**Динамическое выделение памяти**

**Для динамического выделения памяти используется системный вызов** **sys\_brk().**

Эта системная функция позволяет установить наибольший доступный адрес в секции data. Этот адрес памяти является параметром, который передается системному вызову в регистре **ebx.**

В случае возникновения любой ошибки, системный вызов возвращает **sys\_brk().** -1 или фактический код ошибки.

Номер функции системного вызова **45**, которая, как всегда записывается в регистр eax, а в регистр ebx записываем количество выделяемых байтов.

Например, выделим 100 байтов памяти

global \_start

section .data

a dw 22,33,44

section .text

\_start:

mov eax,45 ; **sys\_brk()**

xor ebx,ebx

int 0x80 ; в eax получили наибольший доступный адрес

add eax,100 ; добавили к нему количество байтов, которые мы хотим выделить

mov ebx,eax

move ax,45

int 0x80

cmp eax,0

jl p

……………………

p: PRINT “ERROR”

FINISH

**Линейные односвязные списки**

Для работы с массивами с постоянно изменяющимся, в процессе выполнения программы, количеством элементов используются линейные односвязные списки. Линейные односвязные списки это - последовательность звеньев, которые могут размещаться в произвольных местах памяти и в каждом из которых указывается элемент списка и ссылка на следующее звено.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **List** |  |  |  |  |  |  |  |  | **….** |  |  | NIL |

В последнем звене размещается “пустая” ссылка NIL. Ссылка на первое звено списка хранится в переменной List, если список пуст, то ее значение равно NIL.

***Представление списков***

Для хранения звеньев списка обычно отводят специальную область памяти, называемую областью списков (list space) или “куча”. Размер кучи зависит от того как много списков будет использовать программа.

Ссылка на звено это - адрес звена, следовательно, ссылка представляет собой 32-х разрядный адрес.

Под каждое звено отводим несколько соседних байтов, в первых из которых размещаем элемент списка ELEM, в остальных ссылку на следующее звено NEXT.

|  |  |
| --- | --- |
| **ELEM** | **NEXT** |

Размер звена зависит от элементов списка. Для определенности, будем считать, что элементами списка являются данные размером в слово. Поэтому на элемент отводим два байта, а поскольку ссылка занимает 4 байта, то для хранения звена нам потребуется 6 байтов.

Звенья списка удобно рассматривать как структуру:

struc node

.elem: resw 1

.next: resd 1

endstruc

Для пустой ссылки NIL, обычно используется адрес 0.

NIL equ 0

Ссылку на первое звено списка будем хранить в переменной list в секции данных.

list dd NIL ; список изначально пуст.

***Операции над списками***

При работе со списком приходится последовательно рассматривать его звенья. Поэтому нам надо знать адрес текущего звена, которое обрабатывается в данный момент. Для его хранения, будем использовать, например, регистр ebx.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **List** |  |  |  |  |  | **…** |  |  |  |  | **…** |  |  | NIL |
|  |  |  |  |  |  | |  |  |  |  |  |  |  |  |
|  |  |  |  |  | **ebx** |  |  |  |  |  |  |  |  |  |

Если нужно сделать текущим какое-то звено, то мы должны занести его **адрес в ebx.**

* **Обращение к элементу текущего звена**

[ebx+node.elem]

* **Переход к следующему звену списка**

mov ebx,[ebx+node.next]

* **Проверка на конец списка**

cmp ebx,NIL

je end

………..

end:

…………

* **Поиск элемента в списке (co значением, из ячейки a)**

mov ax,[a] ; в ax значение переменной a

mov ebx,[list] ; в ebx адрес первого звена

m: cmp ebx,NIL

je end ; переход на печать об отсутствии такого елемента

cmp ax,[ebx+node.elem]

je m1

mov ebx,[ebx+node.next]

jmp m

m1:

…………… ; у нас в ebx – адрес искомого звена

…………….

jmp m2

end:

PRINT “no elem”

m2:

FINISH

* **Вставка звена в начало списка**

для выделения места для нового звена, воспользуемся процедурой NEW, которую опишем чуть позже. Эта процедура отыскивает в куче свободное место для нового звена и его адрес возвращает в регистре edi.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **List** |  |  |  |  |  |
|  |  |  |  |  |  | |  |  |  |  |  |  |  |  |
| **edi** | |  | **X** |  |

* **Удаление звена из списка**

Требуется удалить из списка звено, следующее за звеном, адрес которого записан в ebx.

