

StrongForth 3.1 Glossary: assembler

. (data-type --)	asm.sf
Send the register attributes and the name of <code>data-type</code> as a character string plus a trailing space to the default output stream. If <code>data-type</code> is a reference, send 1st, 2nd, 3rd or <i>n</i> th, depending on the value <i>n</i> of the offset. An exception is thrown if <code>data-type</code> is not a valid data type.	
. (definition --)	asm.sf
Send the name, the stack diagram including register attributes and the attributes of <code>definition</code> to the default output stream.	
.instruction (--)	asm.sf
Send a disassembly of the assembler instruction starting at the disassembler's location counter to the default output stream. Advance the location counter to the next assembler instruction.	
.location (--)	asm.sf
Send the disassembler's location counter in an eight-digit hexadecimal format with a trailing colon and a space character to the default output stream.	
.params (address -> data-type unsigned --)	asm.sf
Send a list of unsigned data types starting at <code>address -> data-type</code> , including register, prefix and reference attributes, to the default output stream.	
.word (single --)	asm.sf
Send the 16-bit value <code>single</code> in a four-digit hexadecimal format with no trailing space to the default output stream.	
16-bit-address: (--)	asm.sf
Compile the next assembler instruction with a 16-bit address prefix.	
16-bit-operand: (--)	asm.sf
Compile the next assembler instruction with a 16-bit operand prefix.	
aaa, (--)	asm.sf
Compile an <code>aaa</code> assembler instruction.	
aad, (--)	asm.sf
Compile an <code>aad</code> assembler instruction.	

aad, (unsigned --) asm.sf

Compile an `aad` assembler instruction with `unsigned` being the number base.

aam, (--) asm.sf

Compile an `aam` assembler instruction.

aam, (unsigned --) asm.sf

Compile an `aam` assembler instruction with `unsigned` being the number base.

aas, (--) asm.sf

Compile an `aas` assembler instruction.

adc, (mode mode --) asm.sf

Compile an `adc` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

adc, (mode single --) asm.sf

Compile an `adc` assembler instruction with `mode` being the destination and `single` being the immediate source operand.

add, (mode mode --) asm.sf

Compile an `add` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

add, (mode single --) asm.sf

Compile an `add` assembler instruction with `mode` being the destination and `single` being the immediate source operand.

again, (destination-address --) asm.sf

Compile a `jmp` assembler instruction with `destination-address` being the jump destination.

ah (-- mode) asm.sf

`mode` is the register direct `ah` addressing mode.

ahead, (-- origin-address) asm.sf

Compile a `jmp` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

al (-- mode) asm.sf

mode is the register direct al addressing mode.

and, (mode mode --) asm.sf

Compile an and assembler instruction with the first mode being the destination and the second mode being the source.

and, (mode single --) asm.sf

Compile an and assembler instruction with mode being the destination and single being the immediate source operand.

arpl, (mode mode --) asm.sf

Compile an arpl assembler instruction with the first mode being the destination and the second mode being the source.

ax (-- mode) asm.sf

mode is the register direct ax addressing mode.

begin, (-- destination-address) asm.sf

destination-address is the next free location of the code space.

bh (-- mode) asm.sf

mode is the register direct bh addressing mode.

bl (-- mode) asm.sf

mode is the register direct bl addressing mode.

bound, (mode mode --) asm.sf

Compile a bound assembler instruction with the first mode being the array index and the second mode being the bounds operand.

bp (-- mode) asm.sf

mode is the register direct bp addressing mode.

bsf, (mode mode --) asm.sf

Compile a bsf assembler instruction with the first mode being the destination and the second mode being the source.

bsr, (mode mode --) asm.sf

Compile a `bsr` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

bswap, (mode --) asm.sf

Compile a `bswap` assembler instruction with `mode` being the destination register.

bt, (mode mode --) asm.sf

Compile a `bt` assembler instruction with the first `mode` being the bit string and the second `mode` being the bit offset in a register.

bt, (mode unsigned --) asm.sf

Compile a `bt` assembler instruction with `mode` being the bit string and `unsigned` being the immediate bit offset.

btc, (mode mode --) asm.sf

Compile a `btc` assembler instruction with the first `mode` being the bit string and the second `mode` being the bit offset in a register.

btc, (mode unsigned --) asm.sf

Compile a `btc` assembler instruction with `mode` being the bit string and `unsigned` being the immediate bit offset.

btr, (mode mode --) asm.sf

Compile a `btr` assembler instruction with the first `mode` being the bit string and the second `mode` being the bit offset in a register.

btr, (mode unsigned --) asm.sf

Compile a `btr` assembler instruction with `mode` being the bit string and `unsigned` being the immediate bit offset.

bts, (mode mode --) asm.sf

Compile a `bts` assembler instruction with the first `mode` being the bit string and the second `mode` being the bit offset in a register.

bts, (mode unsigned --) asm.sf

Compile a `bts` assembler instruction with `mode` being the bit string and `unsigned` being the immediate bit offset.

bx (-- mode) asm.sf

mode is the register direct bx addressing mode.

byte[eax] (-- mode) asm.sf

mode is the register indirect [eax] addressing mode for byte operands.

byte[eax]+ (single -- mode) asm.sf

mode is the register indirect with displacement [eax] + addressing mode for byte operands.
single is the displacement.

byte[ebp] (-- mode) asm.sf

mode is the register indirect [ebp] addressing mode for byte operands.

byte[ebp]+ (single -- mode) asm.sf

mode is the register indirect with displacement [ebp] + addressing mode for byte operands.
single is the displacement.

byte[ebx] (-- mode) asm.sf

mode is the register indirect [ebx] addressing mode for byte operands.

byte[ebx]+ (single -- mode) asm.sf

mode is the register indirect with displacement [ebx] + addressing mode for byte operands.
single is the displacement.

byte[ecx] (-- mode) asm.sf

mode is the register indirect [ecx] addressing mode for byte operands.

byte[ecx]+ (single -- mode) asm.sf

mode is the register indirect with displacement [ecx] + addressing mode for byte operands.
single is the displacement.

byte[edi] (-- mode) asm.sf

mode is the register indirect [edi] addressing mode for byte operands.

byte[edi]+ (single -- mode) asm.sf

mode is the register indirect with displacement [edi] + addressing mode for byte operands.
single is the displacement.

byte[edx] (-- mode)	asm.sf
mode is the register indirect [edx] addressing mode for byte operands.	
byte[edx]+ (single -- mode)	asm.sf
mode is the register indirect with displacement [edx]+ addressing mode for byte operands. single is the displacement.	
byte[esi] (-- mode)	asm.sf
mode is the register indirect [esi] addressing mode for byte operands.	
byte[esi]+ (single -- mode)	asm.sf
mode is the register indirect with displacement [esi]+ addressing mode for byte operands. single is the displacement.	
byte[esp] (-- mode)	asm.sf
mode is the register indirect [esp] addressing mode for byte operands.	
byte[esp]+ (single -- mode)	asm.sf
mode is the register indirect with displacement [esp]+ addressing mode for byte operands. single is the displacement.	
byte[] (single -- mode)	asm.sf
mode is the direct addressing mode for byte operands. single is the memory address.	
call, (address --)	asm.sf
Compile a call instruction to destination address.	
call, (mode --)	asm.sf
Compile a call instruction to destination mode.	
callf, (mode --)	asm.sf
Compile a far call instruction to destination mode.	
callf, (segment address --)	asm.sf
Compile a far call instruction to the destination specified by segment and address.	
cbw, (--)	asm.sf
Compile a cbw assembler instruction.	

cdq, (--)	asm.sf
Compile a <code>cdq</code> assembler instruction.	
ch (-- mode)	asm.sf
mode is the register direct <code>ch</code> addressing mode.	
cl (-- mode)	asm.sf
mode is the register direct <code>cl</code> addressing mode.	
clc, (--)	asm.sf
Compile a <code>clc</code> assembler instruction.	
cld, (--)	asm.sf
Compile a <code>cld</code> assembler instruction.	
cli, (--)	asm.sf
Compile a <code>cli</code> assembler instruction.	
clts, (--)	asm.sf
Compile a <code>clts</code> assembler instruction.	
cmc, (--)	asm.sf
Compile a <code>cmc</code> assembler instruction.	
cmova, (mode mode --)	asm.sf
Compile a <code>cmova</code> assembler instruction with the first mode being the destination register and the second mode being the source.	
cmovae, (mode mode --)	asm.sf
Compile a <code>cmovae</code> assembler instruction with the first mode being the destination register and the second mode being the source.	
cmovb, (mode mode --)	asm.sf
Compile a <code>cmovb</code> assembler instruction with the first mode being the destination register and the second mode being the source.	
cmovbe, (mode mode --)	asm.sf

Compile a `cmovbe` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

`cmovc, (mode mode --)` asm.sf

Compile a `cmovc` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

`cmove, (mode mode --)` asm.sf

Compile a `cmove` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

`cmovg, (mode mode --)` asm.sf

Compile a `cmovg` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

`cmovge, (mode mode --)` asm.sf

Compile a `cmovge` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

`cmovl, (mode mode --)` asm.sf

Compile a `cmovl` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

`cmovle, (mode mode --)` asm.sf

Compile a `cmovle` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

`cmovna, (mode mode --)` asm.sf

Compile a `cmovna` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

`cmovnae, (mode mode --)` asm.sf

Compile a `cmovnae` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

`cmovnb, (mode mode --)` asm.sf

Compile a `cmovnb` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmovnbe, (mode mode --) asm.sf

