Лабораторная работа № 5

**Подпрограммы и стек**

Цель работы: изучение организации программ с использованием подпрограмм

Краткие сведения:

В программировании часто встречаются ситуации, когда одинаковые действия необходимо выполнить многократно в разных частях программы (например, вычисление функции sin x). При этом с целью экономии памяти не следует многократно повторять одну и ту же последовательность команд – достаточно один раз написать так называемую *подпрограмму* (в терминах языков высокого уровня - процедуру) и обеспечить правильный вызов этой подпрограммы и возврат в точку вызова по завершению подпрограммы.

Для *вызова* подпрограммы необходимо указать ее начальный адрес в памяти и передать (если необходимо) параметры — те исходные данные, с которыми будут выполняться предусмотренные в подпрограмме действия. Адрес под­программы указывается в команде вызова **CALL**, а параметры могут переда­ваться через определенные ячейки памяти, регистры или стек. *Возврат* в точку вызова обеспечивается сохранением адреса текущей коман­ды (содержимого регистра PC) при вызове и использованием в конце подпро­граммы команды возврата **RET**, которая возвращает сохраненное значение ад­реса возврата в PC.

Для реализации механизма вложенных подпрограмм (возможность вызова подпрограммы из другой подпрограммы и т.д.) адреса возврата целесообразно сохранить в стеке. *Стек* («магазин») – особым образом организованная безадресная память, доступ к которой осуществляется через единственную ячейку, называемую *верхушкой стека.* При записи слово помещается в верхушку стека, предварительно все находящиеся в нем слова смещаются вниз на одну позицию; при чтении извлекается содержимое верхушки стека (оно при этом из стека исчезает), а все оставшиеся слова смещаются вверх на одну позицию. Такой механизм напоминает действие магазина стрелкового оружия (отсюда и второе название). В программировании называют такую дисциплину обслуживания **LIFO** (Last In First Out, последним пришел - первым вышел) в отличие от дисциплины типа *очередь –* **FIFO** (First In First Out, первым пришел – первым вышел).

В обычных ОЗУ нет возможности перемещать слова между ячейками, поэтому при организации стека перемещается не массив слов относительно неподвижной верхушки, а верхушка относительно неподвижного массива. Под стек отводится некоторая область ОЗУ, причем адрес верхушки хранится в специальном регистре процессора – указателе стека SP.

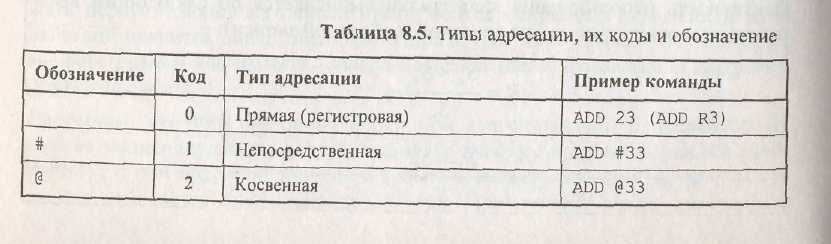
В стек можно поместить содержимое регистра общего назначения по команде **PUSH** или извлечь содержимое верхушки в регистр общего назначения по команде **POP**. Кроме того, по команде вызова подпрограммы **CALL** значение программного счетчика PC (адрес следующей команды) помещается в верхушку стека, а по команде **RET** содержимое верхушки стека извлекается в PC. При каждом обращении в стек указатель SP автоматически модифицируется.

В большинстве ЭВМ стек «растет» в сторону меньших адресов, поэтому перед каждой записью содержимое SP уменьшается на 1, а после каждого извлечения содержимое SP увеличивается на 1. Таким образом, SP всегда указывает на верхушку стека.

Цель настоящей лабораторной работы – изучение организации программ с использованием подпрограмм. Кроме того, в процессе организации циклов мы будем использовать новые возможности системы команд модели ЭВМ, которые позволяют работать с новым классом памяти – сверхоперативной (регистры общего назначения - РОН). В реальны ЭВМ доступ в РОН занимает значительно меньше время, чем в ОЗУ; кроме того, команды обращения с регистрами короче команд обращения к памяти. Поэтому в РОН размещаются наиболее часто используемые в программе данные, промежуточные результаты, счетчики циклов, косвенные адреса и т.п.

В системе команд учебной ЭВМ для работы с РОН используются специаль­ные команды, мнемоники которых совпадают с мнемониками соответствую­щих команд для работы с ОЗУ, но в адресной части содержат символы реги­стров R0—R9.

Кроме обычных способов адресации (прямой и косвенной) в регистровых командах используются два новых — постинкрементная и предекрементная (см. табл. 8.5). Кроме того, к регистровым относится команда организации цикла JRNZ R,M. По этой команде содержимое указанного в команде регист­ра уменьшается на 1, и если в результате вычитания содержимого регистра не равно 0, то управление передается на метку м. Эту команду следует ставить в конце тела цикла, метку м— в первой команде тела цикла, а в регистр R по­мещать число повторений цикла.



