**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по практической работе №3**

**по дисциплине «Операционные системы»**

**Тема**: **Исследование организации управления основной памятью**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8381 |  | Муковский Д.В. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2020

**Цель работы.**

Для исследования организации управления памятью необходимо ориентироваться на тип основной памяти, реализованный в компьютере и способ организации, принятый в ОС. В лабораторной работе рассматривается нестраничная память и способ управления динамическими разделами. Для реализации управления памятью в этом случае строится список занятых и свободных участков памяти. Функции ядра, обеспечивающие управление основной памятью, просматривают и преобразуют этот список.

В лабораторной работе исследуются структуры данных и работа функций управления памятью ядра операционной системы.

**Основные теоретические положения.**

Учет занятой и свободной памяти ведется при помощи списка блоков управления памятью MCB (Memory Control Block). MCB занимает 16 байт (параграф) и располагается всегда с адреса кратного 16 (адрес сегмента ОП) и находится в адресном пространстве непосредственно перед тем участком памяти, которым он управляет.

MCB имеет следующую структуру:

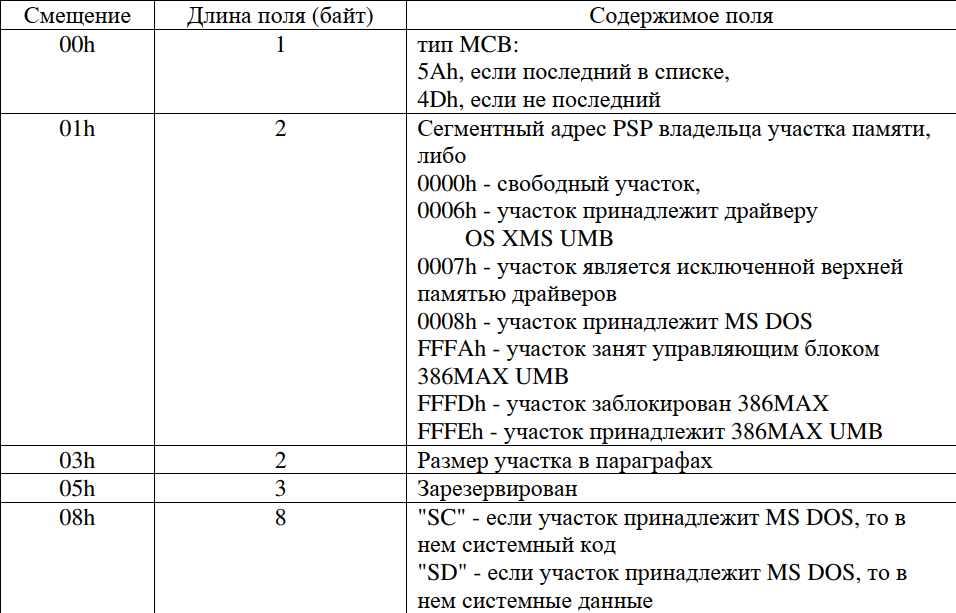
****

Рисунок 1 – Структура MCB

По сегментному адресу и размеру участка памяти, контролируемого этим MCB можно определить местоположение следующего MCB в списке.

Адрес первого MCB хранится во внутренней структуре MS DOS, называемой "List of Lists" (список списков). Доступ к указателю на эту структуру можно получить, используя функцию 52h "Get List of Lists" int 21h. В результате выполнения этой функции ES:BX будет указывать на список списков. Слово по адресу ES:[BX-2] и есть адрес самого первого MCB.

Размер расширенной памяти находится в ячейках 30h, 31h CMOS. CMOS это энергонезависимая память, в которой хранится информация о конфигурации ПЭВМ. Объем памяти составляет 64 байта. Размер расширенной памяти в Кбайтах можно определить, обращаясь к ячейкам CMOS следующим образом:

mov AL,30h ; запись адреса ячейки CMOS

out 70h,AL

in AL,71h ; чтение младшего байта

mov BL,AL ; размера расширенной памяти

mov AL,31h ; запись адреса ячейки CMOS

out 70h,AL

in AL,71h ; чтение старшего байта размера расширенной памяти

**Выполнение работы.**

Написан текст исходного .COM модуля, который выбирает и распечатывает следующую информацию:

* Количество доступной памяти (в килобайтах)
* Размер расширенной памяти (в килобайтах)
* Выводит цепочку блоков управления памятью

Полученный исходный модуль был отлажен. Результаты выполнения программы представлены на рис. 2.

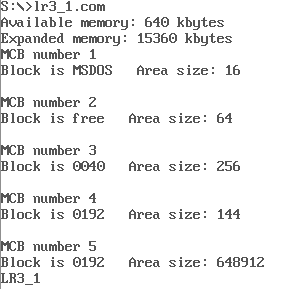


Рисунок 2 – Вывод программы lr3\_1.com

Программа занимает всю доступную память.

Далее в программу была внесена процедура, которая освобождала всю память, которую программа не занимает. Была использована функция 4Ah прерывания 21h. Результат выполнения программы представлен на рис. 3.

