stored.

  EDX Control word. Bits[1:0]==0 means output raw data generated by hardware random number generators; Bits[1:0]~=0 means output random number which raw data after postprocessing.

  EDI Pointer to the memory for the random number of bytes to be stored. Assumes ES segment. The memory must be writable.

- **Output Registers**

  ECX 0

  EDX Bits[1:0] are unchanged, other bits are zero.

  EDI Incremented by bytes for the random number to be stored. The memory will contain the random number.

2 ADVANCED CRYPTOGRAPHY ENGINE

Control word pointed by EDX define:

Bits[3:0]: Round, 10 round for 128bits key; 12 round for 192bits key; 14 round for 256bits key.

Bit[4]: 1 means support message authentication code(MAC); 0 means don't support MAC.

Bit[7]: 1 means load key from memory for 192bits and 256bits key; 0 means to generate key by hardware for 128bits key.

Bit[9]: 0 means encryption; 1 means decryption.

Bit[11:10]: key size, 2'b00 means 128bits, 2'b01 means 192bits, 2'b10 means 256bits, don't support 2'b11.

2.1 REP XCRYPTECB

Encoding: 0xF3 0x0F 0xA7 0xC8

Modes: REAL, VIRTUAL 8086, COMPAT, PROTECT, LONG

Description: encrypt/decrypt multiple 16-byte blocks plaintext/ciphertext using electronic code book mode

- **Input Registers**
  - **EBX** Pointer to the memory for encryption or decryption key. Assumes ES segment.
  - **ECX** Number of 16-byte blocks to encrypt or decrypt.
  - **EDX** Pointer to the memory for control word. Assumes ES segment
  - **ESI** Input pointer to the memory for plaintext when encrypting or ciphertext when decrypting. Assumes ES segment.
  - **EDI** Output pointer to the memory for ciphtertext when encrypting or plaintext when decrypting. Assumes ES segment. The memory must be writable.
- **Output Registers**
  - **ECX** 0
  - **ESI** Incremented by the number of bytes for plaintext when encrypting or ciphertext when decrypting.
  - **EDI** Incremented by the number of bytes ciphtertext when encrypting or plaintext when decrypting. The memory will contain the ciphtertext when encrypting or plaintext when decrypting.

2.2 REP XCRYPTCBC

Encoding: 0xF3 0x0F 0xA7 0xD0

Modes: REAL, VIRTUAL 8086, COMPAT, PROTECT, LONG

Description: encrypt/decrypt multiple 16-byte blocks plaintext/ciphertext using cipher block chaining mode

- **Input Registers**
  - **EAX** Pointer to the memory for initialization vector. Assumes ES segment. The memory must be writable.
  - **EBX** Pointer to the memory for encryption or decryption key. Assumes ES segment.
  - **ECX** Number of 16-byte blocks to encrypt or decrypt.
  - **EDX** Pointer to the memory for control word. Assumes ES segment
  - **ESI** Input pointer to the memory for plaintext when encrypting or ciphertext when decrypting. Assumes ES segment.
  - **EDI** Output pointer to the memory for ciphtertext when encrypting or plaintext when decrypting. Assumes ES segment. The memory must be writable.
- **Output Registers**
  - **EAX** Unchanged. The memory will contain the updated initialization vector, to be used for the next sequential 16-byte blocks input.
  - **ECX** 0
  - **ESI** Incremented by the number of bytes for plaintext when encrypting or ciphertext when decrypting.
  - **EDI** Incremented by the number of bytes ciphtertext when encrypting or plaintext when decrypting. The memory will contain the ciphtertext when encrypting or plaintext when decrypting.

2.3 REP XCRYPTCTR

Encoding: 0xF3 0x0F 0xA7 0xD8

Modes: REAL, VIRTUAL 8086, COMPAT, PROTECT, LONG

Description: encrypt/decrypt multiple 16-byte blocks plaintext/ciphertext using counter mode

- **Input Registers**
  - **EAX** Pointer to the memory for initialization vector. Assumes ES segment. The memory must be writable.
  - **EBX** Pointer to the memory for encryption or decryption key. Assumes ES segment.
  - **ECX** Number of 16-byte blocks to encrypt or decrypt.
  - **EDX** Pointer to the memory for control word. Assumes ES segment
  - **ESI** Input pointer to the memory for plaintext when encrypting or ciphertext when decrypting. Assumes ES segment.
  - **EDI** Output pointer to the memory for ciphtertext when encrypting or plaintext when decrypting. Assumes ES segment. The memory must be writable.
- **Output Registers**
  - **EAX** Unchanged. The memory will contain the updated initialization vector, to be used for the next sequential 16-byte blocks input.
  - **ECX** 0
  - **ESI** Incremented by the number of bytes for plaintext when encrypting or ciphertext when decrypting.
  - **EDI** Incremented by the number of bytes ciphtertext when encrypting or plaintext when decrypting. The memory will contain the ciphtertext when encrypting or plaintext when decrypting.

