Команди на целочисленото устройство от микропроцесора Intel 80486

(Кратко описание)

Забележка. Тук не са описани различни особености на командите, свързани с подробности за въздействието върху флагове, с възможности за възникването на особени случаи и за обработката на прекъсвания, с режимите на работа на микропроцесора (например при смесване на 16-разрядна и 32-разрядна адресация в защитен режим), с машинните формати и кодове, с броя на машинните тактове, за които се изпълнява една отделна команда, със зависимостта на изпълнението от нивото на привилегии, с поддръжката на псевдопаралелизъм, със сегментацията на паметта и контрола на достъпа до нея, с използването на системните регистри и някои др.

Използвани означения

<-- – отляво на стрелката се присвоява (записва) стойността на израза, намиращ се отдясно на стрелката;
if... then... else... endif – обичайният смисъл;
and, or, not – побитови операции над стойностите на изразите;
>, <, = – обичайните сравнения;
+, –, *, /, mod – обичайните аритметични действия (mod – остатък);
H – шестнадесетично число (буквата стои след шестнадесетични цифри).

AAA – ASCII Adjust after Addition
Корекция след събиране (след ADD) (за 2-10-ична аритметика с непакетиран формат). Няма явен операнд. По подразбиране работи с регистъра ax по следния начин:

```
if ( ( AL and 0fH ) > 9 ) or ( AF = 1 )
then AL <-- ( AL + 6 ) and 0fH
     AH <-- AH + 1;
     AF <-- 1
     CF <-- 1
else CF <-- 0
     AF <-- 0
endif
```

AAD – ASCII Adjust AX before Division
Корекция на регистъра AX преди делене (за 2-10-ична аритметика с непакетиран формат). Няма явен операнд. По подразбиране работи с регистъра ax по следния начин:

```
AL <-- AH * 10 + AL
AH <-- 0
```

AAM – ASCII Adjust AX after Multiply
Корекция на регистъра AX след умножение (след MUL) (за 2-10-ична аритметика с непакетиран формат). Няма явен операнд. По подразбиране работи с регистъра ax по следния начин:

```
AH <-- AL / 10
AL <-- AL mod 10
```

Например командите

```
mov al,217
aam
```

записват 21 (т. е. 15H) в ah и 7 в al.

AAS – ASCII Adjust after Subtraction
Корекция след изваждане (след SUB) (за 2-10-ична аритметика с непакетиран формат). Няма явен операнд. По подразбиране работи с регистъра ax по следния начин:

```
if ( ( AL and 0fH ) > 9 ) or ( AF = 1 )
then AL <-- AL - 6
     AL <-- AL and 0fH
     AH <-- AH -1
     AF <-- 1
     CF <-- 1
else CF <-- 0
     AF <-- 0
endif
```

ADC – ADd with Carry
Събиране заедно с преноса (т. е., +CF). Има два явни операнда. Записва на мястото на първия операнд сумата от двата операнда и стойността на флага CF. Например след командите

```
mov esi,3
mov edi,4
stc ; записване на 1 във флага CF
adc esi,edi
```

в регистъра esi има стойност 8.

ADD – ADD
Събиране. Има два явни операнда. Записва на мястото на 1-я операнд сумата на двата операнда.

AND – logical AND
Логическо И. Има два явни операнда. Записва на мястото на 1-я операнд резултата от поразрядно логическо И над двата операнда.

ARPL – Adjust RPL field of selector
Корекция на полето RPL (за привилегия) в селектор (на сегмент). Има два явни операнда.

BOUND – check array index against BOUNDs
Контрол, дали индексът на масив се намира в границите. Има два явни операнда. Проверява дали 1-ят операнд (съдържание на регистър), т. е. проверяваният индекс, се намира в границите, зададени чрез 2-операнд. Границите са 2 отмествания и са записани последователно (по-малка – по-голяма) в паметта. 2-ят операнд задава адреса им. При несъответствие на индекса възниква прекъсване номер 5.

BSF – Bit Scan Forward
Сканиране бита напред. Има два явни операнда, които трябва да са с еднаква разрядност – или 16- или 32-разрядни. 1-ят операнд е в регистър, а 2-ят може да бъде в регистър или в паметта. Когато 2-ят операнд има само нулеви битове – записва 1 във флага ZF, а стойността на 1-я е неопределена. Иначе – записва 0 във флага ZF, а в 1-я операнд записва номера (считано от 0 до 31) на най-десния (най-младшия) бит, различен от нула, във 2-я операнд. Например командите

```
mov ecx,0f0H
bsf eax,ecx
```

записват в eax числото 4.

BSR – Bit Scan Reverse
Сканиране бита назад. Има два явни операнда, които трябва да са с еднаква разрядност – или 16- или 32-разрядни. 1-ят операнд е в регистър, а 2-ят може да бъде в регистър или в паметта. Когато 2-ят операнд има само нулеви битове – записва 1 във флага ZF, а стойността на 1-я е неопределена. Иначе – записва 0 във флага ZF, а в 1-я операнд записва номера (считано от 0 до 31) на най-левия (най-старшия) бит, различен от нула, във 2-я операнд. Например командите

```
mov ecx,0f0H
bsr eax,ecx
```

записват в eax числото 7.

BSWAP – Byte SWAP
Обмен на байт. Има един 32-разряден явен операнд регистър. Разменя местата на байтовете му в точно обратен ред. Например командите

```
mov ecx,0a1a2a3a4H
bswap ecx
```

записват в ecx числото 0a4a3a2a1H.

BT – Bit Test
Проверка на бита. Има два явни операнда. Записва във флага CF бита с номер 2-я операнд от стойността на 1-ия операнд. Например командите

```
mov eax,0f0H
bt eax,4
```

записват 1 във флага CF.

BTC – Bit Test and Complement
Проверка на бит и допълнение (инвертиране). Има два явни операнда. Записва във флага CF бита с номер 2-я операнд от стойността на 1-ия операнд. След това инвертира същия бит от 1-я операнд. Например командите

```
mov eax,0f0H
btc eax,4
```

записват 1 във флага CF и 0e0H в eax.

BTR – Bit Test and Reset
Проверка на бит и нулиране. Има два явни операнда. Записва във флага CF бита с номер 2-я операнд от стойността на 1-ия операнд. След това записва 0 в същия бит от 1-я операнд. Например командите

```
mov eax,0f0H
btr eax,4
```

записват 1 във флага CF и 0e0H в eax.

BTS – Bit Test and Set
Проверка на бит и установяване (записване на 1). Има два явни операнда. Записва във флага CF бита с номер 2-я операнд от стойността на 1-ия операнд. След това записва 1 в същия бит от 1-я операнд. Например командите

```
mov eax,0f0H
bts eax,3
```

записват 0 във флага CF и 0e8H в eax.

CALL – CALL
Обръщение към (извикване на) подпрограма. Има един явен операнд, адрес на команда. Съхранява в стека точката на връщане и предава управлението към адреса, определен с операнда (към входната точка на процедура). Според начина на извикване на процедурата, може в стека да се записва или да не се записва и сегментният регистър cs.

CBW – Convert Byte to Word
Преобразуване на байт в дума. Без явни операнди. 8-разрядната знакова стойност на al преобразува в 16-разрядна знакова стойност в ax.

