

```
.HEAD 1 '@UAPXИТЕКТУРА          ОПИСАНИЕ СИСТЕМЫ КОМАНД@u'
.HEAD 0 '@UAPXИТЕКТУРА          ОПИСАНИЕ СИСТЕМЫ КОМАНД@u'
.TAIL 1 '@A%62d@a'
.TAIL 0 '@A%62d@a'
```

ЧАСТЬ III. ОПИСАНИЕ СИСТЕМЫ КОМАНД

Данное описание не является самостоятельным документом. Можно рассматривать его как подробный и пространный комментарий к интерпретатору системы команд процессоров КРОНОС. Здесь опущены определения основных понятий архитектуры семейства КРОНОС, их можно найти в разделе "Виртуальная Модуль-2 машина".

Код команды

Исполнение любой команды начинается с выборки ее кода. Код команды имеет размер 8 бит. Он интерпретируется как номер команды и полностью определяет алгоритм исполнения команды. В нем нет никаких полей или способов адресации. Код команды - это только байт и больше ничего.

Операнды

После выборки кода команды определен алгоритм ее исполнения. Некоторые операнды команд располагаются на стеке выражений (А-стек), другие могут следовать непосредственно за кодом команды, как последовательность байтов. Последние называются непосредственными операндами. Они никогда не интерпретируются как абсолютные адреса.

Выборка кода

Выборка кода производится с помощью процедур Next, Next2, Next4 (см. Интерпретатор системы команд). Эти процедуры выбирают из кода байт, два байта и слово, увеличивая, соответственно, счетчик команд PC на один, два или четыре. В терминах Модуль-2 эти процедуры выглядят так:

```
PROCEDURE Next(): BYTE;
BEGIN INC(PC); RETURN INTEGER(F^[PC-1])
END Next;
```

```
PROCEDURE Next2(): WORD16;
BEGIN RETURN Next()+Next()*100h
END Next2;
```

```
PROCEDURE Next4(): WORD;
BEGIN RETURN Next2()+Next2()*10000h
END Next4;
```

Стек выражений

Стек выражений используется для вычисления выражений и хранения аргументов команд и результатов их выполнения. Работа со стеком определяется процедурами Pop (взять со стека) и Push (положить на стек).

```

PROCEDURE Push(X: WORD);
BEGIN A[sp]:=X;
      IF sp<ESdepth THEN INC(sp) ELSE Ipt:=TRUE; IptNo:=4Ch END;
END Push;

```

```

PROCEDURE Pop(): INTEGER;
BEGIN
    IF sp=0 THEN Ipt:=TRUE; IptNo:=4Ch ELSE DEC(sp) END;
    RETURN A[sp];
END Pop;

```

Везде в дальнейшем под словами "взять со стека", "брать со стека" следует понимать действия:

1) проверяется, не пуст ли стек выражений; если пуст, то возбуждается прерывание с номером 4Ch;

2) иначе с вершины стека выражений считается слово и счетчик стека уменьшается на единицу; таким образом, слово оказывается счеркнутым.

Под словами "поместить на стек", "загрузить на стек" следует понимать действия:

- 1) элементу стека выражений, на который указывает счетчик, присваивается то значение, которое загружается;

2) если счетчик меньше границы стека выражений, то он увеличивается на единицу, иначе вызывается прерывание с номером 4Ch.

Если не указано, о каком стеке идет речь, то имеется в виду стек выражений.

4-битовые непосредственные операнды

Поскольку система команд Кронаса очень маленькая (менее 128 команд), было сочтено возможным использовать 4 младших бита в некоторых часто исполняемых командах для 4-битового непосредственного операнда. Такая команда может быть описана либо как множество из 16-ти различных команд (см. "Интерпретатор"), либо как просто 4-битовая команда с четырьмя битами непосредственного операнда.

Заметим, что все числа, встречающиеся далее - шестнадцатеричные (и поэтому снабжены символом 'h' - 'hex').

```

.PAGE
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          LI0..LI0F          00h..0Fh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          LI0..LI0F          00h..0Fh@u"
@ALI0..LI0F          00h..0Fh@a
Load Immediate

```

Код операции:	4 бита	0h
Непосредственные операнды:	4 бита	0h..0Fh
Длина команды:	1 байт	
Действие:		

Загружает на стек значение, соответствующее 4-м младшим битам команды (из отрезка 00h..0Fh), в виде 32-битового слова с нулями в старших 28 битах (ведущие нули).

PC увеличивается на 1.

В терминах интерпретатора:

Push(IR MOD 10h);

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LIB	10h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LIB	10h@u"
@ALIB	10h@a	

Load Immediate Byte

Код операции:	1 байт	10h
Непосредственные операнды:	1 байт	0h..0FFh
Длина команды:	2 байта	

Действие:

Загружает значение непосредственного операнда из диапазона 00h..0FFh (1 байта кода, следующих за командой) и помещает его на стек в виде 32-битового слова с ведущими нулями.

PC увеличивается на 2.

В терминах интерпретатора:

Push(Next())

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LID	11h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LID	11h@u"
@ALID	11h@a	

Load Immediate Double Byte

Код операции:	1 байт	11h
Непосредственные операнды:	2 байта	0h..0FFFFh
Длина команды:	3 байта	

Действие:

Загружает значение непосредственного операнда из диапазона 00h..0FFFFh (2 байта кода, следующих за командой) и помещает его на стек в виде 32-битового слова с ведущими нулями.

PC увеличивается на 3.

В терминах интерпретатора:

Push(Next2())

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LIW	12h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LIW	12h@u"
@ALIW	12h@a	

Load Immediate Word

Код операции:	1 байт	12h
Непосредственные операнды:	4 байта	0h..0FFFFFFFFh
Длина команды:	5 байт	

Действие:

Загружает значение непосредственного операнда из диапазона 00h..0FFFFFFFh (4 байта кода, следующих за командой) и помещает его на стек в виде 32-битового слова.

PC увеличивается на 5.

Замечание. Числа представляются в двоично-дополнительном коде.

Диапазон целых:

$-2^{31} \leq x \leq 2^{31}-1$,

или, в шестнадцатеричном представлении,

80000000h $\leq x \leq$ 7FFFFFFFh.

"-1" представляется как 0FFFFFFFh.

Замечание. Только команда LIW позволяет положить на стек отрицательное число. Небольшие отрицательные числа порождаются по-другому: загружаются их абсолютные величины и командой NEG изменяется знак.

В терминах интерпретатора:

Push(Next4())

.HEAD 0 "@ОПИСАНИЕ СИСТЕМЫ КОМАНД LIN 13h@u"

.HEAD 1 "@ОПИСАНИЕ СИСТЕМЫ КОМАНД LIN 13h@u"

@ALIN 13h@a

Load Immediate Nil

Код операции: 1 байт 13h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Загружает на стек несуществующий адрес (NIL).

PC увеличивается на 1.

Замечание. Значение NIL зависит от модели процессора. Это не влияет на переносимость программного обеспечения. Так, например, в Кронос 2.2 NIL равен FFFFFh. В этой модели этот адрес проверяется аппаратно, в остальных моделях может быть проверен явно командой CHKNIL.

В терминах интерпретатора:

Push(NIL)

.HEAD 0 "@ОПИСАНИЕ СИСТЕМЫ КОМАНД LLA 14h@u"

.HEAD 1 "@ОПИСАНИЕ СИСТЕМЫ КОМАНД LLA 14h@u"

@ALLA 14h@a

Load Local Address

Код операции: 1 байт 14h

Непосредственные операнды: 1 байт 0h..0FFh

Длина команды: 2 байта

Действие:

Прибавляет к содержимому L-регистра непосредственный операнд (байт, следующий за кодом команды) и помещает полученную сумму на стек. PC увеличивается на 2. Эта команда позволяет загрузить на стек адрес локальной переменной. Если

смещение переменной относительно L-регистра больше 0FFh, задача решается комбинированием команд LLA, LI, LIB, LID или LIW с ADD.

В терминах интерпретатора:
Push(L+Next())

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LGA	15h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LGA	15h@u"
@ALGA	15h@a	
Load Global Address		
Код операции:	1 байт	15h
Непосредственные операнды:	1 байт	0h..0FFh
Длина команды:	2 байта	
Действие:		

Прибавляет к содержимому G-регистра непосредственный операнд (байт, следующий за кодом команды) и помещает полученную сумму на стек. PC увеличивается на 2. Эта команда позволяет загрузить на стек адрес глобальной переменной. Если смещение переменной относительно G-регистра больше 0FFh, задача решается комбинированием команд LGA, LI, LIB, LID или LIW с ADD.

В терминах интерпретатора:
Push(G+Next())

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LSA	16h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LSA	16h@u"
@ALSA	16h@a	
Load Stack Address		
Код операции:	1 байт	16h
Непосредственные операнды:	1 байт	0h..0FFh
Длина команды:	2 байта	
Действие:		

Берет слово со стека, прибавляет к нему непосредственный операнд (байт, следующий за кодом команды), и помещает полученную сумму на стек. PC увеличивается на 2. Эта команда позволяет заменить адрес, лежащий на стеке, на адрес, смещенный относительно него. Это используется для получения адресов полей в тех случаях, когда адрес записи уже загружен на стек. Когда смещение поля больше 0FFh, задача решается комбинированием команд LSA, LI, LIB, LID или LIW с ADD.

Замечание. LSA XX семантически эквивалентно последовательности LIB XX ADD.

В терминах интерпретатора:
Push(Pop()+Next())

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LEA	17h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LEA	17h@u"
@ALEA	17h@a	
Load External Address		
Код операции:	1 байт	17h

Непосредственные операнды: 2 байта 0h..0FFh
 Длина команды: 3 байта
 Действие:

Выбирается однобайтовый непосредственный операнд, который интерпретируется как номер внешнего модуля, по которому в DFT модуля ищется ссылка на адрес ОГД внешнего модуля. Слово из ОГД внешнего модуля со смещением, взятым из следующего однобайтового непосредственного операнда, загружается на стек. PC увеличивается на 3.

В терминах интерпретатора:

```
i:=G-Next()-1;
adr:=MEM[i];
Push(MEM[adr]+Next())
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД JFLC 18h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД JFLC 18h@u"
@AJFLC 18h@a
```

Jump Forward Long Condition

Код операции: 1 байт 18h
 Непосредственные операнды: 2 байта 0h..0FFh
 Длина команды: 3 байта
 Действие:

Выбирает двухбайтовый непосредственный операнд - размер (N) возможного перехода к следующей команде. PC увеличивается на 3. Со стека берется значение. Если это 0, то PC увеличивается на N, иначе (при любом не равном нулю значении) далее будет выполняться следующая команда.

В терминах интерпретатора:

```
IF Pop()=0 THEN PC:=Next2()+PC
ELSE INC(PC,2)
END
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД JFL 19h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД JFL 19h@u"
@AJFL 19h@a
```

Jump Forward Long

Код операции: 1 байт 19h
 Непосредственные операнды: 2 байта 0h..0FFFFh
 Длина команды: 3 байта
 Действие:

Выбирается двухбайтовый непосредственный операнд - размер (N) перехода к следующей команде. PC увеличивается на 3+N.

В терминах интерпретатора:

```
PC:=Next2()+PC;
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД JFSC 1Ah@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД JFSC 1Ah@u"
@AJFSC 1Ah@a
```

Jump Forward Short Condition

Код операции: 1 байт 1Ah
 Непосредственные операнды: 1 байт 0h..0FFh

Длина команды: 2 байта
Действие:

Выбирает однобайтовый непосредственный операнд - размер (N) возможного перехода к следующей команде. PC увеличивается на 2. Со стека берется значение. Если это 0, то PC увеличивается на N, иначе (при любом ненулевом значении) далее будет выполняться следующая команда.

В терминах интерпретатора:
IF Pop()=0 THEN PC:=Next()+PC
ELSE INC(PC)
END

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	JFS	1Bh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	JFS	1Bh@u"
@AJFS	1Bh@a	

Jump Forward Short

Код операции:	1 байт	1Bh
Непосредственные операнды:	1 байт	0h..0FFh
Длина команды:	2 байта	

Действие:
Выбирается однобайтовый непосредственный операнд - размер (N) перехода к следующей команде. PC увеличивается на 2+N.

В терминах интерпретатора:
PC:=Next()+PC;

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	JBLC	1Ch@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	JBLC	1Ch@u"
@AJBLC	1Ch@a	

Jump Back Long Condition

Код операции:	1 байт	1Ch
Непосредственные операнды:	2 байта	0h..0FFFFh
Длина команды:	3 байта	

Действие:
PC увеличивается на 3.
При условии, что значение, взятое со стека, равно нулю, PC уменьшается на столько, сколько указано в двухбайтовом непосредственном операнде. Иначе выполняется следующая команда.

В терминах интерпретатора:
IF Pop()=0 THEN PC:=-Next2()+PC
ELSE INC(PC,2) END

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	JBL	1Dh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	JBL	1Dh@u"
@AJBL	1Dh@a	

Jump Back Long

Код операции:	1 байт	1Dh
Непосредственные операнды:	2 байта	0h..0FFFFh
Длина команды:	3 байта	

Действие:
PC увеличивается на 3 (в результате выборки кода и

двухбайтового непосредственного операнда), после чего уменьшается на столько, сколько указано в непосредственном операнде.

В терминах интерпретатора:

PC:=-Next2()+PC;

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	JBSC	1Eh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	JBSC	1Eh@u"
@AJBSC	1Eh@a	

Jump Back Short Condition

Код операции:	1 байт	1Eh
Непосредственные операнды:	1 байт	0h..0FFh
Длина команды:	2 байта	

Действие:

PC увеличивается на 2 в результате выборки команды и непосредственного операнда.