2.4 REP XCRYPTCFB

Encoding: 0xF3 0x0F 0xA7 0xE0

Modes: REAL, VIRTUAL 8086, COMPAT, PROTECT, LONG

Description: encrypt/decrypt multiple 16-byte blocks plaintext/ciphertext using cipher feedback mode

- **Input Registers**
  - **EAX** Pointer to the memory for initialization vector. Assumes ES segment. The memory must be writable.
  - **EBX** Pointer to the memory for encryption or decryption key. Assumes ES segment.
  - **ECX** Number of 16-byte blocks to encrypt or decrypt.
  - **EDX** Pointer to the memory for control word. Assumes ES segment
  - **ESI** Input pointer to the memory for plaintext when encrypting or ciphertext when decrypting. Assumes ES segment.
  - **EDI** Output pointer to the memory for ciphtertext when encrypting or plaintext when decrypting. Assumes ES segment. The memory must be writable.
- **Output Registers**
  - **EAX** Unchanged. The memory will contain the updated initialization vector, to be used for

the next sequential 16-byte blocks input.

ECX 0

ESI Incremented by the number of bytes for plaintext when encrypting or ciphertext when decrypting.

EDI Incremented by the number of bytes ciphtertext when encrypting or plaintext when decrypting. The memory will contain the ciphtertext when encrypting or plaintext when decrypting.

2.5 REP XCRYPTOFB

Encoding: 0xF3 0x0F 0xA7 0xE8

Modes: REAL, VIRTUAL 8086, COMPAT, PROTECT, LONG

Description: encrypt/decrypt multiple 16-byte blocks plaintext/ciphertext using output feedback mode

- **Input Registers**

  EAX Pointer to the memory for initialization vector. Assumes ES segment. The memory must be writable.

  EBX Pointer to the memory for encryption or decryption key. Assumes ES segment.

  ECX Number of 16-byte blocks to encrypt or decrypt.

  EDX Pointer to the memory for control word. Assumes ES segment

  ESI Input pointer to the memory for plaintext when encrypting or ciphertext when decrypting. Assumes ES segment.

  EDI Output pointer to the memory for ciphtertext when encrypting or plaintext when decrypting. Assumes ES segment. The memory must be writable.

- **Output Registers**

  EAX Unchanged. The memory will contain the updated initialization vector, to be used for the next sequential 16-byte blocks input.

  ECX 0

  ESI Incremented by the number of bytes for plaintext when encrypting or ciphertext when decrypting.

  EDI Incremented by the number of bytes ciphtertext when encrypting or plaintext when decrypting. The memory will contain the ciphtertext when encrypting or plaintext when decrypting.

3 PADLOCK HASH ENGINE

3.1 REP XSHA1

Encoding: 0xF3 0x0F 0xA6 0xC8

Modes: REAL, VIRTUAL 8086, COMPAT, PROTECT, LONG

Description: Calculate SHA-1 hash algorithm as specified by FIPS 180-3

- **Input Registers**

  EAX If EAX == 0 performs padding for input data stream specified by FIPS 180-3, and ECX means byte counter of input data stream.

If EAX == -1 does not perform padding for input data stream, and ECX means the number of 64-byte blocks of input data stream.

ECX Size of the input data stream:
If EAX == 0: means in bytes
If EAX == -1: means in 64-byte blocks

ESI Pointer to the memory for input data stream. Assumes ES segment.

EDI Pointer to the memory for initial hash constants. Assumes ES segment. The memory must be writable.

- **Output Registers**

EAX If the input EAX == 0, EAX will be equal to ECX, otherwise it was 0.

ECX If the input EAX == 0, ECX will be unchanged, otherwise it was 0.

ESI Incremented by the number of bytes input data stream to perform the hash calculation.

EDI Unchanged. The memory will contain the result of the hash calculation.

3.2 REP XSHA256

Encoding: 0xF3 0x0F 0xA6 0xD0

Modes: REAL, VIRTUAL 8086, COMPAT, PROTECT, LONG

Description: Calculate SHA-256 hash algorithm as specified by FIPS 180-3

- **Input Registers**

EAX If EAX == 0 performs padding for input stream specified by FIPS 180-3, and ECX means byte counter of input data stream.
If EAX == -1 does not perform padding for input stream, and ECX means the number of 64-byte blocks of input data stream.

