Дисциплина

«РАЗРАБОТКА ЭФФЕКТИВНЫХ АЛГОРИТМОВ»

Лекция 13

МЕХАНИЗМ ВЫЗОВА ФУНКЦИИ И ОБСЛУЖИВАНИЯ РЕКУРСИИ

Лектор: Михаил Васильевич Ульянов,

muljanov@mail.ru, **8 916 589 94 04**

0. Функционал вызова / возврата и его реализация

При вызове функции необходимо:

— передать в функцию параметры (по значению или по ссылке);

— передать адрес возврата в вызывающий модуль;

— сохранить состояние машины (регистры и доп. информацию).

При возврате необходимо:

— передать вычисленное значение;

— восстановить состояние машины;

— вернуть управление в точку вызова.

Реализация функционала — «соглашение о связях» в операционной системе.

Кто сохраняет состояние машины — вызывающий модуль или вызываемая функция? Как устроена область памяти для сохранения состояния машины и обмена информацией? Как устроена связь по управлению (собственно вызов и возврат в точку вызова? Это регламентируется операционной системой в соглашении о связях. Откуда компилятор знает адрес тела функции?

Реализации соглашения о связях различны. Область сохранения состояния машины — единая для всех или отдельная для каждого модуля?

1. Рекуррентное соотношение и рекурсивная функция

Рекуррентное соотношение, определяющее рекурсивно заданную функцию факториала от целочисленного аргумента n, имеет вид

$$\begin{cases} f(0) = 1; f(1) = 1; \\ f(n) = n \cdot f(n-1), \ n \geq 2. \end{cases}$$

Рекурсивный алгоритм в виде рекурсивной функции F(n) с остановом при значении $n = 1$ (справа указано количество базовых операций в строке).

```
F(n)
    If (n=1)          (проверка останова рекурсии)   1
        then
            F ← 1                  (останов рекурсии)  1
        else
            F ← n*F(n-1)  (рекурсивный вызов)    3
Return (F)
End.
```

2. Модель программного стека

Рассмотрение механизма организации рекурсивных вызовов мы начнем с модели программного стека. Будем предполагать, что операционная система поддерживает специальную область оперативной памяти — программный стек, используя специальный регистр или ячейку памяти — указатель стека, хранящий адрес некоторой ячейки (слова) в этой области.

Две специальные команды — «записать в стек» и «читать из стека» организуют работу с этой областью так, что логически мы считаем, что имеем дело со стеком, как классической структурой, хотя указанные операции просто изменяют адрес указателя стека. Например, команда «записать в стек» уменьшает текущий адрес и записывает по этому адресу содержимое своего операнда. Такой подход гарантирует, что любая операция со стеком не приводит к перезаписи со сдвигом его содержимого.

3. Механизм вызова процедуры / функции

Механизм должен обеспечить сохранение регистров процессора, т. к. при возврате в точку вызова мы должны восстановить их значения, кроме того, в процедуру должны быть некоторым образом переданы фактические параметры.

Для обеспечения этих действий мы и будем использовать описанный выше программный стек и обслуживающие его команды. Поскольку значения должны помещаться в программный стек в порядке обратном их выборке, то вначале мы помещаем в стек адрес возврата в точку вызова, затем значения необходимых регистров и передаваемые в процедуру фактические параметры.

Кто должен сохранять регистры — вызывающая программа или процедура / функция, которой передается управление? Это регулируется «соглашением о связях», принятым в данной операционной системе.

