

Министерство образования и науки России  
Санкт-Петербургский Государственный Технический Университет

УДК Н .....

Инв. Н ...

Рукопись одобрена кафедрой .....  
Зав. кафедрой Черноруцкий И.Г.

РУКОПИСНЫЙ ФОНД  
кафедры информационных и управляющих систем

Учебное пособие по курсу: "Системы программирования"

РАЗРАБОТКА ЭЛЕМЕНТОВ УЧЕБНОЙ СИСТЕМЫ ПРОГРАММИРОВАНИЯ.  
БАЗОВЫЕ КОНСТРУКЦИИ ЯЗЫКОВОГО ИНТЕРФЕЙСА МЕЖДУ  
ЭЛЕМЕНТАМИ СИСТЕМЫ  
(справочник)

Для студентов кафедры информационных и управляющих систем  
Семестр 10

Автор ..... Расторгуев В.Я.

С.-Петербург

2011 г.

Министерство образования и науки России  
Санкт-Петербургский Государственный Технический Университет  
Кафедра информационных и управляющих систем

Расторгуев В.Я.

РАЗРАБОТКА ЭЛЕМЕНТОВ УЧЕБНОЙ СИСТЕМЫ ПРОГРАММИРОВАНИЯ.  
БАЗОВЫЕ КОНСТРУКЦИИ ЯЗЫКОВОГО ИНТЕРФЕЙСА МЕЖДУ  
ЭЛЕМЕНТАМИ СИСТЕМЫ  
(справочник)

Учебное пособие

С.-Петербург

2011 г.

## АННОТАЦИЯ

Настоящее методическое пособие является справочником, содержащим минимальный набор сведений об языках ПЛ1, АССЕМБЛЕР и регистровой модели целевого процессора IBM 370. Эти сведения могут быть полезны студентам для понимания особенностей как личного задания на выполнение курсовой работы, так и контрольного примера, который является образцом выполнения личного задания.

## СОДЕРЖАНИЕ

1. Базовый набор конструкций языка ПЛ1 . . . . .	6
1.1. Операторы пролога и эпилога программы . . . . .	6
1.1.1 Оператор PROCEDURE . . . . .	6
1.1.2 Оператор END . . . . .	6
1.2. Декларативные конструкции . . . . .	7
1.2.1. Константы языка ПЛ1 . . . . .	7
1.2.2. Скалярные переменные языка ПЛ1 . . . . .	7
1.2.3. Агрегатированные переменные языка ПЛ1 . . . . .	8
1.2.4. Индексированные переменные языка ПЛ1 . . . . .	8
1.2.5. Совмещение переменных в ОЗУ . . . . .	8
1.3. Императивные операторы . . . . .	9
1.3.1. Оператор присваивания . . . . .	9
1.3.2. Оператор условного разветвления . . . . .	9
1.3.4. Оператор безусловной передачи управления . . . . .	9
1.3.5. Оператор цикла . . . . .	9
1.3.6. Встроенная функция SUBSTR . . . . .	10
1.4. Правила преобразования данных назначением . . . . .	10
1.4.1. Преобразование символьной строки в битовую строку	11
1.4.2. Преобразование битовой строки в символьную . . .	11
1.4.3. Преобразование битовой строки в двоичное целое с Фиксированной точкой . . . . .	12
1.4.4. Преобразование двоичного целого с фиксированной точкой в битовую строку. . . . .	12
1.4.5. Технологическая цепочка элементарных преобразований при переходе от строковых данных к арифметическим . . .	12
1.4.6. Технологическая цепочка элементарных преобразований при переходе от арифметических данных к строковым . . .	13

2. Базовый набор конструкций языка АССЕМБЛЕР . . . . .	14
2.1. Псевдокоманды START и END . . . . .	14
2.2. Псевдокоманда USING . . . . .	15
2.3. Команда BALR . . . . .	16
2.4. Псевдокоманда EQU . . . . .	16
2.5. Команда BR . . . . .	17
2.6. Команда L . . . . .	17
2.7. Команда ST . . . . .	18
2.8. Команда A . . . . .	18
2.9. Команда S . . . . .	19
2.10. Псевдокоманда DC . . . . .	19
2.11. Псевдокоманда DS . . . . .	20
3. Характеристики эмулируемой ЭВМ . . . . .	21
3.1. Память . . . . .	21
3.2. Регистры . . . . .	21
3.3. Допустимые форматы данных . . . . .	21
3.4. Форматы команд ЭВМ . . . . .	22

## 1. БАЗОВЫЙ НАБОР КОНСТРУКЦИЙ ЯЗЫКА ПЛ1

Макет компилятора с ПЛ1, входящий в состав учебной системы программирования, способен обрабатывать те исходные тексты программ на ПЛ1, которые состоят из языковых конструкций, представленных в следующем примере:

```
EXAMP: PROC OPTIONS (MAIN);
      DCL A BIN FIXED (31) INIT ( 11B );
      DCL B BIN FIXED (31) INIT ( 100B );
      DCL C BIN FIXED (31) INIT ( 101B );
      DCL D BIN FIXED (31);
      D = A + B - C;
END EXAMP;
```

Рассмотрим этот минимальный набор операторов ПЛ1, необходимый для дальнейшего обсуждения и использования в процессе выполнения лабораторных работ.

