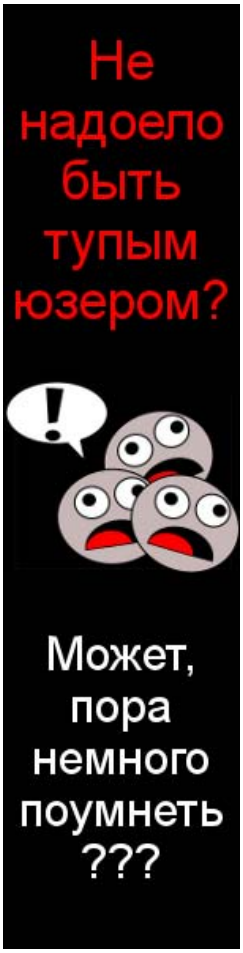

Полный набор команд процессора 8086

Скачать бесплатно справочную систему для emu8086 на русском языке

Быстрые ссылки:

	CMPSB				MOV		
AAA	CMPSW	JAE	JNBE	JPO	MOVSB	RCR	SCASB
AAD	CWD	JB	JNC	JS	MOVSW	REP	SCASW
AAM	DAA	JBE	JNE	JZ	MUL	REPE	SHL
AAS	DAS	JC	JNG	LAHF	NEG	REPNE	SHR
ADC	DEC	JCXZ	JNGE	LDS	NOP	REPNZ	STC
ADD	DIV	JE	JNL	LEA	NOT	REPZ	STD
AND	HLT	JG	JNLE	LES	OR	RET	STI
CALL	IDIV	JGE	JNO	LODSB	OUT	RETF	STOSB
CBW	IMUL	JL	JNP	LODSW	POP	ROL	STOSW
CLC	IN	JLE	JNS	LOOP	POPA	ROR	SUB
CLD	INC	JMP	JNZ	LOOPE	POPF	SAHF	TEST
CLI	INT	JNA	JO	LOOPNE	PUSH	SAL	XCHG
CMC	INTO	JNAE	JP	LOOPNZ	PUSHA	SAR	XLATB
CMP	IRET	JNB	JPE	LOOPZ	PUSHF	SBB	XOR
	JA				RCL		

Типы операндов:

REG: AX, BX, CX, DX, AH, AL, BL, BH, CH, CL, DH, DL, DI, SI, BP, SP.

SREG: DS, ES, SS, и только как второй операнд: CS.

memory: [BX], [BX+SI+7], переменная и т.п...(см. **Доступ к памяти**).

immediate: 5, -24, 3Fh, 10001101b, и т.п...

Примечания:

- Если для команды требуются два операнда, то они разделяются запятой. Например:

  REG, memory

- Если это два операнда, то они должны иметь одинаковый размер (кроме команд сдвига и циклического сдвига). Например:

  ```
  AL, DL
  DX, AX
  m1 DB ?
  AL, m1
  m2 DW ?
  AX, m2
  ```

- Некоторые команды допускают несколько комбинаций операндов. Например:

  memory, immediate
  REG, immediate

  memory, REG
  REG, SREG

- Некоторые примеры содержат макросы, поэтому желательно использовать "горячую клавишу" **Shift + F8**, чтобы **Пропустить шаг** (чтобы выполнить код макроса с максимальной скоростью, установите **step delay - задержку между выполнением команд** на ноль), иначе эмулятор будет выполнять каждую команду макроса. Ниже приведен пример, который использует макрос PRINTN:

  ```
  #make_COM#
  include 'emu8086.inc'
  ORG 100h
  MOV AL, 1
  MOV BL, 2
  PRINTN 'Hello World!'   ; макрос.
  MOV CL, 3
  PRINTN 'Welcome!'       ; макрос.
  RET
  ```

Эти знаки используются для отображения состояния флагов:

1 - команда устанавливает этот флаг в **1**.
0 - команда устанавливает этот флаг в **0**.
r - значение флага зависит от результата выполнения команды.
? - значение флага не определено (может быть **1** или **0**).

Некоторые команды генерируются в абсолютно одинаковый машинный код, поэтому дизассемблер может иметь проблемы при декодировании вашего оригинального кода. Это особенно важно в командах условного перехода

(см. "Управление ходом программы").

Команды в алфавитном порядке:

Команда	Операнды	Описание
AAA	Нет операндов	Коррекция ASCII-формата после сложения. Корректирует результат в регистрах AH и AL после сложения при работе с BCD-значениями. Она работает согласно следующему алгоритму Если младшие (правые) четыре бита регистра AL > 9 или флаг AF = 1, то: • AL = AL + 6 • AH = AH + 1 • AF = 1 • CF = 1 иначе • AF = 0 • CF = 0 в любом случае: очистить старшие четыре бита регистра AL. Пример: MOV AX, 15 ; AH = 00, AL = 0Fh AAA ; AH = 01, AL = 05 RET C Z S O P A r ? ? ? ? r
AAD	Нет операндов	Коррекция ASCII-формата перед делением. Подготавливает два BCD-значения для деления. Алгоритм: • AL = (AH * 10) + AL • AH = 0 Пример: MOV AX, 0105h ; AH = 01, AL = 05 AAD ; AH = 00, AL = 0Fh (15) RET C Z S O P A ? r r ? r ?
AAM	Нет операндов	Коррекция ASCII-формата после умножения. Корректирует результат умножения двух BCD-значений. Алгоритм: • AH = AL / 10 • AL = остаток

Пример:

```
MOV AL, 15   ; AL = 0Fh
AAM          ; AH = 01, AL = 05
RET
```

C	Z	S	O	P	A
?	r	r	?	r	?

Коррекция ASCII-формата после вычитания.
Корректирует результат в регистрах AH и AL после вычитания при работе с BCD-значениями.

Алгоритм:

если младшие четыре бита
регистра AL > 9 или AF = 1, то:

- AL = AL - 6
- AH = AH - 1
- AF = 1
- CF = 1

иначе

AAS Нет операндов

- AF = 0
- CF = 0

в любом случае:
очистить старшие четыре бита регистра AL.

Пример:

```
MOV AX, 02FFh  ; AH = 02, AL = 0FFh
AAS            ; AH = 01, AL = 09
RET
```

C	Z	S	O	P	A
r	?	?	?	?	r

Сложение с переносом.

Алгоритм:

operand1 = operand1 + operand2 + CF

ADC
REG, memory
memory, REG
REG, REG
memory, immediate
REG, immediate

Пример:

```
STC       ; установить CF = 1
MOV AL, 5  ; AL = 5
ADC AL, 1  ; AL = 7
RET
```

C	Z	S	O	P	A
r	r	r	r	r	r

ADD
REG, memory
memory, REG
REG, REG
memory, immediate
REG, immediate

Сложение.

Алгоритм:

operand1 = operand1 + operand2

Пример:

```
MOV AL, 5   ; AL = 5
ADD AL, -3  ; AL = 2
RET
```

C	Z	S	O	P	A
r	r	r	r	r	r

Логическое И между всеми битами двух операндов. Результат записывается в 1-й операнд.