При условии, что значение, взятое со стека, равно нулю, PC уменьшается на столько, сколько указано в однобайтовом непосредственном операнде. Иначе выполняется следующая команда.

В терминах интерпретатора:

IF Pop()=0 THEN PC:=-Next()+PC
ELSE INC(PC) END

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	JBS	1Fh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	JBS	1Fh@u"
@AJBS	1Fh@a	

Jump Back Short

Код операции:	1 байт	1Fh
Непосредственные операнды:	1 байт	0h..0FFh
Длина команды:	2 байта	

Действие:

PC увеличивается на 2 в результате выборки кода и однобайтового непосредственного операнда, после чего уменьшается на столько, сколько указано в непосредственном операнде.

В терминах интерпретатора:

PC:=-Next()+PC;

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LLW	20h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LLW	20h@u"
@ALLW	20h@a	

Load Local Word

Код операции:	1 байт	20h
Непосредственные операнды:	1 байт	0h..0FFh
Длина команды:	2 байта	

Действие:

Загружает на стек слово, взятое по адресу, смещенному относительно L-регистра на величину, равную значению непосредственного операнда. команда позволяет загрузить на стек значение локальной переменной. Если смещение переменной

относительно L-регистра больше 0FFh, задача решается комбинированием команд LLA, LIW, LSW с ADD.

PC увеличивается на 2.

В терминах интерпретатора:

Push(MEM[L+Next()])

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LGW	21h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LGW	21h@u"
@ALGW	21h@a	

Load Global Word

Код операции:	1 байт	21h
Непосредственные операнды:	1 байт	0h..0FFh
Длина команды:	2 байта	

Действие:

Загружает на стек слово, взятое по адресу, смещенному относительно G-регистра на величину, равную значению непосредственного операнда. Эта команда позволяет загрузить на стек значение глобальной переменной. Если смещение переменной относительно G-регистра больше 0FFh, задача решается комбинированием команд LGA, LI, LIB, LID или LIW с ADD и LSW.

PC увеличивается на 2.

В терминах интерпретатора:

Push(MEM[G+Next()])

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LEW	22h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LEW	22h@u"
@ALEW	22h@a	

Load External Word

Код операции:	1 байт	22h
Непосредственные операнды:	2 байта	0h..0FFh
Длина команды:	3 байта	

Действие:

Из сегмента кода выбираются два операнда. Первый соответствует номеру внешнего модуля, второй - смещению относительно G-регистра этого модуля. На стек загружается внешнее глобальное слово из указанного модуля с указанным смещением.

PC увеличивается на 3.

В терминах интерпретатора:

i:=G-Next()-1; adr:=MEM[MEM[i]];
Push(MEM[adr+Next()])

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LSW	23h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LSW	23h@u"
@ALSW	23h@a	

Load Stack addressed Word

Код операции:	1 байт	23h
Непосредственные операнды:	1 байт	0h..0FFh
Длина команды:	2 байта	

Действие:

Загружает на стек слово с адресом, смещенным относительно

значения, взятого со стека на величину, равную значению непосредственного операнда.

PC увеличивается на 2.

Замечание. Эта команда может быть использована для доступа к первым 256 словам записи (RECORD) при компиляции с языков высокого уровня.

В терминах интерпретатора:

Push(MEM[Pop()+Next()])

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LLW4..LLW0F	24h..24h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LLW4..LLW0F	24h..24h@u"
@ALLW4..LLW0F		24h..2Fh@a	

Load Local Word

Код операции:	4 бита	2h
Непосредственные операнды:	4 бита	4h..0Fh
Длина команды:	1 байт	

Действие:

Загружает на стек локальное слово со смещением в диапазоне от 4h до 0Fh относительно L-регистра.

PC увеличивается на 1.

Замечание. Первые четыре слова локальной области содержат специальную информацию, поэтому команд LLW0..LLW3 нет.

Замечание. Данные команды являются, по существу, более компактной версией команды LLW. Такое упрощение оправдано исключительно частым обращением к первым 12-ти локальным переменным процедур.

В терминах интерпретатора:

Push(MEM[L+IR MOD 10h])

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SLW	30h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SLW	30h@u"
@ASLW		30h@a	

Store Local Word

Код операции:	1 байт	30h
Непосредственные операнды:	1 байт	0h..0FFh
Длина команды:	2 байта	

Действие:

Присваивает значение, взятое со стека, локальному слову со смещением, указанным в непосредственном операнде.

PC увеличивается на 2.

В терминах интерпретатора:

MEM[L+Next()]:=Pop()

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SGW	31h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SGW	31h@u"
@ASGW		31h@a	

Store Global Word

Код операции:	1 байт	31h
---------------	--------	-----

Непосредственные операнды: 1 байт 0h..0FFh
Длина команды: 2 байта
Действие:

Присваивает значение, взятое со стека, глобальному слову со смещением, указанным в непосредственном операнде.
PC увеличивается на 2.

В терминах интерпретатора:
MEM[G+Next()]:=Pop()

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД SEW 32h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД SEW 32h@u"
@ASEW 32h@a

Store External Word

Код операции: 1 байт 32h
Непосредственные операнды: 2 байта 0h..0FFh
Длина команды: 3 байта
Действие:

Из кода выбираются два операнда - номер внешнего модуля и смещение относительно его G-регистра. Значение, взятое со стека, присваивается внешнему глобальному слову из указанного модуля с указанным смещением.
PC увеличивается на 3.

В терминах интерпретатора:
i:=G-Next()-1; adr:=MEM[MEM[i]]; (* external G *)
MEM[adr+Next()]:=Pop()

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД SSW 33h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД SSW 33h@u"
@ASSW 33h@a

Store Stack addressed Word

Код операции: 1 байт 33h
Непосредственные операнды: 1 байт 0h..0FFh
Длина команды: 2 байта
Действие:

Со стека берутся значение и адрес. Присваивает указанное значение слову, смещенному относительно указанного адреса на величину, равную значению непосредственного операнда.
PC увеличивается на 2.

В терминах интерпретатора:
i:=Pop(); MEM[Pop()+Next()]:=i

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД SLW4..SLW0F 34h..3Fh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД SLW4..SLW0F 34h..3Fh@u"
@ASLW4..SLW0F 34h..3Fh@a

Store Local Word

Код операции: 4 бита 3h
Непосредственные операнды: 4 бита 4h..0Fh
Длина команды: 1 байт
Действие:

Присваивает значение, взятое со стека, локальному слову со смещением в диапазоне от 4h до 0Fh.

PC увеличивается на 1.

В терминах интерпретатора:

MEM[L+IR MOD 10h]:=Pop()

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LXB	40h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LXB	40h@u"
@ALXB	40h@a	
Load indexEd Byte		
Код операции:	1 байт	40h
Непосредственные операнды:	0 байт	
Длина команды:	1 байт	
Действие:		

На стеке - два операнда. Нижний - словный адрес, верхний - байтовое смещение относительно этого адреса. Оба операнда счеркиваются со стека, и на стек загружается байт, взятый из памяти по указанному словному адресу с указанным байтовым смещением, дополненный нулями в старших разрядах.

PC увеличивается на 1.

В терминах интерпретатора:

i:=Pop(); Push(ByteMEM[Pop()*4+i]);

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LXW	41h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LXW	41h@u"
@ALXW	41h@a	
Load indexEd Word		
Код операции:	1 байт	41h
Непосредственные операнды:	0 байт	
Длина команды:	1 байт	
Действие:		

На стеке - два операнда. Нижний - словный адрес, верхний - словное смещение относительно этого адреса. Оба операнда счеркиваются со стека, и на стек загружается слово, взятое из памяти по указанному словному адресу с указанным смещением.

PC увеличивается на 1.

В терминах интерпретатора:

i:=Pop(); Push(MEM[Pop()+i])

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LGW2..LGW0F	42h..4Fh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LGW2..LGW0F	42h..4Fh@u"
@ALGW2..LGW0F	42h..4Fh@a	
Load Global Word		
Код операции:	4 бита	4h
Непосредственные операнды:	4 бита	2h..0Fh
Длина команды:	1 байт	
Действие:		

Загружает на стек глобальное слово со смещением из диапазона от 2h до 0Fh.

PC увеличивается на 1.

В терминах интерпретатора:

Push(MEM[G+IR MOD 10h])

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SXB	50h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SXB	50h@u"
@ASXB	50h@a	

Store indEXed Byte

Код операции: 1 байт 50h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

На стеке - три операнда. Нижний - словный адрес, выше - байтовое смещение относительно этого адреса. Верхний операнд - слово, старшие 24 бита которого игнорируются, а оставшийся байт записывается в память по указанному словному адресу с указанным байтовым смещением. Все три операнда счеркиваются со стека.

PC увеличивается на 1.

В терминах интерпретатора:

j:=Pop(); i:=Pop(); ByteMEM[Pop()*4+i]:=j;

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SXW	51h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SXW	51h@u"
@ASXW	51h@a	

Store indEXed Word

Код операции: 1 байт 51h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

На стеке - три операнда. Нижний - словный адрес, выше - словное смещение относительно этого адреса. Верхний операнд - слово, которое записывается в память по указанному словному адресу с указанным смещением. Все три операнда счеркиваются со стека.

PC увеличивается на 1.

В терминах интерпретатора:

j:=Pop(); i:=Pop(); MEM[Pop()+i]:=j

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SGW2..SGW0F	52h..5Fh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SGW2..SGW0F	52h..5Fh@u"
@ASGW2..SGW0F	52h..5Fh@a	

Store Global Word

Код операции: 4 бита 5h

Непосредственные операнды: 4 бита 2h..0Fh

Длина команды: 1 байт

Действие:

Присваивает значение, взятое со стека, глобальному слову со смещением из диапазона от 2h до 0Fh.

PC увеличивается на 1.

В терминах интерпретатора:

MEM[G+IR MOD 10h]:=Pop()

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LSW0..LSW0F	60h..6Fh@u"
------------------------------------	-------------	-------------

```
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      LSW0..LSW0F      60h..6Fh@u"
@ALSW0..LSW0F      60h..6Fh@a
```

Load Stack addressed Word

Код операции: 4 бита 6h

Непосредственные операнды: 4 бита 0h..0Fh

Длина команды: 1 байт

Действие:

Загружает на стек из памяти слово со смещением из диапазона 0..0Fh относительно адреса, взятого со стека.

PC увеличивается на 1.

В терминах интерпретатора:

Push(MEM[Pop()+IR MOD 10h])

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      SSW0..SSW0F      70h..7Fh@u"
```

```
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      SSW0..SSW0F      70h..7Fh@u"
```

```
@ASSW0..SSW0F      70h..7Fh@a
```

Store Stack addressed Word

Код операции: 4 бита 7h

Непосредственные операнды: 4 бита 0h..0Fh

Длина команды: 1 байт

Действие:

Слово, взятое со стека, записывает в память по адресу, смещенному на 0..0Fh относительно адреса, взятого со стека.

PC увеличивается на 1.

В терминах интерпретатора:

i:=Pop(); MEM[Pop()+IR MOD 10h]:=i

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      QUIT      81h@u"
```

```
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      QUIT      81h@u"
```

```
@AQUIT      81h@a
```

stop processor

Код операции: 1 байт 81h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Останов процессора.

PC увеличивается на 1 байт.

В терминах интерпретатора:

ConsolMicroProgram

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      GETM      82h@u"
```

```
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      GETM      82h@u"
```