Compile a `cmovnbe` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmovnc, (mode mode --) asm.sf

Compile a `cmovnc` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmovne, (mode mode --) asm.sf

Compile a `cmovne` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmovng, (mode mode --) asm.sf

Compile a `cmovng` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmovnge, (mode mode --) asm.sf

Compile a `cmovnge` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmovnl, (mode mode --) asm.sf

Compile a `cmovnl` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmovnle, (mode mode --) asm.sf

Compile a `cmovnle` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmovno, (mode mode --) asm.sf

Compile a `cmovno` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmovnp, (mode mode --) asm.sf

Compile a `cmovnp` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmovns, (mode mode --) asm.sf

Compile a `cmovns` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmovnz, (mode mode --) asm.sf

Compile a `cmovnz` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmovo, (mode mode --) asm.sf

Compile a `cmovo` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmovp, (mode mode --) asm.sf

Compile a `cmovp` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmovpe, (mode mode --) asm.sf

Compile a `cmovpe` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmovpo, (mode mode --) asm.sf

Compile a `cmovpo` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmovs, (mode mode --) asm.sf

Compile a `cmovs` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmovz, (mode mode --) asm.sf

Compile a `cmovz` assembler instruction with the first `mode` being the destination register and the second `mode` being the source.

cmp, (mode mode --) asm.sf

Compile a `cmp` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

cmp, (mode single --) asm.sf

Compile a `cmp` assembler instruction with `mode` being the destination and `single` being the immediate source operand.

cmpsb, (--) asm.sf

Compile a `cmpsb` assembler instruction.

cmpsd, (--)	asm.sf
Compile a <code>cmpsd</code> assembler instruction.	
cmpsw, (--)	asm.sf
Compile a <code>cmpsw</code> assembler instruction.	
cmpxchg, (mode mode --)	asm.sf
Compile a <code>cmpxchg</code> assembler instruction with the first <code>mode</code> being the destination and the second <code>mode</code> being the source.	
cmpxchg8b, (mode --)	asm.sf
Compile a <code>cmpxchg8b</code> assembler instruction with <code>mode</code> being the destination.	
cpuid, (--)	asm.sf
Compile a <code>cpuid</code> assembler instruction.	
cr0 (-- creg)	asm.sf
<code>creg</code> is control register 0.	
cr1 (-- creg)	asm.sf
<code>creg</code> is control register 1.	
cr2 (-- creg)	asm.sf
<code>creg</code> is control register 2.	
cr3 (-- creg)	asm.sf
<code>creg</code> is control register 3.	
cr4 (-- creg)	asm.sf
<code>creg</code> is control register 4.	
creg (stack-diagram -- 1st)	asm.sf
When used in a stack diagram, specifies an input or output parameter with data type <code>creg</code> .	
cs (-- sreg)	asm.sf
<code>sreg</code> is segment register <code>cs</code> .	

cs: (--)	asm.sf
Compile a cs segment override instruction prefix.	
cwd, (--)	asm.sf
Compile a cwd assembler instruction.	
cwde, (--)	asm.sf
Compile a cwde assembler instruction.	
cx (-- mode)	asm.sf
mode is the register direct cx addressing mode.	
daa, (--)	asm.sf
Compile a daa assembler instruction.	
das, (--)	asm.sf
Compile a das assembler instruction.	
db, (single --)	
Reserve space for one byte in the code space and store the least significant byte of single in it. An exception is thrown if the code space overflows.	
dd, (single --)	
Reserve space for one cell in the code space and store single in the cell. An exception is thrown if the code space overflows.	
dec, (mode --)	asm.sf
Compile a dec assembler instruction with mode being the operand.	
destination-address (stack-diagram -- 1st)	asm.sf
When used in a stack diagram, specifies an input or output parameter with data type destination-address.	
dh (-- mode)	asm.sf
mode is the register direct dh addressing mode.	
di (-- mode)	asm.sf
mode is the register direct di addressing mode.	

disassemble (address --) asm.sf

Send an assembly code representation of the machine code instructions starting at `address` to the default output stream. The disassembly ends when a `ret` instruction is encountered that is not being skipped over by a conditional or unconditional jump instruction within the disassembled code.

div, (mode --) asm.sf

Compile a `div` assembler instruction with `mode` being the source operand.

dl (-- mode) asm.sf

`mode` is the register direct `dl` addressing mode.

dq, (double --)

Reserve space for two cells in the code space and store `double` in the two cells. An exception is thrown if the code space overflows.

dr0 (-- dreg) asm.sf

`dreg` is debug register 0.

dr1 (-- dreg) asm.sf

`dreg` is debug register 1.

dr2 (-- dreg) asm.sf

`dreg` is debug register 2.

dr3 (-- dreg) asm.sf

`dreg` is debug register 3.

dr4 (-- dreg) asm.sf

`dreg` is debug register 4.

dr5 (-- dreg) asm.sf

`dreg` is debug register 5.

dr6 (-- dreg) asm.sf

`dreg` is debug register 6.

dr7 (-- dreg)	asm.sf
dreg is debug register 7.	
dreg (stack-diagram -- 1st)	asm.sf
When used in a stack diagram, specifies an input or output parameter with data type dreg.	
ds (-- sreg)	asm.sf
sreg is segment register ds.	
ds: (--)	asm.sf
Compile a ds segment override instruction prefix.	
dw, (single --)	
Reserve space for two bytes in the code space and store the lower 16 bits of single in the two bytes. An exception is thrown if the code space overflows.	
dx (-- mode)	asm.sf
mode is the register direct dx addressing mode.	
ea (mode -- unsigned)	asm.sf
Splits a double-cell item mode into two single-cell items. unsigned is the most significant cell of mode, which specifies the effective address of an addressing mode.	
eax (-- mode)	asm.sf
mode is the register direct eax addressing mode.	
ebp (-- mode)	asm.sf
mode is the register direct ebp addressing mode.	
ebx (-- mode)	asm.sf
mode is the register direct ebx addressing mode.	
ecx (-- mode)	asm.sf
mode is the register direct ecx addressing mode.	
edi (-- mode)	asm.sf
mode is the register direct edi addressing mode.	

edx (-- mode) asm.sf

mode is the register direct edx addressing mode.

else, (origin-address -- 1st) asm.sf

Compile a `jmp` assembler instruction with a dummy jump destination. `1st` is a pointer to the code space location after this instruction. Resolve a forward jump by compiling the jump offset from `origin-address` to the current location of the code space.

endcode (code-definition --) asm.sf

Makes `code-definition` the definition most recently added to the current compilation vocabulary. Remove the `assembler` vocabulary from the context vocabulary list and make it the head of the hidden vocabulary list.

enter, (unsigned unsigned --) asm.sf

Compile an `enter` assembler instruction with the first `unsigned` being the size of the stack frame in bytes and the second `unsigned` being the nesting level.

es (-- sreg) asm.sf

`sreg` is segment register `es`.

es: (--) asm.sf

Compile an `es` segment override instruction prefix.

esi (-- mode) asm.sf

mode is the register direct `esi` addressing mode.

esp (-- mode) asm.sf

mode is the register direct `esp` addressing mode.

exit, (unsigned --)

Compile up to 6 `pop` instructions for registers `eax`, `edx`, `ebx`, `ecx`, `esi` and `edi`. A `pop` assembler instruction for a register is compiled if this register has been marked as preserved in the stack diagram of the current definition.

If `unsigned` is zero, compile a `nop` assembler instruction and a `ret` assembler instruction. If `unsigned` is not equal to zero, compile a `leave` assembler instruction and a `ret` assembler instruction.

f2xm1, (--) asm.sf

Compile a `f2xm1` assembler instruction.

fabs, (--) asm.sf

Compile a `fabs` assembler instruction.

fadd, (mode --) asm.sf

Compile a `fadd` assembler instruction with `st0` being the destination and `mode` being the source floating-point register.

fadd, (mode mode --) asm.sf

Compile a `fadd` assembler instruction with the first `mode` being the destination floating-point register and the second `mode` being the source floating-point register.

faddp, (--) asm.sf

Compile a `faddp` assembler instruction with `st1` being the destination and `st0` being the source.

faddp, (mode mode --) asm.sf

Compile a `faddp` assembler instruction with the first `mode` being the destination floating-point register and the second `mode` being the source floating-point register.

fbld, (mode --) asm.sf

Compile a `fbld` assembler instruction with `mode` being the source.

fbstp, (mode --) asm.sf

Compile a `fbstp` assembler instruction with `mode` being the destination.

fchs, (--) asm.sf

Compile a `fchs` assembler instruction.

fclex, (--) asm.sf

Compile a `fclex` assembler instruction.

fcmovb, (mode mode --) asm.sf

Compile a `fcmovb` assembler instruction with the first `mode` being the source and the second `mode` being the destination.

fcmovbe, (mode mode --) asm.sf

Compile a `fcmovbe` assembler instruction with the first `mode` being the source and the second `mode` being the destination.

fcmovbe, (mode mode --) asm.sf

Compile a `fcmovbe` assembler instruction with the first mode being the source and the second mode being the destination.

