



# **Язык инструкций ЛИР-986**

**Версия 2.1**  
29 октября 2008 г.

## Оглавление

1. Типы данных.....	3
2. Команды записи/чтения и модификации данных.....	4
3. Логические операции.....	8
4. Арифметические операции.....	9
5. Операции сравнения.....	11
6. Команды работы с таймерами/счетчиками.....	14
7. Команды индексирования.....	16
8. Команды перехода.....	17
9. Карта команд виртуальной машины.....	18
10. Архитектура.....	19
10.1. Массивы состояния входов и выходов.....	20
10.2. Память данных.....	21
10.3. Память констант.....	-
10.4. Массив таймеров/счетчиков.....	22
11. Сводная таблица команд виртуальной машины.....	23

## Типы данных

Тип	Обозначение	Разрядность
Входы	X0...X255	1 бит
Выходы	Y0...Y255	1 бит
Маркеры	M0...M1023	1 бит
Байты	B0...B127	1 байт
Слова	W0...W63	2 байта
Константы	K0...K255	2 байта
Входы таймеров/счетчиков	TX0...TX255	1 бит
Выходы таймеров/счетчиков	TY0...TY255	1 бит
Значения таймеров/счетчиков	T0...T255	2 байта

## Команды записи/чтения и модификации данных

Команды push/pop работают со Стекком данных. Стек данных растет сверху вниз, его максимальная длина – 8 машинных слов (16 бит). В общем случае, команда push помещает значение в стек, а команда pop его от туда извлекает. SP – указатель на вершину стека. В зависимости от типа помещаемых и извлекаемых из стека данных, команды push/pop могут быть:

### PUSH Xn

Операция	SP = SP - 1, [SP] = Xn	
КОП	1000 0000 nnnn nnnn	2 байта
Операнды	0 ≤ n ≤ 255 - номер входа	
Описание	Помещает n-й вход в стек	
Пример	PUSH X0 ; Поместить вход 0 в стек	

### PUSHR Xn

Операция	SP = SP - 1, [SP] = Rn	
КОП	1000 0001 nnnn nnnn	2 байта
Операнды	0 ≤ n ≤ 255 - номер входа	
Описание	Помещает состояние фронта n-го входа в стек	
Пример	PUSHR X0 ; Поместить положительный фронт по входу 0 в стек	

### PUSHF Xn

Операция	SP = SP - 1, [SP] = Fn	
КОП	1000 0010 nnnn nnnn	2 байта
Операнды	0 ≤ n ≤ 255 - номер входа	
Описание	Помещает состояние спада n-го входа в стек	
Пример	PUSHF X0 ; Поместить отрицательный фронт по входу 0 в стек	

### PUSH Yn

Операция	SP = SP - 1, [SP] = Yn	
КОП	1000 1101 nnnn nnnn	2 байта
Операнды	0 ≤ n ≤ 255 - номер выхода	
Описание	Помещает n-й выход в стек	
Пример	PUSH Y0 ; Поместить выход 0 в стек	

### PUSHR Yn

Операция	SP = SP - 1, [SP] = Yn	
КОП	1000 1110 nnnn nnnn	2 байта
Операнды	0 ≤ n ≤ 255 - номер выхода	
Описание	Помещает состояние фронта n-го выхода в стек	
Пример	PUSHR Y0 ; Поместить положительный фронт по выходу 0 в стек	

### PUSHF Yn

Операция	SP = SP - 1, [SP] = Yn	
КОП	1000 1111 nnnn nnnn	2 байта
Операнды	0 ≤ n ≤ 255 - номер выхода	
Описание	Помещает состояние спада n-го выхода в стек	
Пример	PUSHF Y0 ; Поместить спад по выходу 0 в стек	

### POP Yn

Операция	Yn = [SP], SP = SP + 1	
КОП	1101 0000 nnnn nnnn	2 байта
Операнды	0 ≤ n ≤ 255 - номер выхода	
Описание	Помещает в n-й выход младший бит значения в стеке	
Пример	POP Y0 ; Отправить на выход 0 значение из стека	

### PUSH Mn

Операция	SP = SP - 1, [SP] = Mn	
КОП	1001 00nn nnnn nnnn	2 байта
Операнды	0 ≤ n ≤ 1023 - номер маркера	
Описание	Помещает n-й маркер в стек	
Пример	PUSH M0 ; Поместить маркер 0 в стек	

**PUSHR Mn**

Операция	SP = SP - 1, [SP] = Mn	
КОП	1001 01nn nnnn nnnn	2 байта
Операнды	0 ≤ n ≤ 1023 - номер маркера	
Описание	Помещает состояние фронта n-го маркера в стек	
Пример	PUSHR M0 ; Поместить фронт маркера 0 в стек	

**PUSHF Mn**

Операция	SP = SP - 1, [SP] = Mn	
КОП	1001 10nn nnnn nnnn	2 байта
Операнды	0 ≤ n ≤ 1023 - номер маркера	
Описание	Помещает состояние спада n-го маркера в стек	
Пример	PUSHF M0 ; Поместить спад маркера 0 в стек	

**POP Mn**

Операция	Mn = [SP], SP = SP + 1	
КОП	1100 00nn nnnn nnnn	2 байта
Операнды	0 ≤ n ≤ 1023 - номер бита	
Описание	Помещает в n-й маркер младший бит значения в стеке	
Пример	POP M0 ; Записать в маркер 0 значение из стека	

**PUSH Wn**

Операция	SP = SP - 1, [SP] = Wn	
КОП	1000 0011 000n nnnn	2 байта
Операнды	0 ≤ n ≤ 31 - номер слова	
Описание	Помещает n-е слово памяти в стек	
Пример	PUSH W0 ; Поместить слово 0 памяти данных в стек	

**POP Wn**

Операция	Wn = [SP], SP = SP + 1	
КОП	1101 0011 000n nnnn	2 байта
Операнды	0 ≤ n ≤ 31 - номер слова	
Описание	Помещает в n-е слово памяти значение из стека	
Пример	POP W0 ; Извлечь в слово 0 значение из стека	

**PUSH Bn**

Операция	SP = SP - 1, [SP] = Bn	
КОП	1000 0100 00nn nnnn	2 байта
Операнды	0 ≤ n ≤ 63 - номер байта	
Описание	Помещает n-й байт памяти в стек	
Пример	PUSH B0 ; Поместить байт 0 памяти данных в стек	

**POP Bn**

Операция	BYTEn = [SP], SP = SP + 1	
КОП	1101 0100 00nn nnnn	2 байта
Операнды	0 ≤ n ≤ 63 - номер байта	
Описание	Помещает в n-й байт памяти младший байт значения в стеке	
Пример	POP B0 ; Извлечь в байт 0 значение из стека	