### 1.1. Операторы пролога/эпилога программы

#### 1.1.1. О п е р а т о р PROCEDURE

Оператор PROCEDURE или PROC:

- предназначен для обозначения начала процедурного блока,
- имеет формат:

```
имя_прогр: PROCEDURE OPTIONS (MAIN);           /*для головной */
                                                    /* программы   */
```

```
имя_прогр: PROCEDURE( пар_1, ... , пар_N ); /* для           */
                                                    /*подпрограммы */
```

где:

пар\_1, ... , пар\_N - формальные параметры подпрограммы.

#### 1.1.2. О п е р а т о р END

Оператор END:

- предназначен для обозначения конца процедурного блока,
- имеет формат:

```
END имя_прогр;
```

Имя\_прогр должно совпадать с тем, которое было указано в соответствующем операторе PROCEDURE.

## 1.2. Декларативные конструкции

### 1.2.1. К о н с т а н т ы языка ПЛ1

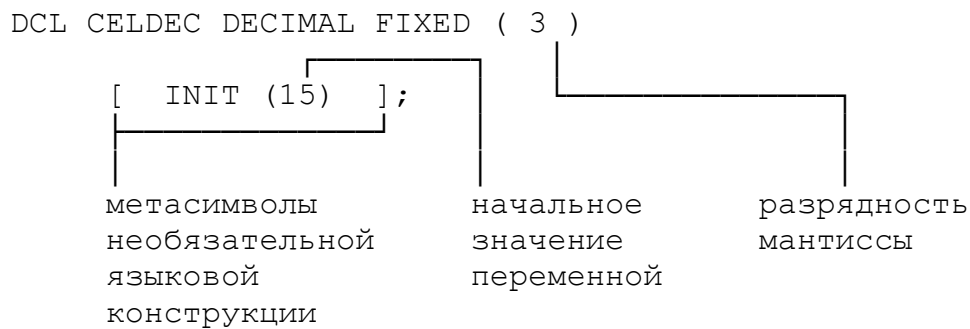
Для объявления констант в языке ПЛ1 нет специальных декларативных операторов. Константы объявляются контекстуально следующим образом:

- целые десятичные: 1, +1, 10, -27 и т.д.,
- целые двоичные: 1В, 1011В, -110В и т.д.,
- целые в зонном формате: 1, 3, 7 и т.д.,
- символьные строки: 'ABC', 'AAAA', (4)'A' и т.д.,
- битовые строки: '101'В, '1111'В, (4)'1'В и т.д.,
- метки: LABEL: , МЕТКА1: и т.д.

### 1.2.2. С к а л я р н ы е п е р е м е н н ы е языка ПЛ1

Скалярные переменные объявляются с помощью декларативного оператора DECLARE или DCL следующим образом:

- **целые десятичные**



- целые двоичные

DCL CELDVO BINARY FIXED ( 15 )

[ INIT (1010В) ];

- целые в зонном формате

DCL CELZON PICTURE '99...9'

[ INIT (187) ];

каждый символ '9' соответствует одному разрешенному разряду мантиссы переменной CELZON

- метки

```
DCL METKA LABEL
```

```
    [ INIT ( L1 ) ]; метка_константа
```

- символьная строка

```
DCL SYMSTR CHARACTER ( 36 )
```

```
    [ INIT ( (36)'A' ) ];
```

- битовая строка

```
DCL BITSTR BIT ( 40 )
```

```
    [ INIT ( '101 ... 1'B ) ];
```

### 1.2.3. А г р е г а т и р о в а н н ы е   п е р е м е н н ы е языка ПЛ1

```
DCL 1 ANKETA,                /* Анкета          */
    2 FAMIL CHAR (20),       /* Фамилия         */
    2 IMJA  CHAR (20),       /* Имя             */
    2 OTCH  CHAR (25),       /* Отчество        */
    2 GODR  DEC FIXED (4),   /* Год рождения    */
    2 BEC   BIN FIXED (15);  /* Вес             */
```

### 1.2.4. И н д е к с и р о в а н н ы е   переменные языка ПЛ1

```
DCL 1 ANKET_S (25),          /* Анкеты          */
    2 FAMIL CHAR (20),       /* Фамилия         */
    2 IMJA  CHAR (20),       /* Имя             */
    2 OTCH  CHAR (25),       /* Отчество        */
    2 GODR  DEC FIXED (4),   /* Год рождения    */
    2 BEC   BIN FIXED (15);  /* Вес             */
```