fcmovnb, (mode mode --) asm.sf

Compile a `fcmovnb` assembler instruction with the first mode being the source and the second mode being the destination.

fcmovnbe, (mode mode --) asm.sf

Compile a `fcmovnbe` assembler instruction with the first mode being the source and the second mode being the destination.

fcmovne, (mode mode --) asm.sf

Compile a `fcmovne` assembler instruction with the first mode being the source and the second mode being the destination.

fcmovnu, (mode mode --) asm.sf

Compile a `fcmovnu` assembler instruction with the first mode being the source and the second mode being the destination.

fcmovu, (mode mode --) asm.sf

Compile a `fcmovu` assembler instruction with the first mode being the source and the second mode being the destination.

fcom, (--) asm.sf

Compile a `fcom` assembler instruction with `st1` being the destination and `st0` being the source.

fcom, (mode --) asm.sf

Compile a `fcom` assembler instruction with mode being the destination and `st0` being the source.

fcom, (mode mode --) asm.sf

Compile a `fcom` assembler instruction with the first mode being the destination and the second mode being the source.

fcomi, (mode mode --) asm.sf

Compile a `fcomi` assembler instruction with the first mode being the destination and the second mode being the source.

fcomip, (mode mode --) asm.sf

Compile a `fcomip` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

fcomp, (--) asm.sf

Compile a `fcomp` assembler instruction with `st1` being the destination and `st0` being the source.

fcomp, (mode --) asm.sf

Compile a `fcomp` assembler instruction with `mode` as the destination and `st0` being the source.

fcomp, (mode mode --) asm.sf

Compile a `fcomp` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

fcompp, (--) asm.sf

Compile a `fcompp` assembler instruction with `st1` being the destination and `st0` being the source.

fcos, (--) asm.sf

Compile a `fcos` assembler instruction.

fdecstp, (--) asm.sf

Compile a `fdecstp` assembler instruction.

fdiv, (mode --) asm.sf

Compile a `fdiv` assembler instruction with `mode` being the destination and `st0` being the source.

fdiv, (mode mode --) asm.sf

Compile a `fdiv` assembler instruction with the first `mode` being the destination floating-point register and the second `mode` being the source floating-point register.

fdivp, (--) asm.sf

Compile a `fdivp` assembler instruction with `st1` being the destination and `st0` being the source.

fdivp, (mode mode --) asm.sf

Compile a `fdivp` assembler instruction with the first `mode` being the destination floating-point register and the second `mode` being the source floating-point register.

fdivr, (mode --)	asm.sf
Compile a <code>fdivr</code> assembler instruction with <code>mode</code> being the destination floating-point register and <code>st0</code> being the source.	
fdivr, (mode mode --)	asm.sf
Compile a <code>fdivr</code> assembler instruction with the first <code>mode</code> being the destination floating-point register and the second <code>mode</code> being the source floating-point register.	
fdivrp, (--)	asm.sf
Compile a <code>fdivrp</code> assembler instruction with <code>st1</code> being the destination and <code>st0</code> being the source.	
fdivrp, (mode mode --)	asm.sf
Compile a <code>fdivrp</code> assembler instruction with the first <code>mode</code> being the destination floating-point register and the second <code>mode</code> being the source floating-point register.	
ffree, (mode --)	asm.sf
Compile a <code>ffree</code> assembler instruction with <code>mode</code> being the affected floating-point register.	
fiadd, (mode --)	asm.sf
Compile a <code>fiadd</code> assembler instruction with <code>st0</code> being the destination and <code>mode</code> being the source.	
ficom, (mode --)	asm.sf
Compile a <code>ficom</code> assembler instruction with <code>st0</code> being the destination and <code>mode</code> being the source.	
ficomp, (mode --)	asm.sf
Compile a <code>ficomp</code> assembler instruction with <code>st0</code> being the destination and <code>mode</code> being the source.	
fidiv, (mode --)	asm.sf
Compile a <code>fidiv</code> assembler instruction with <code>st0</code> being the destination and <code>mode</code> being the source.	
fidivr, (mode --)	asm.sf
Compile a <code>fidivr</code> assembler instruction with <code>st0</code> being the destination and <code>mode</code> being the source.	
fild, (mode --)	asm.sf

Compile a `fild` assembler instruction with `mode` being the source.

`fimul, (mode --)`

asm.sf

Compile a `fiadd` assembler instruction with `st0` being the destination and `mode` being the source.

`fincstp, (--)`

asm.sf

Compile a `fincstp` assembler instruction.

`finit, (--)`

asm.sf

Compile a `finit` assembler instruction.

`fist, (mode --)`

asm.sf

Compile a `fist` assembler instruction with `mode` being the destination.

`fistp, (mode --)`

asm.sf

Compile a `fistp` assembler instruction with `mode` being the destination.

`fisttp, (mode --)`

asm.sf

Compile a `fisttp` assembler instruction with `mode` being the destination.

`fisub, (mode --)`

asm.sf

Compile a `fisub` assembler instruction with `st0` being the destination and `mode` being the source.

`fisubr, (mode --)`

asm.sf

Compile a `fisubr` assembler instruction with `st0` being the destination and `mode` being the source.

`fld, (mode --)`

asm.sf

Compile a `fld` assembler instruction with `mode` being the source.

`fld1, (--)`

asm.sf

Compile a `fld1` assembler instruction.

`fldcw, (mode --)`

asm.sf

Compile a `fldcw` assembler instruction with `mode` being the source.

fldenv, (mode --)	asm.sf
Compile a <code>fldenv</code> assembler instruction with <code>mode</code> being the source.	
fldenvw, (mode --)	
Compile a <code>fldenv</code> assembler instruction with <code>mode</code> being the 16-bit source.	
fldl2e, (--)	asm.sf
Compile a <code>fldl2e</code> assembler instruction.	
fldl2t, (--)	asm.sf
Compile a <code>fldl2t</code> assembler instruction.	
fldlg2, (--)	asm.sf
Compile a <code>fldlg2</code> assembler instruction.	
fldln2, (--)	asm.sf
Compile a <code>fldln2</code> assembler instruction.	
fldpi, (--)	asm.sf
Compile a <code>fldpi</code> assembler instruction.	
fldz, (--)	asm.sf
Compile a <code>fldz</code> assembler instruction.	
fmul, (mode --)	asm.sf
Compile a <code>fmul</code> assembler instruction with <code>mode</code> being the destination and <code>st0</code> being the source.	
fmul, (mode mode --)	asm.sf
Compile a <code>fmul</code> assembler instruction with the first <code>mode</code> being the destination floating-point register and the second <code>mode</code> being the source floating-point register.	
fmulp, (--)	asm.sf
Compile a <code>fmulp</code> assembler instruction with <code>st1</code> being the destination and <code>st0</code> being the source.	
fmulp, (mode mode --)	asm.sf
Compile a <code>fmulp</code> assembler instruction with the first <code>mode</code> being the destination floating-point register and the second <code>mode</code> being the source floating-point register.	

fncler , (--)	asm.sf
Compile a <code>fncler</code> assembler instruction.	
fninit , (--)	asm.sf
Compile a <code>fninit</code> assembler instruction.	
fnop , (--)	asm.sf
Compile a <code>fnop</code> assembler instruction.	
fnsave , (mode --)	asm.sf
Compile a <code>fnsave</code> assembler instruction with <code>mode</code> being the destination.	
fnsavew , (mode --)	asm.sf
Compile a <code>fnsave</code> assembler instruction with <code>mode</code> being the 16-bit destination.	
fnstcw , (mode --)	asm.sf
Compile a <code>fnstcw</code> assembler instruction with <code>mode</code> being the destination.	
fnstenv , (mode --)	asm.sf
Compile a <code>fnstenv</code> assembler instruction with <code>mode</code> being the destination.	
fnstenvw , (mode --)	asm.sf
Compile a <code>fnstenv</code> assembler instruction with <code>mode</code> being the 16-bit destination.	
fnstsw , (mode --)	asm.sf
Compile a <code>fnstsw</code> assembler instruction with <code>mode</code> being the 16-bit destination.	
fpatan , (--)	asm.sf
Compile a <code>fpatan</code> assembler instruction.	
fprem , (--)	asm.sf
Compile a <code>fprem</code> assembler instruction.	
fprem1 , (--)	asm.sf
Compile a <code>fprem1</code> assembler instruction.	

fptan, (--)	asm.sf
Compile a <code>fptan</code> assembler instruction.	
frndint, (--)	asm.sf
Compile a <code>frndint</code> assembler instruction.	
frstor, (mode --)	asm.sf
Compile a <code>frstor</code> assembler instruction with <code>mode</code> being the source.	
frstorw, (mode --)	asm.sf
Compile a <code>frstor</code> assembler instruction with <code>mode</code> being the 16-bit source.	
fs (-- sreg)	asm.sf
<code>sreg</code> is segment register <code>fs</code> .	
fs: (--)	asm.sf
Compile a <code>fs</code> segment override instruction prefix.	
fsave, (mode --)	asm.sf
Compile a <code>fsave</code> assembler instruction with <code>mode</code> being the destination.	
fscale, (--)	asm.sf
Compile a <code>fscale</code> assembler instruction.	
fsin, (--)	asm.sf
Compile a <code>fsin</code> assembler instruction.	
fsincos, (--)	asm.sf
Compile a <code>fsincos</code> assembler instruction.	
fsqrt, (--)	asm.sf
Compile a <code>fsqrt</code> assembler instruction.	
fst, (mode --)	asm.sf
Compile a <code>fst</code> assembler instruction with <code>mode</code> being the destination.	
fstcw, (mode --)	asm.sf

Compile a `fstcw` assembler instruction with `mode` being the destination.