**PUSH Kn**

Операция	SP = SP - 1, [SP] = Kn	
КОП	1000 0101 nnnn nnnn	2 байта
Операнды	0 ≤ n ≤ 255 - номер константы	
Описание	Помещает n-ю константу в стек	
Пример	PUSH K0 ; Поместить константу 0 в стек	

**POP Kn**

Операция	Kn = [SP], SP = SP + 1	
КОП	1101 0101 nnnn nnnn	2 байта
Операнды	0 ≤ n ≤ 255 - номер константы	
Описание	Помещает в n-ю константу значение из стека для хранения	
Пример	POP K0 ; Переписать константу 0 значением из стека	

**PUSH n**

Операция	SP = SP - 1, [SP] = n	
КОП	1000 1100 nnnn nnnn nnnn nnnn	3 байта
Операнды	$0 \leq n \leq 65535$	
Описание	Помещает число n в стек	
Пример	PUSH 10 ; Поместить в стек число 10	

Существует несколько команд, позволяющих модифицировать значение вершины стека без участия внешних данных:

**DEL**

Операция	SP = SP + 1	
КОП	0000 1001	1 байт
Операнды	нет	
Описание	Удаляет значение из стека	
Пример	DEL ;	

**DUP**

Операция	SP = SP - 1, [SP] = [SP+1]	
КОП	0000 1010	1 байт
Операнды	Нет	
Описание	Копирует вершину стека	
Пример	DUP ;	

Помимо команд, работающих со стеком, имеется ряд команд, позволяющих напрямую модифицировать данные источника:

**SET Yn**

Операция	Если [SP] = 1, то Yn = 1	
КОП	1000 1110 nnnn nnnn	2 байта
Операнды	$0 \leq n \leq 255$ – номер выхода	
Описание	Устанавливает n-й выход в 1	
Пример	SET Y0 ; Установить выход 0	

**RST Yn**

Операция	Если [SP] = 1, то Yn = 0	
КОП	1000 1111 nnnn nnnn	2 байта
Операнды	$0 \leq n \leq 255$ – номер выхода	
Описание	Сбрасывает n-й выход в 0	
Пример	RST Y0 ; Сбросить выход 0	

**SET Mn**

Операция	Если [SP] = 1, то Mn = 1	
КОП	1101 10nn nnnn nnnn	2 байта
Операнды	$0 \leq n \leq 1023$	
Описание	Устанавливает маркер n в 1	
Пример	SET M10 ; Установить маркер 10 в 1	

**RST Mn**

Операция	Если [SP] = 1, то Mn = 0	
КОП	1101 11nn nnnn nnnn	2 байта
Операнды	$0 \leq n \leq 1023$	
Описание	Сбрасывает маркер n в 0	
Пример	RST M10 ; Сбросить маркер 10 в 0	

**CLR Wn**

Операция	Если [SP] = 1, то Wn = 0	
КОП	1110 0011 000n nnnn	2 байта
Операнды	$0 \leq n \leq 31$ - номер слова	
Описание	Обнулить n-е слово	
Пример	CLR W10 ; Обнулить 10-е слово	

**CLR Bn**

Операция	Если [SP] = 1, то Bn = 0	
КОП	1110 0100 00nn nnnn	2 байта
Операнды	$0 \leq n \leq 63$ – номер байта	
Описание	Обнулить n-й байт	
Пример	CLR B10 ; Обнулить 10-й байт	

## Логические операции

Логические операции производятся в стеке над 16-битными словами.

### NOT

Операция	[SP] = ~[SP]	
КОП	0000 1000	1 байт
Операнды	нет	
Описание	Инвертирует значение в стеке	
Пример	PUSH X0 ; Инвертировать значение 0-го входа NOT ;	

### AND

Операция	[SP+1] = [SP+1] AND [SP] SP = SP + 1	
КОП	0000 0010	1 байт
Операнды	нет	
Описание	Выполняет операцию логическое «И» над значениями в стеке.	
Пример	PUSH X0 ; Логическое «И» 0-го входа и 0-го маркера PUSH M0 ; AND ;	

### OR

Операция	[SP+1] = [SP+1] OR [SP] SP = SP + 1	
КОП	0000 0100	1 байт
Операнды	нет	
Описание	Выполняет операцию логическое «ИЛИ» над значениями в стеке	
Пример	PUSH X0 ; Логическое «ИЛИ» 0-го входа и 0-го маркера PUSH M0 ; OR ;	

### XOR

Операция	[SP+1] = [SP+1] XOR [SP] SP = SP + 1	
КОП	0000 0110	1 байт
Операнды	нет	
Описание	Выполняет операцию «Исключающее ИЛИ» над значениями в стеке	
Пример	PUSH X0 ; «Исключающее ИЛИ» 0-го входа и 0-го бита PUSH M0 ; памяти данных XOR ;	

К логическим операциям так же отнесены команды двоичного сдвига:

### SHR n

Операция	[SP] = [SP] >> n	
КОП	0011 nnnn	1 байт
Операнды	0 ≤ n ≤ 15 – величина сдвига	
Описание	Сдвигает вправо значение на вершине стека.	
Пример	PUSH W0 ; Сдвинуть 0-е слово SHR 8 ; на 8 бит POP W0 ; вправо	

### SHL n

Операция	[SP] = [SP] << n	
КОП	0100 nnnn	1 байт
Операнды	0 ≤ n ≤ 15 – величина сдвига	
Описание	Сдвигает влево значение на вершине стека.	
Пример	PUSH X0 ; Поместить в первый бит вершины стека PUSH X1 ; вход 0, а во второй бит – вход 1 SHL 1 ; OR ;	



## Арифметические операции

Отсутствуют в модификации ЛИР-986-03.

Арифметические операции производятся в стеке над 16-битными целыми числами. Числа могут иметь или не иметь знак, в зависимости от этого диапазон допустимых значений будет:

- Для беззнаковых целых чисел: от 0 до 65535
- Для целых чисел со знаком: от -32768 до 32767

Наличие знака у операндов определяется кодом операции. При этом команды, работающие со знаковыми операндами, имеют префикс S, а команды, работающие с беззнаковыми операндами – префикс U. Такие команды, как сложение и вычитание, для которых в силу специфики двоичного представления чисел, информация о наличии знака не является важной, префиксов не имеют.