Действуют следующие правила:

1 AND 1 = 1
1 AND 0 = 0
0 AND 1 = 0
0 AND 0 = 0

AND	REG, memory
	memory, REG
	REG, REG
	memory, immediate
	REG, immediate
		Пример:
		`MOV AL, 'a'  ; AL = 01100001b`
		`AND AL, 11011111b  ; AL = 01000001b  ('A')`
		`RET`

C	Z	S	O	P
0	r	r	0	r

Передает управление процедуре, заносит в стек адрес следующей команды (из IP). **4-х байтовый адрес** может быть введен в следующей форме: 1234h:5678h, первое значение - сегмент, второе значение - смещение (в случае дальнего вызова регистр CS также заносится в стек).

CALL	имя процедуры
	метк
	4-х байтовый адрес
		Пример:
		`#make_COM#`
		`ORG 100h  ; для COM-файла.`
		`CALL p1`
		`ADD AX, 1`
		`RET  ; вернуться в операционную систему.`
		`p1 PROC  ; объявление процедуры.`
		`  MOV AX, 1234h`
		`  RET  ; возвращение в программу.`
		`p1 ENDP`

C	Z	S	O	P	A
не изменяются					

CBW	Нет операндов	Преобразует байт в слово. Алгоритм: Если старший бит AL = 1, то: • AH = 255 (0FFh) иначе • AH = 0 Пример:

```
MOV AX, 0   ; AH = 0, AL = 0
MOV AL, -5  ; AX = 000FBh (251)
CBW         ; AX = 0FFFBh (-5)
RET
```

C	Z	S	O	P	A
не изменяются					

Очищает флаг переноса (CF).

Алгоритм:

CLC — Нет операндов

CF = 0

C
0

Очищает флаг направления (DF). Clear Direction flag. Значения регистров SI и DI будут увеличиваться командами: CMPSB, CMPSW, LODSB, LODSW, MOVSB, MOVSW, STOSB, STOSW.

Алгоритм:

CLD — Нет операндов

DF = 0

D
0

Очищает флаг прерывания (IF). Это отключает аппаратные прерывания.

Алгоритм:

CLI — Нет операндов

IF = 0

I
0

Переключает флаг переноса, т.е. инвертирует значение CF.

Алгоритм:

CMC — Нет операндов

если CF = 1 то CF = 0
если CF = 0 то CF = 1

C
r

CMP — REG, memory / memory, REG / REG, REG / memory, immediate / REG, immediate

Сравнение.

Алгоритм:

operand1 - operand2

результат никуда не записывается, флаги устанавливаются (OF, SF, ZF, AF, PF, CF) в соответствии с результатом.

Пример:

```
MOV AL, 5
MOV BL, 5
CMP AL, BL  ; AL = 5, ZF = 1 (значит равно!)
RET
```

C	Z	S	O	P	A
r	r	r	r	r	r

Сравнивает байты: ES:[DI] из DS:[SI].

Алгоритм:

CMPSB Нет операндов

- DS:[SI] - ES:[DI]
- установить флаги в соответствии с результатом: OF, SF, ZF, AF, PF, CF
- если DF = 0 то
  - SI = SI + 1
  - DI = DI + 1

  иначе
  - SI = SI - 1
  - DI = DI - 1

Пример:
см. **cmpsb.asm** в каталоге Samples.

C	Z	S	O	P	A
r	r	r	r	r	r

Сравнивает слова: ES:[DI] из DS:[SI].

Алгоритм:

CMPSW Нет операндов

- DS:[SI] - ES:[DI]
- установить флаги в соответствии с результатом: OF, SF, ZF, AF, PF, CF
- если DF = 0 то
  - SI = SI + 2
  - DI = DI + 2

  иначе
  - SI = SI - 2
  - DI = DI - 2

Пример:
см. **cmpsw.asm** в каталоге Samples.

C	Z	S	O	P	A
r	r	r	r	r	r

CWD Нет операндов

Преобразует слово в двойное слово.

Алгоритм:

если старший бит AX = 1, то:

- DX = 65535 (0FFFFh)

иначе

- DX = 0

Пример:

```
MOV DX, 0   ; DX = 0
MOV AX, 0   ; AX = 0
MOV AX, -5  ; DX AX = 00000h:0FFFBh
CWD         ; DX AX = 0FFFFh:0FFFBh
RET
```

C	Z	S	O	P	A
не изменяются					

Десятичная коррекция после сложения.
Корректирует результат сложения двух упакованных BCD-значений.

Алгоритм:

если младшие четыре бита AL > 9 или AF = 1, то:

- AL = AL + 6
- AF = 1

если AL > 9Fh или CF = 1, то:

- AL = AL + 60h
- CF = 1

DAA Нет операндов

Пример:

```
MOV AL, 0Fh  ; AL = 0Fh (15)
DAA          ; AL = 15h
RET
```

C	Z	S	O	P	A
r	r	r	r	r	r

Десятичная коррекция после вычитания.
Корректирует результат вычитания двух упакованных BCD-значений.

Алгоритм:

если младшие четыре бита AL > 9 или AF = 1, то:

- AL = AL - 6
- AF = 1

если AL > 9Fh или CF = 1, то:

- AL = AL - 60h
- CF = 1

DAS Нет операндов

Пример:

```
MOV AL, 0FFh  ; AL = 0FFh (-1)
DAS           ; AL = 99h, CF = 1
RET
```

C	Z	S	O	P	A
r	r	r	r	r	r

DEC REG
memory

Декремент.

Алгоритм:

operand = operand - 1

Пример:

```
MOV AL, 255  ; AL = 0FFh (255 или -1)
DEC AL       ; AL = 0FEh (254 или -2)
RET
```

Z	S	O	P	A

		r r r r r

		CF - не изменяется!
		Беззнаковое деление.
		
		Алгоритм:
		
		если операнд - это **байт**:
		AL = AX / операнд
		AH = остаток (модуль)
		
		если операнд - это **слово**:
		AX = (DX AX) / операнд
		DX = остаток (модуль)
		
		Пример:
		
		MOV AX, 203 ; AX = 00CBh
		MOV BL, 4
		DIV BL ; AL = 50 (32h), AH = 3
		RET

DIV	REG memory	
		C Z S O P A ? ? ? ? ? ?
		Останов системы. Пример: MOV AX, 5 HLT
HLT	Нет операндов	
		C Z S O P A unchanged

		Знаковое деление.
		
		Алгоритм:
		
		если операнд - это **байт**:
		AL = AX / операнд
		AH = остаток (модуль)
		
		если операнд - это **слово**:
		AX = (DX AX) / операнд
		DX = остаток (модуль)
		
		Пример:
		
		MOV AX, -203 ; AX = 0FF35h
		MOV BL, 4
		IDIV BL ; AL = -50 (0CEh), AH = -3 (0FDh)
		RET

IDIV	REG memory	
		C Z S O P A ? ? ? ? ? ?
IMUL	REG memory	Знаковое умножение. Алгоритм: если операнд - это **байт**: AX = AL * операнд.

если операнд - это **слово**:
(DX AX) = AX * операнд.

Пример:

```
MOV AL, -2
MOV BL, -4
IMUL BL      ; AX = 8
RET
```

C	Z	S	O	P	A
r	?	?	r	?	?

CF=OF=0 если результат вписывается в операнд IMUL.