```
@AGETM      82h@a
```

GET Mask

Код операции: 1 байт 82h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Загружает на стек слово, содержащее маску прерываний (М-регистр процессора).

PC увеличивается на 1 байт.

В терминах интерпретатора:

Push(M)

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SETM	83h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SETM	83h@u"

@ASETМ 83h@a

SET Mask

Код операции: 1 байт 83h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Заносит маску прерываний, взятую со стека, в М-регистр процессора.

PC увеличивается на 1.

В терминах интерпретатора:

М:=BITSET(Pop());

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	TRAP	84h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	TRAP	84h@u"

@ATRAP 84h@a

interrupt simulation

Код операции: 1 байт 84h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Берет значение со стека и возбуждает прерывание с таким номером.

PC увеличивается на 1.

В терминах интерпретатора:

TRAP(Pop())

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	TRA	85h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	TRA	85h@u"

@ATRA 85h@a

TRAnsfer control between processes

Код операции: 1 байт 85h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Происходит переключение двух процессов.

Состояние процесса определяется состоянием стека и значениями шести регистров (G, L, PC, M, S, H).

На стеке - два слова. Верхнее содержит адрес указателя на дескриптор процесса, который требуется запустить, нижнее - адрес, по которому нужно записать адрес указателя на дескриптор текущего процесса.

Регистры текущего процесса записываются в дескриптор процесса, который лежит в памяти по адресу, указанному в регистре Р.

Р-регистр текущего процесса записывается по адресу, взятому в качестве второго операнда со стека.

Стек выражений спасается на процедурный стек текущего процесса. Для того, чтобы при спасении стека выражений не возникло переполнения процедурного стека, регистр границы процедурного стека - Н - указывает на (глубину стека выражений + 1) слов ниже фактической границы процедурного стека.

В верхнем операнде содержится адрес слова, в котором лежит указатель на дескриптор процесса (Р-регистр), подлежащего запуску. По новому Р-регистру восстанавливаются значения всех регистров процесса. С вершины процедурного стека восстанавливается стек выражений.

РС увеличивается на 1.

В терминах интерпретатора:

i:=Pop(); Transfer(Pop(),i)

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	TR	86h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	TR	86h@u"
@ATR	86h@a		
Test & Reset			
Код операции:	1 байт	86h	
Непосредственные операнды:	0 байт		
Длина команды:	1 байт		
Действие:			

Загружает на стек слово, взятое из памяти по адресу, взятому со стека. По этому адресу помещается 0.

РС увеличивается на 1.

Замечание. Команда выполняется как неделимое действие "чтение с разрушением". В случае работы нескольких процессоров с общей памятью используется для межпроцессорной синхронизации (как, например, в Кронос-2.5).

В терминах интерпретатора:

i:=Pop(); Push(MEM[i]); MEM[i]:=0

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	IDLE	87h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	IDLE	87h@u"
@AIDLE	87h@a		
IDLE process			
Код операции:	1 байт	87h	
Непосредственные операнды:	0 байт		
Длина команды:	1 байт		
Действие:			

Уменьшает РС на 1. Процессор ждет прерывания, не занимая шину.

Поскольку при чтении байта кода РС увеличивался на единицу, после выполнения команды IDLE РС остается на прежнем месте.

Замечание. Любой процесс, выполнивший команду IDLE, будет продолжать ее выполнение вечно, пока какой-либо другой процесс (обработчик прерываний, например)

принудительно не увеличит PC процесса, выполнявшего команду IDLE.

В терминах интерпретатора:

DEC(PC); REPEAT (* не занимая шины *) UNTIL Ipt

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	ADD	88h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	ADD	88h@u"
@AADD	88h@a		
integer	ADDition		
Код операции:	1 байт	88h	
Непосредственные операнды:	0 байт		
Длина команды:	1 байт		
Действие:			

Берет со стека два целых числа, складывает их и помещает сумму на стек. Если при этом происходит переполнение целого, возбуждается прерывание с номером 41h.

PC увеличивается на 1.

Замечание. В процессоре П2.6 после возникновения прерывания по переполнению (независимо от того, разрешено оно или нет) на стек помещаются 32 младших бита результата арифметической операции.

В терминах интерпретатора:

Push(Pop()+Pop())

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SUB	89h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SUB	89h@u"
@ASUB	89h@a		
integer	SUBtraction		
Код операции:	1 байт	89h	
Непосредственные операнды:	0 байт		
Длина команды:	1 байт		
Действие:			

Берет со стека два операнда - сначала вычитаемое, затем уменьшаемое. Производит вычитание и помещает результат на стек. Если произошло переполнение целого, возбуждается прерывание с номером 41h.

PC увеличивается на 1.

В терминах интерпретатора:

i:=Pop(); Push(Pop()-i);

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	MUL	8Ah@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	MUL	8Ah@u"
@AMUL	8Ah@a		
integer	MULtiplication		
Код операции:	1 байт	8Ah	
Непосредственные операнды:	0 байт		
Длина команды:	1 байт		
Действие:			

Берет со стека два целых числа, перемножает их и помещает произведение на стек. Если в результате произошло переполнение

целого, возбуждается прерывание с номером 41h.
PC увеличивается на 1.

В терминах интерпретатора:
Push(Pop()*Pop())

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	DIV	8Bh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	DIV	8Bh@u"
@ADIV	8Bh@a	
integer DIVision		
Код операции:	1 байт	8Bh
Непосредственные операнды:	0 байт	
Длина команды:	1 байт	
Действие:		

Берет со стека два целых числа - сначала делитель, затем делимое, производит деление нацело и помещает результат на стек. Округляет в сторону меньшего. Так, $(-1 \text{ DIV } 2) = -1$. Если в результате произошло переполнение целого, возбуждается прерывание с номером 41h.

PC увеличивается на 1.

Замечание. В Кроносе-2.2 округление происходит в сторону нуля.

В терминах интерпретатора:
i:=Pop(); Push(Pop() DIV i)

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SHL	8Ch@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SHL	8Ch@u"
@ASHL	8Ch@a	
integer SHift Left		
Код операции:	1 байт	8Ch
Непосредственные операнды:	0 байт	
Длина команды:	1 байт	
Действие:		

Арифметический сдвиг влево. Со стека берутся величина сдвига и сдвигаемое слово. Слово сдвигается влево на указанную величину. В младшие разряды дописывается соответствующее число нулей. При несовпадении знаковых разрядов сдвигаемого слова и результата возбуждается прерывание с номером 41h (переполнение целого). Результат помещается на стек.

PC увеличивается на 1.

Замечание. Таким образом, команда SHL выполняет умножение числа со знаком на степень двойки.

В терминах интерпретатора:
i:=Pop(); Push(SHL(Pop(),i))

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SHR	8Dh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SHR	8Dh@u"
@ASHR	8Dh@a	
integer SHift Right		
Код операции:	1 байт	8Dh
Непосредственные операнды:	0 байт	

Длина команды: 1 байт
Действие:

Арифметический сдвиг вправо. Со стека берутся величина сдвига и сдвигаемое слово. Слово сдвигается вправо на величину сдвига. Содержимое знакового разряда восстанавливается, старшие разряды заполняются содержимым знакового разряда. Результат помещается на стек.

РС увеличивается на 1.

Замечание. Таким образом, команда SHR выполняет деление числа со знаком на степень двойки с округлением в сторону меньшего.

В терминах интерпретатора:

i:=Pop(); Push(SHR(Pop(),i))

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	ROL	8Eh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	ROL	8Eh@u"
@AROL		8Eh@a

word ROTate Left

Код операции: 1 байт 8Eh

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Циклический сдвиг влево.

При циклическом сдвиге влево на единицу старший разряд слова переносится вправо и становится его младшим разрядом. При циклическом сдвиге на N эта операция повторяется N раз.

Со стека берутся величина сдвига и сдвигаемое слово. Слово сдвигается влево на число разрядов, равное величине сдвига по модулю 32. Результат помещается на стек.

РС увеличивается на 1.

В терминах интерпретатора:

i:=Pop() MOD 20h; Push(ROL(Pop(),i))

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	ROR	8Fh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	ROR	8Fh@u"
@AROR		8Fh@a

word ROTate Right

Код операции: 1 байт 8Fh

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Циклический сдвиг вправо.

При циклическом сдвиге вправо на единицу младший разряд слова переносится влево и становится его старшим разрядом. При циклическом сдвиге на N эта операция повторяется N раз.

Со стека берутся величина сдвига и сдвигаемое слово. Слово сдвигается вправо на число разрядов, равное указанной величине по модулю 32. Результат помещается на стек.

РС увеличивается на 1.

В терминах интерпретатора:

```
i:=Pop() MOD 20h; Push(ROR(Pop(),i))
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      IO0..IO4      90h..94h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      IO0..IO4      90h..94h@u"
@AI00..IO4      90h..94h@a
```

Input-Output

Код операции: 1 байт 94h

Непосредственные операнды: зависит от модели процессора

Длина команды: зависит от модели процессора

Действие:

Команды работы с шиной. Индивидуальны для каждого процессора.

В терминах интерпретатора:

```
CASE cpu OF
```

```
|Kronos2_2: ioP2_2
```

```
|Kronos2_5: ioP2_5
```

```
|Kronos2_6: ioP2_6
```

```
END
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      ARRCMP      95h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      ARRCMP      95h@u"
@AARRCMP      95h@a
```

ARRay CoMPare

Код операции: 1 байт 95h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Команда предназначена для сравнения словных массивов.

Со стека берется размер массивов, затем адрес начала первого массива, затем адрес начала второго массива.

Если размер оказался отрицательным числом, это число помещается обратно на стек и возбуждается прерывание с номером 4Fh. Если размер равен 0, на стек дважды загружается адрес второго массива.

Иначе массивы сравниваются пословно до тех пор, пока сравнение не дойдет до пары последних слов или до пары неравных слов, после чего адреса этой пары слов грузятся на стек.

PC увеличивается на 1.

Замечание. Команда не реализована на П2.2.

В терминах интерпретатора:

```
sz:=Pop(); adr:=Pop(); adr1:=Pop();
```

```
IF sz<0 THEN Push(sz); TRAP(4Fh)
```

```
ELSIF sz=0 THEN Push(adr1); Push(adr1)
```

```
ELSE LOOP
```

```
  IF (adr^ # adr1^) OR (sz=1) THEN
```

```
    Push(adr1); Push(adr); EXIT
```

```
  END;
```

```
  DEC(sz); INC(adr); INC(adr1);
```

```
END;
```

```
END;
```

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	WM	96h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	WM	96h@u"
@AWM	96h@a		
Word Move			
Код операции:	1 байт	96h	
Непосредственные операнды:	0 байт		
Длина команды:	1 байт		

Действие:

Со стека берется размер словного сегмента, который будет передвинут, затем адрес, начиная с которого двигать, затем адрес, начиная с которого писать этот сегмент. Допускается перекрытие областей источника и приемника.

PC увеличивается на 1.

Замечание. Команда реализована только на Кронос-2.6.

В терминах интерпретатора:

```

sz:=Pop(); f:=Pop(); t:=Pop();
IF t>f THEN
  t:=t+sz-1; f:=f+sz-1;
  WHILE sz>0 DO
    MEM[t]:=MEM[f]; DEC(t); DEC(f); DEC(sz)
  END
ELSE
  WHILE sz>0 DO
    MEM[t]:=MEM[f]; INC(t); INC(f); DEC(sz)
  END
END;

```

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	BM	97h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	BM	97h@u"
@ABM	97h@a		
Bit Move			
Код операции:	1 байт	97h	
Непосредственные операнды:	0 байт		
Длина команды:	1 байт		

Действие:

Со стека берутся размер пересылаемого битового сегмента, битовое смещение и словный адрес пересылаемого сегмента, битовое смещение и словный адрес, куда пересылается сегмент, и производится пересылка. Допускается перекрытие областей источника и приемника.

PC увеличивается на 1.

Замечание. Команда не реализована на П2.2.

В терминах интерпретатора:

```

sz:=Pop(); -- размер пересылаемой области в битах
i:=Pop(); src:=Pop(); -- смещение и адрес источника
j:=Pop(); trg:=Pop(); -- смещение и адрес приемника
src:=src*32+i;
trg:=trg*32+j;
IF src>=trg THEN
  n:=+1

```

```

ELSE INC(src,sz-1); INC(trg,sz-1); n:=-1
END;
FOR k:=0 TO sz-1 DO
  i:=trg DIV 32; j:=trg MOD 32;
  IF (src MOD 32) IN BITSET(MEM[src DIV 32]) THEN
    MEM[i]:=INTEGER( BITSET(MEM[i]) + {j} )
  ELSE
    MEM[i]:=INTEGER( BITSET(MEM[i]) - {j} )
  END;
  INC(src,n); INC(trg,n)
END

```

```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          FADD          98h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          FADD          98h@u"
@AFADD          98h@a
Float ADDition
Код операции:          1 байт          98h
Непосредственные операнды:      0 байт
Длина команды:          1 байт
Действие:

```

Берет со стека два вещественных числа, производит сложение и помещает сумму на стек. Если произошло переполнение вещественного, на стек загружается 0 и возбуждается прерывание с номером 42h.
 PC увеличивается на 1.

В терминах интерпретатора:
 Push(REAL(Pop())+REAL(Pop()))

```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          FSUB          99h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          FSUB          99h@u"
@AFSUB          99h@a
Float SUBtraction
Код операции:          1 байт          99h
Непосредственные операнды:      0 байт
Длина команды:          1 байт
Действие:

```

Берет со стека два вещественных числа - сначала вычитаемое, затем уменьшаемое. Производит вычитание и помещает разность на стек. Если произошло переполнение вещественного, на стек загружается 0 и возбуждается прерывание с номером 42h.
 PC увеличивается на 1.

В терминах интерпретатора:
 X:=REAL(Pop()); Push(REAL(Pop())-X)

```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          FMUL          9Ah@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          FMUL          9Ah@u"
@AFMUL          9Ah@a
Float MULtiplication
Код операции:          1 байт          9Ah
Непосредственные операнды:      0 байт
Длина команды:          1 байт
Действие:

```

Перемножает два вещественных числа, взятых со стека. Произведение помещает на стек. Если произошло переполнение вещественного, возбуждается прерывание с номером 42h.

Если результат меньше 2 в степени -128, возбуждается прерывание с номером 43h (исчезновение порядка).

PC увеличивается на 1.

В терминах интерпретатора:

Push(REAL(Pop()))*REAL(Pop()))

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	FDIV	9Bh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	FDIV	9Bh@u"
@AFDIV	9Bh@a	
Float DIVision		
Код операции:	1 байт	9Bh
Непосредственные операнды:	0 байт	
Длина команды:	1 байт	
Действие:		

Берет со стека два вещественных числа - сначала делитель, затем делимое, производит деление и помещает результат на стек. Если произошло переполнение вещественного, возбуждается прерывание с номером 42h.

Если результат меньше 2 в степени -128, возбуждается прерывание с номером 43h (исчезновение порядка).

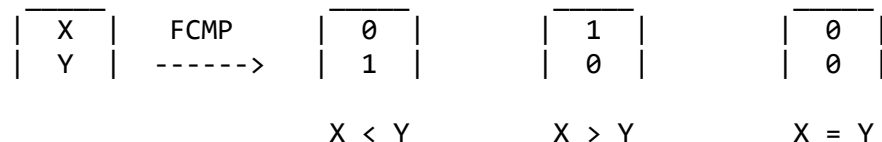
PC увеличивается на 1.

В терминах интерпретатора:

X:=REAL(Pop()); Push(REAL(Pop())/X)

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	FCMP	9Ch@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	FCMP	9Ch@u"
@AFCMP	9Ch@a	
Float CoMPare		
Код операции:	1 байт	9Ch
Непосредственные операнды:	0 байт	
Длина команды:	1 байт	
Действие:		

Сравнивает два взятых со стека вещественных числа. Если верхнее меньше нижнего, на стек помещаются последовательно 1 и 0; если верхнее больше нижнего, на стек помещаются последовательно 0 и 1; если они равны, то на стек помещаются 0 и 0.



PC увеличивается на 1.

Замечание. Комбинации команд

FCMP EQU

FCMP NEG

FCMP LSS

....

дают возможность осуществить любое сравнение вещественных чисел.

В терминах интерпретатора:

```
X:=REAL(Pop()); Y:=REAL(Pop());
IF X<Y THEN Push(1); Push(0)
ELIF X>Y THEN Push(0); Push(1)
ELSE Push(0); Push(0) END
```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	FABS	9Dh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	FABS	9Dh@u"
@AFABS		9Dh@a

Float ABSolute

Код операции: 1 байт 9Dh

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Берет со стека вещественное число и помещает на стек его абсолютное значение.

PC увеличивается на 1.

Замечание. Команда никогда не возбуждает прерываний 42h, 43h.

В терминах интерпретатора:

```
X:=REAL(Pop());
IF X<0.0 THEN Push(-X) ELSE Push(X) END
```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	FNEG	9Eh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	FNEG	9Eh@u"
@AFNEG		9Eh@a

Float NEGative

Код операции: 1 байт 9Eh

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Берет со стека вещественное число и помещает на стек его противоположное значение.

PC увеличивается на 1.

Замечание. Команда никогда не возбуждает прерываний 42h, 43h.

В терминах интерпретатора:

```
Push(-REAL(Pop()))
```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	FFCT	9Fh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	FFCT	9Fh@u"
@AFFCT		9Fh@a

Float FunCTion

Код операции: 1 байт 9Fh

Непосредственные операнды: 1 байт

Длина команды: 2 байта

Действие:

Если значение непосредственного операнда равно 0, то со стека берется целое число, преобразуется в вещественное и помещается на стек; если в следующем за командой байте кода лежит 1, то со стека берется вещественное число, у которого отсекается затем дробная часть. Полученное таким образом целое число помещается на стек.

В этих случаях PC увеличивается на 2.

Если в следующем за командой байте кода не 0 и не 1, PC уменьшается на 1 и возбуждается прерывание с номером 7h.

В терминах интерпретатора:

```
i:=Next();
IF i=0 THEN Push(FLOAT(INTEGER(Pop()))))
ELIF i=1 THEN Push(TRUNC( REAL(Pop()))))
ELSE DEC(PC); TRAP(7h)
END;
```

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LSS	0A0h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LSS	0A0h@u"
@ALSS	0A0h@a		
integer LeSS			
Код операции:	1 байт	0Ah	
Непосредственные операнды:	0 байт		
Длина команды:	1 байт		
Действие:			

Сравниваются два целых числа, взятых со стека. Если нижнее меньше верхнего, на стек помещается 1, иначе 0.

PC увеличивается на 1.

Замечание. Поскольку сравнение не осуществляет вычитания, прерывание 41h (переполнение целого) никогда не возбуждается.

В терминах интерпретатора:

```
i:=Pop(); Push(Pop(<i)
```

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LEQ	0A1h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LEQ	0A1h@u"
@ALEQ	0A1h@a		
integer Less or Equal			
Код операции:	1 байт	0A1h	
Непосредственные операнды:	0 байт		
Длина команды:	1 байт		
Действие:			

Сравниваются два целых числа, взятых со стека. Если нижнее меньше или равно верхнему, на стек помещается 1, иначе 0.

PC увеличивается на 1.

Замечание. Поскольку сравнение не осуществляет вычитания, прерывание 41h (переполнение целого) никогда не возбуждается.

В терминах интерпретатора:

```
i:=Pop(); Push(Pop()<=i)
```

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	GTR	0A2h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	GTR	0A2h@u"
@AGTR	0A2h@a		

integer GreaTeR

Код операции: 1 байт 0A2h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Сравниваются два целых числа, взятых со стека. Если нижнее больше верхнего, на стек помещается 1, иначе 0.

PC увеличивается на 1.

Замечание. Поскольку сравнение не осуществляет вычитания, прерывание 41h (переполнение целого) никогда не возбуждается.

В терминах интерпретатора:

```
i:=Pop(); Push(Pop()>i)
```

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	GEQ	0A3h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	GEQ	0A3h@u"
@AGEQ	0A3h@a		

integer Greater or Equal

Код операции: 1 байт 0A3h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Сравниваются два целых числа, взятых со стека. Если нижнее больше или равно верхнему, на стек помещается 1, иначе 0.

PC увеличивается на 1.

Замечание. Поскольку сравнение не осуществляет вычитания, прерывание 41h (переполнение целого) никогда не возбуждается.

В терминах интерпретатора:

```
i:=Pop(); Push(Pop()>=i)
```

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	EQU	0A4h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	EQU	0A4h@u"
@AEQU	0A4h@a		

integer EQUal

Код операции: 1 байт 0A4h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Сравниваются два целых числа, взятых со стека. Если они равны между собой, на стек помещается 1, иначе 0.

PC увеличивается на 1.

В терминах интерпретатора:
Push(Pop())=Pop())

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	NEQ	0A5h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	NEQ	0A5h@u"
@ANEQ	0A5h@a		
integer NOT equal			
Код операции:	1 байт	0A5h	
Непосредственные операнды:	0 байт		
Длина команды:	1 байт		
Действие:			

Сравниваются два целых числа, взятых со стека. Если они не равны между собой, на стек помещается 1, иначе 0.
PC увеличивается на 1.

В терминах интерпретатора:
Push(Pop())#Pop())

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	ABS	0A6h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	ABS	0A6h@u"
@AABS	0A6h@a		
integer ABSolute			
Код операции:	1 байт	0A6h	
Непосредственные операнды:	0 байт		
Длина команды:	1 байт		
Действие:			

Берет со стека целое число и помещает на стек его абсолютное значение.
PC увеличивается на 1.

В терминах интерпретатора:
Push(ABS(Pop()))

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	NEG	0A7h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	NEG	0A7h@u"
@ANEG	0A7h@a		
integer NEGative			
Код операции:	1 байт	0A7h	
Непосредственные операнды:	0 байт		
Длина команды:	1 байт		
Действие:			

Берет со стека целое число и помещает на стек его противоположное значение.
PC увеличивается на 1.

Замечание. Если операнд на стеке равен min(INTEGER), то в результате выполнения команды возбуждается прерывание с номером 41h (переполнение целого).

В терминах интерпретатора:
Push(-Pop())

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	OR	0A8h@u"
---------	----------------------------	----	---------

```
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          OR          0A8h@u"
@AOR          0A8h@a
```

logical bit per bit OR

Код операции: 1 байт 0A8h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Берет со стека два слова и помещает на стек результат их сложения, то есть слово, в котором биты выставлены в тех позициях, что в первом и/или во втором слове.

Результат действия команды может быть проиллюстрирован следующей таблицей:

	0 1	
----	+	----
0	0 1	
----	+	----
1	1 1	

PC увеличивается на 1.

В терминах интерпретатора:

```
v:=BITSET(Pop()); w:=BITSET(Pop()); Push(w+v)
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          AND          0A9h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          AND          0A9h@u"
```

```
@AAND          0A9h@a
```

logical bit per bit AND

Код операции: 1 байт 0A9h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Берет со стека два слова и помещает на стек результат их перемножения, то есть слово, в котором единицы только в тех позициях, где были единицы в обоих словах.

Результат действия команды может быть проиллюстрирован следующей таблицей:

	0 1	
----	+	----
0	0 0	
----	+	----
1	0 1	

PC увеличивается на 1.

В терминах интерпретатора:

```
v:=BITSET(Pop()); w:=BITSET(Pop()); Push(w*v)
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          XOR          0AAh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          XOR          0AAh@u"
```

```
@AXOR          0AAh@a
```

logical bit per bit XOR

Код операции: 1 байт 0AAh
 Непосредственные операнды: 0 байт
 Длина команды: 1 байт
 Действие:

Берет со стека два слова и помещает на стек результат деления нижнего на верхнее, то есть слово, в котором единицы в тех позициях, где биты не совпали в первом и втором словах, и нули - где совпали.

Результат действия команды может быть проиллюстрирован следующей таблицей:

	0 1	
----	+	----
0	0 1	
----	+	----
1	1 0	

PC увеличивается на 1.

В терминах интерпретатора:

v:=BITSET(Pop()); w:=BITSET(Pop()); Push(w/v)

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД BIC 0ABh@u"
 .HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД BIC 0ABh@u"
 @ABIC 0ABh@a

logical bit per bit BIt Clear

Код операции: 1 байт 0ABh
 Непосредственные операнды: 0 байт
 Длина команды: 1 байт
 Действие:

Берет со стека два слова и помещает на стек результат вычитания верхнего из нижнего, то есть нижнее слово, у которого единицы заменились на нули в тех позициях, где у верхнего были единицы.

Результат действия команды может быть проиллюстрирован следующей таблицей:

	0 1	
----	+	----
0	0 0	
----	+	----
1	1 0	

Обратите внимание, что действие команды несимметрично относительно операндов. В таблице биты верхнего слова изображены в верхней горизонтали, биты нижнего слова - в левой вертикали.

PC увеличивается на 1.

В терминах интерпретатора:

v:=BITSET(Pop()); w:=BITSET(Pop()); Push(w-v)

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД IN 0ACh@u"
 .HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД IN 0ACh@u"

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	MOD	0AFh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	MOD	0AFh@u"
@AMOD	0AFh@a	

integer MODule

Код операции: 1 байт 0AFh

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Берет со стека два целых числа. Результат взятия нижнего числа по модулю верхнего помещает на стек.

PC увеличивается на 1.

Замечание. Здесь взятие по модулю - арифметическая операция, определяемая формулой:

$$X \text{ MOD } N \parallel X - (X \text{ DIV } N) * N,$$

где DIV - деление с округлением в сторону меньшего для П2.6 и в сторону нуля для младших моделей процессора. См. также команду QUOT.

В терминах интерпретатора:

i:=Pop(); Push(Pop() MOD i)

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	DECS	0B0h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	DECS	0B0h@u"
@ADECS	0B0h@a	

DECrement S-register

Код операции: 1 байт 0B0h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Действие, обратное действию ALLOC: берет со стека значение i и на i слов уменьшает S-регистр.

PC увеличивается на 1.

В терминах интерпретатора:

DEC(S,Pop())

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	DROP	0B1h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	DROP	0B1h@u"
@ADROP	0B1h@a	

DROP

Код операции: 1 байт 0B1h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Счеркивает слово со стека.

PC увеличивается на 1.

В терминах интерпретатора:

i:=Pop();

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LODF	0B2h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LODF	0B2h@u"
@ALODF	0B2h@a	

reLOaD expression stack after Function return

Код операции: 1 байт 0B2h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Со стека берется слово, затем из памяти загружается ранее спасенный стек, и сверху на стек грузится взятое слово.

PC увеличивается на 1.

В терминах интерпретатора:

i:=Pop(); RestoreExpStack; Push(i)

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД STORE 0B3h@u"

.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД STORE 0B3h@u"

@ASTORE 0B3h@a

STORE expression stack after function call

Код операции: 1 байт 0B3h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Если величина значения S-регистра, увеличенная на глубину стека выражений + 1, превышает значение границы Р-стека (Н-регистр), то PC уменьшается на единицу и возбуждается прерывание с номером 40h. Такая проверка производится несмотря на то, что Н-регистр дает заниженную на размер стека выражений плюс единица границу Р-стека, поскольку этот запас предназначается для сохранения стека выражений в случае переключения процессов.

Иначе стек выражений записывается в память, начиная с адреса S, и следом записывается размер сохраненного стека. В этом случае в результате выполнения команды PC увеличивается на 1.

В терминах интерпретатора:

IF S+ESdepth+1>H THEN DEC(PC); TRAP(40h)

ELSE SaveExpStack

END

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД STOFV 0B4h@u"

.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД STOFV 0B4h@u"

@ASTOFV 0B4h@a

STORE expression stack with Formal function value on top

Код операции: 1 байт 0B4h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Если величина значения S-регистра, увеличенная на глубину стека выражений и еще на двойку, превышает значение границы Р-стека (Н-регистр), то PC уменьшается на единицу и возбуждается прерывание с номером 40h.

Иначе со стека берется значение, затем стек выражений записывается в память, начиная с адреса S, следом записывается размер спасенного стека, а за ним взятое со стека значение. В этом случае в результате выполнения команды PC увеличивается

на 1.

Замечание. Команда STOFV необходима для корректного использования команды CF (см.).

В терминах интерпретатора:

```
IF S+ESdepth+2>H THEN DEC(PC); TRAP(40h)
ELSE i:=Pop(); SaveExpStack; MEM[S]:=i; INC(S) END
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          COPT          0B5h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          COPT          0B5h@u"
```

```
@ACOPT          0B5h@a
```

COPY Top of expression stack

Код операции: 1 байт 0B5h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Берет со стека значение и затем загружает его на стек дважды (дублирует вершину стека).

PC увеличивается на 1.

В терминах интерпретатора:

```
i:=Pop(); Push(i); Push(i)
```

(* Будем понимать под словами "откат команды" следующие действия:

1) операнды, взятые со стека в ходе выполнения команды, помещаются обратно на стек;

2) PC возвращается в положение, предшествовавшее выборке исполняемой команды;

3) возбуждается прерывание с указанным номером.

*)

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          CPCOP          0B6h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          CPCOP          0B6h@u"
```

```
@ACPCOP          0B6h@a
```

Character array Parameter COPY

Код операции: 1 байт 0B6h

Непосредственные операнды: 1 байт 0h..0FFh

Длина команды: 2 байта

Действие:

Со стека берется верхняя граница байтового массива и вычисляется размер массива в словах. Если значение S-регистра, увеличенное на размер массива в словах, превышает границу P-стека, происходит откат команды с прерыванием номер 40h.