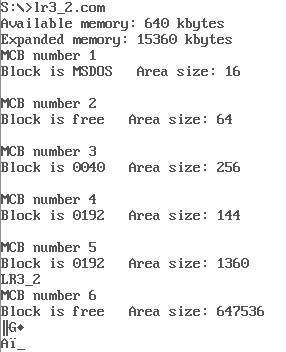


Рисунок 3 – Вывод программы lr3\_2.com

В данном случае была высвобождена память. Как видно из скриншота, освобожденная память относится к шестому блоку управления памятью, который является свободным.

Далее в программу была внесена процедура, которая запрашивает 64Кб памяти (функцией 48h прерывания 21h) после освобождения памяти, которую она не занимает. Результат выполнения программы представлен на рис. 4.

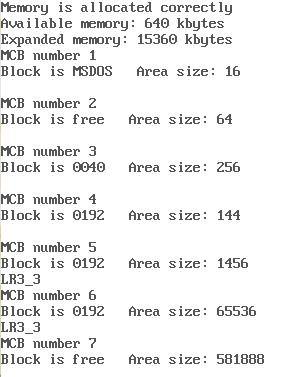


Рисунок 4 – Вывод программы lr3\_3.com

В данном случае мы сначала выделяем всю доступную память, потом освобождаем то, что не нужно. Затем запрашиваем блок памяти 64 Кб, в итоге система выделяет нам ещё 65536 б памяти и программа выводит в консоль, что память выделена корректно. Сегментный адрес PSP владельца участка памяти 5 и 6 блока совпадают.

Далее программа была изменена таким образом, чтобы сначала она запрашивала дополнительно 64Кб, а затем освобождала память. Результат выполнения программы представлен на рис. 5.

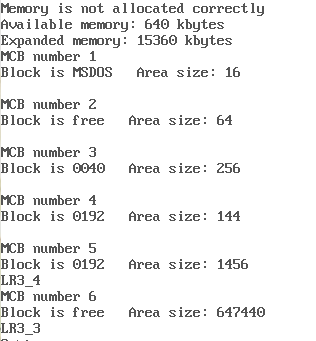


Рисунок 5 – Вывод программы lr3\_4.com

В результате возникает ошибка и дополнительный блок памяти в 64Кб не выделяется, о чем нам сообщает программа первой строчкой. Это происходит так как вся доступная память уже была выделена.

**Контрольные вопросы**

* **Что означает «доступный объём памяти»?**

Максимальный объем памяти, который может использовать программа.

* **Где MCB блок Вашей программы в списке?**

Принадлежность MCB можно определить по сегментной компоненте адреса владельца блок.

* В lr3\_1.com, lr3\_2.com и lr3\_4.com MCB блок программы четвертый и пятый.
* В lr3\_4.com MCB блок программы четвертый, пятый и шестой, так как во время работы программы выделяется дополнительно 64Кб.
* **Какой размер памяти занимает программа в каждом случае?**
* Lr3\_1.com

Программа занимает (648912Б + 144Б) 649056 байт.

* Lr3\_2.com

Программа занимает (1360Б + 144Б) 1504 байт после освобождения памяти.

* Lr3\_3.com

Программа занимает (1376Б + 144Б) 1520 байт после освобождения памяти и дополнительно 65536 байт после выделения.

* Lr3\_4.com

Программа занимает (1376Б + 144Б) 1520 байт.

**Вывод.**

В ходе выполнения данной лабораторной работы был изучен список блоков управления памятью, а также методы выделения и освобождения памяти для программы.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММЫ. LR3\_1.ASM**

LAB SEGMENT

ASSUME CS:LAB, DS:LAB, ES:NOTHING, SS:NOTHING

ORG 100H

MAIN: JMP BEGIN

;данные

AVAILABLE\_MEMORY db 'Available memory: $'

EXPENDED\_MEMORY db 'Expanded memory: $'

MCB\_NUM\_INFO db 'MCB number $'

AREA\_SIZE\_INFO db ' Area size: $'

END\_LINE db 0Dh, 0Ah, "$"

KBYTES db ' kbytes', 13,10,'$'

BLOCK db 0Dh, 0Ah, 'Block is $'

FREE db 'free$'

XMS db 'OS XMS UMB$'

DRIVER\_TOP db 'Excluded top memory of driver$'

DOS db 'MSDOS$'

OCCUP\_386 db '386MAX UMB$'

BLOCK\_386 db '386MAX$'

BELONG\_386 db '386MAX UMB$'

;--------------------------------------------------------------------------------

PRINT PROC NEAR

PUSH AX

MOV AH, 09H

INT 21H

POP AX

RET

PRINT ENDP

;--------------------------------------------------------------------------------

PRINT\_KBYTES PROC NEAR

PUSH DX

MOV DX, OFFSET KBYTES

CALL PRINT

POP DX

RET

PRINT\_KBYTES ENDP

;--------------------------------------------------------------------------------