IN	AL, im.byte AL, DX AX, im.byte AX, DX	Помещает данные из порта в **AL** или **AX**. Второй операнд - это номер порта. Если требуется доступ к порту с номером более 255, то нужно использовать регистр **DX**. Пример: IN AX, 4 ; получить состояние светофора. IN AL, 7 ; получить состояние шагового дигателя.

C	Z	S	O	P	A
не изменяются					

INC	REG memory	Инкремент. Алгоритм: operand = operand + 1 Пример: MOV AL, 4 INC AL ; AL = 5 RET

Z	S	O	P	A
r	r	r	r	r

CF - не изменяется!

INT	immediate byte	Выполняет прерывание программы и передает управление функции, указанной в immediate byte (0..255).
		Алгоритм:
		Поместить в стек:
		◦ флаговый регистр
		◦ CS
		◦ IP
		• IF = 0
		• Передать управление процедуре прерывания
		Пример:
		MOV AH, 0Eh ; телетайп.
		MOV AL, 'A'
		INT 10h ; Прерывание BIOS.
		RET

C	Z	S	O	P	A	I
не изменяются						0

INTO	Нет операндов	Приводит к прерыванию при возникновении переполнения (флаг OF = 1) и выполняет команду IRET 4. Алгоритм:

если OF = 1 то INT 4

Пример:

```
; -5 - 127 = -132 (за пределами диапазона -128..127)
; результат вычитания (SUB) неправильный (124),
; поэтому OF = 1:
MOV AL, -5
SUB AL, 127   ; AL = 7Ch (124)
INTO          ; ошибка процесса.
RET
```

IRET — Нет операндов

Возврат из обработки прерывания.

Алгоритм:

Выгрузить из стека:
- IP
- CS
- регистр флагов

C	Z	S	O	P	A
выгруженные					

JA — метка

Короткий переход по "выше" или "не ниже или равно". Используется после проверки беззнаковых данных для передачи управления по другому адресу.

Алгоритм:

если (CF = 0) и (ZF = 0) то выполнить переход

Пример:

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 250
  CMP AL, 5
  JA label1
  PRINT 'AL не больше 5'
  JMP exit
label1:
  PRINT 'AL больше 5'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

JAE — метка

Короткий переход, если первый операнд "больше или равен" второму операнду. (в результате выполнения команды CMP). Беззнаковый.

Алгоритм:

если CF = 0 то выполнить переход

Пример:

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 5
  CMP AL, 5
```

```
  JAE label1
  PRINT 'AL не больше 5'
  JMP exit
label1:
  PRINT 'AL больше или равен 5'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

Короткий переход, если первый операнд "меньше" второго операнда (в результате выполнения команды CMP). Беззнаковый.

Algorithm:

если CF = 1 то выполнить переход

Пример:

JB метка

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 1
  CMP AL, 5
  JB  label1
  PRINT 'AL не меньше 5'
  JMP exit
label1:
  PRINT 'AL меньше 5'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

Короткий переход, если первый операнд "меньше" или "равен" второму операнду (в результате выполнения команды CMP). Беззнаковый.

Алгоритм:

если CF = 1 или ZF = 1 то выполнить переход

Пример:

JBE метка

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 5
  CMP AL, 5
  JBE  label1
  PRINT 'AL не меньше 5'
  JMP exit
label1:
  PRINT 'AL меньше или равен 5'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

JC метка

Короткий переход если флаг переноса установлен в 1.

Алгоритм:

если CF = 1 то выполнить переход

Пример:

```
include 'emu8086.inc'
#make_COM#
ORG 100h
MOV AL, 255
ADD AL, 1
JC  label1
PRINT 'нет переноса.'
JMP exit
label1:
PRINT 'имеем перенос.'
exit:
RET
```

C	Z	S	O	P	A
не изменяются					

Короткий переход, если регистр CX равен 0.

Algorithm:

если CX = 0 то выполнить переход

Пример:

JCXZ метка

```
include 'emu8086.inc'
#make_COM#
ORG 100h
MOV CX, 0
JCXZ label1
PRINT 'CX не равен нулю.'
JMP exit
label1:
PRINT 'CX равен нулю.'
exit:
RET
```

C	Z	S	O	P	A
не изменяются					

JE метка

Короткий переход, если первый операнд равен второму операнду (в результате выполнения команды CMP). Знаковый/Беззнаковый.

Алгоритм:

если ZF = 1 то выполнить переход

Пример:

```
include 'emu8086.inc'
#make_COM#
ORG 100h
MOV AL, 5
CMP AL, 5
JE  label1
PRINT 'AL не равен 5.'
JMP exit
label1:
PRINT 'AL равен 5.'
exit:
RET
```

C	Z	S	O	P	A

не изменяются

Короткий переход, если первый операнд больше второго операнда (в результате выполнения команды CMP). Знаковый.

Алгоритм:

если (ZF = 0) и (SF = OF) то выполнить переход

Пример:

JG метка

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 5
  CMP AL, -5
  JG  label1
  PRINT 'AL не больше -5.'
  JMP exit
label1:
  PRINT 'AL больше -5.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

Короткий переход, если первый операнд больше или равен второму операнду (в результате выполнения команды CMP). Знаковый.

Алгоритм:

если SF = OF то выполнить переход

Пример:

JGE метка

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 2
  CMP AL, -5
  JGE  label1
  PRINT 'AL < -5'
  JMP exit
label1:
  PRINT 'AL >= -5'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

JL метка

Короткий переход, если первый операнд меньше второго операнда (в результате выполнения команды CMP). Знаковый.

Алгоритм:

если SF <> OF то выполнить переход

Пример:

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, -2
```

```
  CMP AL, 5
  JL  label1
  PRINT 'AL >= 5.'
  JMP exit
label1:
  PRINT 'AL < 5.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

Короткий переход, если первый операнд меньше или равен второму операнду (в результате выполнения команды CMP). Знаковый.

Алгоритм:

если SF <> OF или ZF = 1 то выполнить переход

Пример:

JLE метка

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, -2
  CMP AL, 5
  JLE label1
  PRINT 'AL > 5.'
  JMP exit
label1:
  PRINT 'AL <= 5.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

Безусловный переход. Передает управление другому участку программы. **4-х байтовый адрес** может быть введен в такой форме: 1234h:5678h, первое значение - сегмент, второе значение - смещение.