Иначе из кода выбирается непосредственный операнд - смещение, и в память с этим смещением относительно L-регистра записывается текущее значение S-регистра (для того, чтобы запомнить, откуда будет начинаться копия массива). Затем с начала свободной области памяти (S-регистр) поэлементно копируется массив, адрес которого берется со стека, до тех пор, пока не исчерпается размер массива. В этом случае PC увеличивается на 2, S-регистр увеличивается на размер массива.

В терминах интерпретатора:

```
i:=Pop(); (* High *) sz:=(i+4) DIV 4;
IF S+sz>H THEN Push(i); DEC(PC); TRAP(40h)
ELSE MEM[L+Next()]:=S; adr:=Pop();
  WHILE sz>0 DO MEM[S]:=MEM[adr]; INC(S); INC(adr) END
END
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          PCOP          0B7h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          PCOP          0B7h@u"
@APCOP          0B7h@a
structure Parameter allocate and COPy
Код операции:          1 байт          0B7h
Непосредственные операнды: 1 байт          0h..0FFh
Длина команды:          2 байта
Действие:
```

Со стека берется верхняя граница словного массива, увеличивается на единицу для получения его размера. Если значение S-регистра, увеличенное на размер массива, превышает границу Р-стека (H-регистр), происходит откат команды с прерыванием номер 40h.

Иначе из кода выбирается непосредственный операнд - смещение, и в память с этим смещением относительно L-регистра записывается текущее значение S-регистра (для того, чтобы запомнить, откуда будет начинаться копия массива). Затем с начала свободной области памяти записывается массив, адрес которого берется со стека, до тех пор, пока не исчерпается размер. В этом случае PC увеличивается на 2, а S-регистр увеличивается на размер массива.

В терминах интерпретатора:

```
i:=Pop(); (* High *) sz:=i+1;
IF S+sz>H THEN Push(i); DEC(PC); TRAP(40h)
ELSE MEM[L+Next()]:=S; adr:=Pop();
  WHILE sz>0 DO MEM[S]:=MEM[adr]; INC(S); INC(adr) END
END
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          FOR1          0B8h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          FOR1          0B8h@u"
@AFOR1          0B8h@a
enter FOR statement
Код операции:          1 байт          0B8h
Непосредственные операнды: 3 байта 0h..0FFh, 0h..0FFFFh
Длина команды:          4 байта
Действие:
```

Если S-регистр, увеличенный на 2, превышает границу Р-стека, происходит откат команды с прерыванием номер 40h.

Иначе со стека берется верхняя и нижняя границы цикла, адрес переменной цикла. Из кода выбирается шаг цикла - если в следующем байте 0, шаг положительный, если не 0 - шаг отрицательный. Если условия цикла не выполняются сразу (конечное значение меньше начального при положительном шаге или начальное меньше конечного при отрицательном шаге), PC

изменяется на столько, сколько указано в двухбайтовом непосредственном операнде, плюс 4. Иначе нижняя граница записывается по адресу переменной цикла, адрес и верхняя граница размещаются на Р-стеке для команды FOR2. В этом случае PC увеличивается в результате выполнения команды на 4.

Замечание. Команда реализована только на Кронос-2.2.

В терминах интерпретатора:

```
IF S+2>H THEN DEC(PC); TRAP(40h)
ELSE sz:=Next(); (* =0 up; #0 down *)
  hi:=Pop(); low:=Pop(); adr:=Pop(); k:=Next2()+PC;
  IF ((sz=0) & (low<=hi)) OR ((sz#0) & (low>=hi)) THEN
    MEM[adr]:=low;
    MEM[S]:=adr; INC(S); MEM[S]:=hi; INC(S);
  ELSE (* цикл не исполняется не разу *) PC:=k
END
END
```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	FOR2	0B9h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	FOR2	0B9h@u"
@AFOR2	0B9h@a	
end of FOR statement		
Код операции:	1 байт	0B9h
Непосредственные операнды:	3 байта	0h..0FFh, 0h..0FFFFh
Длина команды:	4 байта	

Действие:
Если значение шага цикла, взятое из однобайтового непосредственного операнда, больше 7Fh, ему переприсваивается отрицательное значение, равное (7Fh-шаг) (шаг цикла всегда в промежутке [-128..127]).

Если возможна еще одна итерация, то есть значение переменной цикла, измененное на шаг, не выходит за верхнюю границу, то значение переменной цикла изменяется на размер шага, PC уменьшается на значение двухбайтового непосредственного операнда.

Иначе S-регистр сдвигается на 2 слова назад, счеркивая с Р-стека значение верхней границы и адрес переменной цикла.

Замечание. Команда реализована только на Кронос-2.2.

В терминах интерпретатора:

```
hi:=MEM[S-1];
adr:=MEM[S-2];
sz:=Next();
IF sz>7Fh THEN
  sz:=7Fh-sz    (* шаг [-128..127] *)
END;
k:=-Next2()+PC;
i:=MEM[adr]+sz;
IF ((sz>=0) & (i>hi)) OR ((sz<0) & (i<hi)) THEN
  DEC(S,2); (* terminate *)
ELSE MEM[adr]:=i; PC:=k (* continue *)
```

END

.HEAD 0 "@ОПИСАНИЕ СИСТЕМЫ КОМАНД	ENTC	0BAh@u"
.HEAD 1 "@ОПИСАНИЕ СИСТЕМЫ КОМАНД	ENTC	0BAh@u"
@AENTC		0BAh@a

ENTER Case statement

Код операции:	1 байт	0BAh
Непосредственные операнды:	2 байта	0h..0FFFFh
Длина команды:	3 байта + размер таблицы	

Действие:

Выбор нужной альтернативы из таблицы альтернатив выбирающего оператора (CASE) и передача управления на ее код.

Если S-регистр, увеличенный на 2, превышает границу Р-стека, происходит откат команды с прерыванием номер 40h.

Иначе выбирает двухбайтовый непосредственный операнд и осуществляет переход на таблицу альтернатив. Далее из кода выбираются минимальное и максимальное значения альтернатив, представленные двухбайтовыми непосредственными операндами; вычисляется точка выхода из оператора CASE по формуле:

$$PC' = PC + 2(high - low),$$

где high и low - максимальное и минимальное значения альтернатив соответственно. Полученное значение PC' записывается на Р-стек.

Затем со стека берется параметр оператора CASE. В случае, если его значение не лежит в границах значений альтернатив, PC остается неизменным. Иначе осуществляется переход в таблицу альтернатив по формуле:

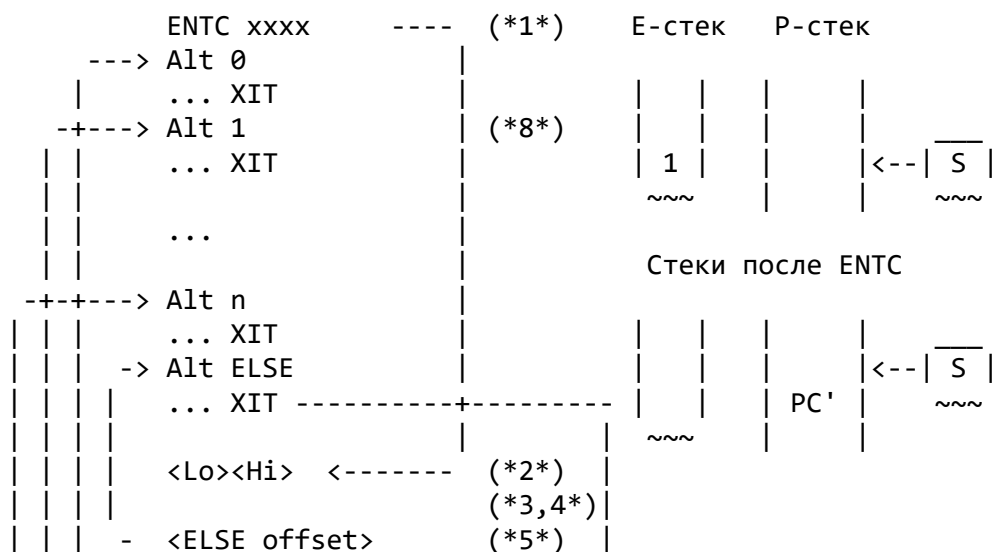
$$PC' = PC + 2(i - low + 1),$$

где i - значение параметра CASE; low - минимальное значение альтернатив.

Далее из кода выбирается двухбайтовое смещение, определяющее переход на код альтернативы. PC изменяется по формуле:

$$PC' = PC - offset.$$

Стеки до ENTC



		----	<Alt0 offset>	
		-----	<Alt1 offset>	(*6*)
			
		-----	<AltN offset>	
			<Continue code>	<-- PC' --

Замечание. Команда реализована только на Кронос-2.2.

В терминах интерпретатора:

```

IF S+1>H THEN DEC(PC); TRAP(40h)
ELSE PC:=Next2()+PC; (* jump to case table *)
  k:=Pop(); low:=Next2(); hi:=Next2();
  MEM[S]:=PC + 2*(hi-low) + 4; INC(S);(*PC for exit*)
  IF (k>=low) & (k<=hi) THEN
    PC:=PC+2*(k-low+1) (* jump into case table *)
  END;
  PC:=-Next2()+PC (* jump back to variant's code *)
END

```

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	XIT	0BBh@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	XIT	0BBh@u"
@AXIT	0BBh@a		
eXIT from case or control structure			
Код операции:	1 байт	0BBh	
Непосредственные операнды:	0 байт		
Длина команды:	1 байт		
Действие:			
Выход в точку в коде, помеченную при входе в структуру управления (ENTC или ENTS). PC устанавливается равным значению, взятому с P-стека.			

Замечание. Команда реализована только на Кронос-2.2.

В терминах интерпретатора:

```

DEC(S); PC:=MEM[S]

```

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	ADDP	0BCh@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	ADDP	0BCh@u"
@AADDP	0BCh@a		
ADD Program Counter			
Код операции:	1 байт	0BCh	
Непосредственные операнды:	0 байт		
Длина команды:	1 байт		
Действие:			
Грузит на стек PC, сложенный с величиной, взятой со стека.			
PC увеличивается на 1.			

Замечание. ADDP не реализована в П2.2.

В терминах интерпретатора:

```

Push(Pop()+PC);

```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      JMP      0BDh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      JMP      0BDh@u"
@AJMP                                0BDh@a
Jump
```

Код операции: 1 байт 0BDh

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

PC присваивается значение, взятое со стека.

Замечание. JMP не реализована в П2.2.

В терминах интерпретатора:

PC:=Pop();

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      ORJP      0BEh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      ORJP      0BEh@u"
@AORJP                                0BEh@a
```

short circuit OR Jump

Код операции: 1 байт 0BEh

Непосредственные операнды: 1 байт

Длина команды: 2 байта

Действие:

PC увеличивается на 1 в результате выборки кода команды.

Если значение, взятое со стека, не равно нулю, помещает на стек единицу. PC увеличивается на столько, сколько указано в однобайтовом непосредственном операнде плюс единица (за счет выборки этого операнда).

Иначе увеличивает PC на единицу. Таким образом, в этом случае PC в процессе выполнения команды увеличивается на два.