`fstenv, (mode --)`

asm.sf

Compile a `fstenv` assembler instruction with `mode` being the destination.

`fstenvw, (mode --)`

asm.sf

Compile a `fstenvw` assembler instruction with `mode` being the 16-bit destination.

`fstop, (mode --)`

asm.sf

Compile a `fstop` assembler instruction with `mode` being the destination.

`fstsw, (mode --)`

asm.sf

Compile a `fstsw` assembler instruction with `mode` being the 16-bit destination.

`fsub, (mode --)`

asm.sf

Compile a `fsub` assembler instruction with `st0` being the destination and `mode` being the source floating-point register.

`fsub, (mode mode --)`

asm.sf

Compile a `fsub` assembler instruction with the first `mode` being the destination floating-point register and the second `mode` being the source floating-point register.

`fsubp, (--)`

asm.sf

Compile a `fsubp` assembler instruction with `st1` being the destination and `st0` being the source.

`fsubp, (mode mode --)`

asm.sf

Compile a `fsubp` assembler instruction with the first `mode` being the destination floating-point register and the second `mode` being the source floating-point register.

`fsubr, (mode --)`

asm.sf

Compile a `fsubr` assembler instruction with `mode` being the destination floating-point register and `st0` being the source.

`fsubr, (mode mode --)`

asm.sf

Compile a `fsubr` assembler instruction with the first `mode` being the destination floating-point register and the second `mode` being the source floating-point register.

`fsubrp, (--)`

asm.sf

Compile a `fsubrp` assembler instruction with `st1` being the destination and `st0` being the source.

`fsubrp, (mode mode --)`

asm.sf

Compile a `fsubrp` assembler instruction with the first `mode` being the destination floating-point register and the second `mode` being the source floating-point register.

`ftst, (--)`

asm.sf

Compile a `ftst` assembler instruction.

`fucom, (--)`

asm.sf

Compile a `fucom` assembler instruction with `st1` being the destination and `st0` being the source.

`fucom, (mode --)`

asm.sf

Compile a `fucom` assembler instruction with `mode` being the destination and `st0` being the source.

`fucomi, (mode mode --)`

asm.sf

Compile a `fucomi` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

`fucomip, (mode mode --)`

asm.sf

Compile a `fucomip` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

`fucomp, (--)`

asm.sf

Compile a `fucomp` assembler instruction with `st1` being the destination and `st0` being the source.

`fucomp, (mode --)`

asm.sf

Compile a `fucomp` assembler instruction with `mode` being the destination and `st0` being the source.

`fucompp, (--)`

asm.sf

Compile a `fucompp` assembler instruction with `st1` being the destination and `st0` being the source.

`fwait, (--)`

asm.sf

Compile a `fwait` assembler instruction.

fxam, (--) asm.sf

Compile a `fxam` assembler instruction.

fxch, (--) asm.sf

Compile a `fxch` assembler instruction with `st0` and `st1` being the two floating-point registers.

fxch, (mode --) asm.sf

Compile a `fxch` assembler instruction with `mode` and `st0` being the two floating-point registers.

fxch, (mode mode --) asm.sf

Compile a `fxch` assembler instruction with the first and the second `mode` being the two floating-point registers.

fxtract, (--) asm.sf

Compile a `fxtract` assembler instruction.

fyl2x, (--) asm.sf

Compile a `fyl2x` assembler instruction.

fyl2xp1, (--) asm.sf

Compile a `fyl2xp1` assembler instruction.

getsec, (--) asm.sf

Compile a `getsec` assembler instruction.

gs (-- sreg) asm.sf

`sreg` is segment register `gs`.

gs: (--) asm.sf

Compile a `gs` segment override instruction prefix.

hlt, (--) asm.sf

Compile a `hlt` assembler instruction.

idiv, (mode --) asm.sf

Compile an `idiv` assembler instruction with `mode` being the source operand.

ifa, (-- origin-address) asm.sf

Compile a jna assembler instruction with a dummy jump destination. origin-address is a pointer to the code space location after this instruction.

ifae, (-- origin-address) asm.sf

Compile a jnae assembler instruction with a dummy jump destination. origin-address is a pointer to the code space location after this instruction.

ifb, (-- origin-address) asm.sf

Compile a jnb assembler instruction with a dummy jump destination. origin-address is a pointer to the code space location after this instruction.

ifbe, (-- origin-address) asm.sf

Compile a jnbe assembler instruction with a dummy jump destination. origin-address is a pointer to the code space location after this instruction.

ifc, (-- origin-address) asm.sf

Compile a jnc assembler instruction with a dummy jump destination. origin-address is a pointer to the code space location after this instruction.

ife, (-- origin-address) asm.sf

Compile a jne assembler instruction with a dummy jump destination. origin-address is a pointer to the code space location after this instruction.

ifg, (-- origin-address) asm.sf

Compile a jng assembler instruction with a dummy jump destination. origin-address is a pointer to the code space location after this instruction.

ifge, (-- origin-address) asm.sf

Compile a jnge assembler instruction with a dummy jump destination. origin-address is a pointer to the code space location after this instruction.

ifl, (-- origin-address) asm.sf

Compile a jnl assembler instruction with a dummy jump destination. origin-address is a pointer to the code space location after this instruction.

ifle, (-- origin-address) asm.sf

Compile a jnle assembler instruction with a dummy jump destination. origin-address is a pointer to the code space location after this instruction.

ifna, (-- origin-address) asm.sf

Compile a `ja` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifnae, (-- origin-address) asm.sf

Compile a `jae` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifnb, (-- origin-address) asm.sf

Compile a `jb` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifnbe, (-- origin-address) asm.sf

Compile a `jbe` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifnc, (-- origin-address) asm.sf

Compile a `jc` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifncxz, (-- origin-address) asm.sf

Compile a `jcxz` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifne, (-- origin-address) asm.sf

Compile a `je` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifnecxz, (-- origin-address) asm.sf

Compile a `jecxz` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifng, (-- origin-address) asm.sf

Compile a `jg` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifnge, (-- origin-address) asm.sf

Compile a `jge` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifnl, (-- origin-address) asm.sf

Compile a `j1` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifnle, (-- origin-address) asm.sf

Compile a `j1e` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifno, (-- origin-address) asm.sf

Compile a `j0` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifnp, (-- origin-address) asm.sf

Compile a `j1p` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifns, (-- origin-address) asm.sf

Compile a `j1s` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifnz, (-- origin-address) asm.sf

Compile a `jz` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifo, (-- origin-address) asm.sf

Compile a `jno` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifp, (-- origin-address) asm.sf

Compile a `jnp` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifpe, (-- origin-address) asm.sf

Compile a `jpo` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifpo, (-- origin-address) asm.sf

Compile a `jpe` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifs, (-- origin-address) asm.sf

Compile a `jns` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

ifz, (-- origin-address) asm.sf

Compile a `jnz` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction.

imul, (mode --) asm.sf

Compile an `imul` assembler instruction with `mode` being the source operand.

imul, (mode mode --) asm.sf

Compile an `imul` assembler instruction with the first `mode` being the destination operand and the second `mode` being the source operand.

imul, (mode mode single --) asm.sf

Compile an `imul` assembler instruction with the first `mode` being the destination operand, the second `mode` being the first source operand and `single` being the second (immediate) source operand.

in, (mode mode --) asm.sf

Compile an `in` assembler instruction with the first `mode` specifying the destination register and the second `mode` specifying register `dx` as the port number. An exception is thrown if the second `mode` is not register `dx`.

in, (mode unsigned --) asm.sf

Compile an `in` assembler instruction with `mode` specifying the destination register and `unsigned` specifying the immediate port number.

inc, (mode --) asm.sf

Compile an `inc` assembler instruction with `mode` being the operand.

index[eax*2] (mode -- mode) asm.sf

`mode` is the base and scaled index addressing mode with the input parameter `mode` being the base and 2 times `[eax]` being the index. An exception is thrown if the input parameter `mode` is not a register indirect or a register indirect with displacement addressing mode.

index[eax*4] (mode -- mode) asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 4 times [eax] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[eax*8] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 8 times [eax] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[eax] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and [eax] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[ebp*2] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 2 times [ebp] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[ebp*4] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 4 times [ebp] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[ebp*8] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 8 times [ebp] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[ebp] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and [ebp] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[ebx*2] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 2 times [ebx] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[ebx*4] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 4 times [ebx] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[ebx*8] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 8 times [ebx] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[ebx] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and [ebx] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[ecx*2] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 2 times [ecx] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[ecx*4] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 4 times [ecx] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[ecx*8] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 8 times [ecx] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[ecx] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and [ecx] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[edi*2] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 2 times [edi] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[edi*4] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 4 times [edi] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[edi*8] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 8 times [edi] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[edi] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and [edi] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[edx*2] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 2 times [edx] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[edx*4] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 4 times [edx] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[edx*8] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 8 times [edx] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[edx] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and [edx] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[esi*2] (mode -- mode)

asm.sf

mode is the base and scaled index addressing mode with the input parameter mode being the base and 2 times [esi] being the index. An exception is thrown if the input parameter mode is not a register indirect or a register indirect with displacement addressing mode.

index[esi*4] (mode -- mode)

asm.sf

`mode` is the base and scaled index addressing mode with the input parameter `mode` being the base and 4 times `[esi]` being the index. An exception is thrown if the input parameter `mode` is not a register indirect or a register indirect with displacement addressing mode.