### 1.2.5. С о в м е щ е н и е   переменных в ОЗУ

```
DCL BUF_ANKETA CHAR (70);    /* Буфер чтен._зап. */
                                /* анкет             */
```

```
DCL 1 ANKETA DEFINED BUF_ANKETA, /* Анкета          */
    2 FAMIL CHAR (20),          /* Фамилия         */
    2 IMJA  CHAR (20),          /* Имя             */
    2 OTCH  CHAR (25),          /* Отчество        */
    2 GODR  DEC FIXED (4),      /* Год рождения    */
    2 BEC   BIN FIXED (15);     /* Вес             */
```



### 1.3. Императивные операторы

#### 1.3.1. Оператор `присваивания` (назначения)

Оператор присваивания:

- предназначен для назначения нового значения переменной, идентификатор которой использован в левой части оператора,
- имеет формат:

`имя_перемен = выражение;`

#### 1.3.2. Оператор `условного разветвления`

Оператор условного разветвления:

- предназначен для организации разветвления программы в зависимости от истинности или ложности заданного условия,
- имеет формат:

`IF условие THEN оператор_1; ELSE оператор_2;`

#### 1.3.3. `Составной` оператор

Составной оператор:

- предназначен для группирования нескольких операторов с целью обеспечения возможности рассмотрения компилятором всей группы как единого оператора,
- имеет формат:

`DO; оператор_1; ... оператор_N; END;`

#### 1.3.4. Оператор `безусловной` передачи управления

Оператор безусловной передачи управления:

- предназначен для изменения естественного порядка выбора очередного исполняемого оператора,
- имеет формат:

`GOTO имя_метки ;`

#### 1.3.5. Оператор `цикла`

Оператор цикла:

- предназначен для циклического повторения внутреннего (возможно составного) оператора с регулярным изменением значения переменной цикла на каждом витке цикла,
- имеет формат:

```

DO имя_перем = выражен_1 [ ВY выражен_2 ] ТО выражен_3;

    оператор;

END;

```

Цикл начинается со стартовым значением переменной цикла имя\_перем, равным выражен\_1, которое далее на каждом витке цикла будет инкрементироваться на величину выражен\_2 (возможно отрицательную). Цикл завершится при равенстве значения переменной цикла имя\_перем выражению выражен\_3.

#### 1.3.6. Встроенная функция SUBSTR

Встроенная функция SUBSTR:

- применяется в строковых выражениях,
- предназначена для выделения из строки (символов или битов) подстроки заданной длины, начинающейся с заданной позиции строки,
- имеет формат:

```
SUBSTR ( имя_строки, выражен_1, выражен_2 )
```

Здесь:

- выражен\_1 – номер позиции строки, с которого начинается подстрока (нумерация начинается с 1),
- выражен\_2 – длина подстроки.

#### 1.4. Правила преобразования данных назначением

При написании оператора присваивания допускается различие типов переменной, указанной в левой, и выражения в правой частях оператора. Кроме того допускается неоднородность типов переменных, являющихся элементами выражения. Т.к. любое выражение м.б. представлено как суперпозиция элементарных одноместных и (или) двухместных операций, операндами которых м.б. разнотипные данные, то для выполнения каждой такой операции необходимо выполнить выравнивание типов. В ПЛ1 такие преобразования выполняются согласно следующих правил:

типы операндов операции	тип операции
строковые и арифметические	арифметические
комплексные и вещественные	комплексные
с фиксированной и плавающей точкой	с плавающей точкой
двоичные и десятичные	двоичные
битовые и символьные	символьные

После вычисления значения и типа выражения в случае использования такого выражения в качестве правой части оператора присваивания может возникнуть необходимость в выравнивании типов левой и правой частей. Это выравнивание в общем случае может оказаться сложным и выполнимым за несколько последовательных элементарных этапов, примеры которых приведены ниже.

#### 1.4.1. Преобразование символьной строки в битовую строку

```
...
DCL A BIT (n);
DCL B CHAR (m);
...
A = B;
```

Данное преобразование возможно только, если B состоит из литер '0' или '1'.

При выполнении этого условия i-й бит в A устанавливается в:

- '0'B, если i-й байт в B равен '0',
- '1'B, если i-й байт в B равен '1'.

#### 1.4.2. Преобразование битовой строки в символьную строку

```
...
DCL B BIT (n);
DCL A CHAR (m);
...
A = B;
```

Данное преобразование возможно всегда. При выполнении оператора присваивания i-й байт в A устанавливается в:

- '0', если i-й бит в B равен '0'B,
- '1', если i-й бит в B равен '1'B.

#### 1.4.3. Преобразование битовой строки в двоичное целое с фиксированной точкой

```

...
DCL B BIT (n);
DCL A BIN FIXED (n-1);
...
A = B;

```

где  $n = 16$  или  $32$ .

Данное преобразование возможно всегда. При выполнении оператора присваивания  $i$ -й бит в  $A$  приравнивается к  $i$ -му биту и полученная в  $A$  комбинация битов воспринимается далее как целое с фиксированной точкой. Например, если  $B$  равно '1011'B, то  $A$  примет значение 1011B.