В терминах интерпретатора:

IF Pop()#0 THEN Push(1); PC:=Next()+PC

ELSE INC(PC)

END

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      ANDJP      0BFh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      ANDJP      0BFh@u"
@AANDJP                                0BFh@a
```

short circuit AND Jump

Код операции: 1 байт 0BFh

Непосредственные операнды: 1 байт 0h..0FFh

Длина команды: 2 байта

Действие:

PC увеличивается на 1 в результате выборки кода команды.

Если значение, взятое со стека, равно нулю, помещает на стек ноль. PC увеличивается на столько, сколько указано в однобайтовом непосредственном операнде плюс единица (за счет выборки этого операнда).

Иначе увеличивает PC на единицу. В этом случае PC в процессе выполнения команды увеличивается на два.

В терминах интерпретатора:

IF Pop()=0 THEN Push(0); PC:=Next()+PC

```
ELSE INC(PC)
END
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      MOVE      0C0h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      MOVE      0C0h@u"
@AMOVE      0C0h@a
```

MOVE block

Код операции: 1 байт 0Ch

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Со стека берется размер словного сегмента, который будет передвинут, затем адрес, начиная с которого двигать, затем адрес, начиная с которого писать этот сегмент. Передвижка происходит пословно, поэтому в случае определенного перекрытия сегментов хвост передвигаемого сегмента может испортиться еще до того, как его перенесли (см. текст на Модуле). Аккуратный сдвиг выполняется командой WM (96h).

Сдвиг перекрывающихся областей памяти в сторону младших адресов производится корректно; сдвиг в сторону старших можно использовать для заполнения области одинаковыми значениями.

PC увеличивается на 1.

В терминах интерпретатора:

```
sz:=Pop();
i:=Pop(); j:=Pop();
WHILE sz>0 DO
  MEM[j]:=MEM[i]; INC(i); INC(j); DEC(sz)
END
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      CHKNIL      0C1h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      CHKNIL      0C1h@u"
@ACHKNIL      0C1h@a
```

CHeck address for NIL

Код операции: 1 байт 0C1h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

PC увеличивается на 1. Если на стеке лежит NIL, то PC уменьшается на 1 и возбуждается прерывание.

Замечание. Команда не реализована на Кронос-2.2.

В терминах интерпретатора:

```
i:=Pop(); Push(i);
IF i=NIL THEN DEC(PC); TRAP(41h) END;
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      LSTA      0C2h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      LSTA      0C2h@u"
@ALSTA      0C2h@a
```

Load STring Address

Код операции: 1 байт 0C2h

Непосредственные операнды: 2 байта 0h..0FFFFh

Длина команды: 3 байта

Действие:

Грузит на стек адрес структуры, полученный сложением смещения, взятого из двухбайтового непосредственного операнда и содержимого 1-го слова области глобальных данных.

PC увеличивается на 3.

В терминах интерпретатора:

```
Push(MEM[G+1]+Next2());
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          COMP          0C3h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          COMP          0C3h@u"
@ACOMP                                     0C3h@a
COMParE strings
Код операции:                            1 байт          0C3h
Непосредственные операнды:              0 байт
Длина команды:                          1 байт
Действие:
```

Берет со стека словные адреса двух строк, преобразует их в байтовые, и читает из памяти по два байта, начиная с указанных адресов, до тех пор, пока один из байтов не окажется равным 0, либо до тех пор, пока они не окажутся разными. Тогда на стек загружается последний прочитанный байт сначала второй, затем первой строки с ведущими нулями. Полученные результаты можно сравнить командами EQU, NEQ, LSS и т.д..

PC увеличивается на 1.

В терминах интерпретатора:

```
i:=Pop()*4; j:=Pop()*4;
REPEAT
  a:=CHAR(ByteMEM[i]); b:=CHAR(ByteMEM[j]);
  INC(i); INC(j)
UNTIL (a=0c) OR (b=0c) OR (a#b); Push(a); Push(b)
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          GB          0C4h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          GB          0C4h@u"
@AGB                                     0C4h@a
Get procedure Base n level down
Код операции:                            1 байт          0C4h
Непосредственные операнды:              1 байт          0h..0FFh
Длина команды:                          2 байт
Действие:
```

Осуществляет проход по статической цепочке на столько уровней, сколько указано в непосредственном операнде, и грузит на стек адрес области локальных данных соответствующей статически объемлющей процедуры.

PC увеличивается на 2.

В терминах интерпретатора:

```
i:=L; k:=Next();
WHILE k>0 DO i:=MEM[i]; DEC(k) END; Push(i)
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          GB1          0C5h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          GB1          0C5h@u"
@AGB1                                     0C5h@a
```


Get procedure Base 1 level down

Код операции: 1 байт 0C5h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Берет из памяти и грузит на стек начало области локальных данных статически объемлющей процедуры.

PC увеличивается на 1.

В терминах интерпретатора:

Push(MEM[L])

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД

CHK

0C6h@u"

.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД

CHK

0C6h@u"

@ACHK

0C6h@a

range boundary CHeck

Код операции: 1 байт 0C6h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Берет со стека два значения, которые рассматривает как верхнюю и нижнюю границу соответственно; берет и грузит обратно следующий элемент стека, чтобы узнать его значение; если оно не лежит в пределах границ, помещает границы обратно на стек и возбуждает прерывание 4Ah.

PC увеличивается на 1.

В терминах интерпретатора:

hi:=Pop(); low:=Pop(); i:=Pop(); Push(i);

IF (i<low) OR (i>hi) THEN

Push(low); Push(hi); TRAP(4Ah)

END

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД

CHKZ

0C7h@u"

.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД

CHKZ

0C7h@u"

@ACHKZ

0C7h@a

array boundary CHeck (low=Zero)

Код операции: 1 байт 0C7h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Берет со стека значение, которое интерпретирует как верхнюю границу; в качестве нижней границы подразумевается 0. Берет и грузит обратно следующий элемент стека, чтобы узнать его значение; если оно не лежит в пределах границ, помещает верхнюю границу обратно на стек и возбуждает прерывание 4Ah.

PC увеличивается на 1.

В терминах интерпретатора:

hi:=Pop(); i:=Pop(); Push(i);

IF (i<0) OR (i>hi) THEN Push(hi); TRAP(4Ah) END

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД

ALLOC

0C8h@u"

.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД

ALLOC

0C8h@u"

@AALLOC	0C8h@a	
ALLOCate block		
Код операции:	1 байт	0C8h
Непосредственные операнды:	0 байт	
Длина команды:	1 байт	
Действие:		

Берет со стека размер отводимой памяти в словах. Если S-регистр, увеличенный на размер, превышает границу Р-стека, происходит откат команды с прерыванием номер 40h.

Иначе на стек грузится текущее значение S-регистра (начало отводимой памяти), S-регистр увеличивается на столько слов, каков размер отведенной памяти.

В этом случае PC увеличивается на 1.

В терминах интерпретатора:

```
sz:=Pop();
IF S+sz>H THEN Push(sz); DEC(PC); TRAP(40h)
ELSE Push(S); INC(S,sz) END
```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	ENTR	0C9h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	ENTR	0C9h@u"
@AENTR	0C9h@a	
ENTeR procedure		
Код операции:	1 байт	0C9h
Непосредственные операнды:	1 байт	0h..0FFh
Длина команды:	2 байта	
Действие:		

Выбирает из кода размер сегмента локальных данных в словах. Если S-регистр, увеличенный на размер, превышает границу Р-стека, происходит откат команды с прерыванием номер 40h.

Иначе увеличивает S-регистр на размер локального сегмента. В этом случае в результате выполнения команды PC увеличивается на 2.

В терминах интерпретатора:

```
sz:=Next();
IF S+sz>H THEN DEC(PC,2); TRAP(40h)
ELSE INC(S,sz) END
```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	RTN	0CAh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	RTN	0CAh@u"
@ARTN	0CAh@a	
ReTurN from procedure		
Код операции:	1 байт	0CAh
Непосредственные операнды:	0 байт	
Длина команды:	1 байт	
Действие:		

Восстанавливаются значение S-регистра до вызова процедуры (т.е. освобождается память, занятая в ходе выполнения процедуры), значение L-регистра процедуры, из которой была вызвана данная процедура, значение PC и, если вызывалась процедура из внешнего модуля, восстанавливается указатель на область глобальных данных исполняемого модуля и по нему

F-регистр.

В терминах интерпретатора:

```
S:=L; L:=MEM[S+1]; PC:=WORD(BITSET(MEM[S+2])*{0..0Fh});
IF ExternalBit IN BITSET(MEM[S+2]) THEN
  (* called from external module *)
  G:=MEM[S]; F:=CodePtr(MEM[G])
END;
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      NOP      0CBh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      NOP      0CBh@u"
@ANOP      0CBh@a
```

No OPeration

Код операции: 1 байт 0CBh

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

PC увеличивается на 1.

При исполнении этой команды больше ничего не делается.

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      CX      0CCh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      CX      0CCh@u"
@ACX      0CCh@a
```

Call eXternal

Код операции: 1 байт 0CCh

Непосредственные операнды: 2 байта 0h..0FFh

Длина команды: 3 байта

Действие:

Вызов внешней процедуры.

Если значение S-регистра, увеличенное на 4, превосходит границу Р-стека, происходит откат команды с прерыванием номер 40h.

Из кода выбираются два однобайтовых непосредственных операнда. Первый интерпретируется как номер внешнего модуля в локальной DFT, из которого вызывается процедура, а второй - как номер процедуры. После этого производится разметка Р-стека для вызова указанной процедуры, то есть в нулевое слово области локальных данных заносится G-регистр (для последующего восстановления командой RTN), в первое слово - указатель на область локальных данных той процедуры, из которой произошел вызов (выстраивается динамическая цепочка), во второе слово - PC точки вызова с пометкой в 31-м бите о том, что вызов был внешним (для последующего восстановления PC). Затем по ссылке восстанавливается G-регистр внешнего модуля, по нему - F-регистр, PC устанавливается на начало вызванной процедуры.

В терминах интерпретатора:

```
IF S+4<=H THEN j:=MEM[G-Next()-1]; (* big DFT *)
  i:=Next(); Mark(G,TRUE);
  G:=MEM[j]; F:=CodePtr(MEM[G]); PC:=GetPc(i);
ELSE DEC(PC); TRAP(40h)
END
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД      CI      0CDh@u"
```

```
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          CI          0CDh@u"
@ACI          0CDh@a
```

Call procedure at Intermediate level

```
Код операции:          1 байт          0CDh
Непосредственные операнды: 1 байт          0h..0FFh
Длина команды:          2 байта
```

Действие:

Если значение S-регистра, увеличенное на 4, превосходит границу Р-стека, происходит откат команды с прерыванием номер 40h.

Иначе выбирается непосредственный операнд, интерпретируемый как номер вызываемой процедуры, после чего производится разметка Р-стека для вызова процедуры, то есть в нулевое слово области локальных данных заносится слово, взятое со стека (например, указатель на область локальных данных процедуры, статически объемлющей ту, из которой происходит вызов, для обеспечения доступа к ее локальным данным - строится статическая цепочка), в первое слово - указатель на область локальных данных той процедуры, из которой произошел вызов (строится динамическая цепочка), во второе слово - РС точки вызова (для последующего восстановления РС при возврате из процедуры). Затем РС устанавливается на начало вызываемой процедуры.

В терминах интерпретатора:

```
IF S+4<=H THEN
    i:=Next(); Mark(Pop(),FALSE); PC:=GetPc(i);
ELSE DEC(PC); TRAP(40h) END
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          CF          0CEh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          CF          0CEh@u"
@ACF          0CEh@a
```

Call Formal procedure

```
Код операции:          1 байт          0CEh
Непосредственные операнды: 0 байт
Длина команды:          1 байт
```

Действие:

Если значение S-регистра, увеличенное на 3, превосходит границу Р-стека, происходит откат команды с прерыванием номер 40h.

Иначе с вершины Р-стека берется слово, в котором содержится процедурное значение вызываемой процедуры, после чего производится разметка Р-стека для вызова процедуры, то есть в нулевое слово области локальных данных заносится G-регистр (выстраивается статическая цепочка), в первое слово - указатель на область локальных данных той процедуры, из которой произошел вызов (динамическая цепочка), во второе слово - РС точки вызова с пометкой в 31-м бите о том, что вызов был внешним, поскольку формальная процедура может быть как локальной, так и внешней (для последующего восстановления РС). Затем РС устанавливается на начало вызываемой процедуры. Старший байт процедурного значения интерпретируется как номер процедуры, три младшие - как адрес входа в глобальную DFT. По этому адресу вычисляется адрес глобальных данных модуля, в

котором содержится вызываемая процедура, по нему - F-регистр. PC устанавливается на начало процедуры.

Замечания. Ссылка на адрес своей области глобальных данных в 0-м слове локальной DFT модуля необходима, чтобы команда CF выполнялась корректно для собственных локальных процедур модуля.

В терминах интерпретатора:

```
IF S+3<=H THEN i:=MEM[S-1]; DEC(S); Mark(G,TRUE);
  k:=i DIV 1000000h; i:=i MOD 1000000h;
  G:=MEM[i]; F:=CodePtr(MEM[G]); PC:=GetPc(k);
ELSE DEC(PC); TRAP(40h)
END
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          CL          0CFh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          CL          0CFh@u"
@ACL                                          0CFh@a
```

Call Local procedure

Код операции:	1 байт	0CFh
Непосредственные операнды:	1 байт	0h..0FFh
Длина команды:	2 байта	

Действие:

Если значение S-регистра, увеличенное на 4, не превышает границы P-стека (H-регистр), выбирается непосредственный операнд, интерпретируемый как номер вызываемой процедуры, затем производится разметка P-стека для вызова локальной процедуры: в нулевое и первое слова, начиная с адреса S, заносится дважды значение L-регистра; во второе слово - текущее значение PC, после чего в L-регистр заносится значение S-регистра, затем значение S-регистра увеличивается на 4, PC устанавливается на начало процедуры, номер которой был взят из кода.

Иначе происходит откат команды с прерыванием номер 40h.

В терминах интерпретатора:

```
IF S+4<=H THEN i:=Next(); Mark(L,FALSE); PC:=GetPc(i);
ELSE DEC(PC); TRAP(40h) END
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          CL0..CL0E          0D0h..0DFh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          CL0..CL0E          0D0h..0DFh@u"
@ACL0..CL0F                                0D0h..0DFh@a
```

Call Local procedure

Код операции:	4 бита	0Dh
Непосредственные операнды:	4 бита	0h..0Fh
Длина команды:	1 байт	

Действие:

Команды для вызова локальных процедур с номерами из диапазона 0h..0Fh (см. CL).

В терминах интерпретатора:

```
IF S+4<=H THEN Mark(L,FALSE); PC:=GetPc(IR MOD 10h);
ELSE DEC(PC); TRAP(40h) END
```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	INCL	0E0h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	INCL	0E0h@u"
@AINCL	0E0h@a	

INCLude in set

Код операции: 1 байт 0E0h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Со стека берется значение N, которое интерпретируется как номер бита. Он может превышать 31.

Затем со стека берется адрес слова, и в N-й битовой позиции относительно этого адреса выставляется единица. PC увеличивается на 1.

