Assembler

≤3 (if(2) приёмник, src)
[метка:] код [операнды][; комментарии]

ФА - физический адрес

режимы ЦП
1. реальный (i8086)
2. защищенный
3. виртуальный i8086 (V86)

▷ - сегмент
🔊 - прерывание
Д - данные
АП - адрес памяти
А - адрес
П - память
У[/У] - указател[ь/и]
К - команда
Р - регистр
🖫 - режим

[Bb / bB] - [ст./мл.] бит

Общего назначения (АЛУ) РОН
для save:
1) операндов лог. и ариф. операций
2) компонентов А.
3) УУ. на ячейки

Сегментные Ps (0..15)
Save АА. сегментов в П.
c - кода (А. к ▷с машинными КК)
d - ДД. (к тек. ПО)
s - стека
[e/g/f] - доп. ▷ Д.
(при use - явно указывать их АА.)

ePx
Px
Ph Pl
31 15 7 0

a / d - аккумулятор / Д. (save промежуточных Д.)
b - базовый (save базового А. некоторого объекта в П.)
c - счётчик (повторов)

eP
P
31 15 0

для 🔗КК.
[s/d]i - тек. А. ([источ/приём]ника)
стек
bp - вершина стека (в тек. сегменте стека)
sp - У. базы кадра стека

🔗К. - цепочная К.
fnS[b/w/d] [оп[,оп2]] У [lod / sto / sca]s нет не 🔗К.
mov es:(e)di, ds:(e)si
cmp (r: 🖫)
sca es:(e)di - 🔗 с
sto - save из al/(e)ax
lod ds:(e)si - load в
i/o **in** es:(e)di, dx
out dx, ds:(e)si

rep[|n][e/z] - повтор 🔗К. (e)cx раз
df=[0/1] [или zf=[0|1]] ([x. / для] **cmp/sca**)
(e)[s/d]i [+/-]= ✎ элемента 🔗

i/o
in al/(e)ax, порт - ввод из порта i/o.
out порт, al/(e)ax - вывод в порт i/o.
порт - цифра8/dx

Состояния и управления

[e]ip - ⇥ до next команды (от **cs**)(недоступен)

Системные регистры

lahf [E]Flags 7..0 → ah

EFlags (0..31)

31 ...	21	20	19	18	17	16	15	14	12..13	11	10	9	8	7	6	5	4	3	2	1	0
0	ID	VIP	VIF	AC	VM	RF	0	NT	IOPL	OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	CF

Flags (0..15)

состояния (итог ариф./лог. КК)
0 **CF** - переноса (из [31/15/7] бита)
1 1
(num '1')%2 2 **PF** - чётности (для 8 мл. бита)
3 0
4 **AF** - переноса$_{10}$ (т.е. для BCD)
5 0
0 6 **ZF** - ==0
(-) 7 **SF** - знака ([31/15/7] бит)
11 **OF** - переполнения (из [31/...] бита)

use в защ. режиме
1[2/3] **IOPL** - level доступа I/O
14 **NT** - вложенности задачи

control (для 🔗КК.)
10 **DF** - направления
[0/1] - А.[++/--] (т.е. к [старшему/мл.])

[CL/ST] n set [0/1] для nF
0,10,9) **C**, **D**, **I**

системные
8 **TF** - трассировки (пошагово 🔊 '1')
9 **IF** - разрешение аппаратных 🔊 (🔊 по входу INTR)
15 0
16 **RF** - возобновления (use при обработке 🔊 от рег. отладки)
17 **VM** - режим вирт. проц. 8086 (else: в реальном или защищённом)
18 **AC** - check 🔔 (при обращениях к памяти)
(и в определённых условиях (Н. в v-режиме))

use вместе
19 **VIF** - вирт. 🔊 (аналог IF)
20 **VIP** - отложенного вирт. 🔊

21 **ID** - идентификации (для check поддержки инструкции CPUID)
(т.е. ПО can ✎ этот флаг)
22..31 0

Режимы адресации

по способу set
1. Регистровая (**H:** mov ax, bx)
2. Непосредственная (**H:** mov ax, 1234h)

1. Прямая - нет доп. src/P
1) относительная - ⇥ от Р. [E]IP (в КК. перехода) **H:** jc m1 / m1:
2) абсолютная - ⇥ из val.
H: mov ax, [x1 ; =ds:0000 / word **ptr** [0000]]

2. Косвенная
со ⇥ [dx+3]
базовая (ds:P) [dx]; ds:dx
P [+ s/d i+val] [⇥]
+ индексная tab[si+1]
со ⇥ [si][dx]

