Экспериментальное исследование методов распределения памяти

Все переменные, объявленные в программе, размещаются в одной непрерывной области оперативной памяти, которая называется сегментом данных. Длина сегмента данных составляет 64К.

Глобальным переменным программы память отводится в начале ее выполнения. Они существуют в течение всего периода работы программы. Для локальных переменных, описанных в подпрограмме, память отводится при вызове подпрограммы. При выходе из нее эта память освобождается, а переменные прекращают свое существование. Переменные, память под которые распределяется описанным образом, называются статическими. Использование одних лишь статических переменных лишает возможности писать программы, обрабатывающие достаточно большие информационные массивы.

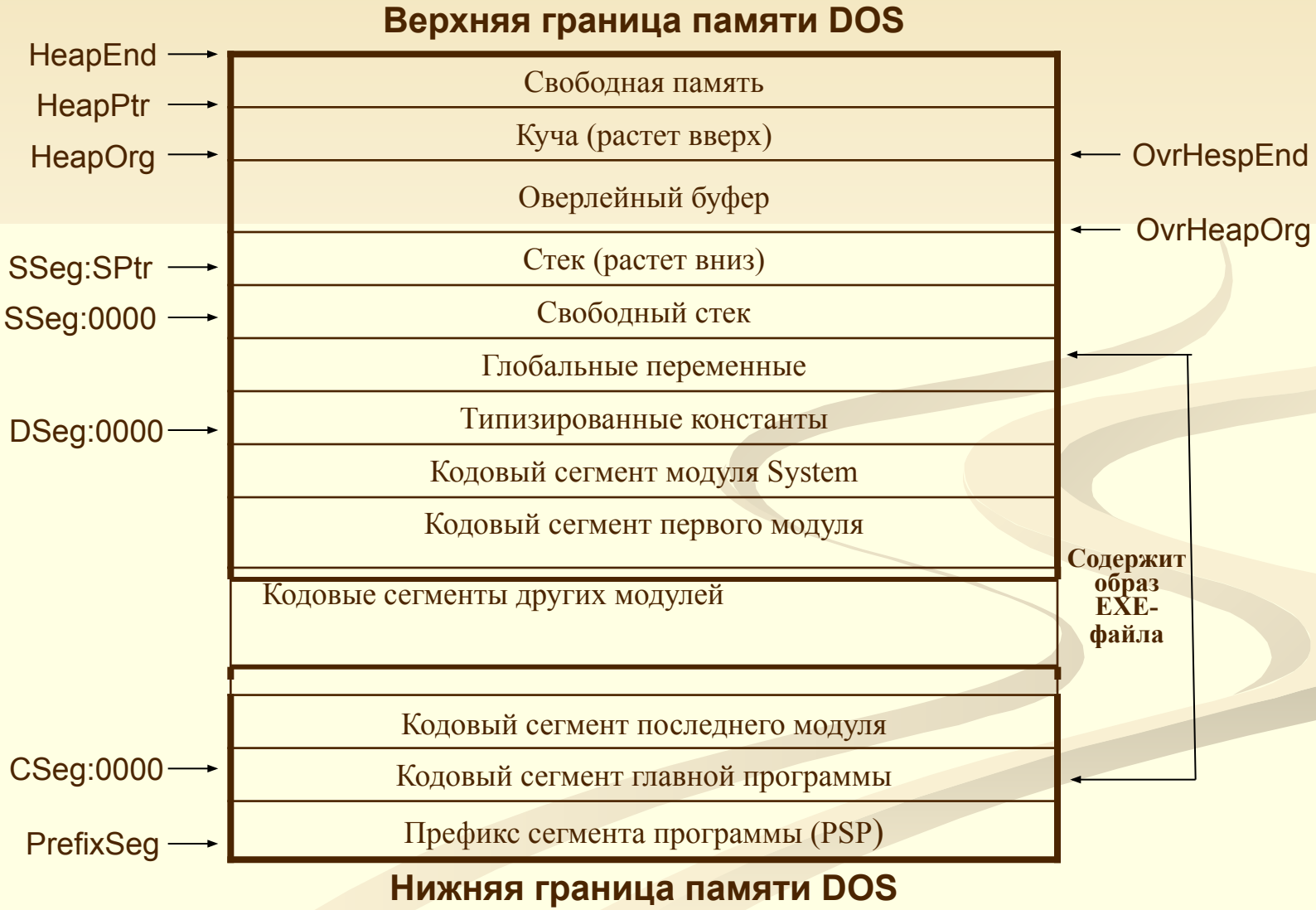
Распределение оперативной памяти ПЭВМ для программ на Pascal и C.

<table>
<tr><td>Старшие адреса</td><td colspan="2">Системная область</td></tr>
<tr><td></td><td colspan="2">Область динамической памяти (heap, куча)</td></tr>
<tr><td></td><td>Стек</td><td rowspan="3">Область программы</td></tr>
<tr><td></td><td>Данные</td></tr>
<tr><td></td><td>Код</td></tr>
<tr><td>Младшие адреса</td><td colspan="2">Системная область</td></tr>
</table>

<table>
<tr><td>Старшие адреса</td><td>Системная область</td></tr>
<tr><td></td><td>Область стека</td></tr>
<tr><td></td><td>Область динамической памяти (heap, куча)</td></tr>
<tr><td></td><td>Область глобальных переменных</td></tr>
<tr><td></td><td>Область программы</td></tr>
<tr><td>Младшие адреса</td><td>Системная область</td></tr>
</table>