4. Механизм вызова процедуры с использованием стека

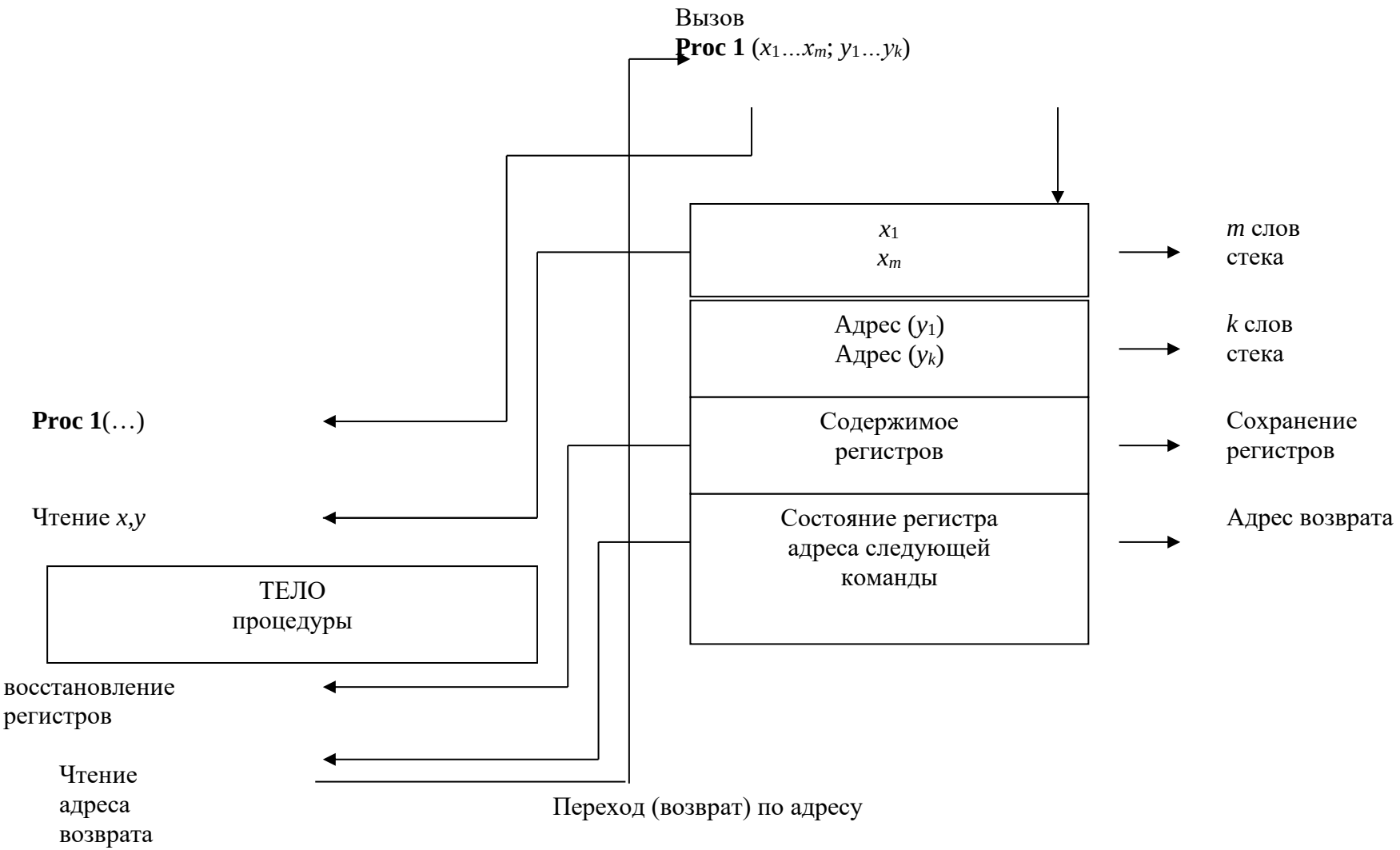

Рисунок 1. Механизм вызова процедуры с использованием стека.

При вызове процедуры вызывающая программа помещает в стек адрес возврата, состояние необходимых регистров процессора, адреса возвращаемых значений и передаваемые параметры, пользуясь командой «записать в стек». После этого выполняется переход по адресу на вызываемую процедуру, которая извлекает переданные фактические параметры (командой «читать из стека»), выполняет необходимые вычисления и помещает результаты по указанным в стеке адресам. При завершении работы вызываемая процедура восстанавливает значения регистров, выталкивает из стека адрес возврата и осуществляет переход по этому адресу, возвращая управление в точку своего вызова. При вызове функции основное отличие заключается в том, что вычисленное значение передается в вызывающую программу по имени функции. Реализация аналогична — при возврате в точку вызова в указанную ячейку выталкивается значение из стека.