пример set ▷ Д.
data segment
x1 dw 5; ФА=ds:0000
tab1 db 5h, 0Ah, 3h
data ends
mov ax data
mov ds, ax

set[n]c оп - set **1/0**.
'c' как в **j[n]**c

>> Сдвига

l (Pr) rDsc, У. src
x.**cs**

lea rDsc, У. src
эффек. А. ⇥

код оп1, num ⇥
H: shl, 4
shl cf 7 6 ... 1 0 ← 0
shr 0 → 7 6 ... 1 0 cf
sar знак
rol cf 7 6 ... 1 0
ror 0 1 ... 6 7
rcl cf 7 6 ... 1 0
rcr 0 1 ... 6 7

код оп1, оп2, num ⇥
(2-ая точность)
shld cf ... 0 ... 0
shrd 0 оп1 0 оп2

[o/s/z/a/p/c]f
x. rc[l/r]: af=?

shr: 'old Bb'
др: (⇥=1 и [sar]) ? [(cf ✎?)] : [?/0]

Битовые 10010
bs[f/r] оп1, оп2
r: id самой [пра/ле]вой '1'
zf=(have '1')
bt[c|r|s] src, id
1. **cf** = src.id
2. src.id = [!val | 1 | 0]
🖫: **of/sf/zf/af/pf/cf**
(p.194)

Логические
[**and | test**] / [**x**]**or** оп, 🖫 - [И [|(✗)] / ИЛИ]
🖫:
[c/o]f=0 **[z/s/p]f**=r **af**=?
not оп - НЕ (🖫✗)

Переход

куда - rA/метка

Безусловный
jmp[short ptr] куда
кроткий
(≤ 35÷40 КК.(127 байт))

Условный
cmp оп1, оп2 - сравнение (только 🖫)

переход
j[n]c куда
с/без знака: **e** - =
[g / a][e] - >
[l / b] - <
[=]
по flags
c p z
s o - cf

j[e]cxz метка - **[e]cx**=0

формат=**ds**
id (x.**[e]sp**)
(Pr) : [РОН + РОН * **1/2/4/8** + ⇥]
база

loop[|n][e/z] метка ([e]cx)
1. --[e]cx
2. if([e]cx>0 [и zf=[1|0]])
на метку (-128..+127 байт)

Ариф. операции

of/sf/zf/af/pf/cf=r
[**inc/dec**] - [++/--] не r. **cf**
⊕ **ad[d/c]** - rОп1 +/- (оп2 [+ cf])
⊖ **s[u/b]b**
! **neg** rОп **cf**=оп

оп1
[i]**mul** [r, |] оп1, оп2 [|r]
оп2: al/ax/[e/dx:]ax
r: ax / dx:ax / edx:eax
i - с учётом (-)

[i]**div** оп - РОН(ax / dx:ax / edx:eax)/оп
частное (max):
r: al (255) / ax (65535) / eax (2^{32}-1)
остаток: ah / dx / edx

коррекция BCD
[(упакованных)]
⊕ [a/d]**aa** ⊖ [a/d]**as**
use РОН(ax)
aam
'ah.al' = al/10
aad - до /.
al +=(ah*10)
ah=00h

↑ ✎ в 2 раза (🔊 битом знака)
c[bw/wde] - a[l/x] в [e]ax
c[wd/dq] - [e]ax в [e]dx:[e]ax
mov[s/z] rОп, оп2 - 🔊[бит зн. / '0']
✎ rОп > оп2

xadd оп1, оп2
оп2 = оп1
оп1 += оп2

cmpxchg оп1, оп2
if(al/ax/eax = оп1)
zf=1, оп1=оп2
else
zf=0, al/ax/eax=оп1

bswap r32 - инверсия порядка байтов.

опВыражения (от компилятора)
1. length, size width, mask, (,), [,], <, >
2. '.' 3. ':' 4. ptr, offset, seg, type, this
5. high, low [6/8]. '+/-' ([унар/би]ные)
остаток от '/' «/» =/≠, </≤, >/≥
7. '*', '/', mod, shl/shr 9. eq/ne, lt/le, gt/ge
10. not 11. and 12. or, xor 13. short, type

тип **ptr** выражение - тип.
тип: [t]byte, [/d/q]word near/far
H: t1 dd 0
mov al, **byte ptr** t1+1
//со 2-ого байта 't1'

seg оп - А. ▷
метка, var,
имя[/ ▷ /Группы]
offset выражение - от ▷
H: mov ax, **seg** txt1 //А. ▷
mov es, ax
mov dx, **offset** txt1 //от ▷
//в 'es:dx' - полный А. txt1