CDQ – Convert Doubleword to Quadword
Преобразуване на двойна дума в четворна дума. Без явни операнди. 32-разрядната знакова стойност на eax преобразува в 64-разрядна знакова стойност с младша дума в eax и старша дума в edx.

CLC – CLear Carry flag
Нулиране на флага за пренос (флага CF). Без явен операнд.

CLD – CLear Direction flag
Нулиране на флага за направление (флага DF). Без явен операнд.

CLI – CLear Interrupt flag
Нулиране на флага за прекъсване (флага IF). Без явен операнд.

CLTS – CLear Task-Switched flag in CR0
Нулиране на флага за превключване на задачата в регистъра CR0 (флага TS). Без явен операнд.

CMC – CoMplement Carry flag
Допълване (инвертиране) на флага за пренос (флага CF). Без явен операнд.

CMP – CoMPare two operands
Сравняване на два операнда. Има точно два явни операнда. От 1-я изважда 2-я. Резултатът не се съхранява, но при изчислението се актуализират флаговете.

CMPS – CoMPare String operands
Сравняване на низови операнди. Има два явни операнда с еднаква разрядност. Изчислява разликата DS:[ESI] – ES:[EDI]. Размерът на обработваните елементи на низове се определя от написаните операнди. Резултатът не се съхранява, но при изчислението се актуализират флаговете.

CMPSB – CoMPare String operands of Bytes
Сравняване на низови операнди байтове. Няма явен операнд. Изчислява разликата DS:[ESI] – ES:[EDI] от байтове. Резултатът не се съхранява, но при изчислението се актуализират флаговете.

CMPSD – CoMPare String operands of Doubleword
Сравняване на низови операнди двойни думи. Няма явен операнд. Изчислява разликата DS:[ESI] – ES:[EDI] от двойни думи. Резултатът не се съхранява, но при изчислението се актуализират флаговете.

CMPSW – CoMPare String operands of Word
Сравняване на низови операнди думи. Няма явен операнд. Изчислява разликата word ptr DS:[ESI] – word ptr ES:[EDI]. Резултатът не се съхранява, но при изчислението се актуализират флаговете.

CMPXCHG – CoMPare and eXCHanGe
Сравняване и обмен. Има два еднакво разрядни операнда, които могат да бъдат 8-, 16- или 32-разрядни. Сравнява акумулатора (al, ax или eax), имащ разрядност като на операндите (например al при 8-разрядни операнди), с 1-я операнд. Когато са равни – записва 2-я операнд на мястото на 1-я. Когато не са равни – записва 1-я операнд в акумулатора (имащ същата разрядност). Флаговете CF, OF, AF, SF, PF и ZF се обновяват точно както при команда CMP с 1-и операнд, същия като този на CMPXCHG, и 2-и операнд акумулатора със съответната разрядност.

CWD – Convert Word to Doubleword
Преобразуване на дума в двойна дума. Няма явен операнд. 16-разрядната знакова стойност на ax преобразува в 32-разрядна знакова стойност с младша дума в ax и старша дума в dx.

CWDE – Convert Word to Doubleword Extended
Преобразуване на дума в двойна дума с (използване на) разширение. Няма явен операнд. 16-разрядната знакова стойност на ax преобразува в 32-разрядна знакова стойност в eax.

DAA – Decimal Adjust AL after Addition
Десетична корекция на AL след събиране. (За 2-10-ична аритметика с пакетиран формат.) Няма явен операнд. По подразбиране работи с регистъра ax по следния начин:

```
if ( ( AL and 0fH ) > 9 ) or ( AF = 1 )
then AL <-- ( AL + 6 )
     AF <-- 1
else AF <-- 0
endif
if ( AL > 9fH ) or ( CF = 1 )
then AL <-- AL + 60H
     CF <-- 1
else CF <-- 0
endif
```

DAS – Decimal Adjust AL after Substraction
Десетична корекция на AL след изваждане. (За 2-10-ична аритметика с пакетиран формат.) Няма явен операнд. По подразбиране работи с регистъра ax по следния начин:

```
if ( ( AL and 0fH ) > 9 ) or ( AF = 1 )
then AL <-- ( AL - 6 )
     AF <-- 1
else AF <-- 0
endif
if ( AL > 9fH ) or ( CF = 1 )
then AL <-- AL - 60H
     CF <-- 1
else CF <-- 0
endif
```

DEC – DECrement by 1
Декремент (намаляване) с 1. Има един явен операнд. Намалява го с 1. НЕ променя флага CF при заем отляво на старшия бит.

DIV – unsigned DIVide
Беззнаково делене. Предполага се, че операндите и резултатите са беззнакови числа. Командата има един явен операнд с разрядност 8, 16 или 32. Той трябва да бъде в регистър или в паметта (но не в самата команда, т. е. не може div 7). Когато операндът е 8-разряден, тогава се разделя ax на операнда, полученото частно (по-точно –цялата част от него) се записва в al, а остатъкът се записва в ah. Когато операндът е 16-разряден, тогава на него се дели 32-разрядното число, чиято старша дума е в dx и младша дума е в ax, частното се записва в ax и остатъкът се записва в dx. Когато операндът е 32-разряден, тогава на него се дели 64-разрядното число, чиято старша двойна дума е в edx и младша двойна дума е в eax, частното се записва в eax и остатъкът се записва в регистъра edx.

ENTER – make stack frame for procedure parameters
Формиране на стеков кадър за параметрите на процедура. (Обикновено локалните данни на процедурата в стека се премахват чрез LEAVE.) Има два явни операнда. 1-ят определя размера на кадъра в брой байтове. 2-ят определя колко указателя на (вложени) стекови кадри се копират от стария стеков кадър в новия. Указател на кадъра е регистъра ebp. Когато 2-ят операнд е 0, тогава: командата включва в стека указателя на кадър (т. е. ebp), после изважда от указателя към върха на стека (т. е. от esp) 1-я параметър и, накрая, записва текущият указател към върха на стека (стойността на esp) в указателя на кадъра (в регистъра ebp).

FWAIT – WAIT
Изчакване. Няма явен операнд. Командата е алтернатива на WAIT (а не на ESC). Кара процесора, преди да продължи, да провери за чакащи немаскирани особени случаи (вид прекъсвания).

HLT – HaLT
Спиране. Няма явен операнд. Процесорът преминава в състояние на изчакване (спира) докато възникне заявка за разрешено прекъсване, за немаскируемо прекъсване или сигнал за "реинициализиране".

IDIV – sIgned DIVide
Знаково делене. Има един явен операнд. Предполага се, че операндите и резултатите са знакови числа. Техните местонахождение и размерности са както в командата DIV.

IMUL – sIgned MULtirly
Знаково умножение. Може да има 1, 2 или 3 операнда.

Когато има 1 явен операнд: Той трябва да бъде в регистър или в паметта (т. е., не може да се използва непосредствена адресация). Размерността му може да бъде 8, 16 или 32 разряда. При 8-разряден операнд, в ax се записва произведението му с al. При 16-разряден операнд, неговото произведение с ax се записва в 32-разряден резултат с младша дума в ax и старша дума в dx. При 32-разряден операнд, неговото произведение с eax се записва в 64-разряден резултат с младша двойна дума в eax и старша двойна дума в edx.

