Основные инструкции

MOV	r∗	r/m∗	$\text{arg}_1 = \text{arg}_2$
	r/m∗	r∗	
	r/m∗	imm∗	
XCHG	r∗	r/m∗	$\text{arg}_1 \leftrightarrow \text{arg}_2$
	r/m∗	r∗	
MOVZX	r32	r/m16	беззнаковое расширение
	r32	r/m8	
	r16	r/m8	
MOVSX	r32	r/m16	знаковое расширение
	r32	r/m8	
	r16	r/m8	
LEA	r32	m	$\text{arg}_1 =$ адрес arg_2
CWD			знаковое расширение AX до DX:AX
CDQ			знаковое расширение EAX до EDX:EAX
LEAVE			$\Longleftrightarrow$ MOV ESP, EBP $\rightarrow$ POP EBP

Арифметические операции

ADD	r∗	r/m∗		$\text{arg}_1 \mathrel{+}= \text{arg}_2$
	r/m∗	r∗		
	r/m∗	imm∗		
	r/m∗	imm8		
SUB	r∗	r/m∗		$\text{arg}_1 \mathrel{-}= \text{arg}_2$
	r/m∗	r∗		
	r/m∗	imm∗		
	r/m∗	imm8		
ADC	r∗	r/m∗		$\text{arg}_1 \mathrel{+}= \text{arg}_2 + \text{CF}$
	r/m∗	r∗		
	r/m∗	imm∗		
	r/m∗	imm8		
SBB	r∗	r/m∗		$\text{arg}_1 \mathrel{-}= \text{arg}_2 + \text{CF}$
	r/m∗	r∗		
	r/m∗	imm∗		
	r/m∗	imm8		
IMUL	r/m∗			eDX:eAX = eAX $\cdot$ arg_1 (знаковое)
MUL	r/m∗			eDX:eAX = eAX $\cdot$ arg_1 (беззнаковое)
IMUL	r16	r/m16		$\text{arg}_1 \mathrel{*}= \text{arg}_2$
	r32	r/m32		
IMUL	r16	r/m16	imm8	$\text{arg}_1 = \text{arg}_2 \cdot \text{arg}_3$
	r32	r/m32	imm8	
	r16	r/m16	imm16	
	r32	r/m32	imm32	
IDIV	r/m∗			eAX = eDX:eAX / arg_1; eDX = eDX:eAX % arg_1 (знаковое)
DIV	r/m∗			eAX = eDX:eAX / arg_1; eDX = eDX:eAX % arg_1 (беззнаковое)
NEG	r/m∗			$\text{arg}_1 = -\text{arg}_1$

Битовые операции

AND	r∗	r/m∗	$\text{arg}_1 = \text{arg}_1 \,\&\, \text{arg}_2$
	r/m∗	r∗	
	r/m∗	imm∗	
	r/m∗	imm8	
OR	r∗	r/m∗	$\text{arg}_1 = \text{arg}_1 \vee \text{arg}_2$
	r/m∗	r∗	
	r/m∗	imm∗	
	r/m∗	imm8	
XOR	r∗	r/m∗	$\text{arg}_1 = \text{arg}_1 \oplus \text{arg}_2$
	r/m∗	r∗	
	r/m∗	imm∗	
	r/m∗	imm8	
NOT	r/m∗		$\text{arg}_1 = \neg\text{arg}_1$
SHL	r/m∗	imm8	сдвиг arg_1 влево на arg_2
	r/m∗	CL	
SHR	r/m∗	imm8	неарифметический сдвиг arg_1 вправо на arg_2
	r/m∗	CL	
SAR	r/m∗	imm8	арифметический сдвиг arg_1 вправо на arg_2
	r/m∗	CL	
ROL	r/m∗	imm8	циклический сдвиг arg_1 влево на arg_2
	r/m∗	CL	
ROR	r/m∗	imm8	циклический сдвиг arg_1 вправо на arg_2
	r/m∗	CL	
RCL	r/m∗	imm8	циклический сдвиг CF:arg_1 влево на arg_2
	r/m∗	CL	
RCR	r/m∗	imm8	циклический сдвиг arg_1:CF вправо на arg_2
	r/m∗	CL	
BT	r/m16/32	imm8	записывает в CF бит с номером (arg_2 % size) из arg_1
	r/m16	r16	
	r/m32	r32	
BTC	r/m16/32	imm8	записывает в CF бит с номером (arg_2 % size) из arg_1,
	r/m16	r16	после чего инвертирует его
	r/m32	r32	
BTR	r/m16/32	imm8	записывает в CF бит с номером (arg_2 % size) из arg_1,
	r/m16	r16	после чего обнуляет его
	r/m32	r32	
BTS	r/m16/32	imm8	записывает в CF бит с номером (arg_2 % size) из arg_1,
	r/m16	r16	после чего устанавливает его в 1
	r/m32	r32	
BSF	r16	r/m16	записывает в arg_1 номер самого младшего
	r32	r/m32	единичного бита в arg_2 (если таких нет, ZF = 1)
BSR	r16	r/m16	записывает в arg_1 номер самого старшего
	r32	r/m32	единичного бита в arg_2 (если таких нет, ZF = 1)