Алгоритм:

выполнить переход в любом случае

Пример:

JMP метка
4-х байтовый адрес

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 5
  JMP label1    ; "перешагнуть" через две строки!
  PRINT 'Нет перехода!'
  MOV AL, 0
label1:
  PRINT 'Добрались сюда!'
  RET
```

C	Z	S	O	P	A
не изменяются					

JNA метка

Короткий переход, если первый операнд не больше второго операнда (в результате выполнения команды CMP). Беззнаковый.

Алгоритм:

если CF = 1 или ZF = 1 то выполнить переход

Пример:

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 2
  CMP AL, 5
  JNA label1
  PRINT 'AL больше 5.'
  JMP exit
label1:
  PRINT 'AL не больше 5.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

JNAE метка

Короткий переход, если первый операнд не больше и не равен второму операнду (в результате выполнения команды CMP). Беззнаковый.

Алгоритм:

если CF = 1 то выполнить переход

Пример:

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 2
  CMP AL, 5
  JNAE label1
  PRINT 'AL >= 5.'
  JMP exit
label1:
  PRINT 'AL < 5.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

JNB метка

Короткий переход, если первый операнд не меньше второго операнда (в результате выполнения команды CMP). Беззнаковый.

Алгоритм:

если CF = 0 то выполнить переход

Пример:

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 7
  CMP AL, 5
  JNB label1
  PRINT 'AL < 5.'
  JMP exit
label1:
```

```
  PRINT 'AL >= 5.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

Короткий переход, если первый операнд не меньше и не равен второму операнду (в результате выполнения команды CMP). Беззнаковый.

Алгоритм:

если (CF = 0) и (ZF = 0) то выполнить переход

Пример:

JNBE метка

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 7
  CMP AL, 5
  JNBE label1
  PRINT 'AL <= 5.'
  JMP exit
label1:
  PRINT 'AL > 5.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

Короткий переход, если флаг переноса установлен в ноль.

Алгоритм:

если CF = 0 то выполнить переход

Пример:

JNC метка

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 2
  ADD AL, 3
  JNC  label1
  PRINT 'имеем перенос.'
  JMP exit
label1:
  PRINT 'нет переноса.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

JNE метка

Короткий переход, если первый операнд не равен второму операнду (в результате выполнения команды CMP). Знаковый/Беззнаковый.

Алгоритм:

если ZF = 0 то выполнить переход

Пример:

```
include 'emu8086.inc'
#make_COM#
ORG 100h
MOV AL, 2
CMP AL, 3
JNE  label1
PRINT 'AL = 3.'
JMP exit
label1:
PRINT 'Al <> 3.'
exit:
RET
```

C	Z	S	O	P	A
не изменяются					

Короткий переход, если первый операнд не больше второго операнда (в результате выполнения команды CMP). Знаковый.

Алгоритм:

если (ZF = 1) и (SF <> OF) то выполнить переход

Пример:

JNG метка

```
include 'emu8086.inc'
#make_COM#
ORG 100h
MOV AL, 2
CMP AL, 3
JNG  label1
PRINT 'AL > 3.'
JMP exit
label1:
PRINT 'Al <= 3.'
exit:
RET
```

C	Z	S	O	P	A
не изменяются					

JNGE метка

Короткий переход, если первый операнд не больше и не равен второму операнду (в результате выполнения команды CMP). Знаковый.

Алгоритм:

если SF <> OF то выполнить переход

Пример:

```
include 'emu8086.inc'
#make_COM#
ORG 100h
MOV AL, 2
CMP AL, 3
JNGE  label1
PRINT 'AL >= 3.'
JMP exit
label1:
PRINT 'Al < 3.'
exit:
RET
```

C	Z	S	O	P	A
не изменяются					

Короткий переход, если первый операнд не меньше второго операнда (в результате выполнения команды CMP). Знаковый.

Алгоритм:

если SF = OF то выполнить переход

Пример:

JNL метка

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 2
  CMP AL, -3
  JNL label1
  PRINT 'AL < -3.'
  JMP exit
label1:
  PRINT 'Al >= -3.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

Короткий переход, если первый операнд не меньше и не равен второму операнду (в результате выполнения команды CMP). Знаковый.

Алгоритм:

если (SF = OF) и (ZF = 0) то выполнить переход

Пример:

JNLE метка

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 2
  CMP AL, -3
  JNLE label1
  PRINT 'AL <= -3.'
  JMP exit
label1:
  PRINT 'Al > -3.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

JNO метка

Короткий переход, если нет переполнения.

Алгоритм:

если OF = 0 то выполнить переход

Пример:

```
; -5 - 2 = -7 (в пределах -128..127)
; результат команды SUB (вычитание) правильный,
; поэтому OF = 0:

include 'emu8086.inc'
#make_COM#
```

```
ORG 100h
  MOV AL, -5
  SUB AL, 2   ; AL = 0F9h (-7)
JNO  label1
  PRINT 'переполнение!'
JMP exit
label1:
  PRINT 'нет переполнения.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяется					

Короткий переход, если нет паритета, или паритет нечетный. Проверяются только младшие 8 битов результата. Устанавливается командами CMP, SUB, ADD, TEST, AND, OR, XOR.

Алгоритм:

если PF = 0 то выполнить переход

Example:

JNP метка

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 00000111b   ; AL = 7
  OR  AL, 0           ; только установка флагов.
  JNP label1
  PRINT 'паритет четный.'
  JMP exit
label1:
  PRINT 'паритет нечетный.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

Короткий переход, если нет знака (если положительный). Устанавливается командами CMP, SUB, ADD, TEST, AND, OR, XOR.

Алгоритм:

если SF = 0 то выполнить переход

Пример:

JNS метка

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 00000111b   ; AL = 7
  OR  AL, 0           ; только установка флагов.
  JNS label1
  PRINT 'есть знак.'
  JMP exit
label1:
  PRINT 'нет знака.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

JNZ метка

Короткий переход, если "не ноль". Устанавливается командами CMP, SUB, ADD, TEST, AND, OR, XOR.

Алгоритм:

если ZF = 0 то выполнить переход

Пример:

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 00000111b   ; AL = 7
  OR  AL, 0           ; только установка флагов
  JNZ label1
  PRINT 'ноль.'
  JMP exit
label1:
  PRINT 'не ноль.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

Короткий переход по переполнению.

Алгоритм:

если OF = 1 то выполнить переход

Пример:

```
; -5 - 127 = -132 (вне диапазона -128..127)
; результат вычитания неправильный (124),
; поэтому OF = 1:

include 'emu8086.inc'
#make_COM#
org 100h
  MOV AL, -5
  SUB AL, 127   ; AL = 7Ch (124)
JO  label1
  PRINT 'нет переполнения.'
JMP exit
label1:
  PRINT 'переполнение!'
exit:
  RET
```

JO метка

C	Z	S	O	P	A
не изменяются					

JP метка

Короткий переход, если есть паритет или паритет четный. Проверяются только младшие 8 битов результата. Устанавливается командами CMP, SUB, ADD, TEST, AND, OR, XOR.

Алгоритм:

если PF = 1 то выполнить переход

Пример:

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
```

```
  MOV AL, 00000101b   ; AL = 5
  OR  AL, 0           ; только установка флагов.
  JP label1
  PRINT 'паритет нечетный.'
  JMP exit
label1:
  PRINT 'паритет четный.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

Короткий переход, если есть паритет или паритет четный. Проверяются только младшие 8 битов результата. Устанавливается командами CMP, SUB, ADD, TEST, AND, OR, XOR.

Алгоритм:

если PF = 1 то выполнить переход

Пример:

JPE метка

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 00000101b   ; AL = 5
  OR  AL, 0           ; только установка флагов.
  JPE label1
  PRINT 'паритет нечетный.'
  JMP exit
label1:
  PRINT 'паритет четный.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

Короткий переход, если нет паритета, или паритет нечетный. Проверяются только младшие 8 битов результата. Устанавливается командами CMP, SUB, ADD, TEST, AND, OR, XOR.