Этот вид преобразования играет роль шлюза при переходе от строковых величин (непосредственно от типа BIT и транзитивно от типа CHAR) к арифметическим (непосредственно к типу BIN FIXED и транзитивно к типам DEC FIXED и PIC или DEC FLOAT).

#### 1.4.4. Преобразование двоичного целого с фиксированной точкой в битовую строку

```

...
DCL A BIT (n);
DCL B BIN FIXED (n-1);
...
A = B;

```

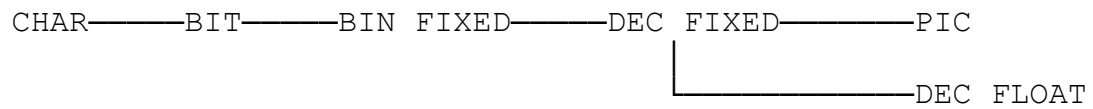
где  $n = 16$  или  $32$ .

Данное преобразование возможно всегда. При выполнении оператора присваивания  $i$ -й бит в  $A$  приравнивается к  $i$ -му биту и полученная в  $A$  комбинация битов воспринимается далее битовая строка. Например, если  $B$  равно 1011B, то  $A$  примет значение '1011'B.

Этот вид преобразования играет роль шлюза при переходе от арифметических величин (непосредственно от типа BIN FIXED и транзитивно от типов DEC FIXED и PIC или DEC FLOAT) к строковым величинам (непосредственно к типу BIT и транзитивно к типу CHAR).

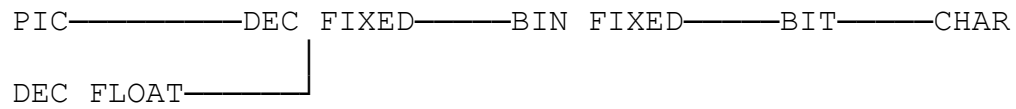
#### 1.4.5. Технологическая цепочка элементарных преобразований при переходе от строковых данных к арифметическим

При построении арифметических эквивалентов строковых данных допускается следующая последовательность элементарных преобразований (читаемая слева направо):



1.4.6. Технологическая цепочка элементарных преобразований при переходе от арифметических данных к строковым

При построении строковых эквивалентов арифметических данных допускается следующая последовательность элементарных преобразований (читаемая слева направо) :



## 2. БАЗОВЫЙ НАБОР КОНСТРУКЦИЙ ЯЗЫКА АСSEMBЛЕР

Макет компилятора с АСSEMBЛЕРА, входящий в состав учебной системы программирования, способен обрабатывать те исходные тексты программ на АСSEMBЛЕРЕ, которые состоят из языковых конструкций, представленных в следующем примере:

МЕТКА	КОП	ОП-НД	ПОЯСНЕНИЯ
EXAMP	START	0	Начало программы
	BALR	RBASE, 0	Загрузить регистр базы
	USING	*, RBASE	Назначить регистр базой
	L	RRAB, A	Загрузка переменной в регистр
	A	RRAB, B	Формирование промежуточного значения
	S	RRAB, C	Формирование промежуточного значения
	ST	RRAB, D	Формирование значения арифм. выражения
	BCR	15, 14	Выход из программы
A	DC	F'3'	Определение переменной
B	DC	F'4'	Определение переменной
C	DC	F'5'	Определение переменной
D	DC	F'0'	Определение переменной
RBASE	EQU	15	
RRAB	EQU	5	
	END	EXAMP	Конец программы

Рассмотрим этот минимальный набор операторов АСSEMBЛЕРА, необходимый для дальнейшего обсуждения и использования в процессе выполнения лабораторных работ.

### З а м е ч а н и е

Все операторы языка АСSEMBЛЕРА делятся на две группы:

1. Команды.
2. Псевдокоманды.

Вторая группа (т.е. псевдокоманды) делятся, в свою очередь, на:

- псевдокоманды, влияющие на размер компилируемой программы,
- псевдокоманды, не влияющие на размер компилируемой программы.

#### 2.1. П с е в д о к о м а н д ы START и END

Каждая компилируемая программа, написанная на АСSEMBЛЕРЕ, должна иметь начало (пролог) и конец (эпилог), которые определяются соответственно псевдокомандами START и END, имеющими следующие форматы:

МЕТКА	МНЕМОКОД	ОПЕРАНДЫ
имя или пробел	START	самоопределенный терм или пробел
---"---	END	перемещаемое выражение или пробел

#### П р и м е ч а н и е:

Перемещаемым является выражение, значение которого зависит от начального адреса данной программы в зоне загрузки ОЗУ. Иначе выражение называется абсолютным.

