1. Name: SGDT/SIDT -- Store Global/Interrupt Descriptor Table Register

Usage: **SGDT mem (48 bit)**

SIDT mem (48 bit)

Description

SGDT/SIDT copies the contents of the descriptor table register the six bytes of memory indicated by the operand. The LIMIT field of the register is assigned to the first word at the effective address. If the operand-size attribute is 32 bits, the next three bytes are assigned the BASE field of the register, and the fourth byte is written with zero. The last byte is undefined. Otherwise, if the operand-size attribute is 16 bits, the next four bytes are assigned the 32-bit BASE field of the register.

SGDT and SIDT are used only in operating system software; they are not used in application programs.

Flags Affected: None

2. SLDT -- Store Local Descriptor Table Register

Usage: **SLDT reg/mem (16 bit)**

Description

SLDT stores the Local Descriptor Table Register (LDTR) in the two-byte register or memory location indicated by the effective address operand. This register is a selector that points into the Global Descriptor Table.

SLDT is used only in operating system software. It is not used in application programs.

Flags Affected: None

3. SMSW -- Store Machine Status Word

Usage: **SMSW reg/mem (16 bit)**

Description

SMSW stores the machine status word (part of CR0) in the two-byte register or memory location indicated by the effective address operand.

Flags Affected: None

4. LGDT/LIDT -- Load Global/Interrupt Descriptor Table Register

Usage: **LGDT mem (48 bit)**

LIDT mem (48 bit)

Description

The LGDT and LIDT instructions load a linear base address and limit value from a six-byte data operand in memory into the GDTR or IDTR, respectively. If a 16-bit operand is used with LGDT or LIDT, the register is loaded with a 16-bit limit and a 24-bit base, and the high-order eight bits of the six-byte data operand are not used. If a 32-bit operand is used, a 16-bit limit and a 32-bit base is loaded; the high-order eight bits of the six-byte operand are used as high-order base address bits.

The SGDT and SIDT instructions always store into all 48 bits of the six-byte data operand. With the 80286, the upper eight bits are undefined after SGDT or SIDT is executed. With the 80386, the upper eight bits are written with the high-order eight address bits, for both a 16-bit operand and a 32-bit operand. If LGDT or LIDT is used with a 16-bit operand to load the register stored by SGDT or SIDT, the upper eight bits are stored as zeros.

LGDT and LIDT appear in operating system software; they are not used in application programs. They are the only instructions that directly load a linear address (i.e., not a segment relative address) in 80386 Protected Mode.

Flags Affected: None

5. LLDT -- Load Local Descriptor Table Register

Usage: **LLDT reg/mem (16 bit)**

Description

LLDT loads the Local Descriptor Table register (LDTR). The word operand (memory or register) to LLDT should contain a selector to the Global Descriptor Table (GDT). The GDT entry should be a Local Descriptor Table. If so, then the LDTR is loaded from the entry. The descriptor registers DS, ES, SS, FS, GS, and CS are not affected. The LDT field in the task state segment does not change.

The selector operand can be 0; if so, the LDTR is marked invalid. All descriptor references (except by the LAR, VERR, VERW or LSL instructions) cause a #GP fault.

LLDT is used in operating system software; it is not used in application programs.

Flags Affected: None

6. LMSW -- Load Machine Status Word

Usage: **SMSW reg/mem (16 bit)**

Description

LMSW loads the machine status word (part of CR0) from the source operand. This instruction can be used to switch to Protected Mode; if so, it must be followed by an intrasegment jump to flush the instruction queue. LMSW will not switch back to Real Address Mode.

LMSW is used only in operating system software. It is not used in application programs.

Flags Affected: None

7. LTR -- Load Task Register

Usage: **LTR reg/mem (16 bit)**

Description Loads the source operand into the segment selector field of the task register.

Flags Affected: None

8. STR -- Store Task Register

Usage: **STR reg/mem (16 bit)**

Description Stores the segment selector field of the task register to operand.

Flags Affected: None

Key to Codes	
T	Instruction tests flag
M	Instruction modifies flag (either sets or resets depending on operands)
0	Instruction resets flag
—	Instruction's effect on flag is undefined
Blank	Instruction does not affect flag

Instruction	OF	SF	ZF	AF	PF	CF
AAS	—	—	—	TM	—	M
AAD	—	M	M	—	M	—
AAM	—	M	M	—	M	—
DAA	—	M	M	TM	M	TM
DAS	—	M	M	TM	M	TM
ADC	M	M	M	M	M	TM
ADD	M	M	M	M	M	M
SBB	M	M	M	M	M	TM
SUB	M	M	M	M	M	M
CMP	M	M	M	M	M	M
CMPS	M	M	M	M	M	M
SCAS	M	M	M	M	M	M
NEG	M	M	M	M	M	M
DEC	M	M	M	M	M	
INC	M	M	M	M	M	
IMUL	M	—	—	—	—	M
MUL	M	—	—	—	—	M
RCL/RCR 1	M					TM
RCL/RCR count	—					TM
ROL/ROR 1	M					M
ROL/ROR count	—					M
SAL/SAR/SHL/SHR 1	M	M	M	—	M	M
SAL/SAR/SHL/SHR count	—	M	M	—	M	M
SHLD/SHRD	—	M	M	—	M	M
BSF/BSR	—	—	M	—	—	—
BT/BTS/BTR/BTC	—	—	—	—	—	M
AND	0	M	M	—	M	0
OR	0	M	M	—	M	0
TEST	0	M	M	—	M	0
XOR	0	M	M	—	M	0