Алгоритм:

если PF = 0 то выполнить переход

Пример:

JPO метка

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 00000111b   ; AL = 7
  OR  AL, 0           ; только установка флагов.
  JPO label1
  PRINT 'паритет четный.'
  JMP exit
label1:
  PRINT 'паритет нечетный.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

JS метка

Переход по знаку (если отрицательный). Устанавливается командами CMP, SUB, ADD,

TEST, AND, OR, XOR.

Алгоритм:

если SF = 1 то выполнить переход

Пример:

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 10000000b   ; AL = -128
  OR  AL, 0           ; только установка флагов.
  JS label1
  PRINT 'нет знака.'
  JMP exit
label1:
  PRINT 'есть знак.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

JZ	метка	Короткий переход по "нулю". Устанавливается командами CMP, SUB, ADD, TEST, AND, OR, XOR.

Алгоритм:

если ZF = 1 то выполнить переход

Пример:

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AL, 5
  CMP AL, 5
  JZ  label1
  PRINT 'AL не равно 5.'
  JMP exit
label1:
  PRINT 'AL равно 5.'
exit:
  RET
```

C	Z	S	O	P	A
не изменяются					

LAHF	Нет операндов	Загрузка младших 8 байтов регистра флагов в регистр AH.

Алгоритм:

AH = флаговому регистру

Биты AH: 7 6 5 4 3 2 1 0
[SF] [ZF] [0] [AF] [0] [PF] [1] [CF]

биты 1, 3, 5 зарезервированы.

C	Z	S	O	P	A
не изменяются					

Загружает в необходимые регистры четыре байта памяти (двойное слово), содержащей относительный адрес и сегментный адрес. Сегментный адрес помещается в регистр DS, а относительный адрес (смещение) - в любой из общих или индексных регистров или в регистровый указатель.

Алгоритм:

- REG = первое слово
- DS = второе слово

Пример:

LDS REG, memory

```
#make_COM#
ORG 100h

LDS AX, m

RET

m  DW  1234h
   DW  5678h

END
```

В AX записано значение 1234h, в DS записано значение 5678h.

C	Z	S	O	P	A
не изменяются					

LEA REG, memory

Загрузка исполнительного адреса.
Команда LEA загружает в регистр, указанный в команде в качестве первого операнда, относительный адрес второго операнда (не значение операнда!). В качестве первого операнда следует указывать регистр общего назначения (не сегментный), в качестве второго - ячейку памяти. Команда

LEA reg, mem

эквивалентна команде

MOV reg,offset mem

но у первой команды больше возможностей описания адреса интересующей нас ячейки. Команда не воздействует на флаги процессора.

Алгоритм:

- REG = адрес памяти (смещение)

Обычно эту команду заменяют командой MOV, если это возможно.

Пример:

```
#make_COM#
ORG 100h

LEA AX, m

RET

m  DW  1234h
```

END

В AX записано значение: 0104h.
Команда LEA занимает 3 байта, команда RET занимает 1 байт, мы начинаем с адреса 100h, поэтому адрес 'm' - это 104h.

C	Z	S	O	P	A
не изменяются					

Загрузка указателя с использованием регистра ES.
Считывает из памяти по указанному адресу двойное слово (32 бит), содержащее указатель (полный адрес некоторой ячейки), и загружает младшую половину указателя (т.е. относительный адрес) в указанный в команде регистр, а старшую половину указателя (т.е. сегментный адрес) в регистр ES. Таким образом, команда

LES reg, mem

эквивалентна следующей группе команд:

MOV reg, word ptr mem
MOV ES, word ptr mem+2

В качестве первого операнда команды LES указывается регистр общего назначения; в качестве второго - ячейка памяти с двухсловным содержимым. Указатель, содержащийся в этой ячейке, может быть адресом как процедуры, так и поля данных.

LES REG, memory

Алгоритм:

- REG = первое слово
- ES = второе слово

Пример:

```
#make_COM#
ORG 100h

LES AX, m

RET

m  DW  1234h
   DW  5678h

END
```

В AX записано значение 1234h, В ES записано значение 5678h.

C	Z	S	O	P	A
не изменяются					

LODSB Нет операндов

Загружает байт из DS:[SI] в регистр AL. Изменяет SI.

Алгоритм:

- AL = DS:[SI]
- если DF = 0 то
  - SI = SI + 1

  иначе
  - SI = SI - 1

Пример:

```
#make_COM#
ORG 100h

LEA SI, a1
MOV CX, 5
MOV AH, 0Eh

m: LODSB
INT 10h
LOOP m

RET

a1 DB 'H', 'e', 'l', 'l', 'o'
```

C	Z	S	O	P	A
не изменяются					

Загрузить слово из DS:[SI] в регистр AX. Изменяет SI.

Алгоритм:

- AX = DS:[SI]
- если DF = 0 то
  - SI = SI + 2

  иначе
  - SI = SI - 2

Пример:

LODSW Нет операндов

```
#make_COM#
ORG 100h

LEA SI, a1
MOV CX, 5

REP LODSW   ; в итоге в AX будет значение 555h.

RET

a1 dw 111h, 222h, 333h, 444h, 555h
```

C	Z	S	O	P	A
не изменяются					

LOOP метка

Уменьшает CX, переходит на метку, если CX не равен нулю.

Алгоритм:

- CX = CX - 1
- если CX <> 0 то
  - выполнить переход

  иначе
  - не выполнять переход, продолжить цикл

Пример:

```
  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV CX, 5
label1:
  PRINTN 'цикл!'
```

```
  LOOP label1
  RET
```

C	Z	S	O	P	A
не изменяются					

Уменьшить CX, выполнить переход, если CX "не ноль" или "равно". (ZF = 1).

Алгоритм:

- CX = CX - 1
- если (CX <> 0) и (ZF = 1) то
  - выполнить переход

  иначе
  - не выполнять переход, продолжить цикл

Пример:

```
; Цикл выполняется 5 раз или до тех пор, пока
; результат в регистре AL лежит в пределах байта.
; Результат превысит значение 255 на третьем "заходе" (100+100+100),
; поэтому цикл будет завершен.

  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AX, 0
  MOV CX, 5
label1:
  PUTC '*'
  ADD AX, 100
  CMP AH, 0
  LOOPE label1
  RET
```

LOOPE метка

C	Z	S	O	P	A
не изменяются					

LOOPNE метка

Уменьшает CX, выполняет переход к метке если CX не равен нулю и выполняется условие "не равно" (ZF = 0).

Алгоритм:

- CX = CX - 1
- если (CX <> 0) и (ZF = 0) то
  - выполнить переход

  иначе
  - не выполнять переход, продолжить цикл

Пример:

```
; Цикл выполняется 5 раз, или до тех пор,
; пока не будет найдено число '7'.

  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV SI, 0
  MOV CX, 5
label1:
  PUTC '*'
  MOV AL, v1[SI]
  INC SI        ; следующий байт (SI=SI+1).
  CMP AL, 7
  LOOPNE label1
```

```
  RET
  v1 db 9, 8, 7, 6, 5
```

C	Z	S	O	P	A
не изменяются					

Уменьшает CX, выполняет переход к метке, если CX не равен нулю и ZF = 0.

Алгоритм:

- CX = CX - 1
- если (CX <> 0) и (ZF = 0) то
  - выполнить переход

  иначе
  - не выполнять переход, продолжить цикл

Пример:

```
; Цикл выполняется 5 раз, или до тех пор,
; пока не будет найдено число '7'.

