<?xml version="1.0" encoding="UTF-8"?>

<!-- Visit http://ref.x86asm.net -->

<!-- revision 1.11 -->

<!-- Elements -->

<!-- 1st level -->

<!-- x86reference: root element -->

<!ELEMENT x86reference (one-byte, two-byte, gen\_notes, ring\_notes)>

<!-- 2nd level -->

<!-- one-byte and two-byte: using this emelents is the reference -->

<!-- divided among two main parts, one-byte opcodes and two-byte -->

<!-- opcodes -->

<!ELEMENT one-byte (pri\_opcd+)>

<!ELEMENT two-byte (pri\_opcd+)>

<!-- gen\_notes: general notes, referenced when some content needs -->

<!-- further explanation -->

<!ELEMENT gen\_notes (gen\_note+)>

<!-- ring notes: further determinative conditions referenced from the -->

<!-- entry element, ring\_ref attribute with the value "f" -->

<!ELEMENT ring\_notes (ring\_note+)>

<!-- 3rd level-->

<!-- pri\_opcd: wraps all informations about single primary opcode -->

<!-- note: proc\_start and proc\_end could't be attributes of -->

<!-- pri\_opcode, because they can be overriden by the same -->

<!-- elements in entry element -->

<!-- note: proc\_end at this level can never be overriden by -->

<!-- entry/proc\_end with latter processor -->

<!ELEMENT pri\_opcd (proc\_start?, proc\_end?, entry+)>

<!-- gen\_note and ring\_note: single entry for -->

<!-- gen\_notes and ring\_notes element -->

<!ELEMENT gen\_note EMPTY>

<!ELEMENT ring\_note EMPTY>

<!-- 4th level-->

<!-- entry: wraps all informations about single opcode extension, -->

<!-- single mod variant {to be revised} and single semantic (i.e. single -->

<!-- prefix or single secondary opcode);-->

<!-- more syntaxes are inside one entry element -->

<!ELEMENT entry (

opcd\_ext?,

pref?, sec\_opcd?,

proc\_start?, proc\_end?,

syntax+,

instr\_ext?, grp1?, grp2\*, grp3\*,

test\_f?, modif\_f?, def\_f?, undef\_f?, f\_vals?,

test\_f\_fpu?, modif\_f\_fpu?, def\_f\_fpu?, undef\_f\_fpu?, f\_vals\_fpu?,

note?)>

<!-- 5th level -->

<!-- quote: quoted text -->

<!ELEMENT quote (#PCDATA)>

<!-- opcd\_ext: opcode extension; if present, only one per entry -->

<!-- element -->

<!ELEMENT opcd\_ext (#PCDATA)>

<!-- pref: prefix; doesn't always match the meaning of a prefix from -->

<!-- the Intel manuals. This is simply the value which always -->

<!-- precede the primary opcode -->

<!ELEMENT pref (#PCDATA)>

<!-- sec\_opcd: secondary opcode; fixed, appended value to the primary -->

<!-- opcode -->

<!ELEMENT sec\_opcd (#PCDATA)>

<!-- proc\_start: opcode's introductory processor -->

<!ELEMENT proc\_start (#PCDATA)>

<!-- proc\_end: opcode's terminating processor -->

<!ELEMENT proc\_end (#PCDATA)>

<!-- processor codes:

00 - 8086

01 - 80186

02 - 80286

03 - 80386

04 - 80486

05 - Pentium 1

06 - Pentium with MMX

07 - Pentium Pro

08 - Pentium II

09 - Pentium III

10 - Pentium 4

11 - Core 1

12 - Core 2

13 - Core i7

99 - Itanium

-->

<!-- instr\_ext: instruction extension, which the opcode belongs to -->

<!ELEMENT instr\_ext (#PCDATA)>

<!-- grp1, grp2, grp3: main group, sub-group, sub-sub-group: -->

<!-- classifies the opcode among groups; these groups don't match the -->

<!-- instruction groups given by the Intel manual; one instruction may -->

<!-- fit into more groups -->

<!ELEMENT grp1 (#PCDATA)>

<!ELEMENT grp2 (#PCDATA)>

<!ELEMENT grp3 (#PCDATA)>

<!-- test\_f, modif\_f, def\_f, and undef\_f: tested, modified, defined, -->

<!-- and undefined flags in rFlags register; marks these flags using -->

<!-- case-sensitive "odiszapc" flag pattern. Present flag fits in with -->

<!-- the appropriate group -->

<!ELEMENT test\_f (#PCDATA)>

<!ELEMENT modif\_f (#PCDATA)>

<!ELEMENT def\_f (#PCDATA)>

<!ELEMENT undef\_f (#PCDATA)>

<!-- f\_vals: flag values in rFlags register; marks the values of flags, -->

<!-- which are always set or cleared, using case-sensitive odiszapc -->

<!-- flag pattern -->

<!ELEMENT f\_vals (#PCDATA)>

<!-- test\_f\_fpu, modif\_f\_fpu, def\_f\_fpu, and undef\_f\_fpu: tested, -->

<!-- modified, defined, and undefined flags in x87 fpu flags; marks -->

<!-- these flags using 0123 fpu flag pattern; present flag fits in -->

<!-- with the appropriate group -->

<!ELEMENT test\_f\_fpu (#PCDATA)>

<!ELEMENT modif\_f\_fpu (#PCDATA)>

<!ELEMENT def\_f\_fpu (#PCDATA)>

<!ELEMENT undef\_f\_fpu (#PCDATA)>

<!-- f\_vals\_fpu: flag values in x87 fpu flags; marks these flags using -->

<!-- 0123 fpu flag pattern; present flag holds its value -->

<!ELEMENT f\_vals\_fpu (#PCDATA)>

<!-- note: description and notes (brief and detailed) -->

<!ELEMENT note (brief, det?)>

<!-- syntax: instruction syntax; there may be more syntaxes for one -->

<!-- opcode; the syntax may be empty -->

<!ELEMENT syntax (

mnem?,

dst\*,

src\*)>

<!ELEMENT brief (#PCDATA|sup|sub)\*>

<!ELEMENT det (#PCDATA|op1|op2|op3|op4)\*> <!-- op\* intended for future use -->

<!-- 6th level -->

<!-- mnem: instruction mnemonic -->

<!ELEMENT mnem (#PCDATA)>

<!-- dst: destination (modified) operand -->

<!ELEMENT dst (#PCDATA|a|t)\*>

<!-- src: source (not modified) operand -->

<!ELEMENT src (#PCDATA|a|t)\*>

<!-- note/brief/sup and note/brief/sub: superscript and subscript -->

<!ELEMENT sup (#PCDATA)>

<!ELEMENT sub (#PCDATA)>

<!ELEMENT op1 EMPTY>

<!ELEMENT op2 EMPTY>

<!ELEMENT op3 EMPTY>

<!ELEMENT op4 EMPTY>

<!-- 7th level -->

<!ELEMENT a (#PCDATA)>

<!ELEMENT t (#PCDATA)>

<!-- Attributes -->

<!-- x86reference@version: current version (completeness) of the reference -->

<!ATTLIST x86reference version NMTOKEN #REQUIRED>

<!-- two-byte@id: used only as an anchor for 0F opcode -->

<!ATTLIST two-byte xml:id ID #REQUIRED>

<!-- gen\_note@id, and ring\_note@id: anchors for notes -->

<!ATTLIST gen\_note id ID #REQUIRED>

<!ATTLIST ring\_note id ID #REQUIRED>

<!-- pri\_opcd@value: primary opcode value itself -->

<!ATTLIST pri\_opcd value NMTOKEN #REQUIRED>

<!-- DEPRECATED since 1.11, unnecessary, will be removed in future:

proc\_start@post: is valid also on posterior processors? -->

<!-- proc\_start@lat\_step: is valid only in latter steppings of the processor? -->

<!ATTLIST proc\_start post (no) #IMPLIED> <!-- "yes" is implied -->

<!ATTLIST proc\_start lat\_step (yes) #IMPLIED> <!-- "no" is implied -->

<!-- entry -->

<!-- - bit fields present in the primary opcode -->

<!-- entry@direction: means bit d (bit index 1, Direction); may be combined with bit w -->

<!-- entry@sign-ext: means bit s (bit index 1, Sign-extend); may be combined with bit w -->

<!-- entry@op\_size: means bit w (bit index 0, operand size) is present; may be combined with bits d and s -->

<!-- entry@tttn: means bit field tttn (4 bits, bit index 0, condition); used only with conditional instructions -->

<!-- entry@mem\_format: means bit field MF - memory format (2 bits, bit index 1); used only with x87 FPU instructions -->

<!-- coded with second floating-point instruction format -->

<!-- entry@r: does the ModR/M byte of the instruction contain a register operand and an r/m operand? -->

<!-- entry@mod: mod is always 11bin (nomem) or different from 11bin (mem); there is also a default value because -->

<!-- it simplifies its testing in XSL -->

<!-- entry@ref: referrer (for example, it refers to two-byte element from the opcode 0F) -->

