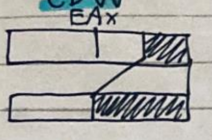
MOV - ...
MOVZX - MOV zero extend
MOVSX - MOV sign extend

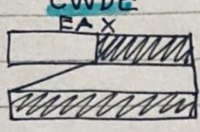
ADD a, b - a+=b
ADC a, b - a+=sign_extend(b)
SUB a, b - a-=b
SBB a, b - a-=sign_extend(b)

INC a - a++
DEC a - a--
NEG a - a=-a

CBW



CWDE



CWD
AX → DX:AX

CDQ
EAX → EDX:EAX

XCHG a, b - vymění 'a a b'

AND/OR/NOT/XOR

AND a, b	a &= b
OR a, b	a |= b
NOT a	a = ~a
XOR a, b	a ^= b

TEST
a&b, pouze nastaví příznaky

CMP
a-b, pouze nastaví příznaky

MUL
MUL 8bit - AL → AX
MUL 16bit - AX → DX:AX
MUL 32bit - EAX → EDX:EAX
D - horní část
A - dolní část

PUSH
- Implicitně 32 bit
PUSH a - SUB ESP, 4
MOV [ESP], a

POP
POP a - MOV a, [esp]
ADD esp, 4

IMUL
IMUL a, b - a *= sign_ext(b)
IMUL a, b, c a = b * sign_extend(c)

CALL
- Zavolá funkci

RET
Ret n - ADD esp, n

ENTER
ENTER n, 0
- Vytvoří o n bajtů stackframe

LEAVE
- Uklidí stack-frame

DIV
DIV x 8bit - AL = AX/X
CBW AH = AX%X
DIV x 16bit - AX = (DX:AX)/X
CWD DX = (DX:AX)/X
DIV x 32bit - EAX = (EDX:EAX)/X
CDQ EDX = (EDX:EAX)%X

REPCC
REP - while(--ECX) > 0
REPE - REP while equal
REPZ - REP while zero
REPNE - REP while not equal
REPNZ - REP while not zero

LEA
LEA a, [b]
- Nahraje adresu 'b' do 'a'

Skoky
JMP - nepodmíněný

Unsigned		Signed	
J(N)E	==	J(N)E	==
A	>	G	>
B	<	L	<
AE	>=	GE	>=
BE	<=	LE	<=

! A pro floaty

LOOPCC
LOOP ... JMP while (--ECX) > 0
LOOPE and equal
LOOPNE and not equal

SHL/SHR
-Posun s nulami
00 |0|1|1|0|1|0| 0 0 0 ...
L ⟷ R

SAL/SAR
- Jako SHc
ale se
znaménkem

RCR/RCL
-Rotate
trough
carry

ROR/ROL
-Rotate

MOVSc (B|W|D)
ESI → EDI
a inkrement obou
DF = 0 - normální směr
DF = 1 - opačný směr

CMPSc (B|W|D)
Porovnává
ESI ⟷ EDI
a inkrement
DF=0 - normální směr
DF=1 - opačný směr

SCASc (B|W|D)
Hledá znak uložený
v EAX v EDI

LODSc(B|W|D)
ESI → EAX
(a inkrement)

STOSc (B|W|D)
Kopíruje z EAX do
EDI (a inkrement)

STD - DF = 1
CLD - DF = 0 (nutné zavolat
před ukončením)

Stackframe

PUSH ebp
MOV ebp, esp
sub esp, local_data_size

ENTER local-data_size

POP ebp

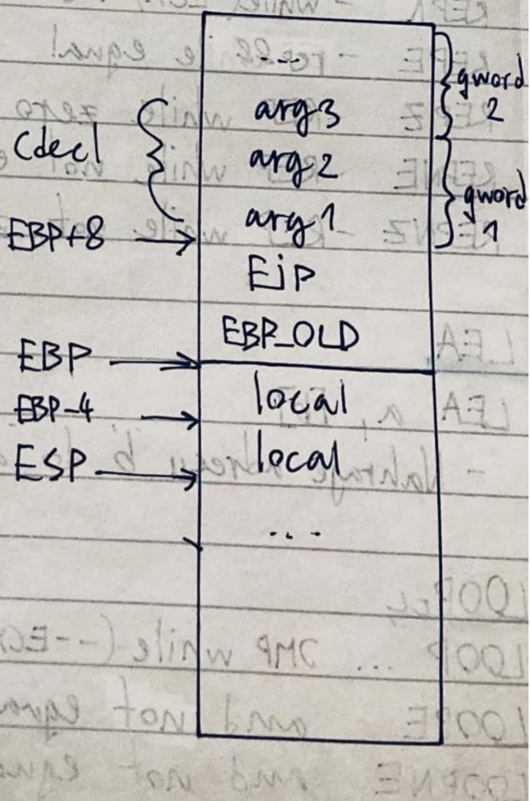
leave

pascal (Func)
- Zleva doprava,
uklízí volaný

fastcall
ECX, EDX, zbytek
zprava doleva
-uklízí volaný

cdecl (_func)
-Zprava doleva,
uklízí volající

stdcall (_func@4)
-Zprava doleva,
uklízí volaný



FLD
- Načte float do st0

FILD
Načte int do st0

FINIT
- Připraví FPU
V testu nevolat

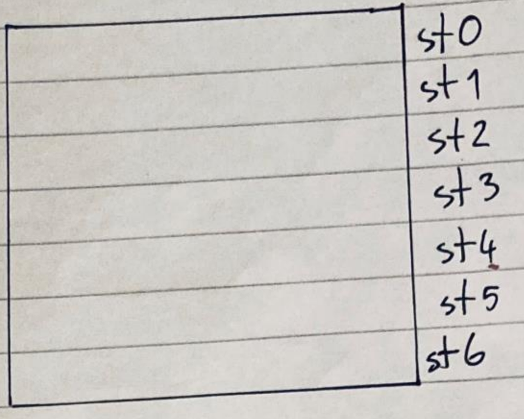
Konstanty.

FLD1	1.0
FLDPi	π
FLDZ	0.0

FST(P)
- Save float (and pop)

FIST(P)
- Save int (and pop)

FXCH
- Prohodí obsah st0 a st(i) (st0 a st1 default)



F(i)ADD(P)
- Add float/int (and pop)
st(0) += st(1)

F(i)SUB(R)(P)
- Sub float/int (and pop) (Reverse)
st(0) -=

FABS	st(i) = |st(i)|
FCHS	st(i) = - st(i)
FSQRT	$st(0) = \sqrt{st(0)}$
FSCALE	$st(0) = (st(0))^2$
FSIN	st0 = sin(st0)
FCOS	st0 = cos(st0)
FSINCOS	cos do st0 a sin do st1

F(I)MUL(P)
- Mul float/int (and pop)

st(0) *= st(1)

F(i)DIV(R)
- Div float/int (and pop)

st(0) /= st(1)

FCOM(P)(P)
- Porovná (a popne 1x/2x)

FCOMi(P) ! Porovnávání unsigned
- Nahraje do FLAGS registru

FICOM(P)
- Porovná int s st0

FTST
FSTSW ax
SAHF

FTST
- Porovná st0 s nulou

FXAM
- Zjistí typ čísla

FSTSW
- Uloží stavové slovo FPU do EAX → SAHF (Pošle EAX do FLAGS)

```
PUSH eax
FST dword [esp]
```
- Nahraje st0 do eax

```
sub esp, 8
FST qword [esp]
```
- Nahraje double jako argument

! Pozor, add je větší