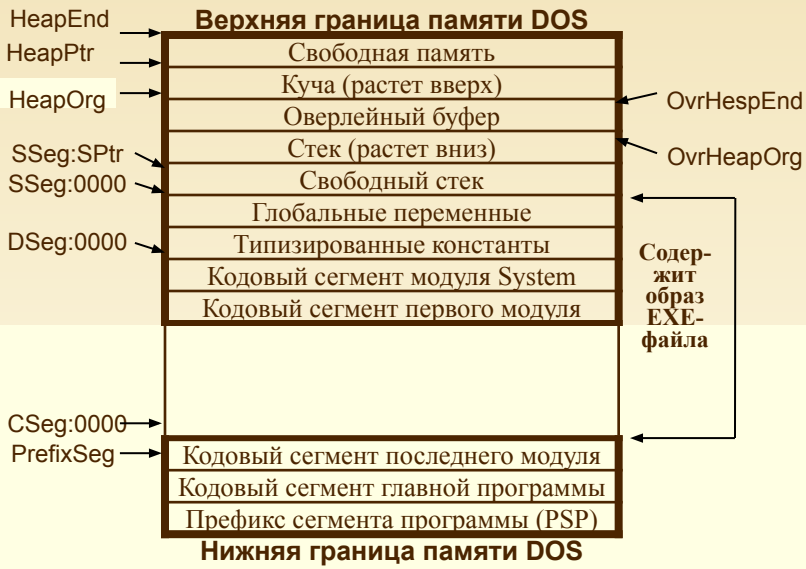
Экспериментальное исследование методов распределения памяти

Карта оперативной памяти программ на Borland Pascal



Экспериментальное исследование методов распределения памяти

Замечания к карте оперативной памяти для программ на Pascal



Префикс сегмента программы (Program Segment Prefix - PSP) -это 256-ти байтовая область, создаваемая DOS при загрузке программы. Адрес сегмента PSP хранится в переменной PrefixSeg.

Главная программа, и каждый модуль имеют свой кодовый сегмент. Главная программа занимает первый кодовый сегмент; кодовые сегменты, которые следуют за ним, занимают модули (в порядке, обратном тому, как они следовали в операторе uses), и последний кодовый сегмент занимает библиотека времени выполнения (модуль System). Размер одного кодового сегмента не может превышать 64К, но общий размер кода ограничен только имеющейся памятью.

Сегмент данных (адресуемый через DSeg) содержит все глобальные переменные и затем все типизованные константы. Регистр DS никогда не изменяется во время выполнения программы. Размер сегмента данных не может превышать 64К.

При запуске программы регистр сегмента стека (SSeg) и указатель стека (SP) устанавливаются так, что SS:SP указывает на первый байт после сегмента стека. Регистр SS никогда не изменяется во время выполнения программы, а SP может передвигаться вниз пока не достигнет конца сегмента. Размер стекового сегмента не может превышать 64К; размер по умолчанию - 16К, он может быть изменен директивой компилятора $M.

Буфер оверлеев используется стандартным модулем Overlay для хранения оверлейного кода. Размер оверлейного буфера по умолчанию соответствует размеру наибольшего оверлея в программе; если в программе нет оверлеев, размер буфера оверлеев равен 0. Размер буфера оверлеев может быть увеличен с помощью вызова программы OvrSetBuf модуля Overlay; в этом случае размер кучи соответственно уменьшается, смещением вверх HeapOrg.

Экспериментальное исследование методов распределения памяти

Исследовать:

- **Размещение в оперативной памяти ПЭВМ переменных разного типа (в соответствии с их машинным представлением) – целочисленных, символьных, логических; в каком сегменте размещаются глобальные и локальные переменные; определить динамику распределения памяти;**
  Проанализировать видимость и доступность одноимённых и разноименных глобальных и локальных переменных разных типов данных.
- **Выравнивание на границу слова (включая-выключая опцию в оболочке или используя директивы компилятора);**
- **Размещение в памяти процедур и функций: параллельных и вложенных; определить в каком сегменте и в каких местах его размещаются подпрограммы, какова динамика распределения памяти (для подпрограмм с 2-я и 3-я уровнями вложенности);**

Инструменты исследования:

- **Программы на Pascal и C, использующие переменные в описанных выше режимах.**
- **Режим отладки оболочки Borland для изучение адресов переменных (Watches).**
- **Ваша голова и её содержимое для анализа получаемых результатов.**

Экспериментальное исследование методов распределения памяти

ПРИМЕРЫ программ (написать комментарии)

```
program raspr_pamyati;
var n:integer; c:char;
      pn:^integer; pc:^char;

procedure A1;
var n:char; x:real;
      pn:^char; px:^real;


procedure B1;
var n:boolean; x:integer;
      pn:^boolean; px:^integer;
begin
 pn:=@n;
 px:=@x;
end;
begin
 pn:=@n;
 px:=@x;
B1;
end;
```

```
procedure C1;
var n:char;
      pn:^char;
begin
 pn:=@n;
end;


begin
 pn:=@n;
 pc:=@c;
 A1;
 C1;
end.
```

Экспериментальное исследование методов распределения памяти

ПРИМЕРЫ программ (написать комментарии)

```
program mehanizm_vyravn;
var n:integer; c:char;k:integer;
      pn:^integer; pc:^char; pk:^integer;

procedure A1;
var n:char; x:real; k:integer;
      pn:^char; px:^real; pk:^integer;

procedure B1;
var n:boolean; x:integer; k:integer;
      pn:^boolean; px:^integer; pk:^integer;
begin
 pn:=@n;
 px:=@x;
 pk:=@k;
end;
begin
 pn:=@n;
 px:=@x;
 pk:=@k;
B1;
end;
```

```
procedure C1;
var n:char; k:integer;
      pn:^char; pk:^integer;
begin
 pn:=@n;
 pk:=@k;
end;


begin
 pn:=@n;
 pc:=@c;
 pk:=@k;
 A1;
 C1;
end.
```