  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV SI, 0
  MOV CX, 5
label1:
  PUTC '*'
  MOV AL, v1[SI]
  INC SI        ; следующий байт (SI=SI+1).
  CMP AL, 7
  LOOPNZ label1
  RET
  v1 db 9, 8, 7, 6, 5
```

LOOPNZ метка

C	Z	S	O	P	A
не изменяются					

LOOPZ метка

Уменьшает CX, выполняет переход к метке, если CX не равен нулю и ZF = 1.

Алгоритм:

- CX = CX - 1
- если (CX <> 0) и (ZF = 1) то
  - выполнить переход

  иначе
  - не выполнять переход, продолжить цикл

Пример:

```
; Цикл выполняется 5 раз или до тех пор,
; пока значение в регистре AL не выходит за пределы байта.
; Результат превысит значение 255 на третьем цикле (100+100+100),
; поэтому цикл будет завершен после третьего прохода.

  include 'emu8086.inc'
  #make_COM#
  ORG 100h
  MOV AX, 0
  MOV CX, 5
label1:
  PUTC '*'
  ADD AX, 100
  CMP AH, 0
  LOOPZ label1
  RET
```

C	Z	S	O	P	A
не изменяются					

Копирует operand2 в operand1.

Команда MOV не может:

- записывать данные в регистры CS и IP.
- копировать данные из одного сегментного регистра в другой сегментный регистр (сначала нужно скопировать данные в регистр общего назначения).
- копировать непосредственное значение в сегментный регистр (сначала нужно скопировать данные в регистр общего назначения).

MOV

REG, memory
memory, REG
REG, REG
memory, immediate
REG, immediate

SREG, memory
memory, SREG
REG, SREG
SREG, REG

Алгоритм:

operand1 = operand2

Пример:

```
#make_COM#
ORG 100h
MOV AX, 0B800h    ; установить AX = B800h (память VGA).
MOV DS, AX        ; копировать значение из AX в DS.
MOV CL, 'A'       ; CL = 41h (ASCII-код).
MOV CH, 01011111b ; CL = атрибуты цвета.
MOV BX, 15Eh      ; BX = позиция на экране.
MOV [BX], CX      ; w.[0B800h:015Eh] = CX.
RET               ; вернуться в операционную систему.
```

C	Z	S	O	P	A
не изменяются					

Копирует байт из DS:[SI] в ES:[DI]. Изменяет регистры SI и DI.

Алгоритм:

- ES:[DI] = DS:[SI]
- если DF = 0 то
  - SI = SI + 1
  - DI = DI + 1

  иначе
  - SI = SI - 1
  - DI = DI - 1

Пример:

MOVSB Нет операндов

```
#make_COM#
ORG 100h

LEA SI, a1
LEA DI, a2
MOV CX, 5
REP MOVSB

RET

a1 DB 1,2,3,4,5
a2 DB 5 DUP(0)
```

C	Z	S	O	P	A
не изменяются					

MOVSW Нет операндов

Копирует **слово** из DS:[SI] в ES:[DI]. Изменяет регистры SI и DI.

Алгоритм:

- ES:[DI] = DS:[SI]
- если DF = 0 то
  - SI = SI + 2
  - DI = DI + 2

  иначе
  - SI = SI - 2
  - DI = DI - 2

Пример:

```
#make_COM#
ORG 100h

LEA SI, a1
LEA DI, a2
MOV CX, 5
REP MOVSW

RET

a1 DW 1,2,3,4,5
a2 DW 5 DUP(0)
```

C	Z	S	O	P	A
не изменяются					

Беззнаковое умножение.

Алгоритм:

если операнд - **byte**:
AX = AL * операнд.

если операнд - **word**:
(DX AX) = AX * операнд.

MUL — REG, memory

Пример:

```
MOV AL, 200   ; AL = 0C8h
MOV BL, 4
MUL BL        ; AX = 0320h (800)
RET
```

C	Z	S	O	P	A
r	?	?	r	?	?

CF=OF=0 если старшая секция результата - ноль.

Отрицание. Делает операнд отрицательным (дополнение до двух).

Алгоритм:

- Инвертировать все биты операнда
- Прибавить единицу к инвертированному операнду

NEG — REG, memory

Пример:

```
MOV AL, 5   ; AL = 05h
NEG AL      ; AL = 0FBh (-5)
NEG AL      ; AL = 05h (5)
RET
```

C	Z	S	O	P	A
r	r	r	r	r	r

NOP	Нет операндов	Нет операции. Обычно используется для небольшой задержки программы. Алгоритм: • Ничего не делать Пример: ; ничего не делать три раза: NOP NOP NOP RET C Z S O P A не изменяются
NOT	REG memory	Инвертирует каждый бит операнда. Алгоритм: • если бит равен 1, переключить его в 0. • если бит равен 0, переключить его в 1. Пример: MOV AL, 00011011b NOT AL ; AL = 11100100b RET C Z S O P A не изменяются
OR	REG, memory memory, REG REG, REG memory, immediate REG, immediate	Логическое ИЛИ между всеми битами двух операндов. Результат записывается в первый операнд. Выполняются следующие правила: 1 OR 1 = 1 1 OR 0 = 1 0 OR 1 = 1 0 OR 0 = 0 Пример: MOV AL, 'A' ; AL = 01000001b OR AL, 00100000b ; AL = 01100001b ('a') RET C Z S O P A 0 r r 0 r ?

OUT	im.byte, AL
	im.byte, AX
	DX, AL
	DX, AX
		Выводит данные из регистра **AL** или **AX** в порт.
		Первый операнд - номер порта. Если требуется доступ к порту, номер которого превышает 255, то должен быть использован регистр **DX**.
		
		Пример:
		
		MOV AX, 0FFFh ; Включить все
		OUT 4, AX ; светофоры.
		
		MOV AL, 100b ; Включть третий магнит
		OUT 7, AL ; шагового двигателя.
		
		┌┬┬┬┬┬┐

C	Z	S	O	P	A
не изменяются					

Получает 16-битовое значение из стека.

Алгоритм:

- операнд = SS:[SP] (вершина стека)
- SP = SP + 2

POP	REG SREG memory	Пример: MOV AX, 1234h PUSH AX POP DX ; DX = 1234h RET

C	Z	S	O	P	A
не изменяются					

Выгружает все регистры общего назначения DI, SI, BP, SP, BX, DX, CX, AX из стека. (Значение SP игнорируется, оно выгружается, но не записывается в регистр SP).

Примечание: эта команда работает только на процессорах **80186** и выше!

Алгоритм:

POPA	Нет операндов	• POP DI • POP SI • POP BP • POP xx (значение SP игнорируется) • POP BX • POP DX • POP CX • POP AX

C	Z	S	O	P	A
не изменяются					

Получает регистр флагов из стека.

Алгоритм:

POPF	Нет операндов	• флаги = SS:[SP] (вершина стека) • SP = SP + 2

C	Z	S	O	P	A
выгружаются из стека					

PUSH	REG SREG memory immediate	Записывает 16-битовое значение в стек. Примечание: **PUSH immediate** работает только на процессорах 80186 и выше! Алгоритм: • SP = SP - 2 • SS:[SP] (вершина стека) = операнд Пример:

```
MOV AX, 1234h
PUSH AX
POP  DX     ; DX = 1234h
RET
```

C	Z	S	O	P	A
не изменяются					

Помещает в стек все регистры общего назначения: AX, CX, DX, BX, SP, BP, SI, DI.
Использует оригинальное значение регистра SP (перед выполнением PUSHA).

Примечание: эта команда работает только на процессорах 80186 и выше!

Алгоритм:

PUSHA — Нет операндов

- PUSH AX
- PUSH CX
- PUSH DX
- PUSH BX
- PUSH SP
- PUSH BP
- PUSH SI
- PUSH DI

C	Z	S	O	P	A
не изменяются					

Записывает регистр флагов в стек.

Алгоритм:

PUSHF — Нет операндов

- SP = SP - 2
- SS:[SP] (вершина стека) = флаги

C	Z	S	O	P	A
не изменяются					

Циклический сдвиг (ротация) влево через перенос. Количество ротаций устанавливается во втором операнде.
Если **immediate** больше единицы, ассемблер генерирует несколько команд **RCL xx, 1**, потому что 8086 имеет машинный код только для этой команды (тот же принцип работы используют все команды сдвига/ротации).

Алгоритм:

RCL — memory, immediate; REG, immediate; memory, CL; REG, CL

самый левый бит записать во флаг CF, сдвинуть все биты влево, значение флага CF записать в самый правый бит (бит 0).

Пример:

```
STC             ; установить перенос (CF=1).
MOV AL, 1Ch     ; AL = 00011100b
RCL AL, 1       ; AL = 00111001b,  CF=0.
RET
```

C	O
r	r

OF=0 если первый операнд сохраняет первоначальный знак (+ или -).

RCR — memory, immediate; REG, immediate

Циклический сдвиг (ротация) вправо через перенос. Количество ротаций устанавливается во втором операнде.

	memory, CL REG, CL	Алгоритм: самый правый бит (бит 0) записать во флаг CF, сдвинуть все биты вправо, значение флага CF записать в самый левый бит. Пример: STC ; установить перенос (CF=1). MOV AL, 1Ch ; AL = 00011100b RCR AL, 1 ; AL = 10001110b, CF=0. RET C O r r OF=0 если первый операнд сохраняет первоначальный знак (+ или -).
REP	нужная команда	Повторяет следующие команды MOVSB, MOVSW, LODSB, LODSW, STOSB, STOSW. Количество повторов указано в CX. Алгоритм: check_cx: если CX <> 0 то • выполнить нужную команду • CX = CX - 1 • вернуться к метке check_cx иначе • выйти из REP-цикла

REPE	нужная команда	Z
		r
		
		Повторяет следующие команды CMPSB, CMPSW, SCASB, SCASW, пока ZF = 1, но не более CX раз.
		
		Алгоритм:
		
		check_cx:
		
		если CX <> 0 то
		
		• выполнить нужную команду
		• CX = CX - 1
		• если ZF = 1 то:
		  ◦ вернуться к метке check_cx
		иначе
		  ◦ выйти из REPE-цикла
		
		иначе
		
		• выйти из REPE-цикла
		
		Пример:
		см. **cmpsb.asm** в каталоге Samples.

REPNE	нужная команда	Z r Повторяет следующие команды CMPSB, CMPSW, SCASB, SCASW, пока ZF = 0, но не более CX раз. Алгоритм:

		check_cx:
		
		если CX <> 0 то
		
		• выполнить нужную команду
		• CX = CX - 1
		• если ZF = 0 то:
		  ○ вернуться к метке check_cx
		иначе
		  ○ выйти из REPNE-цикла
		
		иначе
		
		• выйти из REPNE-цикла
		
		 

REPNZ	нужная команда	Повторяет следующие команды CMPSB, CMPSW, SCASB, SCASW, пока ZF = 0, но не более CX раз.
		
		Алгоритм:
		
		check_cx:
		
		если CX <> 0 то
		
		• выполнить нужную команду
		• CX = CX - 1
		• если ZF = 0 то:
		  ○ вернуться к метке check_cx
		иначе
		  ○ выйти из REPNZ-цикла
		
		иначе
		
		• выйти из REPNZ-цикла
		
		 

REPZ	нужная команда	Повторяет следующие команды CMPSB, CMPSW, SCASB, SCASW, пока ZF = 1, но не более CX раз.
		
		Алгоритм:
		
		check_cx:
		
		если CX <> 0 то
		
		• выполнить нужную команду
		• CX = CX - 1
		• если ZF = 1 то:
		  ○ вернуться к метке check_cx
		иначе
		  ○ выйти из REPZ-цикла
		
		иначе
		
		• выйти из REPZ-цикла
		
		 

RET	Нет операндов	Возврат из ближней процедуры.

	или четное
	immediate
		Алгоритм:
		• Получить из стека:
		◦ IP
		• если имеется операнд immediate: SP = SP + операнд
		Пример:
		#make_COM#
		ORG 100h ; для COM-файла.
		CALL p1
		ADD AX, 1
		RET ; вернуться в операционную систему.
		p1 PROC ; объявление процедуры.
		MOV AX, 1234h
		RET ; вернуться в программу.
		p1 ENDP
		C Z S O P A
		не изменяются

RETF	Нет операндов или четное immediate	Возврат из дальней процедуры. Алгоритм: • Получить из стека: ◦ IP ◦ CS • если имеется операнд immediate: SP = SP + операнд C Z S O P A не изменяются
ROL	memory, immediate REG, immediate memory, CL REG, CL	Циклический сдвиг (ротация) влево. Количество ротаций устанавливается во втором операнде. Алгоритм: самый левый бит записать во флаг CF, сдвинуть все биты влево, в самый правый бит записать флаг CF. Пример: MOV AL, 1Ch ; AL = 00011100b ROL AL, 1 ; AL = 00111000b, CF=0. RET C O r r OF=0 если первый операнд сохраняет первоначальный знак (+ или -).
ROR	memory, immediate REG, immediate memory, CL REG, CL	Циклический сдвиг (ротация) вправо. Количество ротаций устанавливается во втором операнде. Алгоритм: самый правый бит записать во флаг CF, сдвинуть все биты вправо, в самый левый бит записать флаг CF.

Пример:

```
MOV AL, 1Ch       ; AL = 00011100b
ROR AL, 1         ; AL = 00001110b,  CF=0.
RET
```

C	O
r	r

OF=0 если первый операнд сохраняет первоначальный знак (+ или -).

Записать данные из регистра AH в младшие 8 битов регистра флагов.

Алгоритм:

регистр флагов = AH

SAHF — Нет операндов

Биты AH: 7 6 5 4 3 2 1 0
[SF] [ZF] [0] [AF] [0] [PF] [1] [CF]

биты 1, 3, 5 зарезервированы.

C	Z	S	O	P	A
r	r	r	r	r	r

SAL — memory, immediate; REG, immediate; memory, CL; REG, CL

Арифметический сдвиг влево. Количество сдвигов записывается во второй операнд.

Алгоритм:

- самый левый бит записать в CF, сдвинуть все биты влево,
- в самый правый бит записать ноль.

Пример:

```
MOV AL, 0E0h      ; AL = 11100000b
SAL AL, 1         ; AL = 11000000b,  CF=1.
RET
```

C	O
r	r

OF=0 если первый операнд сохраняет первоначальный знак (+ или -).

SAR — memory, immediate; REG, immediate; memory, CL; REG, CL

Арифметический сдвиг вправо. Количество сдвигов записывается во второй операнд.

Алгоритм:

- самый правый бит записать в CF, сдвинуть все биты вправо,
- в самый правый бит записать ноль.
- Бит знака, который вставляется в самую левую позицию, имеет то же значение, что и перед сдвигом.

Пример:

```
MOV AL, 0E0h      ; AL = 11100000b
SAR AL, 1         ; AL = 11110000b,  CF=0.