С помощью этих псевдокоманд (невлияющих на длину компилируемой программы) можно составить простейшую синтаксически-правильную, программу, лишенную, однако, практического смысла из-за отсутствия в ней семантики:

МЕТКА	МНЕМОКОД	ОПЕРАНДЫ
EXAMPL	START END	0

## 2.2. П с е в д о к о м а н д а USING

Псевдокоманда USING:

- не изменяет длину компилируемой программы,
- используется для передачи компилятору как номера РОН, который должен быть использован в качестве базового (первый операнд псевдокоманды), так и относительного адреса компилируемой программы, который соответствует адресу, полученному с помощью нулевого смещения относительно выбранной базы (второй операнд псевдокоманды),
- имеет следующий формат:

МЕТКА	МНЕМОКОД	ОПЕРАНДЫ	Пояснения
метка или пробел	USING	v, r	выбор базового РОН

где:

$v$  – абсолютное или перемещаемое выражение,  
 $r$  – номер РОН, содержимое которого сопоставляется значению выражения  $v$ .

Применение здесь термина "сопоставляется", а не "равно" не случайно, т.к. ответственность за правильную загрузку РОН  $r$  возлагается на программиста, а псевдокоманда USING лишь информирует компилятор о необходимости безусловного учета указанного сопоставления при дальнейшей обработке программы.

### 2.3. Команда BALR

Команда BALR:

- увеличивает длину компилируемой программы на два байта,
- в общем случае используется для организации передачи управления подпрограмме с последующим возвратом в вызвавшую программу,
- имеет следующий формат:

МЕТКА	МНЕМОКОД	ОПЕРАНДЫ	Пояснения
метка или пробел	BALR	R1, R2	передача управления на подпрограмму с последующим возвратом

где:

R1 – номер РОН, содержащий адрес возврата;  
 R2 – номер РОН, содержащий адрес передачи управления (в случае, если R2=0, то передача будет выполняться на команду, следующую за данной командой BALR).

### 2.4. Псевдокоманда EQU

Псевдокоманда EQU:

- не изменяет длину компилируемой программы,
- используется для передачи компилятору указания об отождествлении во всем последующем тексте компилируемой программы идентификатора, указанного в поле МЕТКА, с абсолютным или перемещаемым выражением, записанным в поле ОПЕРАНДЫ,
- имеет следующий формат:



МЕТКА	МНЕМОКОД	ОПЕРАНДЫ	Пояснения
метка или пробел	EQU	v	установление эквивалентности между левой и правой частями псевдооперации EQU

где:

v – абсолютное или перемещаемое выражение.

## 2.5. Команда BR

Команда BR:

- увеличивает длину компилируемой программы на два байта,
- используется для организации безусловной передачи управления по адресу, указанному в РОН, выбранном единственным параметром,
- имеет следующий формат:

МЕТКА	МНЕМОКОД	ОПЕРАНДЫ	Пояснения
метка или пробел	BR	R	безусловная передача управления

где:

R – номер РОН, содержащий адрес передачи управления.

## 2.6. Команда L

Команда L:

- увеличивает длину компилируемой программы на четыре байта,
- используется для загрузки из помеченной области ОЗУ нового четырехбайтового значения (слова) в РОН с заданным номером,
- имеет следующий формат:

МЕТКА	МНЕМОКОД	ОПЕРАНДЫ	Пояснения
метка или пробел	L	R, m	загрузка в РОН R нового значения

где:

R – номер РОН, в который загружается новое значение,  
 M – метка, соответствующая области ОЗУ, из которой выбирается новое значение для последующей загрузки в РОН R.

## 2.7. Команда ST

Команда ST:

- увеличивает длину компилируемой программы на четыре байта,
- используется для разгрузки четырехбайтового содержимого РОН с заданным номером (слова) в помеченную область ОЗУ,
- имеет следующий формат:

МЕТКА	МНЕМОКОД	ОПЕРАНДЫ	Пояснения
метка или пробел	ST	R, m	разгрузка значения РОН R в ОЗУ

где:

R – номер разгружаемого РОН,  
 m – метка, соответствующая области ОЗУ, в которую запоминается содержимое РОН с номером R.

## 2.8. Команда A

Команда A:

- увеличивает длину компилируемой программы на четыре байта,
- используется для сложения четырехбайтового содержимого РОН с заданным номером (слова) со словом, расположенным в помеченной области ОЗУ, с записью результата в РОН, выбранный для хранения первого операнда сложения,
- имеет следующий формат:

МЕТКА	МНЕМОКОД	ОПЕРАНДЫ	Пояснения
метка или пробел	A	R, m	сложение содержимого РОН R со словом из ОЗУ с меткой m

где:

R – номер РОН, содержащий первый операнд сложения,

m – метка, соответствующая области ОЗУ, содержащей второй операнд сложения.

## 2.9. Команда S

Команда S:

- увеличивает длину компилируемой программы на четыре байта,
- используется для вычитания из четырехбайтового содержимого РОН с заданным номером (слова) второго слова, расположенного в помеченной области ОЗУ с записью результата в РОН, выбранный для хранения первого операнда вычитания,
- имеет следующий формат:

МЕТКА	МНЕМОКОД	ОПЕРАНДЫ	Пояснения
метка или пробел	S	R, m	вычитание из содержимого РОН R слова из ОЗУ с меткой m

где:

R – номер РОН, содержащий первый операнд вычитания,  
m – метка, соответствующая области ОЗУ, содержащей второй операнд сложения.