Когато има 2 явни операнда: 1-ят трябва да бъде 16- или 32-разряден регистър. Резултатът се записва на мястото на 1-я операнд. При 16-разряден 1-и операнд – 2-ят операнд може да бъде 16-разряден регистър, дума в паметта за данни или число в командата (непосредствен операнд), което може да се кодира със знак в 16 разряда. При 32-разряден 1-и операнд – 2-ят може да бъде 32-разряден регистър, двойна дума в паметта за данни или 8-разрядно число в командата (непосредствен операнд).

Когато има 3 явни операнда: 1-ят от тях трябва да бъде 16- или 32-разряден регистър и на неговото място се записва резултатът от умножението на 2-я и 3-я операнди. Тогава 2-ят има същата размерност като 1-я и трябва да бъде или в регистър или в паметта за данни. 3-ят операнд трябва да бъде число в командата (непосредствен), което може да се кодира в не повече разряди от тези на 1-я операнд.

Например командите

```
mov dx,0400H
imul cx,dx,2
```

записват 0800H в cx.

IN – INput from port
Вход от порта. Има два явни операнда. Чете от портовете и записва в 1-я операнд, който трябва да бъде al, ax или eax. 2-ят операнд задава адреса на порта и може да бъде или 8-разрядно число от командата (непосредствен) или регистъра dx.

INC – INCrement by 1
Инкремент (увеличаване) с 1. Има един явен операнд и го увеличава с 1. Не променя флага CF, когато има пренос наляво от старшия разряд.

INS – INput from port to String
Вход от порт в низ. Има два явни операнда. Прехвърля байт, дума или двойна дума (колкото е размерът на 1-я операнд) в (на адрес) ES:[(E)DI] от порта, определен от 2-я операнд (който трябва да бъде dx).

INSB – INput from port to String (Byte)
Вход от порт в низ на един байт. Няма явен операнд. Прехвърля байт в (на адрес) ES:[(E)DI] от порта, чийто адрес е в регистъра dx.

INSD – INput from port to String (Doubleword)
Вход от порт в низ на една двойна дума. Няма явен операнд. Прехвърля двойна дума в (на адрес) ES:[(E)DI] от порта, чийто адрес е в регистъра dx.

INSW – INput from port to String (Word)
Вход от порт в низ на една дума. Няма явен операнд. Прехвърля дума в (на ардес) ES:[(E)DI] от порта, чийто адрес е в регистъра dx.

INT – call to INTerrupt procedure
Извикване на процедура за прекъсване. Има един явен операнд. Извършва обръщение към обработчик на прекъсване с номер (зададен от) операнда така, сякаш отвън е постъпила заявка за прекъсване с този номер.

INTO – call to INTerrupt procedure in Overflow
Предизвикване на прекъсване за препълване. Няма явен операнд. Предизвиква прекъсване 4, точно когато флагът FO=1.

INVD – INValiDate cache
Недостоверност на кеш-паметта. Няма явен операнд. Изчиства вътрешната кеш-памет и подава сигнал за изчистване и на външната кеш-памет.

INVLPG – INVaLidate TLB entry
Недостоверност на елемент от буфера на TLB. Има един явен операнд. Обявява за недостоверен един елемент от буфера на TLB.

IRET – Interrupt Return
Връщане от (обработчик на) прекъсване. Няма явен операнд.

IRETD – синоним на IRET

J# – Jump if condition is met
Преход, когато се удовлетворява условие. Има един явен операнд – адрес на команда. (На Асемблер операндът обикновено е етикет.) Условието се определя по флагове, (на Асемблер се пише мнемонично) по следния начин:

Възможните мнемонични означения, отбелязани с #, са:
Според стойността на САМО ЕДИН ФЛАГ

#	*флаг*	*типична ситуация, когато флагът има посочената стойност*
c	CF=1	имало е пренос или заем от старшия бит на операнда
nc	CF=0	нямало е пренос или заем от старшия бит на операнда
s	ST=1	резултатът е отрицателен (има старши бит 1)
ns	ST=0	резултатът е положителен (има старши бит 0)
z	ZF=1	резултатът е нула
nz	ZF=0	резултатът е различен от нула
o	OF=1	имало е препълване (резултатът не се е побрал в предвидения брой битове) при изчисление със знакови (т. е. в допълнителен код) операнди
no	OF=0	нямало е препълване при изчисление със знакови операнди
p	PF=1	младшите осем байта на резултата съдържат четен брой единици
pe	PF=1	като p
np	PF=0	младшите осем байта на резултата съдържат нечетен брой единици
po	PF=0	като np

Когато стойностите на флаговете се интерпретират като резултат от действие над ***БЕЗЗНАКОВИ операнди***

#	*флагове*	*отношение между* x *и* y, *непосредствено след cmp x,y*	*смисъл на мнемоничното означение*
e	ZF=1	$x = y$	равно
ne	ZF=0	$x \neq y$	не равно
b	CF=1	$x < y$	по-малко
nb	CF=0	$x \geq y$	не по-малко
be	CF=1 или ZF=1	$x \leq y$	по-малко или равно
nbe	CF=0 и ZF=0	$x > y$	не по-малко и не равно
a	CF=0 и ZF=0	$x > y$	по-голямо
na	CF=1 или ZF=1	$x \leq y$	не по-голямо
ae	CF=0	$x \geq y$	по-голямо или равно
nae	CF=1	$x < y$	не по-голямо и не равно

Когато стойностите на флаговете се интерпретират като резултат от действие над ***ЗНАКОВИ операнди***

#	*флагове*	*отношение между* x *и* y, *непосредствено след cmp x,y*	*смисъл на мнемоничното означение*
e	ZF=1	$x = y$	равно
ne	ZF=0	$x \neq y$	не равно
l	SF≠OF	$x < y$	по-малко
nl	SF=OF	$x \geq y$	не по-малко
le	ZF=1 или SF≠OF	$x \leq y$	по-малко или равно
nle	ZF=0 и SF=OF	$x > y$	не по-малко и не равно
g	ZF=0 и SF=OF	$x > y$	по-голямо
ng	ZF=1 или SF≠OF	$x \leq y$	не по-голямо
ge	SF=OF	$x \geq y$	по-голямо или равно
nge	SF≠OF	$x < y$	не по-голямо и не равно

JCXZ – Jump if CX is Zero
Преход, когато регистърът CX=0. Има един явен операнд – адресът, към който може да се премине. (На Асемблер операндът обикновено е етикет.)

JECXZ – Jump if ECX is Zero
Преход, когато регистърът ECX=0. Има един явен операнд – адресът, към който може да се премине. (На Асемблер операндът обикновено е етикет.)

JMP – JuMP
Преход. Безусловен преход към адреса, зададен чрез единствения явен и фактически операнд. (На Асемблер операндът обикновено е етикет.)

LAHF – Load Flags into AH register
Зареждане на флагове в регистъра AH. Няма явен операнд. Прехвърля младшия байт на флаговия регистър FLAGS в регистъра AH.