MOV BL, 4Ch       ; BL = 01001100b
SAR BL, 1         ; BL = 00100110b,  CF=0.

RET
```

C	O
r	r

OF=0 если первый операнд сохраняет первоначальный знак (+ или -).

SBB — REG, memory; memory, REG — Вычитание с заемом.

	REG, REG
	memory, immediate
	REG, immediate
		Алгоритм:
		
		operand1 = operand1 - operand2 - CF
		
		Пример:
		
		STC
		MOV AL, 5
		SBB AL, 3 ; AL = 5 - 3 - 1 = 1
		
		RET
		
		C Z S O P A
		r r r r r r
		
		Сравнивает байты (ищет байт в строке): AL из ES:[DI].
		
		Алгоритм:
		
		• ES:[DI] - AL
		• установить флаги в зависимости от результата: OF, SF, ZF, AF, PF, CF
		• если DF = 0 то
		◦ DI = DI + 1
		иначе
		◦ DI = DI - 1
		
		C Z S O P A
		r r r r r r
		
		Сравнивает слова: AX из ES:[DI].
		
		Алгоритм:
		
		• ES:[DI] - AX
		• установить флаги в зависимости от результата: OF, SF, ZF, AF, PF, CF
		• если DF = 0 то
		◦ DI = DI + 2
		иначе
		◦ DI = DI - 2
		
		C Z S O P A
		r r r r r r
		
		Сдвиг влево. Количество сдвигов указывается во втором операнде. Знаковый бит рассматривается как обычный бит данных.
		
		Algorithm:
		
		• Записать самый левый бит в CF, сдвинуть все биты влево.
		• В самый правый бит записать ноль.
		
		Пример:
		
		MOV AL, 11100000b
		SHL AL, 1 ; AL = 11000000b, CF=1.
		
		RET
		
		C O
		r r
		
		OF=0 если первый операнд сохраняет первоначальный знак (+ или -).

SCASB	Нет операндов	
SCASW	Нет операндов	
SHL	memory, immediate REG, immediate memory, CL REG, CL	

SHR	memory, immediate REG, immediate memory, CL REG, CL	Сдвиг вправо. Количество сдвигов указывается во втором операнде. Знаковый бит рассматривается как обычный бит данных. Algorithm: • Записать самый правый бит в CF, сдвинуть все биты вправо. • В самый левый бит записать ноль. Пример: MOV AL, 00000111b SHR AL, 1 ; AL = 00000011b, CF=1. RET
STC	Нет операндов	 OF=0 если первый операнд сохраняет первоначальный знак (+ или -). Устанавливает флаг переноса (CF). Алгоритм: CF = 1 
STD	Нет операндов	Устанавливает флаг направления (DF). Значения регистров SI и DI уменьшаются командами: CMPSB, CMPSW, LODSB, LODSW, MOVSB, MOVSW, STOSB, STOSW. Алгоритм: DF = 1 
STI	Нет операндов	Устанавливает флаг прерываний. Это включает аппаратные прерывания. Алгоритм: IF = 1 

STOSB	Нет операндов	Записывает байт из AL в ES:[DI]. Изменяет SI.
		
		Алгоритм:
		
		• ES:[DI] = AL
		• если DF = 0 то
		  ◦ DI = DI + 1
		иначе
		  ◦ DI = DI - 1
		
		Пример:
		
		#make_COM#
		ORG 100h
		
		LEA DI, a1
		MOV AL, 12h

MOV CX, 5

REP STOSB

RET

a1 DB 5 dup(0)

C	Z	S	O	P	A
не изменяются					

Записывает слово из AX в ES:[DI]. Изменяет SI.

Алгоритм:

- ES:[DI] = AX
- если DF = 0 то
  - DI = DI + 2

  иначе
  - DI = DI - 2

Пример:

```
#make_COM#
ORG 100h

LEA DI, a1
MOV AX, 1234h
MOV CX, 5

REP STOSW

RET

a1 DW 5 dup(0)
```

C	Z	S	O	P	A
не изменяются					

STOSW	Нет операндов	

SUB	REG, memory
	memory, REG
	REG, REG
	memory, immediate
	REG, immediate
		Вычитание.
		
		Алгоритм:
		
		operand1 = operand1 - operand2
		
		Пример:
		
		MOV AL, 5
		SUB AL, 1 ; AL = 4
		
		RET

C	Z	S	O	P	A
r	r	r	r	r	r

TEST	REG, memory memory, REG REG, REG memory, immediate REG, immediate	Логическое И между всеми битами двух операндов. Не изменяет результирующий операнд, а влияет только на флаги. Задействованы следующие флаги: **ZF, SF, PF.** Выполняются следующие правила: 1 AND 1 = 1 1 AND 0 = 0 0 AND 1 = 0 0 AND 0 = 0

		Пример: MOV AL, 00000101b TEST AL, 1 ; ZF = 0. TEST AL, 10b ; ZF = 1. RET C Z S O P 0 r r 0 r Перестановка двух операндов. Алгоритм: operand1 < - > operand2 Пример:

XCHG	REG, memory
	memory, REG
	REG, REG
		MOV AL, 5
		MOV AH, 2
		XCHG AL, AH ; AL = 2, AH = 5
		XCHG AL, AH ; AL = 5, AH = 2
		RET
		
		C Z S O P A
		не изменяются
		
		Транслирует байт из таблицы.
		Копирует байт из памяти по адресу DS:[BX + беззнаковый AL] в регистр AL.
		
		Алгоритм:
		
		AL = DS:[BX + беззнаковый AL]
		
		Пример:

XLATB	Нет операндов	#make_COM# ORG 100h LEA BX, dat MOV AL, 2 XLATB ; AL = 33h RET dat DB 11h, 22h, 33h, 44h, 55h C Z S O P A не изменяются
XOR	REG, memory memory, REG REG, REG memory, immediate REG, immediate	Логическое XOR (Исключающее ИЛИ)между всеми битами двух операндов. Результат записывается в первый операнд. Выполняются следующие правила: 1 XOR 1 = 0 1 XOR 0 = 1 0 XOR 1 = 1 0 XOR 0 = 0 Пример: MOV AL, 00000111b XOR AL, 00000010b ; AL = 00000101b RET □□□□□□

C	Z	S	O	P	A
0	r	r	0	r	?