**Пример программы**

Даны три массива чисел. Требуется вычислить среднее арифметическое их максимальных элементов. Каждый массив задается двумя параметрами: адресом первого элемента и длиной.

Очевидно, в программе трижды необходимо выполнить поиск максимального элемента массива, поэтому следует написать соответствующую подпрограмму.

Параметры в подпрограмму будем передавать через регистры: R1 – начальный адрес массива, R2 – длина массива.

Рассмотрим конкретную реализацию этой задачи. Пусть первый массив на­чинается с адреса 085 и имеет длину 14 элементов, второй— 100 и 4, тре­тий — 110 и 9.

Программа будет состоять из основной части и подпрограм­мы. Основная программа задает параметры подпрограмме, вызывает ее и со­храняет результаты работы подпрограммы в рабочих ячейках. Затем осуществляет вычисление среднего арифметического и выводит результат на устройство вывода.

В качестве рабочих ячеек используются регистры общего назначения R6 и R7 — для хранения максимальных элементов массивов.

Подпрограмма получает параметры через регистры R1 (начальный адрес мас­сива) и R2 (длина массива). Эти регистры используются подпрограммой в качестве регистра текущего адреса и счетчика цикла соответственно.

Кроме того, R3 используется для хранения текущего максимума, a R4— для вре­менного хранения текущего элемента. Подпрограмма возвращает результат через аккумулятор. В таблице приведен текст основной программы и под­программы.

Обратите внимание, цикл в подпрограмме организован с по­мощью команды **JRNZ**, а модификация текущего адреса — средствами пост­инкрементной адресации.

**Таблица** *Пример программы*

| **Команда** | **Примечания** |
| --- | --- |
| **Основная программа** | |
| RD #95 | Загрузка |
| WR R1 | параметров |
| RD #14 | первого |
| WR R2 | массива |
| CALL M | Вызов подпрограммы |
| WR R6 | Сохранение результатов |
| RD #100 | Загрузка |
| WR R1 | параметров |
| RD #4 | второго |
| WR R2 | массива |
| CALL M | Вызов подпрограммы |
| WR R7 | Сохранение результата |
| RD # 110 | Загрузка |
| WR R1 | параметров |
| RD #9 | третьего |
| WR R2 | массива |
| CALL M | Вызов подпрограммы |
| ADD R7 | Вычисление |
| ADD R6 | среднего |
| DIV #3 | арифметического |
| OUT | Вывод результата |
| **Подпрограмма MAX** | |
| HLT | Стоп |
| M: RD @R1 | Загрузка |
| WR R3 | первого элемента в R3 |
| L2: RD R1+ | Чтение элемента и модификация адреса |
| WR R4 | Сравнение |
| SAB R3 | и замена, |
| JS L1 | если R3 < R4 |
| MOV R3, R4 |  |
| L1: JRNZ R2,L2 | Цикл |
| RD R3 | Чтение результата в Асс |
| RET | Возврат |

**Задание**

Составить и отладить программу учебной модели ЭВМ для решения следующей задачи. Три массива в памяти заданы начальными адресами и длинами. Вычислить и вывести на устройство вывода среднее арифметическое параметров этих массивов. Параметры определяются заданием к предыдущей лабораторной работе (см. табл. 2), причем соответствие между номерами вариантов заданий 4 и 5 устанавливается по таблице 3.

**Таблица 2**

| Номер варианта | Характеристика последовательности чисел С1, С2, …, Сn |
| --- | --- |
| 1 | Количество четных чисел |
| 2 | Номер минимального числа |
| 3 | Произведение всех чисел |
| 4 | Номер первого отрицательного числа |
| 5 | Количество чисел, равных С1 |
| 6 | Количество отрицательных чисел |
| 7 | Максимальное отрицательное число |
| 8 | Номер первого положительного числа |
| 9 | Минимальное положительное число |
| 10 | Номер максимального числа |
| 11 | Количество нечетных чисел |
| 12 | Количество чисел, меньших С1 |
| 13 | Разность сумм четных и нечетных элементов массива |
| 14 | Отношение сумм четных и нечетных элементов массива |

**Таблица 3** *Соответствие между номерами заданий*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер варианта ЛР\_4 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| Номер строки в таблице 2 | 5 | 7 | 13 | 11 | 9 | 12 | 1 | 10 | 14 | 3 | 6 | 8 | 2 | 4 |

**Содержание отчета**

1. Формулировка варианта задания.
2. Распределение памяти (размещение в ОЗУ переменных, программы и не­  
   обходимых констант).
3. Тексты программы и подпрограммы.
4. Результаты программы в виде скриншота.
5. Значения исходных данных и результата выполнения программы.