`index[esi*8] (mode -- mode)`

asm.sf

`mode` is the base and scaled index addressing mode with the input parameter `mode` being the base and 8 times `[esi]` being the index. An exception is thrown if the input parameter `mode` is not a register indirect or a register indirect with displacement addressing mode.

`index[esi] (mode -- mode)`

asm.sf

`mode` is the base and scaled index addressing mode with the input parameter `mode` being the base and `[esi]` being the index. An exception is thrown if the input parameter `mode` is not a register indirect or a register indirect with displacement addressing mode.

`insb, (--)`

asm.sf

Compile an `insb` assembler instruction.

`insd, (--)`

asm.sf

Compile an `insd` assembler instruction.

`insw, (--)`

asm.sf

Compile an `insw` assembler instruction.

`int, (unsigned --)`

asm.sf

Compile an `int` assembler instruction with `unsigned` being the interrupt vector.

`into, (--)`

asm.sf

Compile a `into` assembler instruction.

`invd, (--)`

asm.sf

Compile an `invd` assembler instruction.

`invlpg, (mode --)`

asm.sf

Compile an `invlpg` assembler instruction with `mode` being the source operand.

`iret, (--)`

asm.sf

Compile an `iret` assembler instruction.

`iretd, (--)`

asm.sf

Compile an `iretd` assembler instruction.

`ja, (address --)`

asm.sf

Compile a `ja` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jae, (address --)`

asm.sf

Compile a `jae` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jb, (address --)`

asm.sf

Compile a `jb` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jbe, (address --)`

asm.sf

Compile a `jbe` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jc, (address --)`

asm.sf

Compile a `jc` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jcxz, (address --)`

asm.sf

Compile a `jcxz` assembler instruction with `address` being the jump destination. An exception is thrown if the destination address is not within the range of a relative short jump.

`je, (address --)`

asm.sf

Compile a `je` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jecxz, (address --)`

asm.sf

Compile a `jecxz` assembler instruction with `address` being the jump destination. An exception is thrown if the destination address is not within the range of a relative short jump.

`jg, (address --)`

asm.sf

Compile a `jg` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jge, (address --)`

asm.sf

Compile a `jge` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`j1, (address --)`

asm.sf

Compile a `j1` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jle, (address --)`

asm.sf

Compile a `jle` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jmp, (address --)`

asm.sf

Compile an unconditional `jmp` instruction to destination `address`. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jmp, (mode --)`

asm.sf

Compile a `jmp` instruction to destination `mode`.

`jmpf, (mode --)`

asm.sf

Compile a far `jmp` instruction to destination `mode`.

`jmpf, (segment address --)`

asm.sf

Compile a far `jmp` instruction to the destination specified by `segment` and `address`.

`jna, (address --)`

asm.sf

Compile a `jna` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jnae, (address --)`

asm.sf

Compile a `jnae` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jnb, (address --)`

asm.sf

Compile a `jnb` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jnbe, (address --)`

asm.sf

Compile a `jnbe` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jnc, (address --)`

asm.sf

Compile a `jnc` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jne, (address --)`

asm.sf

Compile a `jne` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jng, (address --)`

asm.sf

Compile a `jng` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jnge, (address --)`

asm.sf

Compile a `jnge` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jnl, (address --)`

asm.sf

Compile a `jnl` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jnle, (address --)`

asm.sf

Compile a `jnle` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

jno, (address --) asm.sf

Compile a `jno` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

jnp, (address --) asm.sf

Compile a `jnp` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

jns, (address --) asm.sf

Compile a `jns` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

jnz, (address --) asm.sf

Compile a `jnz` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

jo, (address --) asm.sf

Compile a `jo` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

jp, (address --) asm.sf

Compile a `jp` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

jpe, (address --) asm.sf

Compile a `jpe` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

jpo, (address --) asm.sf

Compile a `jpo` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

js, (address --) asm.sf

Compile a `jz` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`jz, (address --)`

asm.sf

Compile a `jz` assembler instruction with `address` being the jump destination. If the destination address is within the range of a relative short jump, compile a relative short jump with a byte displacement. Otherwise, compile a relative near jump with a 32-bit displacement.

`lahf, (--)`

asm.sf

Compile a `lahf` assembler instruction.

`lar, (mode mode --)`

asm.sf

Compile a `lar` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

`lds, (mode mode --)`

asm.sf

Compile a `lds` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

`lea, (mode mode --)`

asm.sf

Compile a `lea` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

`leave, (--)`

asm.sf

Compile a `leave` assembler instruction.

`les, (mode mode --)`

asm.sf

Compile a `les` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

`lfs, (mode mode --)`

asm.sf

Compile a `lfs` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

`lgdt, (mode --)`

asm.sf

Compile a `lgdt` assembler instruction with `mode` being the source operand.

`lgs, (mode mode --)`

asm.sf

Compile a `lgs` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

`lidt, (mode --)` asm.sf

Compile a `lidt` assembler instruction with `mode` being the source operand.

`lldt, (mode --)` asm.sf

Compile a `lldt` assembler instruction with `mode` being the source operand.

`lmsw, (mode --)` asm.sf

Compile a `lmsw` assembler instruction with `mode` being the source operand.

`location (-- address -> address)` asm.sf

`address -> address` is the address of a cell containing the disassembler's location counter.

`lock (--)` asm.sf

Compile a `lock` assembler instruction prefix.

`lodsb, (--)` asm.sf

Compile a `lodsb` assembler instruction.

`lodsd, (--)` asm.sf

Compile a `lodsd` assembler instruction.

`lodsw, (--)` asm.sf

Compile a `lodsw` assembler instruction.

`loop, (address --)` asm.sf

Compile a `loop` assembler instruction with `address` being the jump destination. An exception is thrown if the destination address is not within the range of a relative short jump.

`loope, (address --)` asm.sf

Compile a `loope` assembler instruction with `address` being the jump destination. An exception is thrown if the destination address is not within the range of a relative short jump.

`loopne, (address --)` asm.sf

Compile a `loopne` assembler instruction with `address` being the jump destination. An exception is thrown if the destination address is not within the range of a relative short jump.

loopnz, (address --) asm.sf

Compile a `loopnz` assembler instruction with `address` being the jump destination. An exception is thrown if the destination address is not within the range of a relative short jump.

loopz, (address --) asm.sf

Compile a `loopz` assembler instruction with `address` being the jump destination. An exception is thrown if the destination address is not within the range of a relative short jump.

lsl, (mode mode --) asm.sf

Compile a `lsl` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

lss, (mode mode --) asm.sf

Compile a `lss` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

ltr, (mode --) asm.sf

Compile a `ltr` assembler instruction with `mode` being the source operand.

lzcnt, (mode mode --) asm.sf

Compile a `lzcnt` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

m16 (mode -- 1st) asm.sf

Specifies that `mode` refers to a 16-bit memory operand, returning `1st`. An exception is thrown if `mode` is a general-purpose register, a floating-point register or a byte operand.

m32 (mode -- 1st) asm.sf

Specifies that `mode` refers to a 32-bit memory operand, returning `1st`. An exception is thrown if `mode` is a general-purpose register, a floating-point register or a byte operand.

m64 (mode -- 1st) asm.sf

Specifies that `mode` refers to a 64-bit memory operand, returning `1st`. An exception is thrown if `mode` is a general-purpose register, a floating-point register or a byte operand.

m80 (mode -- 1st) asm.sf

Specifies that `mode` refers to a 80-bit memory operand, returning `1st`. An exception is thrown if `mode` is a general-purpose register, a floating-point register or a byte operand.

mode (stack-diagram -- 1st)	asm.sf
When used in a stack diagram, specifies an input or output parameter with data type mode.	
mov, (creg mode --)	asm.sf
Compile a mov assembler instruction with creg being the destination control register and mode being the source register.	
mov, (dreg mode --)	asm.sf
Compile a mov assembler instruction with dreg being the destination debug register and mode being the source register.	
mov, (mode creg --)	asm.sf
Compile a mov assembler instruction with mode being the destination register and creg being the source control register.	
mov, (mode dreg --)	asm.sf
Compile a mov assembler instruction with mode being the destination register and dreg being the source debug register.	
mov, (mode mode --)	asm.sf
Compile a mov assembler instruction with the first mode being the destination and the second mode being the source.	
mov, (mode single --)	asm.sf
Compile a mov assembler instruction with mode being the destination and single being the immediate source operand.	
mov, (mode sreg --)	asm.sf
Compile a mov assembler instruction with mode being the destination and sreg being the source segment register.	
mov, (sreg mode --)	asm.sf
Compile a mov assembler instruction with sreg being the destination segment register and mode being the source.	
movsb, (--)	asm.sf
Compile a movsb assembler instruction.	
movsd, (--)	asm.sf

Compile a `movsd` assembler instruction.