### ADD

Операция	$[SP+1] = [SP+1] + [SP]$ $SP = SP + 1$	
КОП	0000 0000	1 байт
Операнды	нет	
Описание	Складывает два числа из стека.	
Пример	PUSH 10 ; PUSH T0 ; ADD ; Прибавить 10 к значению таймера/счетчика 0 POP T0 ;	

### SUB

Операция	$[SP+1] = [SP+1] - [SP]$ $SP = SP + 1$	
КОП	0000 0001	1 байт
Операнды	Нет	
Описание	Вычитает из одного числа в стеке другое.	
Пример	PUSH 10 ; PUSH T0 ; SUB ; Вычесть из 10 значение таймера/счетчика 0 POP T0 ;	

### UDIV

Операция	$[SP+1] = [SP+1] / [SP]$ $SP = SP + 1$	
КОП	0000 0010	1 байт
Операнды	Нет	
Описание	Делит одно беззнаковое целое число на другое.	
Пример	PUSH T0 ; PUSH 10 ; UDIV ; Разделить на 10 значение таймера/счетчика 0 POP T0 ;	

### SDIV

Операция	$[SP+1] = [SP+1] / [SP]$ $SP = SP + 1$	
КОП	0000 0011	1 байт
Операнды	нет	
Описание	Делит одно целое число со знаком на другое.	
Пример	PUSH B0 ; PUSH -10 ; SDIV ; Разделить на -10 значение байта 0 POP B ;	

**UREM**

Операция	[SP+1] = [SP+1] % [SP] SP = SP + 1	
КОП	0000 0100	1 байт
Операнды	Нет	
Описание	Находит остаток от деления одного целого числа без знака на другое.	
Пример	PUSH T0 ; PUSH 10 ; Найти остаток от деления UREM ; значения таймера/счетчика 0 на число 10 POP T0 ;	

**SREM**

Операция	[SP+1] = [SP+1] % [SP] SP = SP + 1	
КОП	0000 0101	1 байт
Операнды	нет	
Описание	Находит остаток от деления одного целого числа со знаком на другое.	
Пример	PUSH B0 ; PUSH -10 ; Найти остаток от деления SREM ; значения байта 0 на число 10 POP B0 ;	

**UMUL**

Операция	[SP+1] = [SP+1] * [SP] SP = SP + 1	
КОП	0000 0110	1 байт
Операнды	нет	
Описание	Перемножает два беззнаковых целых числа из стека.	
Пример	PUSH 10 ; PUSH T0 ; UMUL ; Умножить на 10 значение таймера/счетчика 0 POP T0 ;	

**SMUL**

Операция	[SP+1] = [SP+1] * [SP] SP = SP + 1	
КОП	0000 0111	1 байт
Операнды	нет	
Описание	Перемножает два целых числа со знаком из стека.	
Пример	PUSH -10 ; PUSH B0 ; SMUL ; Умножить на -10 значение байта 0 POP B0 ;	

**INC**

Операция	[SP] = [SP] + 1	
КОП	0000 1000	1 байт
Операнды	нет	
Описание	Увеличивает на 1 число на вершине стека	
Пример	PUSH T0 ; INC ; Прибавить 1 к значению таймера/счетчика 0 POP T0 ;	

**DEC**

Операция	[SP] = [SP] - 1	
КОП	0000 1001	1 байт
Операнды	Нет	
Описание	Уменьшает на 1 число на вершине стека	
Пример	PUSH T0 ; DEC ; Вычесть 1 из значения таймера/счетчика 0 POP T0 ;	

## Операции сравнения

Отсутствуют в модификации ЛИР-986-03.

Операции сравнения производятся над 16-битными целыми числами из стека. Аналогично арифметическим операциям, наличие знака у операндов определяется префиксом: префикс S означает наличие знака у операндов, а префикс U – его отсутствие.

### UGR

Операция	Если $[SP+1] > [SP]$ , то $[SP+1] = 1$ . Иначе $[SP+1] = 0$ . $SP = SP + 1$	
КОП	0010 0000	1 байт
Операнды	нет	
Описание	Сравнивает два беззнаковых целых числа из стека.	
Пример	PUSH T0 ; PUSH 100 ; UGR ; Сравнить значение таймера 0 с числом 100	

### SGR

Операция	Если $[SP+1] > [SP]$ , то $[SP+1] = 1$ . Иначе $[SP+1] = 0$ . $SP = SP + 1$	
КОП	0010 0001	1 байт
Операнды	нет	
Описание	Сравнивает два целых числа со знаком из стека.	
Пример	PUSH -10 ; PUSH B0 ; SGR ; Сравнить значение байта 0 с числом -10	

### ULS

Операция	Если $[SP+1] < [SP]$ , то $[SP+1] = 1$ . Иначе $[SP+1] = 0$ . $SP = SP + 1$	
КОП	0010 0010	1 байт
Операнды	нет	
Описание	Сравнивает два беззнаковых целых числа из стека.	
Пример	PUSH T0 ; PUSH 100 ; ULS ; Сравнить значение таймера 0 с числом 100	

### SLS

Операция	Если $[SP+1] < [SP]$ , то $[SP+1] = 1$ . Иначе $[SP+1] = 0$ . $SP = SP + 1$	
КОП	0010 0011	1 байт
Операнды	нет	
Описание	Сравнивает два целых числа со знаком из стека.	
Пример	PUSH -10 ; PUSH B0 ; SLS ; Сравнить значение байта 0 с числом -10	

### UGRE

Операция	Если $[SP+1] \geq [SP]$ , то $[SP+1] = 1$ . Иначе $[SP+1] = 0$ . $SP = SP + 1$	
КОП	0010 0100	1 байт
Операнды	нет	
Описание	Сравнивает два беззнаковых целых числа из стека.	
Пример	PUSH T0 ; PUSH 100 ; UGRE ; Сравнить значение таймера 0 с числом 100	

**SGRE**

Операция	Если $[SP+1] \geq [SP]$ , то $[SP+1] = 1$ . Иначе $[SP+1] = 0$ . $SP = SP + 1$	
КОП	0010 0101	1 байт
Операнды	нет	
Описание	Сравнивает два целых числа со знаком из стека.	
Пример	PUSH -10 ; PUSH B0 ; SGRE ; Сравнить значение байта 0 с числом -10	

**ULSE**

Операция	Если $[SP+1] \leq [SP]$ , то $[SP+1] = 1$ . Иначе $[SP+1] = 0$ . $SP = SP + 1$	
КОП	0010 0110	1 байт
Операнды	нет	
Описание	Сравнивает два беззнаковых целых числа из стека.	
Пример	PUSH T0 ; PUSH 100 ; ULSE ; Сравнить значение таймера 0 с числом 100	

**SLSE**

Операция	Если $[SP+1] \leq [SP]$ , то $[SP+1] = 1$ . Иначе $[SP+1] = 0$ . $SP = SP + 1$	
КОП	0010 0111	1 байт
Операнды	нет	
Описание	Сравнивает два целых числа со знаком из стека.	
Пример	PUSH -10 ; PUSH B0 ; SLSE ; Сравнить значение байта 0 с числом -10	