LAR – Load Access Right byte
Зареждане байта на правата за достъп. Има два явни операнда. Записва в регистър на правата за достъп от селектор на сегмент.

LDS – Load full pointer (selector into DS)
Зареждане на пълен адрес; селекторът (се записва) в DS. Има два явни операнда. От паметта с адрес, зададен чрез 2-я операнд, извлича пълен логически адрес и записва селектора на сегмент в сегментния регистър ds, а отместването в 1-я операнд, който трябва да бъде регистър.

LEA – Load Effective Address
Зареждане на ефективен адрес. Има два явни операнда. Записва в 1-я операнд, който трябва да бъде регистър, ефективния адрес (отместването) на 2-я операнд.

LEAVE – high level procedure exit
Излизане от процедура. Без явен операнд. Премахва стековия кадър (например, но не задължително, създаден с ENTER) като копира указателя на кадър (регистъра ebp) в указателя на стека (регистъра esp), а после извлича от стека стария указател на кадър и го записва в регистъра ebp.. Самото връщане от процедура и премахването на параметрите й от стека става чрез RET или RET n.

LES – Load full pointer (selector into ES)
Зареждане на пълен адрес; селекторът (се записва) в ES. Има два явни операнда. От паметта с адрес, зададен чрез 2-я операнд, извлича пълен логически адрес и записва селектора на сегмент в сегментния регистър es, а отместването в 1-я операнд, който трябва да бъде регистър.

LFS – Load full pointer (selector into FS)
Зареждане на пълен адрес; селекторът (се записва) в FS. Има два явни операнда. От паметта с адрес, зададен чрез 2-я операнд, извлича пълен логически адрес и записва селектора на сегмент в регистъра fs, а отместването в 1-я операнд, който трябва да бъде регистър.

LGDT – Load Global Descriptor Table register
Зареждане на регистъра на глобалната дескрипторна таблица. Има един явен операнд. Чете от паметта и записва в регистъра GDTR.

LGS – Load full pointer (selector into GS)
Зареждане на пълен адрес; селекторът (се записва) в GS. Има два явни операнда. От паметта с адрес, зададен чрез 2-я операнд, извлича пълен логически адрес и записва селектора на сегмент в gs, а отместването в 1-я операнд, който трябва да бъде регистър.

LIDT – Load Interrupt Descriptor Table register
Зареждане на регистъра на локалната дескрипторна таблица. Има един явен операнд. Чете от паметта и записва в регистъра LITR.

LLDT – Load Local Descriptor Table register
Зареждане на регистъра на глобалната дескрипторна таблица. Има един явен операнд. Записва стойност в регистъра LDTR.

LMSW – Load Machine Status Word
Зареждане на думата на състоянието на машината. Има един явен операнд. Записва от операнда в думата за състоянието на машината (в част от регистъра CR0).

LOCK – assert LOCK# signal prefix
Префикс за подаване на сигнала LOCK#. Префиксът заставя процесора да издаде сигнал LOCL# за времето на изпълнение на командата след префикса. Може да се поставя само пред някои команди.

LODS – LOaD String operand
Зареждане на низов операнд. Има един явен операнд. Записва в съответния акумулатор (al, ax или eax) байт, дума или двойна дума от адрес DS:[ESI] и насочва esi към следващия елемент на низа (променя esi според размера на елементите на низа и според флага DF). Разрядността на елемента на низа е равна на разрядността на операнда на командата.

LODSB – LOaD String operand Byte
Зареждане на низов операнд байт. Без явен операнд. Записва в акумулатора al байт от адрес DS:[ESI] и насочва esi към следващия байт (променя esi с 1 според флага DF).

LODSD – LOaD String operand Doubleword
Зареждане на низов операнд двойна дума. Без явен операнд. Записва в акумулатора eax двойна дума от адрес DS:[ESI] и насочва esi към следващата двойна дума (променя esi с 4 според флага DF).

LODSW – LOaD String operand Word
Зареждане на низов операнд дума. Без явен операнд. Записва в акумулатора ax дума от адрес DS:[ESI] и насочва esi към следващата дума (променя esi с 2 според флага DF).

LOOP – LOOP control with eCx counter
Управление на цикъл с брояч ECX. Има един явен операнд – адрес на команда. Намалява с 1 ecx. Точно когато след намаляването в ecx се получи стойност, различна от нула, преход към (адреса, зададен с) операнда. Не променя флагове.

LOOPE – LOOP control with eCx counter by condition E
Управление на цикъл с брояч ECX по условие E. Има явен един операнд – адрес на команда. Намалява ecx с 1. Извършва преход към (адреса, зададен с) операнда тогава и само тогава, когато флагът ZF=1 и ECX≠0 след намаляването. Не променя флагове.

LOOPNE – LOOP control with eCx counter by condition NE
Управление на цикъл с брояч ECX по условие NE. Има един явен операнд – адрес на команда. Намалява ecx с 1. Извършва преход към (адреса, зададен с) операнда тогава и само тогава, когато флагът ZF=0 и ECX≠0 след намаляването. Не променя флагове.

LOOPNZ – LOOP control with eCx counter by condition NZ
Синоним на LOOPNE.

LOOPZ – LOOP control with eCx counter by condition Z
Синоним на LOOPE.

LSL – Load Segment Limit
Зареждане на границата на сегмент. Има два явни операнда. Записва в регистъра 1-и операнд границата на сегмента, зададен чрез 2-я операнд.

LSS – Load full pointer (selector into SS)
Зареждане на пълен адрес; селекторът (се записва) в SS. Има два явни операнда. От паметта с адрес, зададен чрез 2-я операнд, извлича пълен логически адрес и записва селектора на сегмент в ss, а отместването в 1-я операнд, който трябва да бъде регистър.

LTR – Load Task Register
Зареждане на регистъра на задачата. Има един явен операнд. Записва единствения операнд в регистъра на задачата TR.

MOV – MOVe data
Предаване на данни. Има два явни операнда. Записва 2-я операнд на мястото на 1-я операнд.

MOVS – MOVe data from String to string
Прехвърляне на данни от низ в низ. Има два явни операнда с еднаква размерност (8, 16 или 32 бита). Копира байт, дума или двойна дума от DS:[ESI] в ES:[EDI] и насочва edi и esi към следващите елементи на низовете (променя esi и edi с 1, 2 или 4 в посоката, определена от флага DF). Размерността на елементите на низовете е еднаква и се определя от двата операнда на командата.

MOVSB – MOVe data from String to string of Bytes
Прехвърляне на данни от низ в низ от байтове. Няма явен операнд. Копира байт от DS:[ESI] в ES:[EDI] и насочва edi и esi към следващите байтове в низовете (промяната е с 1 в посоката, определена от флага DF).

MOVSD – MOVe data from String to string of Doublewords
Прехвърляне на данни от низ в низ от двойни думи. Няма явен операнд. Копира двойна дума от DS:[ESI] в ES:[EDI] и насочва edi и esi към следващите двойни думи в низовете (промяната е с 4 в посоката, определена от флага DF).

MOVSW – MOVe data from String to string of Words
Прехвърляне на данни от низ в низ от думи. Няма явен операнд. Копира дума от DS:[ESI] в ES:[EDI] и насочва edi и esi към следващите думи в низовете (промяната е с 2 в посоката, определена от флага DF).