`movsw, (--)`

asm.sf

Compile a `movsw` assembler instruction.

`movsx, (mode mode --)`

asm.sf

Compile a `movsx` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

`movzx, (mode mode --)`

asm.sf

Compile a `movsx` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

`mul, (mode --)`

asm.sf

Compile a `mul` assembler instruction with `mode` being the source operand.

`neg, (mode --)`

asm.sf

Compile a `neg` assembler instruction with `mode` being the operand.

`nop, (--)`

asm.sf

Compile a `nop` assembler instruction.

`nop, (mode --)`

asm.sf

Compile a `nop` assembler instruction with `mode` being a dummy operand.

`not, (mode --)`

asm.sf

Compile a `not` assembler instruction with `mode` being the operand.

`op (mode -- unsigned)`

asm.sf

Splits a double-cell item `mode` into two single-cell items. `unsigned` is the least significant cell of `mode`, which specifies the operand of an addressing mode.

`or, (mode mode --)`

asm.sf

Compile an `or` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

`or, (mode single --)`

asm.sf

Compile an `or` assembler instruction with `mode` being the destination and `single` being the immediate source.

`origin-address (stack-diagram -- 1st)`

asm.sf

When used in a stack diagram, specifies an input or output parameter with data type `origin-address`.

`out, (mode mode --)`

asm.sf

Compile an `out` assembler instruction with the first `mode` specifying register `dx` as the port number and the second `mode` specifying the source register. An exception is thrown if the first `mode` is not register `dx`.

`out, (unsigned mode --)`

asm.sf

Compile an `out` assembler instruction with `unsigned` specifying the immediate port number and `mode` specifying the source register.

`outsb, (--)`

asm.sf

Compile an `outsb` assembler instruction.

`outsd, (--)`

asm.sf

Compile an `outsd` assembler instruction.

`outsw, (--)`

asm.sf

Compile an `outsw` assembler instruction.

`pop, (data-type --)`

Compile up to 6 `pop` assembler instructions for registers `eax`, `edx`, `ebx`, `ecx`, `esi` and `edi`. A `pop` assembler instruction for a register is compiled if the associated register attribute in `data-type` is set, otherwise nothing is compiled. The order of `pop` assembler instructions is given by the register list above.

`pop, (mode --)`

asm.sf

Compile a `pop` assembler instruction with `mode` being the destination.

`pop, (sreg --)`

asm.sf

Compile a `pop` assembler instruction with `sreg` being the destination segment register.

`popa, (--)`

asm.sf

Compile a `popa` assembler instruction.

popad, (--)	asm.sf
Compile a popad assembler instruction.	
popf, (--)	asm.sf
Compile a popf assembler instruction.	
popfd, (--)	asm.sf
Compile a popfd assembler instruction.	
push, (data-type --)	
Compile up to 6 push assembler instructions for registers edi, esi, ecx, ebx, edx and eax. A push assembler instruction for a register is compiled if the associated register attribute in data-type is set, otherwise nothing is compiled. The order of push assembler instructions is given by the register list above.	
push, (mode --)	asm.sf
Compile a push assembler instruction with mode specifying the source.	
push, (single --)	asm.sf
Compile a push assembler instruction with single specifying the immediate source operand.	
push, (sreg --)	asm.sf
Compile a push assembler instruction with sreg specifying the segment register.	
pusha, (--)	asm.sf
Compile a pusha assembler instruction.	
pushad, (--)	asm.sf
Compile a pushad assembler instruction.	
pushf, (--)	asm.sf
Compile a pushf assembler instruction.	
pushfd, (--)	asm.sf
Compile a pushfd assembler instruction.	
rcl, (mode mode --)	asm.sf

Compile a `rcl` assembler instruction with the first `mode` being the destination and the second `mode` being register `cl` as the count operand. An exception is thrown if the second `mode` is not register `cl`.

`rcl, (mode unsigned --)`

asm.sf

Compile a `rcl` assembler instruction with `mode` being the destination and `unsigned` being the immediate count operand.

`rcr, (mode mode --)`

asm.sf

Compile a `rcr` assembler instruction with the first `mode` being the destination and the second `mode` being register `cl` as the count operand. An exception is thrown if the second `mode` is not register `cl`.

`rcr, (mode unsigned --)`

asm.sf

Compile a `rcr` assembler instruction with `mode` being the destination and `unsigned` being the immediate count operand.

`rdmsr, (--)`

asm.sf

Compile a `rdmsr` assembler instruction.

`rdpmc, (--)`

asm.sf

Compile a `rdpmc` assembler instruction.

`rdtsc, (--)`

asm.sf

Compile a `rdtsc` assembler instruction.

`rep (--)`

asm.sf

Compile a `rep` assembler instruction prefix.

`repe (--)`

asm.sf

Compile a `repe` assembler instruction prefix.

`repeat, (origin-address destination-address --)`

asm.sf

Compile a `jmp` assembler instruction with `destination-address` being the jump destination. Resolve a forward jump by compiling the jump offset from `origin-address` to the current position of the code space.

`repne (--)`

asm.sf

Compile a `repne` assembler instruction prefix.

repnz (--) asm.sf

Compile a `repnz` assembler instruction prefix.

repz (--) asm.sf

Compile a `repz` assembler instruction prefix.

ret, (--) asm.sf

Compile a `ret` assembler instruction.

ret, (unsigned --) asm.sf

Compile a `ret` assembler instruction with `unsigned` being the number of stack bytes to be released.

retf, (--) asm.sf

Compile a far `ret` assembler instruction.

retf, (unsigned --) asm.sf

Compile a far `ret` assembler instruction with `unsigned` being the number of stack bytes to be released.

rol, (mode mode --) asm.sf

Compile a `rol` assembler instruction with the first `mode` being the destination and the second `mode` being register `cl` as the count operand. An exception is thrown if the second mode is not register `cl`.

rol, (mode unsigned --) asm.sf

Compile a `rol` assembler instruction with `mode` being the destination and `unsigned` being the immediate count operand.

ror, (mode mode --) asm.sf

Compile a `ror` assembler instruction with the first `mode` being the destination and the second `mode` being register `cl` as the count operand. An exception is thrown if the second mode is not register `cl`.

ror, (mode unsigned --) asm.sf

Compile a `ror` assembler instruction with `mode` being the destination and `unsigned` being the immediate count operand.

rsm, (--)	asm.sf
Compile a <code>rsm</code> assembler instruction.	
sahf, (--)	asm.sf
Compile a <code>sahf</code> assembler instruction.	
sal, (mode mode --)	asm.sf
Compile a <code>sal</code> assembler instruction with the first <code>mode</code> being the destination and the second <code>mode</code> being register <code>cl</code> as the count operand. An exception is thrown if the second mode is not register <code>cl</code> .	
sal, (mode unsigned --)	asm.sf
Compile a <code>sal</code> assembler instruction with <code>mode</code> being the destination and <code>unsigned</code> being the immediate count operand.	
sar, (mode mode --)	asm.sf
Compile a <code>sar</code> assembler instruction with the first <code>mode</code> being the destination and the second <code>mode</code> being register <code>cl</code> as the count operand. An exception is thrown if the second mode is not register <code>cl</code> .	
sar, (mode unsigned --)	asm.sf
Compile a <code>sar</code> assembler instruction with <code>mode</code> being the destination and <code>unsigned</code> being the immediate count operand.	
sbb, (mode mode --)	asm.sf
Compile a <code>sbb</code> assembler instruction with the first <code>mode</code> being the destination and the second <code>mode</code> being the source.	
sbb, (mode single --)	asm.sf
Compile a <code>sbb</code> assembler instruction with <code>mode</code> being the destination and <code>single</code> being the immediate source.	
scasb, (--)	asm.sf
Compile a <code>scasb</code> assembler instruction.	
scasd, (--)	asm.sf
Compile a <code>scasd</code> assembler instruction.	
scasw, (--)	asm.sf

Compile a `scasw` assembler instruction.

`segment (stack-diagram -- 1st)`

asm.sf

When used in a stack diagram, specifies an input or output parameter with data type `segment`.

`seta, (mode --)`

asm.sf

Compile a `seta` assembler instruction with `mode` being the destination operand.

`setae, (mode --)`

asm.sf

Compile a `setae` assembler instruction with `mode` being the destination operand.

`setb, (mode --)`

asm.sf

Compile a `setb` assembler instruction with `mode` being the destination operand.

`setbe, (mode --)`

asm.sf

Compile a `setbe` assembler instruction with `mode` being the destination operand.

`setc, (mode --)`

asm.sf

Compile a `setc` assembler instruction with `mode` being the destination operand.

`sete, (mode --)`

asm.sf

Compile a `sete` assembler instruction with `mode` being the destination operand.

`setg, (mode --)`

asm.sf

Compile a `setg` assembler instruction with `mode` being the destination operand.

`setge, (mode --)`

asm.sf

Compile a `setge` assembler instruction with `mode` being the destination operand.