**EQ**

Операция	Если $[SP+1] = [SP]$ , то $[SP+1] = 1$ . Иначе $[SP+1] = 0$ . $SP = SP + 1$	
КОП	0010 1000	1 байт
Операнды	нет	
Описание	Сравнивает два беззнаковых целых числа из стека.	
Пример	PUSH T0 ; PUSH 100 ; EQ ; Сравнить значение таймера 0 с числом 100	

**NEQ**

Операция	Если $[SP+1] \neq [SP]$ , то $[SP+1] = 1$ . Иначе $[SP+1] = 0$ . $SP = SP + 1$	
КОП	0010 1001	1 байт
Операнды	Нет	
Описание	Сравнивает два беззнаковых целых числа из стека.	
Пример	PUSH T0 ; PUSH 100 ; NEQ ; Сравнить значение таймера 0 с числом 100	

**GRZ**

Операция	Если $[SP] > 0$ , то $[SP] = 1$ . Иначе $[SP] = 0$ .	
КОП	0010 1101	1 байт
Операнды	нет	
Описание	Сравнивает с нулем целое число со знаком из стека.	
Пример	PUSH T0 ; GRZ ; Сравнить с 0 значение таймера 0	

**EQZ**

Операция	Если $[SP] = 0$ , то $[SP] = 1$ . Иначе $[SP] = 0$ .	
КОП	0010 1110	1 байт
Операнды	Нет	
Описание	Сравнивает с нулем целое число со знаком из стека.	
Пример	PUSH T0 ; EQZ ; Сравнить с 0 значение таймера 0	

**LSZ**

Операция	Если [SP] < 0, то [SP] = 1. Иначе [SP] = 0.	
КОП	0010 1111	1 байт
Операнды	нет	
Описание	Сравнивает с нулем целое число со знаком из стека.	
Пример	PUSH T0 ; LSZ ; Сравнить с 0 значение таймера 0	

**Пример: «Проверка нескольких условий»**

Исходная конструкция - (X0 == 0) && (T0 >= 100)

PUSH X0  
 EQZ  
 PUSH T0  
 PUSH 100  
 UGRE  
 AND

Пусть X0 = 0, а T0 = 99 (63h), тогда:

Команда	Состояние стека
PUSH X0	SP -> 7: 0000h 6: - 5: -
EQZ	SP -> 7: 0001h 6: - 5: -
PUSH T0	7: 0001h SP -> 6: 0063h 5: -
PUSH 100	7: 0001h 6: 0063h SP -> 5: 0064h
UGRE	7: 0001h SP -> 6: 0000h 5: -
AND	SP -> 7: 0000h

## Команды работы с таймерами/счетчиками

### TMR n

Операция	SP = SP - 1, TCONn = 0, TCn = [SP].	
КОП	1101 1001 nnnn nnnn	1 слово (2 байта)
Операнды	0 ≤ n ≤ 255 – номер таймера/счетчика	
Описание	Конфигурирует таймер/счетчик n в режиме таймера и записывает в его регистр сравнения слово из стека.	
Пример	PUSH 100 ; Сконфигурировать таймер/счетчик 0 в режиме TMR 0 ; таймера на 1 с.	

### CNT n

Операция	SP = SP - 1, TCONn = 1, TCn = [SP].	
КОП	1101 1010 nnnn nnnn	1 слово (2 байта)
Операнды	0 ≤ n ≤ 255 – номер таймера/счетчика	
Описание	Конфигурирует таймер/счетчик n в режиме счетчика и записывает в его регистр сравнения слово из стека.	
Пример	PUSH 100 ; Сконфигурировать таймер/счетчик 0 в режиме CNT 0 ; счетчика импульсов по входу.	

### PUSH Tn

Операция	SP = SP - 1, [SP] = Tn	
КОП	1000 0110 nnnn nnnn	1 слово (2 байта)
Операнды	0 ≤ n ≤ 255 - номер таймера	
Описание	Помещает значение n-го таймера в стек	
Пример	PUSH T0 ; Поместить значение таймера 0 в стек	

### POP TXn

Операция	TXn = [SP], SP = SP + 1	
КОП	1101 1000 nnnn nnnn	1 слово (2 байта)
Операнды	0 ≤ n ≤ 255 - номер таймера	
Описание	Отправляет значение в стеке на вход таймера/счетчика n	
Пример	PUSH X10 ; Отправить сигнал на входе 10 POP TX0 ; на вход счета таймера 0	

### PUSH TYn

Операция	SP = SP - 1, [SP] = TYn	
КОП	1000 1000 nnnn nnnn	1 слово (2 байта)
Операнды	0 ≤ n ≤ 255 – номер таймера	
Описание	Помещает значение на выходе таймера n в стек	
Пример	PUSH TY0 ; Поместить флаг таймера 0 в стек	

### PUSHR TYn

Операция	SP = SP - 1, [SP] = ↑TYn	
КОП	1000 1001 nnnn nnnn	1 слово (2 байта)
Операнды	0 ≤ n ≤ 255 – номер таймера	
Описание	Помещает фронт сигнала на выходе таймера n в стек	
Пример	PUSHR TY0 ; Поместить фронт выхода таймера 0 в стек	

### PUSHF TYn

Операция	SP = SP - 1, [SP] = ↓TYn	
КОП	1000 1010 nnnn nnnn	1 слово (2 байта)
Операнды	0 ≤ n ≤ 255 – номер таймера	
Описание	Помещает спад сигнала на выходе таймера n в стек	
Пример	PUSHF TY0 ; Поместить спад выхода таймера 0 в стек	

### RST Tn

Операция	Если [SP] = 1, то Tn = 0.	
КОП	1101 0110 nnnn nnnn	1 слово (2 байта)
Операнды	0 ≤ n ≤ 255 - номер таймера	
Описание	Сбрасывает значение таймера/счетчика n, если в стеке 1.	
Пример	PUSH X1 ; Сбросить таймер/счетчик 0 RST T0 ; если на входе 1 установилась 1.	

Каждый таймер/счетчик имеет собственный вход и выход.

В режиме таймера вход таймера/счетчика управляет его работой. В этом режиме при наличии высокого уровня на входе таймера/счетчика значение таймера/счетчика инкрементируется каждые 10 мс. При достижении таймером/счетчиком значения, записанного в него командой TMR, на выходе таймера/счетчика устанавливается высокий уровень сигнала. При наличии низкого уровня сигнала на входе таймера/счетчика в режиме таймера, таймер/счетчик сброшен.

Если таймер/счетчик сконфигурирован командой CNT в режиме счетчика, он осуществляет счет импульсов по своему входу. Сброс таймера/счетчика в этом режиме производится командой RST.