<!-- [unused, removed] entry@id: an anchor for its aliases -->

<!-- entry@alias: the opcode is just an alias for the referenced opcode (format: -->

<!-- <two-char\_or\_four-char\_hex\_opcode>\_<one-char\_hex\_opcode\_extension>) -->

<!-- entry@part\_alias: the opcode is not true alias for the referenced opcode (format: see @alias) -->

<!-- entry@doc\_part\_alias\_ref: refers to the description of the not-true alias -->

<!-- entry@lock: is the opcode generally valid with the lock prefix? -->

<!-- [unused, removed] entry@rep: is the behavior of the opcode defined with rep/repcc prefix? -->

<!-- entry@attr: other attributes of the opcode: -->

<!-- invd: the opcode is invalid -->

<!-- undef: the behaviour of the opcode is always undefined (e. g., SALC) -->

<!-- null (only prefixes): the prefix has no meaning (no operation) -->

<!-- nop (nop instructions and obsolete x87 FPU instructions): -->

<!-- the instruction is treated as integer NOP instruction; it should contain a reference to and -->

<!-- the title with the source (doc\_ref attribute should be used) -->

<!-- acc: the opcode is optimized for the accumulator (e.g., 0x04 or 0x05) -->

<!-- serial: the opcode is serializing (CPUID; IRET; RSM; MOV Ddqp; WRMSR; INVD, INVLPG, WBINWD; LGDT; -->

<!-- LLDT; LIDT; LTR; LMSW) -->

<!-- serial\_cond: same as serial, but under further conditions (only MOV Cq) -->

<!-- delaysint: the opcode delays recognition of interrupts until after execution of the next instruction -->

<!-- (only POP SS) -->

<!-- delaysint\_cond: same as delaysint, but under further conditions (only STI) -->

<!-- entry@doc: marks how is the instruction documented in the Intel manuals: -->

<!-- d: fully documented; it can contain a reference to and a title with the chapter, where -->

<!-- the instruction is documented, if it may be unclear (doc\_ref attribute should be used) -->

<!-- m: only marginally (e.g., meaning of prefix 66hex when used with SSE instruction extensions) -->

<!-- u: undocumented at all; it should contain a reference to and a title with the source -->

<!-- (e.g., SALC, INT1) (doc\_ref attribute should be used) -->

<!-- entry@doc\_ref: refers to the additional notes about the documentation state, for all modes -->

<!-- entry@doc1632\_ref: same like doc\_ref, applies for outside 64-bit mode -->

<!-- entry@doc64\_ref: same like doc\_ref, applies only for 64-bit mode -->

<!-- entry@ring: The ring level, from which is the instruction valid (3 or 0), or indicates using f mark that -->

<!-- the level depends on a flag(s) (e.g., IN: rFLAGS.IOPL). It should contain a title with and -->

<!-- a reference to the description of that flag, if the flag is not complex (ring\_ref attribute should -->

<!-- be used). -->

<!-- The title is useful in electronic use, the reference is necessary in printed version. -->

<!-- entry@ring\_ref: refers to the description of the flag -->

<!-- entry@mode: Mode of Operation: marks the mode, in which is the opcode valid: -->

<!-- r: valid in real, protected and 64-bit mode; SMM is not taken into account -->

<!-- p: valid only in protected and 64-bit mode; SMM is not taken into account -->

<!-- e: valid only in 64-bit mode; SMM is not taken into account -->

<!-- s: valid only in SMM (only RSM) -->

<!-- This element with @mode 'e' must be always behind -->

<!-- the same sibling element with other @mode value -->

<!-- entry@fpop: Number of pops of the x87 FPU register stack (once, twice). Such explicit way is used because some -->

<!-- x87 FPU mnemonics don't indicate it directly (e. g., FPATAN) -->

<!-- entry@particular: "yes" indicates very specific, or special, meaning of the opcode; such meaning can be omitted -->