`setl, (mode --)`

asm.sf

Compile a `setl` assembler instruction with `mode` being the destination operand.

`setle, (mode --)`

asm.sf

Compile a `setle` assembler instruction with `mode` being the destination operand.

`setna, (mode --)`

asm.sf

Compile a `setna` assembler instruction with `mode` being the destination operand.

setnae, (mode --)	asm.sf
Compile a <code>setnae</code> assembler instruction with <code>mode</code> being the destination operand.	
setnb, (mode --)	asm.sf
Compile a <code>setnb</code> assembler instruction with <code>mode</code> being the destination operand.	
setnbe, (mode --)	asm.sf
Compile a <code>setnbe</code> assembler instruction with <code>mode</code> being the destination operand.	
setnc, (mode --)	asm.sf
Compile a <code>setnc</code> assembler instruction with <code>mode</code> being the destination operand.	
setne, (mode --)	asm.sf
Compile a <code>setne</code> assembler instruction with <code>mode</code> being the destination operand.	
setng, (mode --)	asm.sf
Compile a <code>setng</code> assembler instruction with <code>mode</code> being the destination operand.	
setnge, (mode --)	asm.sf
Compile a <code>setnge</code> assembler instruction with <code>mode</code> being the destination operand.	
setnl, (mode --)	asm.sf
Compile a <code>setnl</code> assembler instruction with <code>mode</code> being the destination operand.	
setnle, (mode --)	asm.sf
Compile a <code>setnle</code> assembler instruction with <code>mode</code> being the destination operand.	
setno, (mode --)	asm.sf
Compile a <code>setno</code> assembler instruction with <code>mode</code> being the destination operand.	
setnp, (mode --)	asm.sf
Compile a <code>setnp</code> assembler instruction with <code>mode</code> being the destination operand.	
setns, (mode --)	asm.sf
Compile a <code>setns</code> assembler instruction with <code>mode</code> being the destination operand.	

setnz, (mode --) asm.sf

Compile a `setnz` assembler instruction with `mode` being the destination operand.

seto, (mode --) asm.sf

Compile a `seto` assembler instruction with `mode` being the destination operand.

setp, (mode --) asm.sf

Compile a `setp` assembler instruction with `mode` being the destination operand.

setpe, (mode --) asm.sf

Compile a `setpe` assembler instruction with `mode` being the destination operand.

setpo, (mode --) asm.sf

Compile a `setpo` assembler instruction with `mode` being the destination operand.

sets, (mode --) asm.sf

Compile a `sets` assembler instruction with `mode` being the destination operand.

setz, (mode --) asm.sf

Compile a `setz` assembler instruction with `mode` being the destination operand.

sgdt, (mode --) asm.sf

Compile a `sgdt` assembler instruction with `mode` being the destination operand.

shl, (mode mode --) asm.sf

Compile a `shl` assembler instruction with the first `mode` being the destination and the second `mode` being register `cl` as the count operand. An exception is thrown if the second `mode` is not register `cl`.

shl, (mode unsigned --) asm.sf

Compile a `shl` assembler instruction with `mode` being the destination and `unsigned` being the immediate count operand.

shld, (mode mode mode --) asm.sf

Compile a `shld` assembler instruction with the first `mode` being the destination, the second `mode` being the source and the third `mode` being register `cl` as the count operand. An exception is thrown if the second `mode` is not register `cl`.

shld, (mode mode unsigned --)	asm.sf
Compile a <code>shld</code> assembler instruction with the first <code>mode</code> being the destination, the second <code>mode</code> being the source and <code>unsigned</code> being the immediate count operand.	
shr, (mode mode --)	asm.sf
Compile a <code>shr</code> assembler instruction with the first <code>mode</code> being the destination and the second <code>mode</code> being register <code>cl</code> as the count operand. An exception is thrown if the second <code>mode</code> is not register <code>cl</code> .	
shr, (mode unsigned --)	asm.sf
Compile a <code>shr</code> assembler instruction with <code>mode</code> being the destination and <code>unsigned</code> being the immediate count operand.	
shrd, (mode mode mode --)	asm.sf
Compile a <code>shrd</code> assembler instruction with the first <code>mode</code> being the destination, the second <code>mode</code> being the source and the third <code>mode</code> being register <code>cl</code> as the count operand. An exception is thrown if the second <code>mode</code> is not register <code>cl</code> .	
shrd, (mode mode unsigned --)	asm.sf
Compile a <code>shrd</code> assembler instruction with the first <code>mode</code> being the destination, the second <code>mode</code> being the source and <code>unsigned</code> being the immediate count operand.	
si (-- mode)	asm.sf
<code>mode</code> is the register direct <code>si</code> addressing mode.	
sidt, (mode --)	asm.sf
Compile a <code>sidt</code> assembler instruction with <code>mode</code> being the destination operand.	
sldt, (mode --)	asm.sf
Compile a <code>sldt</code> assembler instruction with <code>mode</code> being the destination operand.	
smsw, (mode --)	asm.sf
Compile a <code>smsw</code> assembler instruction with <code>mode</code> being the destination operand.	
sp (-- mode)	asm.sf
<code>mode</code> is the register direct <code>sp</code> addressing mode.	
sreg (stack-diagram -- 1st)	asm.sf
When used in a stack diagram, specifies an input or output parameter with data type <code>sreg</code> .	

ss (-- sreg)	asm.sf
sreg is segment register ss.	
ss: (--)	asm.sf
Compile a ss segment override instruction prefix.	
st (-- mode)	asm.sf
mode is the floating point stack register direct st0 addressing mode.	
st0 (-- mode)	asm.sf
mode is the floating point stack register direct st0 addressing mode.	
st1 (-- mode)	asm.sf
mode is the floating point stack register direct st1 addressing mode.	
st2 (-- mode)	asm.sf
mode is the floating point stack register direct st2 addressing mode.	
st3 (-- mode)	asm.sf
mode is the floating point stack register direct st3 addressing mode.	
st4 (-- mode)	asm.sf
mode is the floating point stack register direct st4 addressing mode.	
st5 (-- mode)	asm.sf
mode is the floating point stack register direct st5 addressing mode.	
st6 (-- mode)	asm.sf
mode is the floating point stack register direct st6 addressing mode.	
st7 (-- mode)	asm.sf
mode is the floating point stack register direct st7 addressing mode.	
stc, (--)	asm.sf
Compile a stc assembler instruction.	

std, (--)	asm.sf
Compile a <code>std</code> assembler instruction.	
sti, (--)	asm.sf
Compile a <code>sti</code> assembler instruction.	
stosb, (--)	asm.sf
Compile a <code>stosb</code> assembler instruction.	
stosd, (--)	asm.sf
Compile a <code>stosd</code> assembler instruction.	
stosw, (--)	asm.sf
Compile a <code>stosw</code> assembler instruction.	
str, (mode --)	asm.sf
Compile a <code>str</code> assembler instruction with <code>mode</code> being the destination operand.	
sub, (mode mode --)	asm.sf
Compile a <code>sub</code> assembler instruction with the first <code>mode</code> being the destination and the second <code>mode</code> being the source.	
sub, (mode single --)	asm.sf
Compile a <code>sub</code> assembler instruction with <code>mode</code> being the destination and <code>single</code> being the immediate source.	
syscall, (--)	asm.sf
Compile a <code>syscall</code> assembler instruction.	
sysenter, (--)	asm.sf
Compile a <code>sysenter</code> assembler instruction.	
sysexit, (--)	asm.sf
Compile a <code>sysexit</code> assembler instruction.	
sysret, (--)	asm.sf
Compile a <code>sysret</code> assembler instruction.	

test, (mode mode --) asm.sf

Compile a `test` assembler instruction with the first `mode` being the first source operand and the second `mode` being the second source operand.

test, (mode single --) asm.sf

Compile a `test` assembler instruction with `mode` being the first source operand and `single` being the second (immediate) source operand.

then, (origin-address --) asm.sf

Resolve a forward jump by compiling the jump offset from `origin-address` to the current position of the code space.

tzcnt, (mode mode --) asm.sf

Compile a `tzcnt` assembler instruction with the first `mode` being the destination and the second `mode` being the source.

ud2, (--) asm.sf

Compile an `ud2` assembler instruction.

untila, (destination-address --) asm.sf

Compile a `jna` assembler instruction with `destination-address` being the jump destination.

untilae, (destination-address --) asm.sf

Compile a `jnae` assembler instruction with `destination-address` being the jump destination.

untilb, (destination-address --) asm.sf

Compile a `jnb` assembler instruction with `destination-address` being the jump destination.

untilbe, (destination-address --) asm.sf

Compile a `jnbe` assembler instruction with `destination-address` being the jump destination.

untilc, (destination-address --) asm.sf

Compile a `jnc` assembler instruction with `destination-address` being the jump destination.

untile, (destination-address --) asm.sf

Compile a `jne` assembler instruction with `destination-address` being the jump destination.