### **Пример: «Конфигурирование таймера/счетчика 0»**

Таймер/счетчик 0 конфигурируется в режиме счетчика. Он считает 100 импульсов по входу 0, после чего на выходе 0 формируется одиночный положительный импульс. Код инициализации таймера/счетчика следующий:

PUSH	X0	; Записать сигнал с входа 0
POP	TX0	; на вход таймера/счетчика 0
PUSH	100	; Сконфигурировать таймер/счетчик 0
CNT	0	; как счетчик 100 импульсов
PUSHR	TY0	; При срабатывании таймера/счетчика 0
POP	Y0	; выдать одиночный импульс на выход 0

## Команды индексирования

Отсутствуют в модификации ЛИР-986-03.

Команды индексирования предназначены для динамической модификации адреса приемника/источника в последующей команде. В качестве источника индексного значения используются 16-битные слова (w0-w63). Индексное значение прибавляется к адресу операнда первой после команды индексирования команды, имеющей операнд. В случае, если следующая за командой индексирования команда не имеет операндов, индексное значение игнорируется, но не обнуляется.

Допустимо так же индексирование индексных команд, в этом случае новое значение индекса прибавляется к уже имеющемуся значению индекса.

В настоящий момент имеется всего две команды индексирования:

### IND Wn

Операция	[SP] = [SP] + 1	
КОП	1101 1100 000n nnnn	2 байт
Операнды	$0 \leq n \leq 31$ - номер слова	
Описание	Устанавливает индекс равным значению в Wn	
Пример	IND W0 ; Поместить в стек состояние входа, PUSH X0 ; номер которого содержится в W0	

### INDI Wn

Операция	[SP] = [SP] + 1	
КОП	1101 1101 000n nnnn	2 байт
Операнды	$0 \leq n \leq 31$ - номер слова	
Описание	Устанавливает индекс равным значению в Wn, после чего инкрементирует Wn	
Пример	PUSH X0 ; Сохранить состояние входа X0 INDI W0 ; в маркере, номер которого содержится в W0, POP M0 ; после чего увеличить этот номер на 1.	

Пример использования команд индексирования для задач блочного копирования данных, приведен ниже в описании команды DJNZ.



## Команды перехода

Команды перехода позволяют изменять произвольно номер следующей выполняемой команды (PC). Это позволяет создавать циклы, ветвление и вызывать подпрограммы.

### JMP n

Операция	PC = n	
КОП	1111 0000 nnnn nnnn nnnn nnnn	3 байта
Операнды	$0 \leq n \leq 65535$ – адрес перехода	
Описание	Осуществляет безусловный переход по адресу n	
Пример	JMP 1234 ; Перейти по адресу 1234	

### JZ n

Операция	Если [SP] = 0, то PC = n, иначе PC = PC + 3. SP = SP + 1	
КОП	1111 0001 nnnn nnnn nnnn nnnn	3 байта
Операнды	$0 \leq n \leq 65535$ – адрес перехода	
Описание	Если значение в стеке равно 0, то осуществляет переход по адресу n.	
Пример	JZ 1234 ; Если в стеке 0, перейти по адресу 1234	

### JNZ n

Операция	Если [SP] ≠ 0, то PC = n, иначе PC = PC + 3. SP = SP + 1	
КОП	1111 0010 nnnn nnnn nnnn nnnn	3 байта
Операнды	$0 \leq n \leq 65535$ – адрес перехода	
Описание	Если значение в стеке не равно 0, то осуществляет переход по адресу n.	
Пример	JNZ 1234 ; Если в стеке не 0, перейти по адресу 1234	

### CALL n

Операция	PSP = PSP - 1, [PSP] = PC + 3, PC = n	
КОП	1111 0011 nnnn nnnn nnnn nnnn	3 байта
Операнды	$0 \leq n \leq 65535$ - адрес перехода	
Описание	Вход в подпрограмму расположенную по адресу указанному в операнде. Адрес возврата сохраняется в Стекке возврата.	
Пример	CALL 1234 ; Вызвать подпрограмму по адресу 1234	

### RET

Операция	PC = [PSP], PSP = PSP + 1	
КОП	0111 0111	1 байт
Операнды	Нет операндов	
Описание	Возврат из подпрограммы. Адрес возврата извлекается из Стекка возврата. Если стек возврата пуст, выполняется выход из цикла выполнения программы.	
Пример	RET ; Вернуться из подпрограммы	