Для этого надо изменить ссылку в звене, адрес которого указан в ebx на адрес, который указан в поле next звена, которое необходимо удалить.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **List** | **`** |  | **……** |  |  |  |  |  |  |  |  |  |  | **……** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **ebx** |  |  | **edi** |  |  |  |  |  |  |  |  |

Однако, поменять лишь ссылку недостаточно. Необходимо освободить место в памяти, которое занимало удаленное звено. Это делает процедура dispose, которую мы рассмотрим позже. Эта процедура освобождает память в куче, которая была занята звеном, адрес которого мы передаем ей в регистре edi.

***Организация кучи***

Как уже говорилось, для размещения всех звеньев списка, необходимо отвести специальную область памяти, называемую ***кучей.*** При выполнении программы одни звенья кучи оказываются занятыми, другие - пока свободны. Поскольку, программа может то “захватывать” ячейки кучи, то ”отказываться” от них, то занятые и свободные ячейки кучи идут вперемежку. Рассмотрим как в таком случае определять какие ячейки свободны, а какие заняты.

Для этого надо все свободные ячейки объединить в один список, который называется список свободной памяти ( ССП ). На начало этого списка указывает некоторая фиксированная ячейка с именем ***heap\_ptr*** (указатель кучи). Таким образом, кроме тех списков, с которым работает программа, есть еще один список. ССП используется так: когда программе нужно новое звено, оно выделяется из ССП, а когда программа отказывается от звена, оно добавляется в ССП.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **heap\_ptr** |  |  |  |  |  |  |  |  | **….** |  |  | NIL |

Вообще говоря, heap\_ptr можно размещать где угодно, мы отведем место в секции .bss

**Инициализация кучи**

После выделения необходимого объема памяти, все ячейки кучи свободны, они не объединены в ССП. Поэтому, сначала нужно связать все ячейки (звенья) кучи в единый список и записать в heap\_ptr ссылку на начало списка.

Свяжем звенья сверху вниз, чтобы верхняя ячейка кучи была первым звеном, а нижняя – последним.

|  |  |  |
| --- | --- | --- |
| **heap\_ptr** |  |  |
|  |  |  |
| **+6** |  |  |
|  | **……………** |  |
|  |  |  |
| **+6\*k-6** | NIL |  |

Опишем процедуру **init\_heap:**

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; init\_heap ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

init\_heap:

xor ebx,ebx

mov eax,45

int 0x80

add eax,k\*6

mov ebx,eax

mov eax,45

int 0x80

sub eax,6

mov ecx,heap\_size

mov ebx,NIL

l:mov [eax+node.next],ebx

mov ebx,eax

sub eax,6

loop l

mov [heap\_ptr],ebx

ret

Обратиться к этой процедуре нужно в самом начале программы, а затем уже использовать процедуры new и dispose.

***Процедура new***

К этой процедуре программа обращается, когда нужно место для нового звена. Проще всего “отщепить” первое звено и отдать его программе. Адрес этого звена процедура вернет через регистр edi. Обязательно учтем случай, когда ССП пуст. В этом случае выдадим сообщение об ошибке и в edi запишем 0.

***Процедура dispose***

К этой процедуре программа обращается, когда она отказывается от некоторого звена. Адрес этого звена передается процедуре через регистр edi. Это звено становится свободным и его проще всего вставить в начало ССП.

***Процедура печати списка***

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; print\_spis ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

print\_spis:

mov ebx,[list]

l4: cmp ebx,NIL

je pr\_end

xor eax,eax

mov ax,[ebx+node.elem]

UNSINT eax

PUTCHAR 0xa

mov ebx,[ebx+node.next]

jmp l4

pr\_end:

ret

***Основная программа***

%include "st\_io.inc"

global \_start

struc node

.elem: resw 1

.next: resd 1

endstruc

section .bss

heap\_ptr resd 1

section .data

NIL equ 0

k equ 5

heap\_size equ k

list dd NIL

section .text

\_start:

FINISH