untilg, (destination-address --)	asm.sf
Compile a <code>jng</code> assembler instruction with <code>destination-address</code> being the jump destination.	
untilge, (destination-address --)	asm.sf
Compile a <code>jnge</code> assembler instruction with <code>destination-address</code> being the jump destination.	
untill, (destination-address --)	asm.sf
Compile a <code>jnl</code> assembler instruction with <code>destination-address</code> being the jump destination.	
untille, (destination-address --)	asm.sf
Compile a <code>jnle</code> assembler instruction with <code>destination-address</code> being the jump destination.	
untilna, (destination-address --)	asm.sf
Compile a <code>ja</code> assembler instruction with <code>destination-address</code> being the jump destination.	
untilnae, (destination-address --)	asm.sf
Compile a <code>jae</code> assembler instruction with <code>destination-address</code> being the jump destination.	
untilnb, (destination-address --)	asm.sf
Compile a <code>jnb</code> assembler instruction with <code>destination-address</code> being the jump destination.	
untilnbe, (destination-address --)	asm.sf
Compile a <code>jbe</code> assembler instruction with <code>destination-address</code> being the jump destination.	
untilnc, (destination-address --)	asm.sf
Compile a <code>jnc</code> assembler instruction with <code>destination-address</code> being the jump destination.	
untilncxz, (destination-address --)	asm.sf
Compile a <code>jcxz</code> assembler instruction with <code>destination-address</code> being the jump destination.	
untilne, (destination-address --)	asm.sf
Compile a <code>jne</code> assembler instruction with <code>destination-address</code> being the jump destination.	
untilnecxz, (destination-address --)	asm.sf

Compile a `jecz` assembler instruction with `destination-address` being the jump destination.

untilng, (destination-address --) asm.sf

Compile a `jg` assembler instruction with `destination-address` being the jump destination.

untilnge, (destination-address --) asm.sf

Compile a `jge` assembler instruction with `destination-address` being the jump destination.

untilnl, (destination-address --) asm.sf

Compile a `jle` assembler instruction with `destination-address` being the jump destination.

untilnle, (destination-address --) asm.sf

Compile a `jle` assembler instruction with `destination-address` being the jump destination.

untilno, (destination-address --) asm.sf

Compile a `jo` assembler instruction with `destination-address` being the jump destination.

untilnp, (destination-address --) asm.sf

Compile a `jp` assembler instruction with `destination-address` being the jump destination.

untilns, (destination-address --) asm.sf

Compile a `js` assembler instruction with `destination-address` being the jump destination.

untilnz, (destination-address --) asm.sf

Compile a `jz` assembler instruction with `destination-address` being the jump destination.

untilo, (destination-address --) asm.sf

Compile a `jno` assembler instruction with `destination-address` being the jump destination.

untilp, (destination-address --) asm.sf

Compile a `jnp` assembler instruction with `destination-address` being the jump destination.

untilpe, (destination-address --) asm.sf

Compile a `jpo` assembler instruction with `destination-address` being the jump destination.

untilpo, (destination-address --) asm.sf

Compile a `jpe` assembler instruction with `destination-address` being the jump destination.

untils, (destination-address --) asm.sf

Compile a `jns` assembler instruction with `destination-address` being the jump destination.

untilz, (destination-address --) asm.sf

Compile a `jnz` assembler instruction with `destination-address` being the jump destination.

verr, (mode --) asm.sf

Compile a `verr` assembler instruction with `mode` being the source operand.

verw, (mode --) asm.sf

Compile a `verw` assembler instruction with `mode` being the source operand.

wait, (--) asm.sf

Compile a `wait` assembler instruction.

wbinvd, (--) asm.sf

Compile a `wbinvd` assembler instruction.

whilea, (destination-address -- origin-address 1st) asm.sf

Compile a `jna` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whileae, (destination-address -- origin-address 1st) asm.sf

Compile a `jnae` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whileb, (destination-address -- origin-address 1st) asm.sf

Compile a `jnb` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whilebe, (destination-address -- origin-address 1st) asm.sf

Compile a `jnbbe` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whilec, (destination-address -- origin-address 1st) asm.sf

Compile a `jnc` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

`whilee, (destination-address -- origin-address 1st)` asm.sf

Compile a `jne` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

`whileg, (destination-address -- origin-address 1st)` asm.sf

Compile a `jng` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

`whilege, (destination-address -- origin-address 1st)` asm.sf

Compile a `jnge` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

`whilel, (destination-address -- origin-address 1st)` asm.sf

Compile a `jnl` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

`whilele, (destination-address -- origin-address 1st)` asm.sf

Compile a `jnle` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

`whilena, (destination-address -- origin-address 1st)` asm.sf

Compile a `ja` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

`whilenae, (destination-address -- origin-address 1st)` asm.sf

Compile a `jae` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

`whilenb, (destination-address -- origin-address 1st)` asm.sf

Compile a `jnb` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

`whilenbe, (destination-address -- origin-address 1st)` asm.sf

Compile a `jbe` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whilenc, (destination-address -- origin-address 1st) asm.sf

Compile a `jc` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whilencxz, (destination-address -- origin-address 1st) asm.sf

Compile a `jcxz` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whilene, (destination-address -- origin-address 1st) asm.sf

Compile a `je` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whilenecxz, (destination-address -- origin-address 1st) asm.sf

Compile a `jecxz` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whileng, (destination-address -- origin-address 1st) asm.sf

Compile a `jg` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whilenge, (destination-address -- origin-address 1st) asm.sf

Compile a `jge` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whilenl, (destination-address -- origin-address 1st) asm.sf

Compile a `j1` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whilenle, (destination-address -- origin-address 1st) asm.sf

Compile a `jle` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whileno, (destination-address -- origin-address 1st) asm.sf

Compile a `j0` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whilenp, (destination-address -- origin-address 1st) asm.sf

Compile a `jp` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whilens, (destination-address -- origin-address 1st) asm.sf

Compile a `js` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whilenz, (destination-address -- origin-address 1st) asm.sf

Compile a `jz` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whileo, (destination-address -- origin-address 1st) asm.sf

Compile a `jno` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whilep, (destination-address -- origin-address 1st) asm.sf

Compile a `jnp` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whilepe, (destination-address -- origin-address 1st) asm.sf

Compile a `jpo` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whilepo, (destination-address -- origin-address 1st) asm.sf

Compile a `jpe` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whiles, (destination-address -- origin-address 1st) asm.sf

Compile a `jns` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

whilez, (destination-address -- origin-address 1st) asm.sf

Compile a `jnz` assembler instruction with a dummy jump destination. `origin-address` is a pointer to the code space location after this instruction. `1st` is `destination-address`.

wrmsr, (--) asm.sf

Compile a `wrmsr` assembler instruction.

xadd, (mode mode --) asm.sf

Compile a `xadd` assembler instruction with the first mode being the destination and the second mode being the source.

xchg, (mode mode --)	asm.sf
Compile a <code>xchg</code> assembler instruction with the first <code>mode</code> being the first operand and the second <code>mode</code> being the second operand.	
xlat, (--)	asm.sf
Compile a <code>xlat</code> assembler instruction.	
xlatb, (--)	asm.sf
Compile a <code>xlatb</code> assembler instruction.	
xor, (mode mode --)	asm.sf
Compile a <code>xor</code> assembler instruction with the first <code>mode</code> being the destination and the second <code>mode</code> being the source.	
xor, (mode single --)	asm.sf
Compile a <code>xor</code> assembler instruction with <code>mode</code> being the destination and <code>single</code> being the immediate source.	
[eax] (-- mode)	asm.sf
<code>mode</code> is the register indirect <code>[eax]</code> addressing mode.	
[eax]+ (single -- mode)	asm.sf
<code>mode</code> is the register indirect with displacement <code>[eax]+disp</code> addressing mode. <code>single</code> is the displacement.	
[ebp] (-- mode)	asm.sf
<code>mode</code> is the register indirect <code>[ebp]</code> addressing mode.	
[ebp]+ (single -- mode)	asm.sf
<code>mode</code> is the register indirect with displacement <code>[ebp]+disp</code> addressing mode for. <code>single</code> is the displacement.	
[ebx] (-- mode)	asm.sf
<code>mode</code> is the register indirect <code>[ebx]</code> addressing mode.	
[ebx]+ (single -- mode)	asm.sf
<code>mode</code> is the register indirect with displacement <code>[ebx]+disp</code> addressing mode. <code>single</code> is the displacement.	

[ecx] (-- mode) asm.sf

mode is the register indirect [ecx] addressing mode.

[ecx]+ (single -- mode) asm.sf

mode is the register indirect with displacement [ecx]+disp addressing mode. single is the displacement.

[edi] (-- mode) asm.sf

mode is the register indirect [edi] addressing mode.

[edi]+ (single -- mode) asm.sf

mode is the register indirect with displacement [edi]+disp addressing mode. single is the displacement.

[edx] (-- mode) asm.sf

mode is the register indirect [edx] addressing mode.

[edx]+ (single -- mode) asm.sf

mode is the register indirect with displacement [edx]+disp addressing mode. single is the displacement.

[esi] (-- mode) asm.sf

mode is the register indirect [esi] addressing mode.

[esi]+ (single -- mode) asm.sf

mode is the register indirect with displacement [esi]+disp addressing mode. single is the displacement.

[esp] (-- mode) asm.sf

mode is the register indirect [esp] addressing mode.

[esp]+ (single -- mode) asm.sf

mode is the register indirect with displacement [esp]+disp addressing mode. single is the displacement.

[] (single -- mode) asm.sf

mode is the indirect addressing mode. single is the memory address.