В терминах интерпретатора:

i:=Pop(); j:=Pop() + i DIV 32;

MEM[j]:=INTEGER(BITSET(MEM[j]) + {i MOD 32});

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	EXCL	0E1h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	EXCL	0E1h@u"
@AEXCL	0E1h@a	

EXCLude from set

Код операции: 1 байт 0E1h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Со стека берется значение N, которое интерпретируется как номер бита. Он может превышать 31.

Затем со стека берется адрес слова, и в N-й битовой позиции относительно этого адреса выставляется ноль. PC увеличивается на 1.

В терминах интерпретатора:

i:=Pop(); j:=Pop() + i DIV 32;

MEM[j]:=INTEGER(BITSET(MEM[j]) - {i MOD 32});

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	INL	0E2h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	INL	0E2h@u"
@AINL	0E2h@a	

membership IN Long set

Код операции: 1 байт 0E2h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Со стека берутся размер множества в битах адрес и номер бита. Номер бита может быть произвольным числом. Если номер бита меньше нуля либо не меньше размера множества, то на стек помещается значение 0. Иначе значение, извлеченное из соответствующей битовой позиции относительно взятого адреса, расширяется ведущими нулями до слова и помещается на стек.

PC увеличивается на 1.

Замечание. Команда не реализована на П2.2.

В терминах интерпретатора:

```
k:=Pop(); j:=Pop(); i:=Pop();
IF (i<0) OR (i>=k) THEN Push(0)
ELSE
  Push( (i MOD 32) IN BITSET(MEM[j+i DIV 32]) );
END;
```

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	QUOT	0E3h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	QUOT	0E3h@u"
@AQUOT	0E3h@a		
QUOT	ient commands		
Код операции:	1 байт	0E3h	
Непосредственные операнды:	1 байт	0h..0FFh	
Длина команды:	2 байта		

Действие:

Со стека берутся два операнда и выбирается однобайтовый непосредственный операнд из кода.

PC увеличивается на 2.

Если значение непосредственного операнда:

- 0, то командой осуществляется деление на степень двойки с округлением к нулю; при этом делимое - нижний операнд, степень - верхний.
- 1, то командой осуществляется деление нижнего операнда на верхний с округлением к нулю;

Определим операцию "взятие по модулю" формулой:

$$X \text{ MOD } N = X - (X \text{ QOU } N) * N, \quad (1)$$

где QOU - деление с округлением к нулю.

- 2, то команда является операцией "взятие по модулю", определяемой формулой (1), где N - степень двойки; при этом делимое - нижний операнд, степень - верхний;
- 3, то команда является операцией "взятие по модулю", определяемой формулой (1), при этом нижний операнд берется по модулю верхнего;
- любое другое, то PC откатывается назад и возбуждается прерывание с номером 7h.

Замечание. Команда не реализована для П2.2.

В терминах интерпретатора:

```
i:=Pop(); j:=Pop();
CASE Next() OF
  |0: Push( j SHRQ i ); -- деление на степень двойки
  |1: Push( j QOU i ); -- деление с округлением к нулю
  |2: Push( j ANDQ i ); -- модуль по степени двойки
  |3: Push( j REM i ); -- модуль
ELSE TRAP(7); DEC(PC,2)
END;
```

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	INC1	0E4h@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	INC1	0E4h@u"
@AINC1	0E4h@a		

INCrement by 1

Код операции: 1 байт 0E4h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Увеличивает на единицу значение, лежащее в памяти по адресу, взятому со стека.

PC увеличивается на 1.

В терминах интерпретатора:

INC(MEM[Pop()])

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД

DEC1

0E5h@u"

.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД

DEC1

0E5h@u"

@ADEC1 0E5h@a

DECrement by 1

Код операции: 1 байт 0E5h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Уменьшает на единицу значение, лежащее в памяти по адресу, взятому со стека.

PC увеличивается на 1.

В терминах интерпретатора:

DEC(MEM[Pop()])

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД

INC

0E6h@u"

.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД

INC

0E6h@u"

@AINC 0E6h@a

INCrement

Код операции: 1 байт 0E6h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Берет со стека целое число и адрес. Значение, лежащее в памяти по указанному адресу, увеличивает на вышеупомянутое число.

PC увеличивается на 1.

В терминах интерпретатора:

i:=Pop(); INC(MEM[Pop()],i)

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД

DEC

0E7h@u"

.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД

DEC

0E7h@u"

@ADEC 0E7h@a

DECrement

Код операции: 1 байт 0E7h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Берет со стека целое число и адрес. Значение, лежащее в памяти по указанному адресу, уменьшает на указанное число.

PC увеличивается на 1.

В терминах интерпретатора:

```
i:=Pop(); DEC(MEM[Pop()],i)
```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	STOT	0E8h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	STOT	0E8h@u"

@ASTOT 0E8h@a

STOre Top on procedure stack

Код операции: 1 байт 0E8h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Если значение S-регистра, увеличенное на 1, превышает границу Р-стека, происходит откат команды с прерыванием номер 40h.

Иначе со стека выражений берется слово и помещается на Р-стек.

PC увеличивается на 1.

В терминах интерпретатора:

```
IF S+1>H THEN DEC(PC); TRAP(40h)
```

```
ELSE MEM[S]:=Pop(); INC(S)
```

```
END
```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LODT	0E9h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LODT	0E9h@u"

@ALODT 0E9h@a

LOaD Top of procedure stack

Код операции: 1 байт 0E9h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Значение с вершины Р-стека (элемент с адресом S-1) загружается на стек выражений. Значение S-регистра уменьшается на 1.

PC увеличивается на 1.

В терминах интерпретатора:

```
DEC(S); Push(MEM[S])
```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LXA	0EAh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LXA	0EAh@u"

@ALXA 0EAh@a

Load indeXed Address

Код операции: 1 байт 0EAh

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

По взятым со стека размеру элемента (в словах), номеру элемента и адресу структуры (например, массива) вычисляет адрес элемента и помещает его на стек.

PC увеличивается на 1.

В терминах интерпретатора:

```
sz:=Pop(); i:=Pop(); adr:=Pop(); Push(adr+i*sz)
```

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LPC	0EBh@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LPC	0EBh@u"
@ALPC	0EBh@a		
Load Procedure Constant			
Код операции:	1 байт	0EBh	
Непосредственные операнды:	2 байта	0h..0FFh	
Длина команды:	3 байта		
Действие:			

Из кода выбираются два однобайтовых непосредственных операнда. Первый интерпретируется как номер модуля в локальной DFT, второй - как номер процедуры. Номера упаковываются следующим образом: в старший байт помещается номер процедуры, в младшие три - элемент локальной DFT с указанным номером модуля. Полученное слово грузится на стек.

PC увеличивается на 3.

В терминах интерпретатора:

```
i:=Next(); j:=Next(); Push(j*1000000h+MEM[G-i-1])
```

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	BVU	0ECh@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	BVU	0ECh@u"
@ABVU	0ECh@a		
Bit Block Unpack			
Код операции:	1 байт	0ECh	
Непосредственные операнды:	0 байт		
Длина команды:	1 байт		
Действие:			

Со стека берется размер битовой вырезки. Если он меньше 1 или больше 32, происходит откат команды с прерыванием номер 4Ah.

Иначе формируется битовая вырезка указанного размера, с битовым смещением, взятым со стека, и начинающаяся со словного адреса, взятого со стека. Вырезка дополняется ведущими нулями до слова и грузится на стек.

PC увеличивается на 1.

Замечание. Команда не реализована на Кронос-2.2.

В терминах интерпретатора:

```
sz:=Pop();
IF (sz<1) OR (sz>32) THEN
  Push(sz); DEC(PC); TRAP(4Ah)
END;
i:=Pop(); adr:=Pop();
(* j:=битовая вырезка длиной sz, начиная с
  битового адреса (adr,i)
*)
Push(j);
```

.HEAD 0	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	BVP	0EDh@u"
.HEAD 1	"@УОПИСАНИЕ СИСТЕМЫ КОМАНД	BVP	0EDh@u"

@ABBP	0EDh@a	
Bit Block Pack		
Код операции:	1 байт	0EDh
Непосредственные операнды:	0 байт	
Длина команды:	1 байт	
Действие:		

PC увеличивается на 1.

Со стека берется битовое значение и его размер. Если размер меньше 1 или больше 32, размер грузится на стек, PC уменьшается на 1 и возбуждается прерывание с номером 4Ah.

Иначе указанное размером число младших битов вырезки с битовым смещением, взятым со стека, упаковывается по словному адресу, взятому со стека.

Замечание. Команда не реализована на Кронос-2.2.

В терминах интерпретатора:

```

j:=Pop(); sz:=Pop();
IF (sz<1) OR (sz>32) THEN
  Push(sz); DEC(PC); TRAP(4Ah)
END;
i:=Pop(); adr:=Pop();
(* Упаковывает sz младших битов из j по битовому адресу
(adr,i)*)

```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	BBLT	0EEh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	BBLT	0EEh@u"
@ABBLT	0EEh@a	
Bit Block Transfer		
Код операции:	1 байт	0EEh
Непосредственные операнды:	0 байт	
Длина команды:	1 байт	
Действие:		

Со стека берутся битовый размер пересылаемого сегмента (может быть больше 32), битовое смещение и словный адрес пересылаемого сегмента, битовое смещение и словный адрес, куда пересылается сегмент, и производится пересылка.

PC увеличивается на 1.

Замечание. В случае перекрытия сегментов можно получить самый неожиданный результат. Ожидаемый результат даст использование команды BM (97h).

Замечание. Команда не реализована на Кронос-2.2.

В терминах интерпретатора:

```

sz:=Pop();
i:=Pop(); adr:=Pop();
j:=Pop(); adr1:=Pop();
(* Переслать sz битов (adr,i) -> (adr1,j) *)

```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	PDX	0EFh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	PDX	0EFh@u"
@APDX	0EFh@a	

Prepare Dynamic index

Код операции: 1 байт 0EFh
Непосредственные операнды: 0 байт
Длина команды: 1 байт

Действие:

PC увеличивается на 1.

Со стека берутся два операнда, нижний интерпретируется как указатель на дескриптор массива, верхний - как индекс элемента массива. Дескриптор массива - пара слов в памяти, в первом из них лежит адрес начала массива, во втором - верхняя граница массива.

На стек грузится адрес начала массива и индекс элемента. Если индекс оказался меньше 0 или больше, чем граница, указанная в дескрипторе, возбуждается прерывание с номером 4Ah.

Замечание. Команда не реализована на Кронос-2.2.

В терминах интерпретатора:

```
i:=Pop(); dyn:=Pop();  
Push(dyn^.adr); Push(i);  
IF (i<0) OR (i>dyn^.high)  
  THEN TRAP(4Ah)  
END
```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД SWAP 0F0h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД SWAP 0F0h@u"
@ASWAP 0F0h@a
SWAP

Код операции: 1 байт 0F0h
Непосредственные операнды: 0 байт
Длина команды: 1 байт

Действие:

Меняет местами два верхних элемента стека.
PC увеличивается на 1.

В терминах интерпретатора:

```
i:=Pop(); j:=Pop(); Push(i); Push(j)
```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД LPA 0F1h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД LPA 0F1h@u"
@ALPA 0F1h@a

Load Parameter Address

Код операции: 1 байт 0F1h
Непосредственные операнды: 1 байт 0h..0FFh
Длина команды: 2 байта

Действие:

Грузит на стек величину, равную значению L-регистра минус значение непосредственного операнда минус единица.

PC увеличивается на 2.

В терминах интерпретатора:

```
Push(L-Next()-1);
```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LPW	0F2h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	LPW	0F2h@u"
@ALPW	0F2h@a	

Load Parameter Word

Код операции:	1 байт	0F2h
Непосредственные операнды:	1 байт	0h..0FFh
Длина команды:	2 байта	

Действие:

Грузит на стек слово, лежащее по адресу, вычисленному следующим образом: значение L-регистра минус значение непосредственного операнда минус единица.

PC увеличивается на 2.

В терминах интерпретатора:

Push(MEM[L-Next()-1]);

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SPW	0F3h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SPW	0F3h@u"
@ASPW	0F3h@a	

Store Parameter Word

Код операции:	1 байт	0F3h
Непосредственные операнды:	1 байт	0h..0FFh
Длина команды:	2 байта	

Действие:

Берет со стека слово и записывает в память по адресу, равному значению L-регистра минус значение непосредственного операнда минус единица.

PC увеличивается на 2.

В терминах интерпретатора:

MEM[L-Next()-1]:=Pop();

Замечание. Три вышеописанные команды LPA, LPW и SPW поддерживают способ адресации, позволяющий работать с процедурными параметрами, положенными на P-стек до вызова процедуры.

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SSWU	0F4h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	SSWU	0F4h@u"
@ASSWU	0F4h@a	

Store Stack Word Undestructive

Код операции:	1 байт	0F4h
Непосредственные операнды:	0 байт	
Длина команды:	1 байт	

Действие:

Берет со стека слово и записывает его в память по адресу, взятому со стека, после чего снова загружает это слово на стек.

PC увеличивается на 1.

В терминах интерпретатора:

i:=Pop(); MEM[Pop()]:=i; Push(i)

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	RCHK	0F5h@u"
------------------------------------	------	---------

.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	RCHK	0F5h@u"
@ARCHK	0F5h@a	

Range CHeck

Код операции:	1 байт	0F5h
---------------	--------	------

Непосредственные операнды:	0 байт
----------------------------	--------

Длина команды:	1 байт
----------------	--------

Действие:

Со стека берутся два операнда, которые интерпретируются как верхняя (верхний операнд) и нижняя (нижний операнд) границы отрезка. Если число, лежащее на стеке, выходит за границы отрезка, на стек грузится 0 (FALSE), иначе - 1 (TRUE).