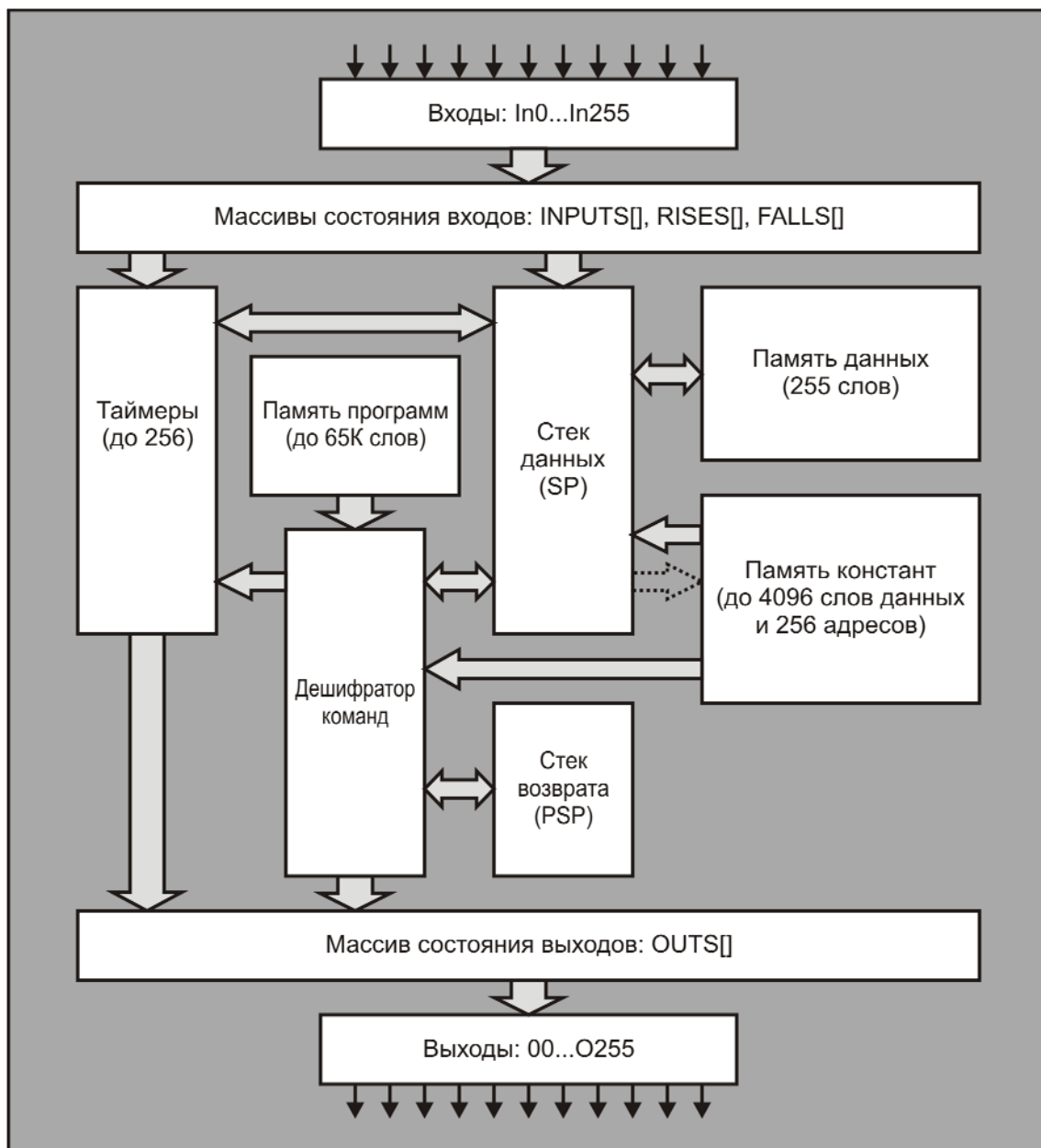
### DJNZ n

Операция	[SP] = [SP] - 1. Если [SP] ≠ 0, то PC = n, иначе PC = PC + 3 и SP = SP + 1	
КОП	1111 1101 nnnn nnnn nnnn nnnn	3 байта
Операнды	$0 \leq n \leq 65535$ – адрес перехода	
Описание	Осуществляет декремент вершины стека, если полученное значение не равно 0, то происходит переход по адресу n (значение из стека <i>не извлекается</i> ), в противном случае происходит переход к выполнению следующей команды (значение из стека <i>извлекается</i> ).	
Пример	<pre> clr w0      ; Передать состояния первых 10 push 10     ; входов (x0-x9) на первые 10 lbl: ind w0  ; выходов (y0-y9). push x0     ; В качестве индексного слова indi w0     ; используется w0. pop y0      ; djnz lbl    ; </pre>	

## Карта команд виртуальной машины

	0h 0000b	1h 0001b	2h 0010b	3h 0011b	4h 0100b	5h 0101b	6h 0110b	7h 0111b	8h 1000b	9h 1001b	Ah 1010b	Bh 1011b	Ch 1100b	Dh 1101b	Eh 1110b	Fh 1111b
0h 0000b	NOP		AND		OR		XOR		NOT	DEL	DUP					
1h 0001b	ADD	SUB	UDIV	SDIV	UREM	SREM	UMUL	SMUL	INC	DEC						
2h 0010b	UGR	SGR	ULS	SLS	UGRE	SGRE	ULSE	SLSE	EQ	NEQ				GRZ	EQZ	LSZ
3h 0011b	SHL n															
4h 0100b	SHR n															
5h 0101b																
6h 0110b																
7h 0111b								RET								
8h 1000b	PUSH Xn	PUSHR Xn	PUSHF Xn	PUSH Wn	PUSH Bn	PUSH Kn	PUSH Tn		PUSH TYn	PUSHR TYn	PUSHF TYn		PUSH n	PUSH Yn	PUSHR Yn	PUSHF Yn
9h 1001b	PUSH Mn				PUSHR Mn				PUSHF Mn							
Ah 1010b																
Bh 1011b																
Ch 1100b	POP Mn								SET Mn				RST Mn			
Dh 1101b	POP Yn			POP Wn	POP Bn	POP Kn	RST Tn		POP TXn	TMR n	CNT n		IND Wn	INDI Wn	SET Yn	RST Yn
Eh 1110b				CLR Wn	CLR Bn											
Fh 1111b	JMP n	JZ n	JNZ n	CALL An										DJNZ n		

## Архитектура



Виртуальная машина имеет стековую архитектуру, то есть все операции над данными производятся через Стек данных. Длина Стекa данных – 8 машинных слов.

Стек возврата используется командой CALL для временного сохранения адреса возврата из подпрограммы. Длина Стекa возврата – 8 машинных слов. Выход из подпрограммы и извлечение адреса возврата из Стекa возврата осуществляется по команде RET.

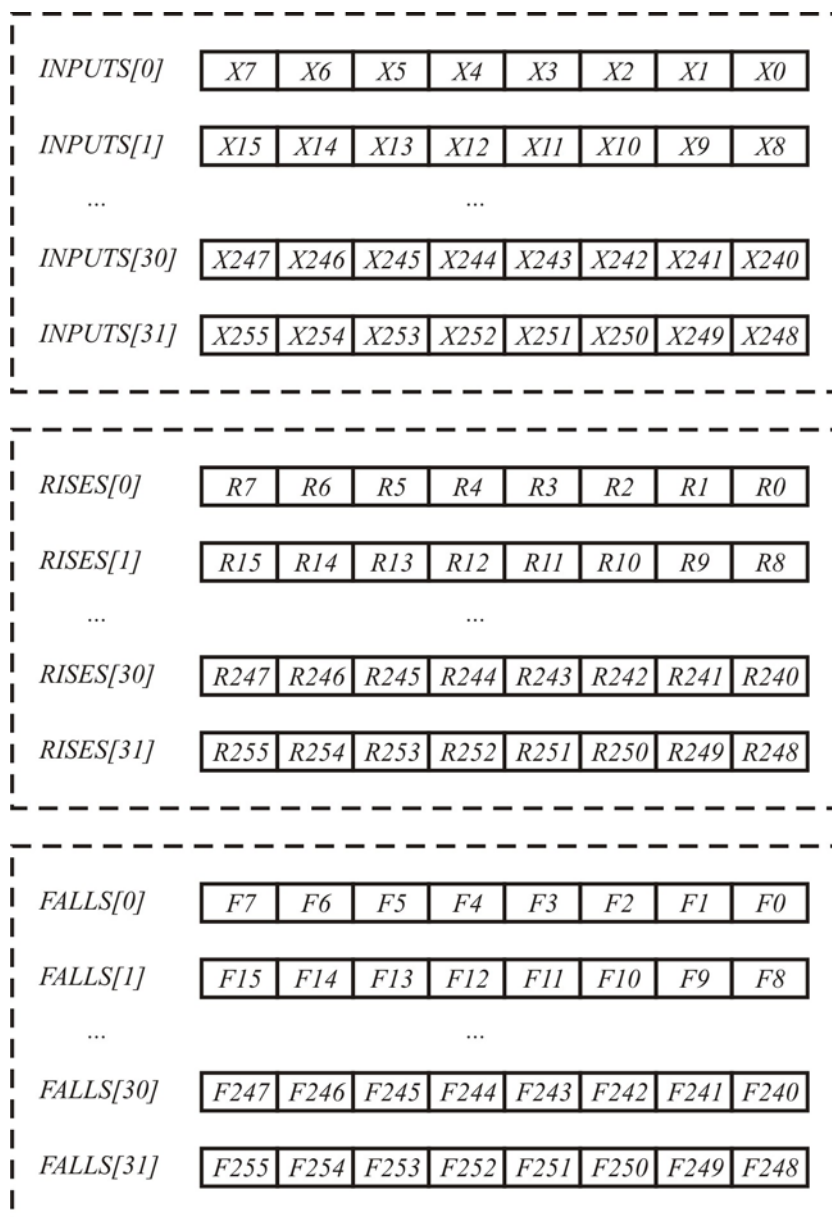
### Цикл работы виртуальной машины следующий:

1. Чтение входов. Формирование массивов INPUTS[], RISES[] и FALLS[].
2. Обновление состояний таймеров.
3. Выполнение программного кода.
4. Запись массива OUTS[] на выход.

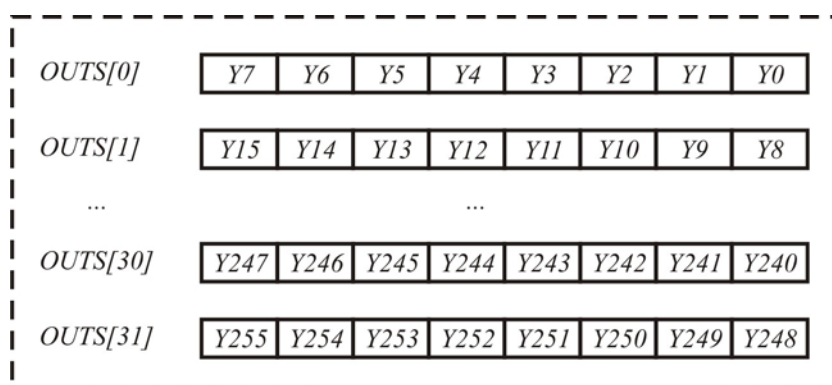
Конкретные значения объема памяти констант и памяти программ, а так же количество входов и выходов зависит от решаемых задач, и может быть меньше указанных значений.

## Массивы состояния входов и выходов

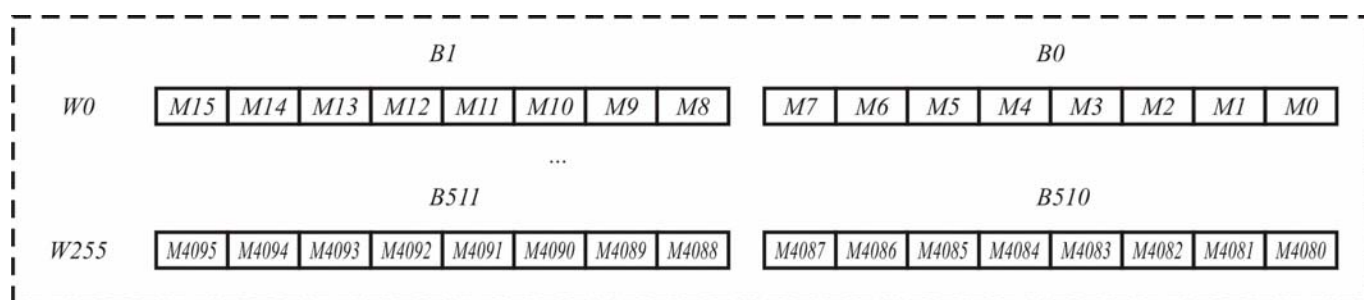
Массив INPUTS[n] содержит биты текущего состояния каждого из входов. Массив RISES[n] содержит биты наличия положительных фронтов сигнала на каждом из входов. Массив FALLS[n] содержит биты наличия отрицательных фронтов сигнала на каждом из входов. Значения битов массивов состояний входов обновляются в начале каждого цикла перед выполнением программного кода.



Массив OUTS[n] содержит биты состояния выходов, которые будут записаны на выход в конце очередного цикла. Биты массивов состояний входов (INPUTS[], RISES[] и FALLS[]) доступны при помощи команды PUSH только для чтения. Биты массива состояния выходов (OUTS[]) доступны при помощи команды POP только для записи.



## Память данных

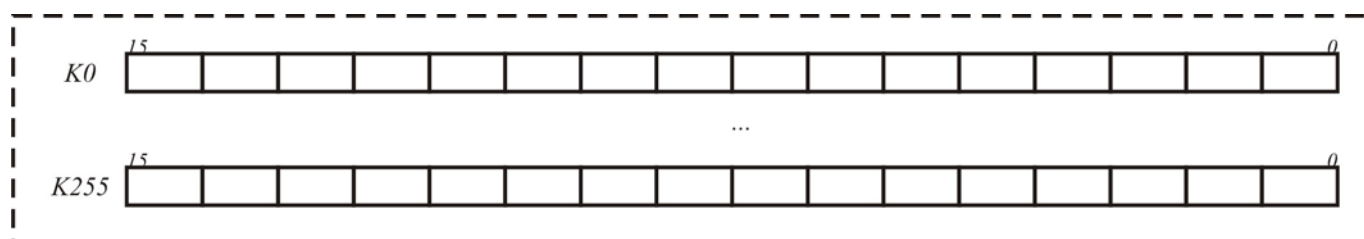


Память данных состоит из 256 16-битных слов. Кроме того, вся область памяти доступна побайтно и побитно. То есть имеется 512 адресуемых байт и 4096 адресуемых бит.

Память данных является энергозависимой и предназначена для временного хранения различных переменных (маркеров) в процессе работы.

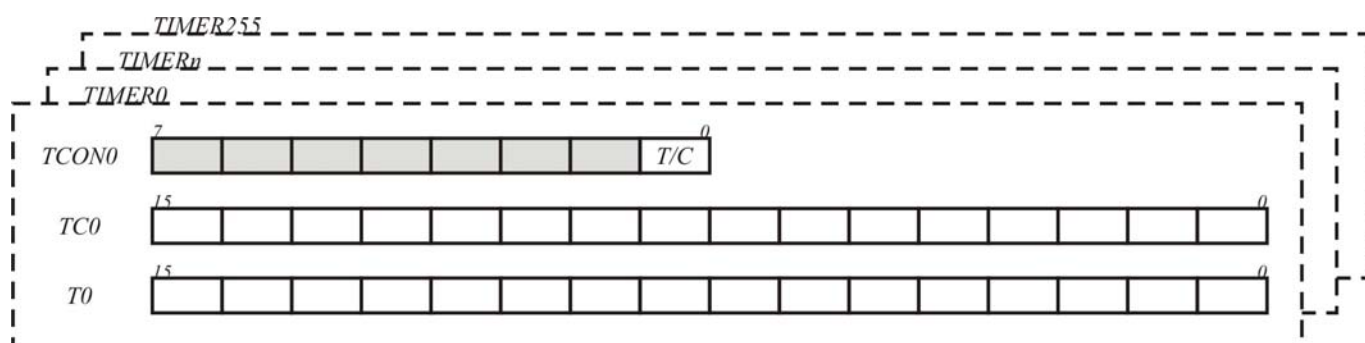
Доступ к памяти данных осуществляется при помощи соответствующих команд PUSH и POP.

