Padlock Instruction Set Reference

Shanghai Zhaoxin Semiconductor Co., Ltd.

Padlock instruction set include three random number generation engine instructions XSTORE/REP XSTORE/REP XRNG2、five REP XCRYPT advanced cryptography engine instructions、four REP XSHA hash engine instructions and three modular multiplication and exponentiation engine instructions REP XMODEXP/REP MONTMUL2/REP MONTMUL.

CPUID flag

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

Bit[2] - XSTORE and REP XSTORE instructions are present[1]

Bit[3] - XSTORE and REP XSTORE instructions are enabled[2]

Bit[6] - REP XCRYPT instructions are present

Bit[7] - REP XCRYPT instructions are enabled

Bit[10] - REP XSHA1 and REP XSHA256 instructions are present

Bit[11] - REP XSHA1 and REP XSHA256 instructions are enabled

Bit[12] - REP MONTMUL instructions are present

Bit[13] - REP MONTMUL instructions are enabled

Bit[22] - REP XRNG2 instructions are present

Bit[23] - REP XRNG2 instructions are enabled

Bit[25] - REP XSHA384 and REP XSHA512 instructions are present

Bit[26] - REP XSHA384 and REP XSHA512 instructions are enabled

Bit[27] - REP XMODEXP and REP MONTMUL2 instructions are present

Bit[28] - REP XMODEXP and REP MONTMUL2 instructions are enabled

Note1: Present bit is used to inform software corresponding instruction is present.

Note2: Enabled bit is used to inform software corresponding instruction is enabled for normal use.

Padlock Instructions Prefix Affect

When executing in protected mode or compatibility mode, depending on the settings of the D flag and the operand-size and address-size prefixes, effective operand-size and address-size attributes as the following:

D Flag in Code Segment Descriptor	0	0	0	0	1	1	1	1
Operand-Size Prefix 66H	N	N	Y	Y	N	N	Y	Y
Address-Size Prefix 67H	N	Y	N	Y	N	Y	N	Y
Effective Operand Size	16	16	32	32	32	32	16	16
Effective Address Size	16	32	16	32	32	16	32	16

Y：This instruction prefix is present.

N：This instruction prefix is not present

When executing in 64-bit mode, depending on the settings of the L flag and the operand-size and address-size prefixes, effective operand-size and address-size attributes as the following:

L Flag in Code Segment Descriptor	1	1	1	1	1	1	1	1
REX.W Prefix	0	0	0	0	1	1	1	1
Operand-Size Prefix 66H	N	N	Y	Y	N	N	Y	Y
Address-Size Prefix 67H	N	Y	N	Y	N	Y	N	Y
Effective Operand Size	32	32	16	16	64	64	64	64
Effective Address Size	64	32	64	32	64	32	64	32

Y：This instruction prefix is present.

N：This instruction prefix is not present

Padlock Instructions Interrupt Handle

Interrupt or exception can be responded to during the REP XSTORE/REP XRNG2/REP XCRYPT/REP XSHA/REP XMODEXP/REP MONTMUL2/REP MONTMUL instructions. The registers and memory contents are modified during REP XSTORE/REP XRNG2/REP XCRYPT/REP XSHA/REP XMODEXP/REP MONTMUL2/REP MONTMUL instructions, these registers and memory status can be saved when interrupt or exception interrupted REP XSTORE/REP XRNG2/REP XCRYPT/REP XSHA/REP XMODEXP/REP MONTMUL2/REP MONTMUL instructions. So it can return to the currently interrupted REP XSTORE/REP XRNG2/REP XCRYPT/REP XSHA/REP XMODEXP/REP MONTMUL2/REP MONTMUL instructions to continue execute according to saved registers and memory status after handling the interrupt or exception.

Padlock Instructions Input Data Size Explanation

REP XSTORE/REP XRNG2/REP XCRYPT/REP XSHA instructions use CX/ECX/RCX according to actual operation mode, the maximum number of input data is limited by the maximum value that register itself can set. However, the actual number of data that can be processed in a single REP XSTORE/REP XRNG2/REP XCRYPT/REP XSHA instruction is also constrained by the physical memory size of the chip and other factors.

1 RANDOM NUMBER GENERATION ENGINE

Random number generation engine include three instructions XSTORE、REP XSTORE and REP XRNG2.

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

Advanced cryptography engine include five instructions REP XCRYPTECB、REP XCRYPTCBC、REP XCRYPTCTR、REP XCRYPTCFB and REP XCRYPTOFB. These five instructions share a control word definition. The control word definition as the following:

Bits[3:0]: Rounds, 10 rounds for 128bits key; 12 rounds for 192bits key; 14 rounds 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 key 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

Hash engine include four instructions REP XSHA1、REP XSHA256、REP XSHA384 and REP XSHA512.

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 ECX == 0, HW doesn't perform any operation like other standard x86 REP prefix instructions since Zhaoxin Wudaokou processor, unlike previous processor.

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. The memory address needs to be 16-byte aligned, and need to allocate 32-byte memory space.

- **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 ECX == 0, HW doesn't perform any operation like other standard x86 REP prefix instructions since Zhaoxin Wudaokou processor, unlike previous processor.

  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. The memory address needs to be 16-byte aligned, and need to allocate 32-byte memory space.

- **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. The memory address needs to be 16-byte aligned, and need to allocate 64-byte memory space.

- **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. The memory address needs to be 16-byte aligned, and need to allocate 64-byte memory space.

- **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 ENGINE

Modular multiplication and exponentiation engine include three instructions REP XMODEXP、REP MONTMUL2 and REP MONTMUL.

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. Show as:

Start address	Length	Content
ESI	0x08	Zero
ESI + 0x08	0x08	Bits size of the bigInts(SIZE)
ESI + 0x10	N	Scratch. At least 3*(SIZE)+64 bytes

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 * R^{-1} (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 Zeroprime、pointer to bigint A、B、T、M and Scratch, show as below:

Start address	Length	Content
ESI	0x04	Zeroprime
ESI + 0x04	0x04	A memory address
ESI + 0x08	0x04	B memory address
ESI + 0x0C	0x04	T memory address
ESI + 0x10	0x04	M memory address
ESI + 0x14	0x20	Scratch