MOVSX – MOVe with Sign-eXtend
Прехвърляне със знаково разширение. Има два явни операнда. Прехвърля 2-я операнд, интерпретиран като число със знак, на мястото на 1-я операнд, също интерпретиран като знаково число. Възможното преобразуване при прехвърлянето е от байт в дума или в двойна дума и от дума в двойна дума.

MOVZX – MOVe with Zero-eXtend
Прехвърляне с нулево разширение. Има два явни операнда. Аналогично на MOVSX, но операндите се интерпретират като беззнакови числа. Т.е. – винаги се добавят водещи нули при преобразуването.

MUL – unsigned MULtiplication af AL, AX or EAX
Беззнаково умножение на AL, AX или EAX. Има един явен операнд. Операндите и резултатът се интерпретират като беззнакови числа. Техните местоположения и разрядности са, точно както при IMUL с единствен явен операнд. (На Асемблер се пише един операнд.)

NEG – two's complement NEGation
Получаване на допълнителен код. (Смяна на знака.) Има един явен операнд. Интерпретира операнда като двоично число и го заменя с допълнителния му код. (Преобразованието е обратимо. Т. е., приложено над кода на отрицателно число, ще даде кода на абсолютната стойност на същото число.) Не променя флагове.

NOP – No OPeration
Празна операция. Няма явен операнд. Заема 1 байт и се изпълнява за 1 такт без да променя нищо.

NOT – one's complement negation
Инвертиране (побитово). Има един явен операнд. Преобразува го чрез поразрядно логическо НЕ.

OR – logical inclusive OR
Логическо включващо ИЛИ. Има два явни операнда. На мястото на 1-я записва резултата от поразрядно логическо ИЛИ над двата операнда..

OUT – OUTput to port
Извеждане в порт. Има два явни операнда. 1-ят от тях (или 8-разряден непосредствен, или dx) задава адреса на порта. 2-ят операнд (al, ax или eax) съдържа извежданата данна.

OUTS – OUTput String to port
Извеждане на низ в порт. Има два явни операнда. 1-ят винаги е регистърът dx. Прехвърля байт, дума или двойна дума от DS:[ESI] в порт и насочва esi към следващия елемент на редицата (променя esi с 1, 2 или 4 според размера на елемента на низа и в посока, определена от флага DF). 1-ят операнд (dx) задава адреса на порта. 2-ят операнд служи само за определяне размерността на елементите на низа.

OUTSB – OUTput String of Bytes to port
Извеждане на низ от байтове в порт. Няма явен операнд. Прехвърля байт от DS:[ESI] в порт и насочва esi към следващия байт от низа (променя esi с 1 в посока, определена от флага DF).

OUTSD – OUTput String of Doublewords to port
Извеждане на низ от двойни думи в порт. Няма явен операнд. Прехвърля една двойна дума от DS:[ESI] в порт и насочва esi към следващата двойна дума от низа (променя esi с 4 в посока, определена от флага DF).

OUTSW – OUTput String of Words to port
Извеждане на низ от думи в порт. Няма явен операнд. Прехвърля една дума от DS:[ESI] в порт и насочва esi към следващата дума от низа (променя esi с 2 в посока, определена от флага DF).

POP – POP from the stack
Извличане от стека. Има един явен операнд. Извлича от стека и записва на мястото на операнда, който може да бъде 16- или 32-разряден. Автоматично пренасочва esp към новия връх на стека.

POPA – POP All general registers
Извличане на всички регистри с общо предназначение. Няма явен операнд. Извлича от стека стойности и ги записва в регистрите DI, SI, BP, SP (тази стойност се игнорира), BX, DX, CX, AX (в този ред). Автоматично обновява esp.

POPAD – POP All general registers
Извличане на всички регистри с общо предназначение. Няма явен операнд. Извлича от стека стойности и ги записва в регистрите EDI, ESI, EBP, ESP (тази стойност се игнорира), EBX, EDX, ECX, EAX (в този ред). Автоматично обновява esp.

POPF – POP stack into FLAGS
Извлича от стека и записва във FLAGS. Няма явен операнд.

POPFD – POP stack into EFLAGS
Извлича от стека и записва във EFLAGS. Няма явен операнд.

PUSH – PUSH operand onto stack
Включване на операнда в стека. Има един явен операнд. Записва операнда в стека и обновява автоматично указателя esp към върха на стека. Операндът може да бъде 16- или 32-разряден в регистър, паметта за данни или в командата (непосредствен).

PUSHA – PUSH All general registers
Включване в стека на всички регистри с общо предназначение. Няма явен операнд. Записва в стека регистрите AX, CX, DX, BX, SP (стойността му преди изпълнение на командата), BP, SI, DI (в този ред) Автоматично актуализира esp.

PUSHAD – PUSH All general registers
Включване в стека на всички регистри с общо предназначение. Няма явен операнд. Записва в стека регистрите EAX, ECX, EDX, EBX, ESP (стойността му преди изпълнение на командата), EBP, ESI, EDI (в този ред) Автоматично актуализира esp.

PUSHF – PUSH Flags register onto stack
Включване в стека на регистъра с флагове FLAGS. Няма явен операнд.

PUSHFD – PUSH EFlags register onto stack
Включване в стека на регистъра с флагове EFLAGS. Няма явен операнд.

RCL – ROtate through CF to Left
Циклично изместване (побитова ротация) през (флага) CF наляво. Има два явни операнда. 1-ят е ротираното число и може да бъде 8-, 16- или 32-разрядна данна в регистър или в паметта за данни. 2-ят е броя на позициите за изместване и може да бъде само 8-разрядна данна или в al, или в самата команда (непосредствен операнд). При всяко изместване с една позиция, младшият бит (отдясно) приема стойността на флага CF, а старшият бит (отляво) се записва в CF. Например, след командите

```
mov edi,1
mov esi,0f000H
mov cl,3
clc ; нулиране на флага CF
rcl edi,cl
stc ; записване на 1 във флага CF
rcl esi,cl
```

в регистъра edi остава 8, в регистъра esi остава 78004H, а флагът CF=0.

RCR – ROtate through CF to Right
Циклично изместване (побитова ротация) през (флага) CF надясно. Има два явни операнда. Аналогично е на RCL, но изместването е в обратна посока. В частност, стойността на CF отива на мястото на старшия бит, а младшият се премества във флага CF. Например, командите

```
mov edi,1
mov esi,0f000H
mov cl,3
clc ; нулиране на флага CF
rcr edi,cl
stc ; записване на 1 във флага CF
rcr esi,cl
```

оставят 40000000H в регистъра edi, стойност 20001e00H в регистъра esi и 0 във флага CF.

REP – REPeat following string operation
(Префикс за) Повторение на следващата низова операция. Низовата команда след префикса се повтаря ECX пъти. Преброяването става чрез декремент на ECX. Т.е.: Командата се изпълнява и, след това, се изважда 1 от ecx. При получаване в ecx на нещо, различно от 0, отново се преминава към изпълнение на командата. Префиксът се съотнася само с една команда. За да се изпълняват в цикъл две или повече команди, трябва да се използва друг начин на зацикляне.