Сравнения и условные операторы

CMP	r∗	$^{r}/_{m}$∗	$arg_1 - arg_2 \rightarrow$ флаги
	$^{r}/_{m}$∗	r∗	
	$^{r}/_{m}$∗	imm∗	
	$^{r}/_{m}$∗	imm8	
TEST	$^{r}/_{m}$∗	r∗	arg_1 & $arg_2 \rightarrow$ флаги
	$^{r}/_{m}$∗	imm∗	
JMP	rel32		переход на arg_1
Jcc	rel32		переход на arg_1, если cc выполняется
SETcc	$^{r}/_{m}$8		$arg_1 =$ cc (0 или 1)
CMOVcc	r16	$^{r}/_{m}$16	$arg_1 = arg_2$, если cc выполняется
	r32	$^{r}/_{m}$32	
JECXZ	rel8		переход на arg_1, если ECX = 0
LOOP	rel8		$\Longleftrightarrow$ DEC ECX $\rightarrow$ CMP ECX, 0 $\rightarrow$ JG arg_1

Строковые инструкции

CLD	DF = 0 (ESI и EDI будут увеличиваться)
STD	DF = 1 (ESI и EDI будут уменьшаться)
CMPS[BWD]	$\sim$ CMP [EDI], [ESI]
LODS[BWD]	$\sim$ MOV eAX, [ESI]
MOVS[BWD]	$\sim$ MOV [EDI], [ESI]
SCAS[BWD]	$\sim$ CMP eAX, [EDI]
STOS[BWD]	$\sim$ MOV [EDI], eAX

Префиксы повторения

```
BEG:
  if ECX==0 then goto END
  ECX = ECX - 1
  выполнение инструкции
  REPE  => if ZF==0 then goto END  // повторять, пока равно
  REPNE => if ZF==1 then goto END  // повторять, пока не равно
  goto BEG
END:
```

Операции над числами с плавающей точкой

FLD	$m^{32}/_{64}$fp	загрузить arg_1 в стек регистров
FILD	$m^{32}/_{64}$int	загрузить arg_1 в стек регистров, перекодировав
FST[P]	$m^{32}/_{64}$fp	выгрузить ST0 в arg_1
FADDP		$ST1 = ST1 + ST0$, pop
FSUBP		$ST1 = ST1 - ST0$, pop
FSUBRP		$ST1 = ST0 - ST1$, pop
FMULP		$ST1 = ST1 * ST0$, pop
FDIVP		$ST1 = ST1 / ST0$, pop
FDIVRP		$ST1 = ST0 / ST1$, pop
FSIN		$ST0 = \sin(ST0)$
FCOS		$ST0 = \cos(ST0)$
FSQRT		$ST0 = \sqrt{ST0}$
F2XM1		$ST0 = 2^{ST0} - 1$
FPTAN		$ST0 = \mathrm{tg}(ST0)$, push 1
FABS		$ST0 = \lvert ST0 \rvert$
FLD1		push +1.0
FLDL2T		push $\log_2 10$
FLDL2E		push $\log_2 e$
FLDPI		push π
FLDLG2		push $\log_{10} 2$
FLDLN2		push $\log_e 2$
FLDZ		push +0.0
FUCOMI[P]	ST0, STi	CMP arg_1, arg_2 $\rightarrow$ EFLAGS