## 2.10. Псевдокоманда DC

Псевдокоманда DC:

- увеличивает длину компилируемой программы на длину константы, определяемой с помощью данной псевдокоманды,
- используется для резервирования и инициализации начальным значением области памяти, отводимой для хранения константы, определяемой данной псевдокомандой,
- имеет следующий формат:

МЕТКА	МНЕМОКОД	ОПЕРАНДЫ	Пояснения
метка или пробел	DC	TV	определить константу

где:

T – тип константы (например: F – двоичное слово, X – шестнадцатеричный байт и т.п.),

V – начальное значение константы (например: 'З', 'FЗ', и т.п.).

### 2.11. П с е в д о к о м а н д а DS

Псевдокоманда DS:

- увеличивает длину компилируемой программы на длину поля ОЗУ, определяемого с помощью данной псевдокоманды;
- используется для резервирования области памяти, отводимой для хранения константы, определяемой данной псевдокомандой,
- имеет следующий формат:

МЕТКА	МНЕМОКОД	ОПЕРАНДЫ	Пояснения
метка или пробел	DS	T	определить поле для хранения константы

где:

T – тип константы (например: F – двоичное слово, X – шестнадцатичный байт и т.п.).

### 3. Характеристики эмулируемой ЭВМ

Исполняемый код, формируемый учебной системой программирования, предназначен для выполнения на целевой ЭВМ из системы IBM-370. Ниже приведены некоторые характеристики этой ЭВМ, которые должны быть учтены при ее эмуляции на технологической ЭВМ IBM PC.

#### 3.1. Память

Адресуемая единица	Длина в байтах	Длина в битах	Особенности
Байт	1	8	нет
Полуслово	2	16	адрес кратен 2
Слово	4	32	адрес кратен 4
Двойное слово	8	64	адрес кратен 8

Диапазон адресации: от 0 до  $2^{24}-1$ , т.е. предельный размер ОЗУ равен 16 мб.

#### 3.2. Регистры

В ЭВМ имеются 16 тридцатидвухразрядных регистров общего назначения (РОН), которые м.б. использованы программистами в качестве быстрой памяти для хранения промежуточных результатов, а также в качестве базы и (или) индекса при вычислении абсолютных адресов операндов.