PC увеличивается на 1.

Замечание. Команда не реализована на Кронос-2.2.

В терминах интерпретатора:

```
hi:=Pop(); low:=Pop(); i:=Pop(); Push(i);
```

```
Push( (i>=low) & (i<=hi) );
```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	RCHZ	0F6h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	RCHZ	0F6h@u"
@ARCHZ	0F6h@a	

Range CHeck (low=Zero)

Код операции:	1 байт	0F6h
---------------	--------	------

Непосредственные операнды:	0 байт
----------------------------	--------

Длина команды:	1 байт
----------------	--------

Действие:

Со стека берется операнд, который интерпретируется как верхняя граница отрезка. Нижней границей служит 0. Если число, лежащее на стеке, выходит за границы отрезка, на стек грузится 0 (FALSE), иначе - 1 (TRUE).

PC увеличивается на 1.

Замечание. Команда не реализована на Кронос-2.2.

В терминах интерпретатора:

```
hi:=Pop(); i:=Pop(); Push(i);
```

```
Push( (i>=0) & (i<=hi) );
```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	CM	0F7h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД	CM	0F7h@u"
@ACM	0F7h@a	

Call procedure from dynamic Module

Код операции:	1 байт	0F7h
---------------	--------	------

Непосредственные операнды:	1 байт	0h..0FFh
----------------------------	--------	----------

Длина команды:	2 байта
----------------	---------

Действие:

Если позволяет размер процедурного стека, из кода выбирается непосредственный операнд, интерпретируемый как номер процедуры. Осуществляется вызов процедуры с указанным номером из модуля, G-регистр которого взят с P-стека (с которого он при этом счеркивается). Разметка P-стека производится так же, как в случае вызова формальной процедуры (см. CF). PC устанавливается на начало вызываемой процедуры.

Иначе происходит откат команды с прерыванием номер 40h.

Замечание. Команда не реализована на Кронос-2.2.

В терминах интерпретатора:

```
IF S+4<=H THEN
  i:=Next();
  DEC(S); j:=MEM[S];
  Mark(G,TRUE);
  G:=j; F:=CodePtr(MEM[G]); PC:=GetPc(i);
ELSE DEC(PC); TRAP(40h)
END;
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          CHB          0F8h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          CHB          0F8h@u"
@ACHKBX                                0F8h@a
```

Сheck Boxes

Код операции: 1 байт 0F8h

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Берет со стека два операнда, которые интерпретируются как указатели на упакованные прямоугольники. Прямоугольник определяется диагональю, проведенной из левого нижнего угла в правый верхний. Координаты каждого конца диагонали упакованы в слово, в котором старшие 16 битов определяют координату по вертикали, младшие 16 - по горизонтали. Все координаты положительные (в диапазоне 0..0FFFFh). Если прямоугольники пересекаются, на стек помещается 1 (TRUE), иначе - 0 (FALSE).

PC увеличивается на 1.

Замечание. Команда реализована только на Кронос-2.6.

В терминах интерпретатора:

```
p0:=Pop(); p1:=Pop();
x0b:=INTEGER( BITSET(p0^)      *{0..15});
y0b:=INTEGER((BITSET(p0^)<<16)*{0..15});
INC(p0);
x0e:=INTEGER( BITSET(p0^)      *{0..15});
y0e:=INTEGER((BITSET(p0^)<<16)*{0..15});
x1b:=INTEGER( BITSET(p1^)      *{0..15});
y1b:=INTEGER((BITSET(p1^)<<16)*{0..15});
INC(p1);
x1e:=INTEGER( BITSET(p1^)      *{0..15});
y1e:=INTEGER((BITSET(p1^)<<16)*{0..15});
Push( (x1b<=x0e) & (y1b<=y0e) & (x0b<=x1e) & (y0b<=y1e) );
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          BMG          0F9h@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          BMG          0F9h@u"
@ABMG                                0F9h@a
```

BitMap Graphic commands

Код операции: 1 байт 0F9h

Непосредственные операнды: 1 байт 0h..0FFh

Длина команды: 2 байта

Действие:

Команда для поддержки bitmap-графики. Поскольку точное описание команды на языке Модуля-2 весьма обширно и представляет интерес только для специального изучения, оно выделено в самостоятельный документ "Интерпретатор графических команд". Здесь приводится лишь поверхностное описание функций команды BMG.

Команда реализована на Кронос-2.6 и с некоторыми ограничениями на Кронос-2.5.

Для применения команды рекомендуется использовать соответствующую библиотеку графики.

Команда выбирает байт из кода. Если полученное целое число выходит из диапазона $[0..9]$, то происходит откат PC на 2 и возбуждается прерывание номером 49h. Иначе число интерпретируется как номер одной из описанных ниже подкоманд и соответствующая подкоманда исполняется.

Все параметры передаются командам либо непосредственно через стек, либо через указатели, размещенные на стеке, которые указывают на массив параметров в памяти. В описании операнды перечисляются в том порядке, в котором они берутся со стека.

Команды осуществляют операции над областью памяти (битовой картой), определенной так называемым bitmap-дескриптором (BMD).

Bitmap-дескриптор представляет собой запись из пяти слов, первое из которых - горизонтальный размер битовой карты в битах, второе - вертикальный размер битовой карты в битах, третье - число слов в одной строке изображения, четвертое - адрес битовой карты, пятое - шаблон, который использует подкоманда DLN (см. ниже).

Код: 0

Подкоманда: IN_RECT (IN RECTangle?)

Действие: проверка принадлежности точки прямоугольнику с диагональю, проведенной из левого нижнего угла в правый верхний, с координатами $(0,0)$ и (w,h) .

Операнды на стеке:

1,2) горизонтальная и вертикальная координаты точки;

3,4) горизонтальная и вертикальная координаты правого верхнего угла прямоугольника.

Результат на стеке:

FALSE (0) - если точка не принадлежит прямоугольнику; TRUE (1) - если принадлежит.

Код: 1

Подкоманда: DVL (Display Vertical Lines)

Действие: построение на битовой карте изображения отрезка вертикальной линии указанной длины в указанной моде от указанной точки в сторону увеличения адресов.

Операнды на стеке:

1) режим, в котором будет строиться изображение отрезка (REPLACE, OR, XOR, BIC), определяется

операндом, равным 0,1,2 или 3 соответственно.

2) указатель на BMD;

3,4) координаты точки по горизонтали и вертикали;

5) длина линии.

Результат на стеке: нет.

Код: 2

Подкоманда: BBLT_G (BBLT-Graphic)

Действие: команда BBLT с режимами: производит логические операции (REPLACE, OR, XOR, BIC) над приемником и источником, являющимися массивами битов.

Операнды на стеке:

1) режим, указывающий, какая операция будет произведена над приемником и источником (REPLACE, OR, XOR, BIC), определяется операндом, равным 0,1,2 или 3 соответственно.

2) операнды, аналогичные операндам команды BBLT.

Результат на стеке: нет.

Код: 3

Подкоманда: DCH (Display CHar)

Действие: построение на битовой карте изображения символа с указанной кодировкой из указанного шрифта.

Операнды на стеке:

1) режим, в котором будет изображаться символ (REPLACE, OR, XOR, BIC), определяется операндом, равным 0, 1, 2 или 3 соответственно;

2) указатель на BMD;

3,4) координаты символа по горизонтали и по вертикали;

5) указатель на дескриптор шрифта;

6) кодировка символа, изображение которого требуется построить.

Результат на стеке: нет.

Код: 4

Подкоманда: CLP (CLiPping)

Действие: клипирование отрезка прямой прямоугольником с диагональю, проведенной из левого нижнего угла в правый верхний, с координатами (0,0) и (w,h).

Операнды на стеке:

1) указатель на массив из 4-х параметров. Перед выполнением команды в эти 4 слова должны быть занесены координаты концов отрезка. Если отрезок пересекается с прямоугольником, то команда заносит в них координаты концов клипированного отрезка.

2,3) горизонтальная и вертикальная координаты правого верхнего угла прямоугольника.

Результат на стеке:

TRUE, если отрезок пересекается с прямоугольником, иначе FALSE.

Код: 5

Подкоманда: DLN (Display LiNe)

Действие: построение на битовой карте изображения отрезка произвольной прямой по двум точкам.

Операнды на стеке:

- 1) режим, в котором будет изображаться отрезок (REPLACE, OR, XOR, BIC), определяется операндом, равным 0,1,2 или 3 соответственно;
- 2) указатель на BMD;
- 3,4) горизонтальная и вертикальная координаты одного конца отрезка;
- 5,6) горизонтальная и вертикальная координаты другого конца отрезка.

Результат на стеке: нет.

Замечание. Подкоманда DLN не отслеживает выход отрезка за границы битовой карты.

Код: 6

Подкоманда: CRC (CiRCus)

Действие: реализует тело цикла изображения на битовой карте окружности с указанными центром и радиусом. Строит на битовой карте 8 точек, принадлежащих этой окружности.

Операнды на стеке:

- 1) режим, в котором будет строиться изображение (REPLACE, OR, XOR, BIC), определяется операндом, равным 0,1,2 или 3 соответственно;
- 2) указатель на BMD;
- 3) указатель на массив параметров (радиус, θ и радиус, поделенный надвое).
- 4,5) горизонтальная и вертикальная координаты центра окружности.

Результат на стеке: нет.

Код: 7

Подкоманда: ARC (ARC)

Действие: реализует тело цикла изображения на битовой карте дуги окружности с указанными центром, радиусом и углом, заданным двумя точками.

Операнды на стеке:

- 1) режим, в котором будет строиться изображение (REPLACE, OR, XOR, BIC), определяется операндом, равным 0,1,2 или 3 соответственно;
- 2) указатель на BMD;
- 3) указатель на массив параметров.

Результат на стеке: нет.

Код: 8

Подкоманда: TRF (TRiangle Filling)

Действие: поддерживает процедуру заполнения области, ограниченной прямыми. Вычисляет координаты точек, между которыми нужно провести горизонтальную линию, чтобы закрасить область.

Операнды на стеке:

указатель на массив параметров. В результате

выполнения команды в него заносятся вычисленные координаты.

Результат на стеке: нет.

Код: 9

Подкоманда: CRF (CiRcus Filling)

Действие: поддерживает процедуру заполнения круга. Вычисляет координаты точек, между которыми нужно провести горизонтальную линию, чтобы закрасить круг.

Операнды на стеке:

указатель на массив параметров. В результате выполнения команды в них заносятся вычисленные координаты.

Результат на стеке: нет.

В терминах интерпретатора:

```
CASE Next() OF
```

```
|0: IN_RECT
```

```
|1: DVL
```

```
|2: BBLT_G
```

```
|3: DCH
```

```
|4: CLP
```

```
|5: DLN
```

```
|6: CRC
```

```
|7: ARC
```

```
|8: TRF
```

```
|9: CRF
```

```
ELSE
```

```
Trap(49h)
```

```
END;
```

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД

ACTIV

0FAh@u"

.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД

ACTIV

0FAh@u"

@AACTIV 0FAh@a

ACTIVE process

Код операции: 1 байт 0FAh

Непосредственные операнды: 0 байт

Длина команды: 1 байт

Действие:

Загружает на стек адрес дескриптора активного процесса (содержимое Р-регистра процессора).

РС увеличивается на 1.

В терминах интерпретатора:

Push(P)

.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД

USR

0FBh@u"

.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД

USR

0FBh@u"

@AUSR 0FBh@a

USeR defined functions

Код операции: 1 байт 0FBh

Непосредственные операнды: 1 байт 0h..0FFh

Длина команды: 2 байта

Действие:

Команда оставлена для дальнейшего возможного расширения системы команд.

PC увеличивается на 1.

В терминах интерпретатора:

i:=Next(); (* *)

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          SYS          0FCh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          SYS          0FCh@u"
@ASYS                                0FCh@a
rarely SYStem functions
Код операции:                        1 байт          0FCh
Непосредственные операнды:          1 байт          0h..0FFh
Длина команды:                      2 байта
Действие:
```

Если в значение непосредственного операнда равно 0, то грузит на стек слово, идентифицирующее процессор. Если 2 - модель процессора. Иначе возбуждается прерывание с номером 7h.

PC увеличивается на 2.

В терминах интерпретатора:

```
CASE Next() OF
|00h: (* PID Processor IDent *)
      (* Push(PID) *)
|02h: (* PM Processor Model *)
      (* Push(PM) *)
(* Остальные могут быть различными в разных моделях *)
ELSE TRAP(7h)
END;
```

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          NII          0FDh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          NII          0FDh@u"
@ANII                                0FDh@a
Never Implemented Instruction
Код операции:                        1 байт          0FDh
Непосредственные операнды:           0 байт
Длина команды:                      1 байт
Действие:
```

При исполнении этой команды возбуждается прерывание с номером 7h. PC увеличивается на 1.

Замечание. Команда не реализована на Кронос-2.2.

В терминах интерпретатора:

TRAP(7h);

```
.HEAD 0 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          INVLD        0FFh@u"
.HEAD 1 "@УОПИСАНИЕ СИСТЕМЫ КОМАНД          INVLD        0FFh@u"
@AINVLD                              0FFh@a
INVAlid command
Код операции:                        1 байт          0FFh
Непосредственные операнды:           0 байт
Длина команды:                      1 байт
Действие:
```

При вызове этой команды возбуждается прерывание с номером 49h. PC увеличивается на 1.

В терминах интерпретатора:
TRAP(49h)