REPE – REPeat following string operation while hte condition E is true
(Префикс за) Повторение на следващата низова операция докато условието E е истина. Низовата команда след префикса се повтаря най-много ECX пъти, но само докато флагът ZF=1. Т. е.: Командата се изпълнява и, след това, се изважда 1 от ecx. При получаване в ecx на нещо, различно от 0, и при условие, че ZF=1, отново се преминава към изпълнение на командата. При получаване в ecx на 0 след изваждането или при ZF=0 след изпълнението на командата се прекратява цикълът за изпълнение на командата. Префиксът се съотнася само с една команда.

REPNE – REPeat following string operation while hte condition NE is true
(Префикс за) Повторение на следващата низова операция докато условието NE е истина. Префиксът е аналогичен на REPE, но повторението се прекратява при ZF=1. Т. е. Командата се изпълнява и, след това, се изважда 1 от ecx. Когато след изваждането в ecx има нещо различно от 0 и при условие, че ZF=0 след изпълнението на командата, тогава отново се преминава към изпълнение на командата. Когато в ecx има 0 след изваждането или при ZF=1 след изпълнението на командата, тогава се прекратява изпълнението на командата. Префиксът се съотнася само с една команда.

REPNZ – REPeat following string operation while hte condition NZ is true
Синоним на REPNE.

REPZ – REPeat following string operation while hte condition Z is true
Синоним на REPE.

RET – RETurn from procedure

Връщане от процедура. Може да няма или да има явен операнд (в двата случая има различни машинни КОП). От стека извлича точката на връщане и прави преход към нея. Дали така се извличат стойност само за eip или и за cs, и за eip – това зависи от КОП. В Асемблер и в език от високо ниво компилаторът има грижата да подбере кода на командата. Т. е., той следи какво е запомнено в стека при извикването на функцията, от която се извършва връщане чрез съответната команда RET. При извличане от стека на стойности и за cs и за eip – първо се извлича стойността на eip. Когато командата има операнд –той е 16-разряден непосредствен. В такъв случай след извличане от стека точката за връщане, към esp се прибавя операндът. Така се премахва част от върха на стека (очаква се, че там има ненужни параметри, подадени по-рано към процедурата).

ROL – ROtate to Left

Циклично изместване (побитова ротация) наляво. Има два явни операнда. 1-ят е ротираното число и може да бъде 8-, 16- или 32-разрядна данна в регистър или в паметта за данни. 2-ят е броя на позициите за изместване и може да бъде само 8-разрядна данна или в al, или в самата команда (непосредствен). При всяко изместване с една позиция, младшият бит (отдясно) приема стойността на старшият бит (отляво). Например, след командите

```
mov edi,10000001H
mov esi,8000f000H
mov cl,3
rol edi,cl
rol esi,cl
```

в edi остава 80000008H и в esi остава 00078004H.

ROR – ROtate to Right

Циклично изместване (побитова ротация) надясно. Аналогично на ROL, но изместването е в обратна посока. В частност, стойността на младшия бит се премества в старшия бит. Например, след командите

```
mov edi,10000001H
mov esi,8000f000H
mov cl,3
ror edi,cl
ror esi,cl
```

има 22000000H в edi и 10001e00H в esi.

SAHF – Store AH into Flags

Съхраняване на регистъра AH във (младшия байт на) флаговия регистър. Няма явен операнд. Така се записват стойности във флаговете SF, ZF, AF, PF и CF.

SAL – Arithmetic Shift to Left

Аритметично изместване наляво. Т.е., не циклично побитово изместване наляво с добавяне на 0 отдясно. Такова изместване с N позиции (също както и SHL) е еквивалентно на умножение по N-тата степен на двойката. Операндите са точно както при ROL, но при изместване с една позиция се губи стойността на старшия разряд, а младшият получава стойност 0.

SAR – Arithmetic Shift to Right

Аритметично изместване надясно. Почти аналогично е на SAL, но е в обратна посока и стойността на старшия бит винаги се запазва непроменена. Такова изместване с N позиции е еквивалентно на делене на N-тата степен на двойката при знакови цели числа в допълнителен код.

SBB – integer SuBtraction wity Borrow

Целочислено изваждане със заем. Има два явни операнда. На мястото на 1-я операнд се записва стойността на израза:

(1-и операнд) - ((2-и операнд) + (флаг CF))

Т. е., отчита се, че във флага CF има единица точно тогава, когато при предишно изваждане е имало заем отляво на старшия разряд на умаляемото. Например, след

```
stc ; записване 1 CF
mov eax,5
mov esi,2
sbb eax,esi
```

в регистъра eax има 2.

SCAS – compare (SCAn) String data

Сравняване (сканиране) на низови данни. Това е сравняване чрез изваждане. Има един явен операнд, който задава разрядността на умаляемото и умалителя. От акумулатора (или AL, или AX, или EAX) се изважда елементът (със същата разрядност) на низа, адресиран чрез ES:[EDI] и автоматично се насочва edi към следващия елемент от низа (според флага DF). Получената разлика не се съхранява, но се актуализират флагове.

SCASB – compare (SCAn) String of Bytes

Сравняване (сканиране) на низове от байтове. Няма явен операнд. Изчислява се разликата AL – byte ptr ES:[EDI] и автоматично се насочва edi към следващия байт от низа (променя се с 1 или с -1, според флага DF). Разликата не се запомня, но се актуализират флагове.

SCASD – compare (SCAn) String of Doublewords

Сравняване (сканиране) на низове от двойни думи. Няма явен операнд. Изчислява се EAX – dword ptr ES:[EDI] и автоматично се насочва edi към следващата двойна дума от низа (променя се с 4 или с -4, според флага DF). Разликата не се запомня, но се актуализират флагове.

SCASW – compare (SCAn) String of Words

Сравняване (сканиране) на низове от думи. Няма явен операнд. Изчислява се AX – word ptr ES:[EDI] и автоматично се насочва edi към следващата дума от низа (променя се с 2 или с -2, според флага DF). Разликата не се запомня, но се актуализират флагове.

SET# – byte SET on condition

Установяване на байт по условие. Има един явен 8-разряден операнд. Записва в него стойността 0 или 1 (стойност с булев смисъл) на условието, означено мнемонично по същия начин, както са означени условията в командите за условен преход j#.

SGDT – Store Global Table

Съхраняване на регистъра на глобалната дескрипторна таблица. Има един явен операнд, задаващ къде да се копира съдържанието на регистъра GDTR.

SHL – Logical Shift to Left

Логическо изместване наляво. Има 2 явни операнда. Циклично побитово изместване наляво с добавяне на 0 отдясно. Такова изместване с N позиции (както и SAL) е еквивалентно на умножение по N-тата степен на двойката. 1-ят операнд (8, 16 или 32 разряда) е числото, чиито разряди се изместват. 2-ят операнд (8-разряден) или е непосредствен, или е регистъра cl. Той задава броя на позициите за изместване. При всяко изместване с една позиция се губи стойността на старшия разряд, а младшият получава стойност 0.