Data Transfer Instructions	
General purpose	
MOV	Move operand
PUSH	Push operand onto stack
POP	Pop operand off stack
PUSHA	Push all registers on stack
POPA	Pop all registers off stack
XCHG	Exchange operand, register
XLAT	Translate
Conversion	
MOVZX	Move Byte or Word, Dword, with zero extension
MOVSX	Move Byte or Word, Dword, sign extended
CBW	Convert Byte to Word, or Word to Dword
CDW	Convert Word to Dword
CDWE	Convert Word to Dword extended
CDQ	Convert Dword to Qword
Input/output	
IN	Input operand from I/O space
OUT	Output operand to I/O space
Address object	
LEA	Load effective address
LDS	Load pointer into D segment register
LES	Load pointer into E segment register
LFS	Load pointer into F segment register
LGS	Load pointer into G segment register
LSS	Load pointer into S (stack) segment register
Flag manipulation	
LAHF	Load AH register from flags
SAHF	Store AH register in flags
PUSHF	Push flags onto stack
POPF	Pop flags off stack
PUSHFD	Push Eflags onto stack
POPFD	Pop Eflags off stack
CLC	Clear carry flag
CLD	Clear direction flag
CMC	Complement carry flag
STC	Set carry flag
STD	Set direction flag

Arithmetic Instructions	
ddition	
ADD	Add operand
ADC	Add with carry
INC	Increment operand by 1
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition
ubtraction	
SUB	Subtract operand
SBB	Subtract with borrow
DEC	Decrement operand by 1
NEG	Negate operand
CMP	Compare operands
AAS	ASCII adjust for subtraction
lultiplication	
MUL	Multiply double/single precision
IMUL	Integer multiply
AAM	ASCII adjust after multiply
Division	
DIV	Divide unsigned
IDIV	Integer divide
AAD	ASCII adjust after division

String Instructions	
MOVS	Move Byte or Word, Dword string
INS	Input string from I/O space
OUTS	Output string to I/O space
CMPS	Compare Byte or Word, Dword string
SCAS	Scan Byte or Word, Dword string
LODS	Load Byte or Word, Dword string
STOS	Store Byte or Word, Dword string
REP	Repeat
REPE/REPZ	Repeat while equal/zero
RENE/REPNZ	Repeat while not equal/not zero

Logical Instructions	
Logicals	
NOT	"NOT" operand
AND	"AND" operand
OR	"Inclusive OR" operand
XOR	"Exclusive OR" operand
TEST	"Test" operand
Shifts	
SHL/SHR	Shift logical left or right
SAL/SAR	Shift arithmetic left or right
SHLD/SHRD	Double shift left or right
Rotates	
ROL/ROR	Rotate left/right
RCL/RCR	Rotate through carry left/right

Bit Manipulation Instructions	
Single bit instructions	
BT	Bit test
BTS	Bit test and set
BTR	Bit test and reset
BTC	Bit test and complement
BSF	Bit scan forward
BSR	Bit scan reverse
Bit string instructions	
IBTS	Insert bit string
XBTS	Exact bit string

Program Control Instructions	
Conditional transfers	
SETCC	Set byte equal to condition code
JA/JNBE	Jump if above/not below nor equal
JAE/JNB	Jump if above or equal/not below
JB/JNAE	Jump if below/not above nor equal
JBE/JNA	Jump if below or equal/not above
JC	Jump if carry
JE/JZ	Jump if equal/zero
JG/JNLE	Jump if greater/not less nor equal
JGE/JNL	Jump if greater or equal/not less
JL/JNGE	Jump if less/not greater nor equal
JLE/JNG	Jump if less or equal/not greater
JNC	Jump if not carry
JNE/JNZ	Jump if not equal/not zero
JNO	Jump if not overflow
JNP/JPO	Jump if not parity/parity odd
JNS	Jump if not sign
JO	Jump if overflow
JP/JPE	Jump if parity/parity even
JS	Jump if sign
Unconditional transfers	
CALL	Call procedure/task
RET	Return from procedure/task
JMP	Jump
Iteration controls	
LOOP	Loop
LOOPE/LOOPZ	Loop if equal/zero
LOOPNE/ LLOPNZ	Loop if not equal/not zero
JCXZ	JUMP if register CX = 0
Interrupts	
INT	Interrupt
INTO	Interrupt if overflow
IRET	Return from interrupt
CLI	Clear interrupt enable
STI	Set interrupt enable

High Level Language Instructions	
BOUND	Check array bounds
ENTER	Setup parameter block for entering procedure
LEAVE	Leave procedure

Protection Model	
SGDT	Store global descriptor table
SIDT	Store interrupt descriptor table
STR	Store task register
SLDT	Store local descriptor table
LGDT	Load global dedcriptor table
LIDT	Load interrupt descriptor table
LTR	Load task register
LLDT	Load local descriptor table
ARPL	Adjust requested privilege level
LAR	Load access rights
LSL	Load segment limit
VERR/VERW	Verify segment for reading or writing
LMSW	Load machine status word (lower 16 bits of CR0)
SMSW	Store machine status word
Processor Control Instructions	
HLT	Halt
WAIT	Wait until BUSY # negated
ESC	Escape
LOCK	Lock bus