<!-- in, for example, less strict disassembler -->

<!ATTLIST entry direction (0|1) #IMPLIED>

<!ATTLIST entry sign-ext (0|1) #IMPLIED>

<!ATTLIST entry op\_size (0|1) #IMPLIED>

<!ATTLIST entry tttn (

0000|0001|0010|0011|0100|0101|

0110|0111|1000|1001|1010|1011|

1100|1101|1110|1111

) #IMPLIED>

<!ATTLIST entry mem\_format (00|01|10|11) #IMPLIED>

<!ATTLIST entry r (yes) #IMPLIED> <!-- "no" is implied -->

<!ATTLIST entry mod (nomem|mem) #IMPLIED>

<!ATTLIST entry ref IDREF #IMPLIED>

<!ATTLIST entry alias NMTOKEN #IMPLIED>

<!ATTLIST entry part\_alias NMTOKEN #IMPLIED>

<!ATTLIST entry doc\_part\_alias\_ref IDREF #IMPLIED>

<!ATTLIST entry lock (yes) #IMPLIED> <!-- "no" is implied -->

<!ATTLIST entry attr (

invd|

undef|

null|

nop|

acc|

serial|

serial\_cond|

delaysint|

delaysint\_cond

) #IMPLIED>

<!ATTLIST entry doc (m|u) #IMPLIED> <!-- "d" is implied -->

<!-- is\_doc, is\_undoc new in 1.02.dev1 -->

<!ATTLIST entry is\_doc (yes) #IMPLIED> <!-- "no" is implied -->

<!ATTLIST entry is\_undoc (yes) #IMPLIED> <!-- "no" is implied -->

<!ATTLIST entry doc\_ref IDREF #IMPLIED>

<!ATTLIST entry doc1632\_ref IDREF #IMPLIED>

<!ATTLIST entry doc64\_ref IDREF #IMPLIED>

<!ATTLIST entry ring (3|2|1|0|f) #IMPLIED>

<!ATTLIST entry ring\_ref IDREF #IMPLIED>

<!ATTLIST entry mode (p|e|s) #IMPLIED> <!-- "r" is implied -->

<!ATTLIST entry fpop (once|twice) #IMPLIED>

<!ATTLIST entry fpush (yes) #IMPLIED>

<!ATTLIST entry particular (yes) #IMPLIED> <!-- "no" is implied -->

<!-- sec\_opcd@escape="yes" indicates three-byte escapes -->

<!ATTLIST sec\_opcd escape (yes) #IMPLIED> <!-- "no" is implied -->

<!-- syntax@mod: similar to entry@mod, enables easier definition of -->

<!-- syntaxes when the instruction acts the same but uses different -->

<!-- syntaxes -->

<!ATTLIST syntax mod (nomem|mem) #IMPLIED>

<!ATTLIST note short IDREF #IMPLIED>

<!-- mnem@sug: author's suggestion in case of no oficial mnemonic -->

<!ATTLIST mnem sug (yes) #IMPLIED> <!-- "no" is implied -->

<!-- src and dst: -->

<!-- @nr: in case of direct register operand, its hardware number, if any -->

<!-- @group: in case of direct register operand, its group -->

<!-- @type: in case of direct register operand, its type code, if explicitly needed -->

<!-- @address: code for addressing method -->

<!-- @depend: does the resulting operand value depend on its previous value? -->

<!-- @displayed: is the implicit operand displayed within the syntax? -->

<!ATTLIST src nr NMTOKEN #IMPLIED>

<!ATTLIST src group (gen|mmx|xmm|seg|x87fpu|ctrl|systabp|msr|debug|xcr) #IMPLIED>

<!ATTLIST src type NMTOKEN #IMPLIED>

<!ATTLIST src address (BA|BB|F|I|X|Y|S2|S33|EST|SC) #IMPLIED>

<!ATTLIST src depend (no) #IMPLIED> <!-- "yes" is implied -->

<!ATTLIST src displayed (no) #IMPLIED> <!-- "yes" is implied -->

<!-- -->

<!ATTLIST dst nr NMTOKEN #IMPLIED>

<!ATTLIST dst group (gen|mmx|xmm|seg|x87fpu|ctrl|systabp|msr|debug|xcr) #IMPLIED>

<!ATTLIST dst type NMTOKEN #IMPLIED>

<!ATTLIST dst address (BD|F|X|Y|S2|S30|S33|EST|SC) #IMPLIED>

<!ATTLIST dst depend (no) #IMPLIED> <!-- "yes" is implied -->

<!ATTLIST dst displayed (no) #IMPLIED> <!-- "yes" is implied -->

<!-- test\_f, modif\_f, def\_f, and undef\_f -->

<!-- @cond: the flags apply only under further conditions -->

<!ATTLIST test\_f cond (yes) #IMPLIED> <!-- "no" is implied -->

<!ATTLIST modif\_f cond (yes) #IMPLIED> <!-- "no" is implied -->

<!ATTLIST def\_f cond (yes) #IMPLIED> <!-- "no" is implied -->

<!ATTLIST undef\_f cond (yes) #IMPLIED> <!-- "no" is implied -->