SHLD – Double precisio SHift Left
Двойно изместване наляво. Има 3 явни операнда.. 1-ят е 16- или 32-разряден и или е в регистър, или е в паметта за данни. Той е числото, подлежащо на изместване. 2-ят операнд е с размерността на 1-я и е в регистър. От него се вземат битовете, които "идват" отдясно в изместваното число. 3-ят операнд е 8-разряден и или е непосредствен (в командата), или е в регистъра cl. Той задава броя на позициите за изместване (от 0 до 31). При 1-то изместване отдясно "идва" старшия бит на 2-я операнд. При 2-то изместване отдясно "идва" следващия по ред бит на 2-я операнд (битът, съседен на старшия) и т.н. Самите 2-и и 3-и операнди не се променят. Например,

```
mov si,1010H
mov di,0f000H
shld si,di,3
```

оставят в si числото 8087H.

SHR – Logical Shift to Right
Логическо изместване надясно. Има два явни операнда. Не циклично побитово логическо (с добавяне на 0 отляво) изместване надясно. Такова изместване с N позиции е еквивалентно на умножение по N-тата степен на двойката САМО при беззнакови или положителни числа. Операндите са както при SHL. При всяко изместване с една позиция се губи стойността на младшия разряд, а старшият получава стойност 0.

SHRD – Double Precision SHift Right
Двойно изместване надясно. Има три явни операнда. Аналогично е на SHLD, но в противоположна посока. Съответно, 1-ят бит, "идващ" отляво е младшият на 2-я операнд. Например, след изпълнение на командите

```
mov si,1010H
mov di,2
shrd si,di,2
```

в регистъра si се намира числото 8404H.

SIDT – Store Interrupt Table
Съхраняване на таблицата на дескрипторите на прекъсванията. Има един явен операнд. В шест байта от паметта се копира съдържанието на регистъра IDTR на дескрипторната таблица на прекъсванията.

SLDT – Store Local Descriptor Table register
Съхраняване на съдържанието на регистъра LDTR на локалната дескрипторна таблица. Има един явен 16-разряден операнд, в който се съхранява.

SMSW – Store Machine Status Word
Съхраняване на думата на състоянието на машината (младшата половина на регистъра CR0). Има един явен 16-разряден операнд, в който се съхранява.

STC – SeT Carry flag
Записване на 1 във флага CF. Без явен операнд.

STD – SeT Direction flag
Записване на 1 във флага DF. Без явен операнд.

STI – SeT Interrupt flag
Записване на 1 във флага IF. Без явен операнд.

STOS – STOre String data
Съхраняване на низови данни. Има един явен операнд, който определя разрядността на елементите на низа. Прехвърля акумулатора (AL, AX или EAX) в ES:[EDI] и насочва edi към следващия елемент на низа (според разрядността на елемента и според флага DF).

STOSB – STOre String of Bytes
Съхраняване на низ от байтове. Без явен операнд. Прехвърля AL в ES:[EDI] и променя edi с 1 според DF.

STOSD – STOre String of Doublewords
Съхраняване на низ от двойни думи. Без явен операнд Прехвърля EAX в ES:[EDI] и променя edi с 4 според DF.

STOSW – STOre String of Words
Съхраняване на низ от думи. Без явен операнд. Прехвърля AX в ES:[EDI] и променя edi с 2 според DF.

STR – Store Task Register
Съхраняване на регистъра TR на задачата. С 1 операнд.

SUB – integer SUBtraction
Целочислено изваждане. Има два явни операнда с еднаква разрядност (8, 16 или 32 бита). На мястото на 1-я операнд записва разликата (1-и операнд) – (2-и операнд).

TEST – logical compare
Логическо сравняване. Има 2 явни операнда (8-, 16- или 32-разрядни). Изчислява логическото поразрядно И над двата операнда. Резултатът не се съхранява, но се актуализират флаговете.

VERR – Verify a segment for Reading
Проверка на сегмент за четене. Има един явен операнд. Записва във флага ZF булевия резултат (0;1) от проверка, дали определеният сегмент е достъпен за четене.

VERW – Verify a segment for Writing
Проверка на сегмент за запис. Има един явен операнд. Записва във флага ZF булевия резултат (0;1) от проверка, дали определеният сегмент е достъпен за запис.

WAIT – WAIT
Изчакване. Без операнд. Указва на микропроцесора, преди да продължи изпълнението, да проконтролира чакащите немаскирани числови особени случаи.

WBINVD – Write-Back and INValiDate cache
Обратен запис и недостоверност на кеш-паметта. Няма явен операнд. Извършва изчистване на вътрешната кеш-памет и формира специален цикъл на шината, показващ на външната кеш-памет да извърши обратен запис на съдържанието си в паметта. После формира и специален цикъл на шината, предизвикващ очистване на външната кеш-памет.

XADD – Exchange and ADD
Размяна и събиране. Има два явни операнда (8-, 16- или 32-разрядни). Записва 1-я на мястото на 2-я, а сумата на операндите, записва на мястото на 1-я.

XCHG – eXCHanGe register/memory with register
Размяна на регистър/памет с регистър. Има два явни операнда (8-, 16- или 32-разрядни). Разменя местата на операндите. Единият от тях трябва да бъде в регистър.

XLAT – table look-up transLATion
Таблично преобразование. Записва в AL байта byte ptr DS:[EBX+AL]. Съдържанието на al се смята за беззнаково. Допуска замяна на DS. За целта може да има един явен 8-разряден операнд в паметта.

XLATB – table look-up transLATion
Работи като XLAT. Използва се, когато таблицата е в сегмента, селектиран от ds. Няма явен операнд.

XOR – logical eXclusive OR
Логическо изключващо ИЛИ. Има два явни операнда (8-, 16- или 32-разрядни). Изпълнява над тях логическо поразрядно изключващо ИЛИ и записва резултата на мястото на 1-я операнд.

Допълнение

Командите на целочисленото устройство (горните), групирани според типове обработки

Аритметични операции

ADC (събиране заедно с флага CF);
ADD (събиране);
CMP (изваждане с цел сравняване);
CMPXCHG (изваждане с цел сравняване и обмен според равенството);
DEC (намаляване с 1);
DIV (делене на цели без знак);
IDIV (делене на цели със знак);
IMUL (умножение на цели със знак);
INC (увеличаване с 1);
MUL (умножение на цели без знак);
NEG (смяна на знака);
SBB (изваждане със заем от флага CF);
SUB (изваждане);
XADD (размяна и събиране).

За двоично-десетична аритметика

а) в непакетиран формат:
AAA (след събиране);
AAD (преди деление);
AAM (след умножение);
AAS (след изваждане).
б) в пакетиран формат:
DAA (след събиране);
DAS (след изваждане).

Логически операции (поразрядни)

AND (логическо и);
NOT (отрицание);
OR (логическо или);
TEST (сравняване чрез логическо и);
XOR (логическо изключващо или).

Измествания

RCL (ротация наляво през CF);
RCR (ротация надясно през CF);
ROL (ротация наляво);
ROR (ротация надясно);
SAL (аритметично наляво);
SAR (аритметично надясно);
SHL (логическо наляво);
SHLD (“двойно” изместване наляво, т. е. с извличане от друга данна);
SHR (логическо надясно);
SHRD (“двойно” изместване надясно, т. е. с извличане от друга данна).