#### 3.3. Допустимые форматы данных

з	цел. двоичн
---	-------------

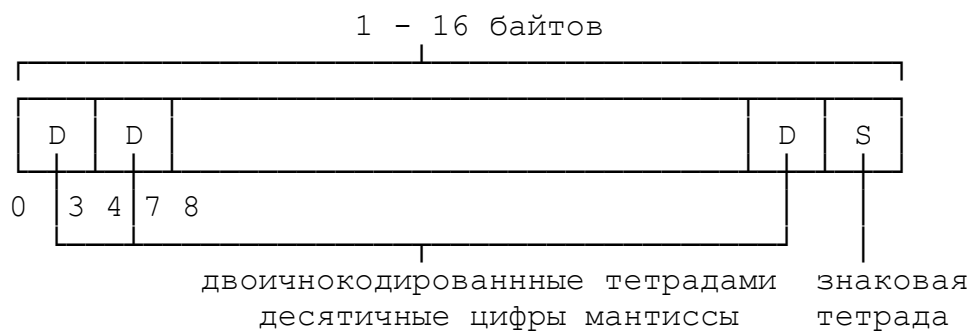
0 1                      15

Короткое  
целое с  
фиксиров.  
точкой

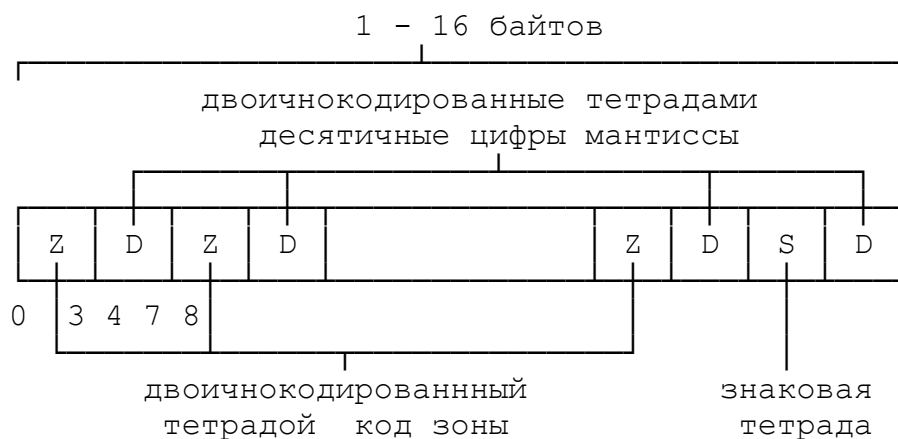
з	целое двоичное
---	----------------

0 1    31

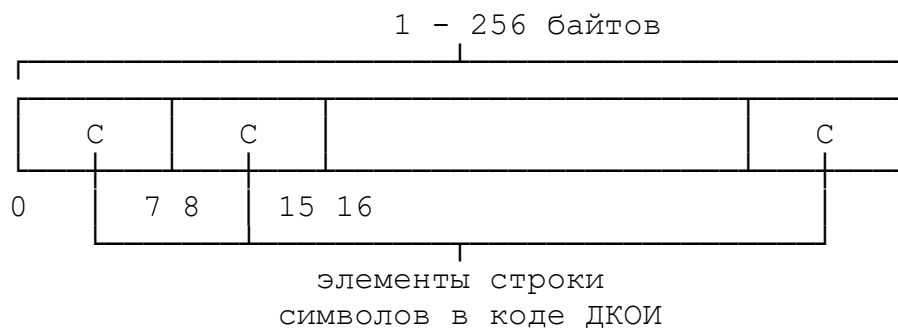
Длинное  
целое с  
фиксиров.  
точкой



Десятичное  
упакованное  
целое



Десятичное  
целое в  
зонном  
формате



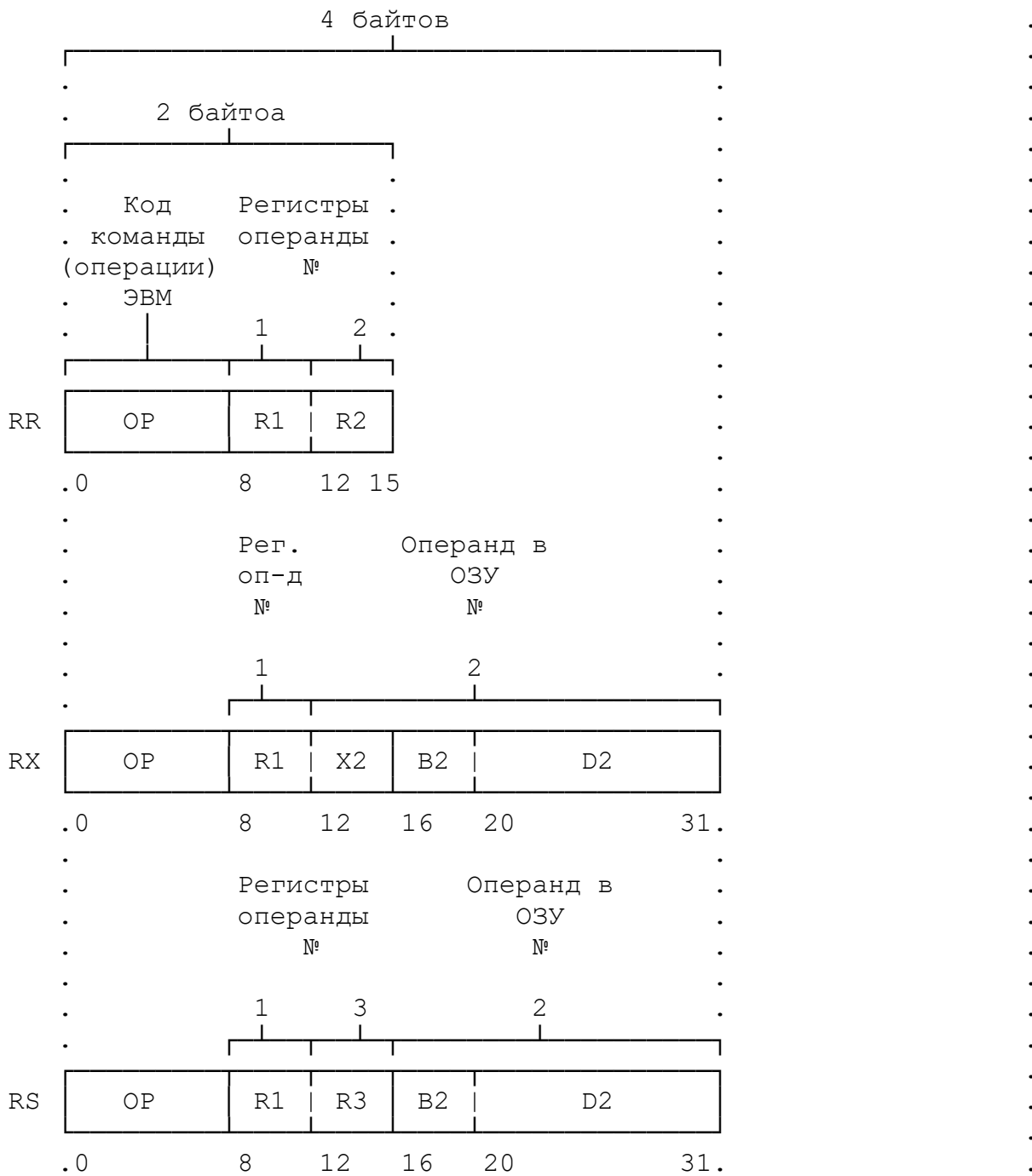
Строчно-  
символьное  
данное

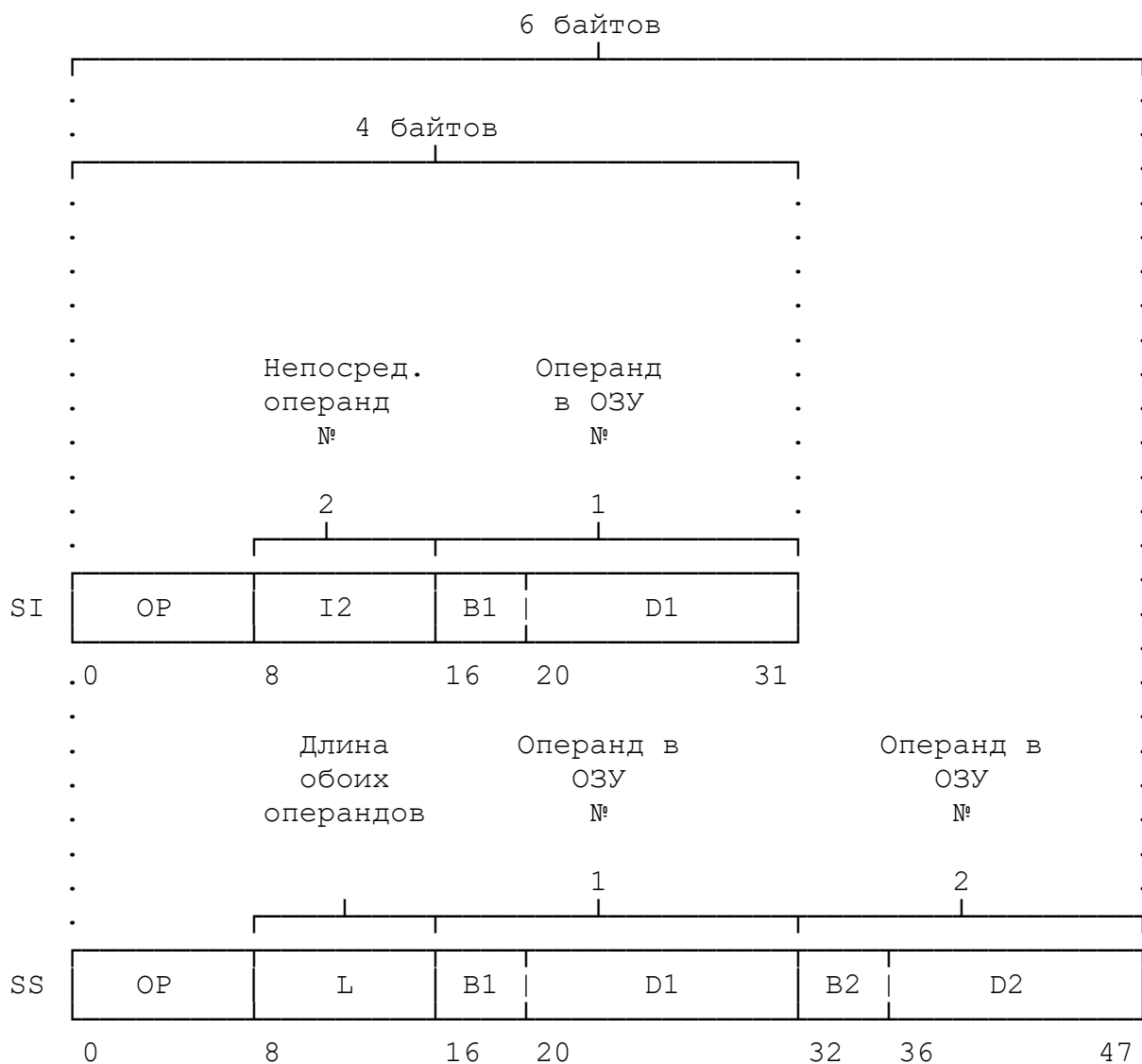
### 3.4. Форматы команд ЭВМ

Команды эмулируемой ЭВМ - представителя системы IBM-370:

- имеют переменную длину в 2, 4 или 6 байтом,
- располагаются в ОЗУ с адреса, кратного 2-м,
- относятся к одному из следующих 5-ти форматов (RR, RX, RS, SI, SS), которые приведены ниже в перечисленном порядке:

6 байтов





Здесь приняты следующие обозначения:

OP - код команды (операции) ЭВМ,

R<sub>i</sub> - регистры общего назначения, содержащие операнд номер *i*,

X<sub>i</sub> - регистры общего назначения, использованные в качестве индексных регистров в *i*-м операнде,

B<sub>i</sub> - регистры общего назначения, использованные в качестве базовых регистров в *i*-м операнде,

D<sub>i</sub> - смещение адреса *i*-го операнда относительно содержимого базового регистра B<sub>i</sub>,

I<sub>i</sub> - непосредственный (хранящийся в теле операции) *i*-й операнд,

L - уменьшенная на единицу длина операндов.