## Память констант



Константы предназначены для хранения данных, которые не изменяются вообще или же изменяются редко. Это могут быть как какие-то конфигурационные данные, так и любые другие данные, которые необходимо сохранить до момента следующего включения контроллера.

## Массив таймеров/счетчиков



*TCOnn* содержит информацию о режиме работы таймера/счетчика *n*

*TCn* содержит значение сравнения (перезагрузки) таймера/счетчика *n*

*Tn* содержит текущее значение таймера/счетчика *n*

### Режимы работы таймера/счетчика:

- *Режим таймера* ( $T/C = 0$ ). При наличии на входе таймера  $TXn = 1$ , регистр  $Tn$  инкрементируется каждые 10 мс. При достижении  $Tn = TCn$  устанавливается выход  $TYn = 1$ . Выход удерживается до тех пор, пока на входе таймера присутствует 1. При  $TXn = 0$  таймер сбрасывается.
- *Режим счетчика* ( $T/C = 1$ ).  $Tn$  инкрементируется с приходом каждого нового импульса на вход счетчика  $TXn$ . При достижении  $Tn = TCn$  устанавливается выход  $TYn = 1$ . Выход удерживается до тех пор, пока счетчик не будет принудительно сброшен командой **RST**  $Tn$ .

## Сводная таблица команд виртуальной машины

Команда	Описание	Стр.
<i>Команды записи/чтения и модификации данных</i>		
CLR Bn	Обнулить n-й байт	7
CLR Wn	Обнулить n-е слово	6
DEL	Удаляет значение из стека	6
DUP	Копирует вершину стека	6
POP Bn	Помещает в n-й байт памяти младший байт значения в стеке	5
POP Kn	Помещает в n-ю константу значение из стека для хранения	5
POP Mn	Помещает в n-й маркер младший бит значения в стеке	5
POP Wn	Помещает в n-е слово памяти значение из стека	5
POP Yn	Помещает в n-й выход младший бит значения в стеке	4
PUSH Bn	Помещает n-й байт памяти в стек	5
PUSH Kn	Помещает n-ю константу в стек	5
PUSH Mn	Помещает n-й маркер в стек	4
PUSH n	Помещает число n в стек	6
PUSH Wn	Помещает n-е слово памяти в стек	5
PUSH Xn	Помещает n-й вход в стек	4
PUSH Yn	Помещает n-й выход в стек	4
PUSHF Mn	Помещает состояние спада n-го маркера в стек	5
PUSHF Xn	Помещает состояние спада n-го входа в стек	4
PUSHF Yn	Помещает состояние спада n-го выхода в стек	4
PUSHR Mn	Помещает состояние фронта n-го маркера в стек	5
PUSHR Xn	Помещает состояние фронта n-го входа в стек	4
PUSHR Yn	Помещает состояние фронта n-го выхода в стек	4
RST Mn	Сбрасывает маркер n в 0	6
RST Yn	Сбрасывает n-й выход в 0	6
SET Mn	Устанавливает маркер n в 1	6
SET Yn	Устанавливает n-й выход в 1	6
<i>Логические операции</i>		
AND	Логическое «И» над значениями в стеке	8
NOT	Инвертирует значение в стеке	8
OR	Логическое «ИЛИ» над значениями в стеке	8
SHL n	Сдвигает влево значение на вершине стека	8
SHR n	Сдвигает вправо значение на вершине стека	8
XOR	«Исключающее ИЛИ» над значениями в стеке	8
<i>Арифметические операции*</i>		
ADD	Складывает два числа из стека	9
DEC	Уменьшает на 1 число на вершине стека	10
INC	Увеличивает на 1 число на вершине стека	10
SDIV	Делит одно целое число со знаком на другое	9
SMUL	Перемножает два целых числа со знаком из стека	10
SREM	Остаток от деления одного целого числа со знаком на другое	10
SUB	Вычитает из одного числа в стеке другое	9
UDIV	Делит одно беззнаковое целое число на другое	9
UMUL	Перемножает два беззнаковых целых числа из стека	10
UREM	Остаток от деления одного целого числа без знака на другое	10
<i>Операции сравнения*</i>		
EQ	Проверка на равенство	12
EQZ	Равно 0	12
GRZ	Больше 0	12
LSZ	Меньше 0	12
NEQ	Проверка на неравенство	12
SGR	"Больше" со знаком	11
SGRE	"Больше или равно" со знаком	11

SLS	"Меньше" со знаком	11
SLSE	"Меньше или равно" со знаком	12
UGR	Беззнаковое "больше"	11
UGRE	Беззнаковое "больше или равно"	11
ULS	Беззнаковое "меньше"	11
ULSE	Беззнаковое "меньше или равно"	12
<i>Команды работы с таймерами/счетчиками</i>		
CNT n	Конфигурирует таймер/счетчик n в режиме счетчика	14
POP TXn	Отправляет значение в стеке на вход таймера/счетчика n	14
PUSH Tn	Помещает значение n-го таймера в стек	14
PUSH TYn	Помещает значение на выходе таймера n в стек	14
PUSHF TYn	Помещает спад сигнала на выходе таймера n в стек	14
PUSHR TYn	Помещает фронт сигнала на выходе таймера n в стек	14
RST Tn	Сбрасывает значение таймера/счетчика n	14
TMR n	Конфигурирует таймер/счетчик n в режиме таймера	14
<i>Команды индексирования</i>		
IND Wn	Устанавливает индекс	16
INDI Wn	Устанавливает индекс, после чего инкрементирует Wn	16
<i>Команды перехода</i>		
CALL n	Вход в подпрограмму расположенную по адресу n	17
DJNZ n	Декремент, затем переход в n, если [SP] ≠ 0	17
JMP n	Безусловный переход по адресу n	17
JNZ n	Осуществляет переход по адресу n, если [SP] ≠ 0	17
JZ n	Осуществляет переход по адресу n, если [SP] = 0	17
RET	Возврат из подпрограммы	17

\*Отсутствуют в модификации ЛИР-986-03.