ECX Size of the input data stream:
If EAX == 0: means in bytes
If EAX == -1: means in 64-byte blocks

ESI Pointer to the memory for input data stream. Assumes ES segment.

EDI Pointer to the memory for initial hash constants. Assumes ES segment. The memory must be writable.

- **Output Registers**

EAX If the input EAX == 0, EAX will be equal to ECX, otherwise it was 0.

ECX If the input EAX == 0, ECX will be unchanged, otherwise it was 0.

ESI Incremented by the number of bytes input data stream to perform the hash calculation.

EDI Unchanged. The memory will contain the result of the hash calculation.

3.3 REP_XSHA384

Encoding: 0xF3 0x0F 0xA6 0xD8

Modes: REAL, VIRTUAL 8086, COMPAT, PROTECT, LONG

Description: Calculate SHA-384 hash algorithm as specified by FIPS 180-3

- **Input Registers**

ECX Size of the input data stream in 128-byte blocks.

ESI Pointer to the memory for input data stream. Assumes ES segment.

EDI Pointer to the memory for initial hash constants. Assumes ES segment. The memory must be writable.

- **Output Registers**

  ECX 0.

  ESI Incremented by the number of bytes input data stream to perform the hash calculation.

  EDI Unchanged. The memory will contain the result of the hash calculation.

3.4 REP_XSHA512

Encoding: 0xF3 0x0F 0xA6 0xE0

Modes: REAL, VIRTUAL 8086, COMPAT, PROTECT, LONG

Description: Calculate SHA-512 hash algorithm as specified by FIPS 180-3

- **Input Registers**

  ECX Size of the input data stream in 128-byte blocks.

  ESI Pointer to the memory for input data stream. Assumes ES segment.

  EDI Pointer to the memory for initial hash constants. Assumes ES segment. The memory must be writable.

- **Output Registers**

  ECX 0.

  ESI Incremented by the number of bytes input data stream to perform the hash calculation.

  EDI Unchanged. The memory will contain the result of the hash calculation.

4 MODULAR MULTIPLICATION AND EXPONENTIATION

4.1 REP_XMODEXP

Encoding: 0xF3 0x0F 0xA6 0xF8

Modes: REAL, VIRTUAL 8086, COMPAT, PROTECT, LONG

Description: Calculates A**B (mod M) where A,B, and M are large integers

- **Input Registers**

  EAX Pointer to the memory for bigInt A. Assumes ES segment.

  EBX Pointer to the memory for bigInt B. Assumes ES segment.

  ECX Bits size of the bigInts. A multiple of 128 at least 256 and less than or equal to 32768.

  EDX Pointer to the memory for bigInt M. Assumes ES segment

  ESI Pointer to the memory for scratch space. The memory must be writable.

  EDI Pointer to the memory for bigInt result. Assumes ES segment. The memory must be writable.

- **Output Registers**

  EDI Unchanged. The memory will contain the result of the exponentiation.

4.2 REP_MONTMUL2

Encoding: 0xF3 0x0F 0xA6 0xF0

Modes: REAL, VIRTUAL 8086, COMPAT, PROTECT, LONG

Description: Calculates A * B (mod M) where A,B, and M are large integers.

- **Input Registers**

  EAX Pointer to the memory for bigInt A. Assumes ES segment.

  EBX Pointer to the memory for bigInt B. Assumes ES segment.

  ECX Bits size of the bigInts. A multiple of 128 at least 256 and less than or equal to 32768.

  EDX Pointer to the memory for bigInt M. Assumes ES segment.

  ESI Pointer to the memory for scratch space. The memory must be writable.

  EDI Pointer to the memory for bigInt result. Assumes ES segment. The memory must be writable.

- **Output Registers**

  EDI Unchanged. The memory will contain the result of the multiplication.

4.3 REP_MONTMUL

Encoding: 0xF3 0x0F 0xA6 0xC0

Modes: COMPAT, 32-bit PROTECT

Description: Calculates A * B (mod M) where A,B, and M are large integers.

- **Input Registers**

  EAX 0.

  ECX Bits size of the bigInts. A multiple of 128 at least 256 and less than or equal to 32768.

  ESI Pointer to the memory for Montgomery context. All pointers for bigInt A, bigInt B, bigInt T and bigInt M are must be 32 bits.

- **Output Registers**

  ESI The result is stored in memory pointed by bigintT in Montgomery context.

Montgomery context is a continuous memory range, include bigint A、B、T、M and Scratch Space address, show as below:

Start address	Length	Content
ESI	0x04	A memory address
ESI + 0x04	0x04	B memory address
ESI + 0x08	0x04	T memory address
ESI + 0x0C	0x04	M memory address
ESI + 0x10	0x20	Scratch Space memory address