5. Механизм обслуживания рекурсивного вызова

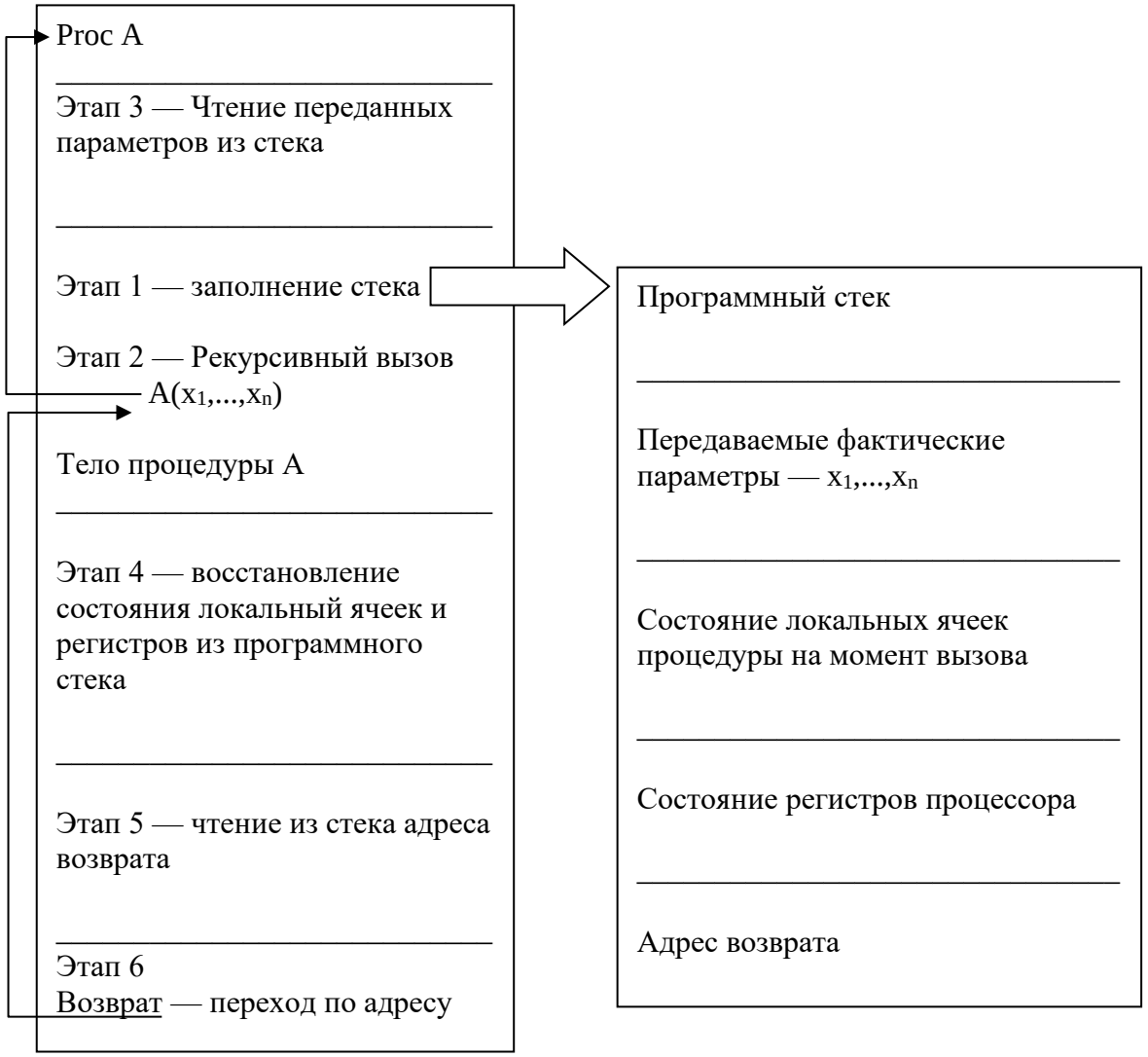

Рисунок 2. Механизм рекурсивного вызова процедуры.

1. Непосредственно перед рекурсивным вызовом процедура помещает в программный стек адрес возврата, состояние регистров, и список передаваемых параметров рекурсивного вызова.

2. Выполняется рекурсивный вызов — процедура вызывает сама себя. Для процессора это всего лишь передача управления на другую машинную команду в программном коде. Этой командой является первая команда процедуры.

3. При получении управления процедура получает доступ к переданным параметрам через программный стек, они либо считываются из стека в регистры, либо процедура имеет прямой адресный доступ к области программного стека.

4. Если этот вызов есть останов рекурсии, процедура формирует результат и восстанавливает состояние локальных ячеек и регистров процессора на момент ее вызова, используя информацию из стека и команду чтения.

Остановимся более подробно на этапе 4. Мы его описали в предположении останова рекурсии. Если это не так, то в теле процедуры формируется новый рекурсивный вызов, включающий сохранение локальных ячеек и регистров на текущий момент и вызов процедуры с новыми параметрами. Очевидно, что в этот момент информация о «новом» вызове сохраняется в стеке выше информации о «старом» вызове. Это приводит к тому, что хотя мы физически имеем один и тот же программный код, но предыстория вызовов сохраняется в стеке, и, следовательно, мы храним всю информацию о последовательности вызовов, на основе которой организуется цепочка возвратов после выполнения условия останова рекурсии.

5. После этапа 4 на верху стека остается адрес возврата, который и считывается для последующей передачи управления.

6. Выполняется команда перехода по адресу — процедура возвращает управление в тот программный модуль, откуда она была вызвана, но это возврат в тело этой же процедуры, и мы оказываемся в теле процедуры A на предыдущем уровне цепочки рекурсии.

Таким образом, механизм программного стека позволяет организовать цепочку рекурсивных вызовов процедур или функций в языке программирования высокого уровня. Хранение в стеке всей текущей информации об этих вызовах позволяет организовать обратную цепочку рекурсивных возвратов. Поэтому, если рассматривать выполнение рекурсивной функции во времени, то в момент останова рекурсии в стеке сохраняется вся информация о предыдущих рекурсивных вызовах, — и в этот момент мы используем наибольший объем памяти в области программного стека.

6. Схема рекурсивных вызовов и возвратов для факториала

Два этапа — 1. Рекурсивные вызовы; 2. Рекурсивные возвраты