Побитови обработки

BSF (търсене на единичен бит напред);
BSR (търсене на единичен бит назад);
BT (извличане);
BTC (извличане и инвертиране);
BTR (извличане и нулиране);
BTS (извличане и записване на единица);
SET# (записва в байт стойността на логическо условие като еднобайтова единица или нула;
замества различните мнемонични съкращения от таблицата на стр. 4).

Празна операция

NOP (само изразходва 1 такт).

Работа с флагове

CLC (нулиране на CF);
CLD (нулиране на DF);
CLI (нулиране на IF; привилегирована);
CMC (инвертиране на CF);
LAHF (младшият байт на FLAGS в AH);
SAHF (AH в младшия байт на FLAGS);
STC (записване на 1 във флага CF);
STD (записване на 1 във флага DF) ;
STI (записване на 1 във флага IF; привилегирована).
Също и POPF, POPFD, PUSHF, PUSHFD.

Предаване на управлението (преходи)

CALL (обръщение към подпрограма);
J# (условен преход; # замества различните мнемонични съкращения от таблицата на стр. 4);
JCXZ (преход при CX=0);
JECXZ (преход при ECX=0);
JMP (безусловен преход);
LOOP (за цикъл по ECX);
LOOPE (за цикъл по ECX и ZF);
LOOPNE (за цикъл по ECX и не ZF);
LOOPNZ (=LOOPNE);
LOOPZ (=LOOPE);
RET (връщане от подпрограма).

Преобразувания и прехвърляния

BSWAP (размяна на байтовете);
CBW (байт в дума);
CDQ (двойна в четворна дума);
CWD (дума в двойна дума с участие на DX);
CWDE (дума в двойна дума само в EAX);
LDS (зареждане на пълен логически адрес, включително в DS);
LEA (зареждане на ефективен адрес);
LES (зареждане на пълен логически адрес, включително в ES);
LFS (зареждане на пълен логически адрес, включително в FS);
LGS (зареждане на пълен логически адрес, включително в GS);
LSL (зареждане на граница на сегмент);
LSS (зареждане на пълен логически адрес, включително в SS);
MOV (записване на избрано място);
MOVSX (прехвърляне със знаково разширяване на разрядността);
MOVZX (прехвърляне с беззнаково разширяване на разрядността);
XCHG (размяна на местата на операндите);
XLAT (извлича байт от таблица от до 256 байта) ;
XLATB (като XLAT, но само с регистъра DS).

Работа със стека

ENTER (формиране на стеков кадър);
LAHF (запис на младшия байт от флагове в регистъра AH);
LEAVE (премахване на стеков кадър);
POP (извличане на една данна);
POPA (извличане на 16-разрядните регистри с общо предназначение);
POPAD (извличане на 32-разрядните регистри с общо предназначение);
POPF (извличане на 16-разрядния флагов регистър);
POPFD (извличане на 32-разрядния флагов регистър; при привилегии 1, 2 и 3 не се променят флаговете VM и RF);
PUSH (записване на единична данна);
PUSHA (записване на 16-разрядните регистри с общо предназначение);
PUSHAD (записване на 32-разрядните регистри с общо предназначение);
PUSHF (записване на 16-разрядния флагов регистър);
PUSHFD (записване на 32-разрядния флагов регистър).

Проверка на индекс или памет

BOUND (дали индекс е в граници);
VERR (дали сегмент е достъпен за четене);
VERW (дали сегмент е достъпен за запис).

За управление на работата с паметта

LOCK (за сигнал LOOK# за монополно владеене на паметта през следващата команда);
WBINVD (обратен запис и недостоверност на кеш-паметта).

Работа с низове (вектори от байтове, думи или двойни думи)

CMPS (изваждане на елементи от низове с цел сравняване);
CMPSB (изваждане за сравняване на байтове от низове);
CMPSD (изваждане за сравняване на двойни думи от низове);
CMPSW (изваждане за сравняване на думи от низове);
INS (въвеждане от порт; влияе се от привилегиите);
INSB (въвеждане от порт на байт; влияе се от привилегиите);
INSD (въвеждане от порт на двойна дума; влияе се от привилегиите);
INSW (въвеждане от порт на дума; влияе се от привилегиите);
LODS (за запис на елемент в акумулатора);
LODSB (за запис на байт в акумулатора AL);
LODSD (за запис на двойна дума в акумулатора EAX);
LODSW (за запис на дума в акумулатора AX);
MOVS (прехвърляне от един към друг низ);
MOVSB (прехвърляне на байт от един към друг низ);
MOVSD (прехвърляне на двойна дума от един към друг низ);
MOVSW (прехвърляне на дума от един към друг низ);
OUTS (записване на един елемент в порт; влияе се от привилегиите);
OUTSB (записване на байт в порт; влияе се от привилегиите);
OUTSD (записване на двойна дума в порт; влияе се от привилегиите);
OUTSW (записване на дума в порт; влияе се от привилегиите);
REP (повторение ECX пъти);
REPE (повторение ECX пъти при ZF);
REPNE (повторение ECX пъти при не ZF);
REPNZ (=REPNE);
REPZ (=REPE);
SCAS (сравнява чрез изваждане акумулатора с елемент на низ);
SCASB (сравнява чрез изваждане на акумулатора AL с байт от низ);
SCASD (сравнява чрез изваждане на акумулатора EAX с двойна дума от низ);
SCASW (сравнява чрез изваждане на акумулатора AX с дума от низ);
STOS (записва акумулатора в елемент на низ) ;
STOSB (записва AL в елемент на низ от байтове) ;
STOSD (записва EAX в елемент на низ от двойни думи);
STOSW (записва AX в елемент на низ от думи).

Команди за системно програмиране

ARPL (промяна привилегия);
CLTS (нулиране на флага TS в регистъра CR0);
FWAIT (преди да продължи процесорът проверява за някои особени случаи);
HLT (спиране на процесора до възникване на прекъсване);
IN (въвеждане от порт);
INT (предизвикване на избрано прекъсване);
INTO (предизвикване на прекъсване 4);
INVD (недостоверност на кеш-паметта);
INVLPG (недостоверност на елемент от TLB);
IRET (връщане от прекъсване);
IRETD (=IRET);
LAR (зареждане на права за достъп);
LGDT (зареждане на регистъра GDTR);
LIDT (зареждане на регистъра IDTR);
LLDT (зареждане на регистъра LDTR);
LMSW (зареждане на думата на състоянието);
LTR (зареждане регистъра на задачата);
OUT (извеждане в порт);
SGDT (извличане на съдържанието на регистъра GDTR);
SIDT (извличане на съдържанието на регистъра IDTR);
SLDT (извличане на съдържанието на регистъра LDTR);
SMSW (извличане на думата на състоянието, т. е. на младшата половина на регистъра CR0);
STR (съхраняване на регистъра TR на задачата);
WAIT (преди да продължи процесорът проверява за някои особени случаи).

Също така от привилегиите зависят и командите:
CLI; CLTS; INS; INSB; INSD; INSW; IRET; IRETD; LSL; OUTS; OUTSB; OUTSD; OUTSW; POPFD; STI.