;ВЫВОДИТ СЛОВО(AX) В 10 С/С

DEC\_WORD\_PRINT PROC

PUSH AX

PUSH CX

PUSH DX

PUSH BX

MOV BX, 10

XOR CX, CX

;В ЦИКЛЕ ДЕЛИМ ЧИСЛО(AX) НА 10(BX) И ОСТАТКИ ОТ ДЕЛЕНИЯ(DX) ЗАНОСИМ В СТЕК

GET\_NUMBERS:

DIV BX

PUSH DX

XOR DX, DX

INC CX

CMP AX, 0

JNZ GET\_NUMBERS

;В ЦИКЛЕ ДОСТАЕМ ИЗ СТЕКА ЧИСЛА В 10 С/С И ВЫВОДИМ

WRITING:

POP DX

OR DL, 48 ;СДВИГ В ASCII ДО ЦИФР

MOV AH, 2

INT 21H

LOOP WRITING

POP BX

POP DX

POP CX

POP AX

RET

DEC\_WORD\_PRINT ENDP

;--------------------------------------------------------------------------------

;ВЫВОДИТ БАЙТ(AL) В 16 С/С

HEX\_BYTE\_PRINT PROC NEAR

PUSH AX

PUSH BX

PUSH DX

MOV AH, 0

MOV BL, 10H

DIV BL

MOV DX, AX ;В DL - ПЕРВАЯ ЦИФРА В DH - ВТОРАЯ

MOV AH, 02H

CMP DL, 0AH

JL PRINT\_1 ;ЕСЛИ В DL - ЦИФРА

ADD DL, 7 ;СДВИГ В ASCII С ЦИФР ДО БУКВ

PRINT\_1:

ADD DL, 48

INT 21H

MOV DL, DH

CMP DL, 0AH

JL PRINT\_2

ADD DL, 7

PRINT\_2:

ADD DL, 48

INT 21H;

POP DX

POP BX

POP AX

RET

HEX\_BYTE\_PRINT ENDP

;--------------------------------------------------------------------------------

;ВЫВОДИТ СЛОВО(AX) В 16 С/С

HEX\_WORD\_PRINT PROC NEAR

PUSH AX

PUSH DX

MOV DX,AX

MOV AL,DH

CALL HEX\_BYTE\_PRINT

MOV AL,DL

CALL HEX\_BYTE\_PRINT

POP DX

POP AX

RET

HEX\_WORD\_PRINT ENDP

;--------------------------------------------------------------------------------

BEGIN:

PUSH AX

PUSH BX

PUSH CX

PUSH DX

PUSH ES

PUSH SI

PRINT\_AVAILABLE\_MEMORY:

XOR AX, AX

INT 12H

MOV DX, OFFSET AVAILABLE\_MEMORY

CALL PRINT

XOR DX, DX

CALL DEC\_WORD\_PRINT

CALL PRINT\_KBYTES

PRINT\_EXPENDED\_MEMORY:

MOV AL, 30H

OUT 70H, AL

IN AL, 71H

MOV BL, AL

MOV AL, 31H

OUT 70H, AL

IN AL, 71H

MOV AH, AL

MOV AL, BL

MOV DX, OFFSET EXPENDED\_MEMORY

CALL PRINT

XOR DX, DX

CALL DEC\_WORD\_PRINT

CALL PRINT\_KBYTES

PRINT\_MCB:

MOV AH, 52H

INT 21H

MOV AX, ES:[BX-2] ;АДРЕСС ПЕРВОГО MCB

MOV ES, AX

XOR CX, CX

NEXT\_MCB:

INC CX

MOV DX, OFFSET MCB\_NUM\_INFO

PUSH CX

CALL PRINT

MOV AX, CX

XOR DX, DX

CALL DEC\_WORD\_PRINT ;ВЫВОДИТ НОМЕР ТЕКУЩЕГО MCB

START:

MOV DX, OFFSET BLOCK

CALL PRINT

XOR AX, AX

MOV AL, ES:[0H]

PUSH AX

MOV AX, ES:[1H]

CMP AX, 0H

JE FREE\_PRINT

CMP AX, 6H

JE XMS\_PRINT

CMP AX, 7H

JE DRIVER\_TOP\_PRINT

CMP AX, 8H

JE DOS\_PRINT

CMP AX, 0FFFAH

JE OCCUP\_386\_PRINT

CMP AX, 0FFFDH

JE BLOCK\_386\_PRINT

CMP AX, 0FFFEH

JE BELONG\_386\_PRINT

XOR DX, DX

CALL HEX\_WORD\_PRINT

JMP AREA\_SIZE\_START

FREE\_PRINT:

MOV DX, OFFSET FREE

JMP PRINTING

XMS\_PRINT:

MOV DX, OFFSET XMS

JMP PRINTING

DRIVER\_TOP\_PRINT:

MOV DX, OFFSET DRIVER\_TOP

JMP PRINTING

DOS\_PRINT:

MOV DX, OFFSET DOS

JMP PRINTING

OCCUP\_386\_PRINT:

MOV DX, OFFSET OCCUP\_386

JMP PRINTING

BLOCK\_386\_PRINT:

MOV DX, OFFSET BLOCK\_386

JMP PRINTING

BELONG\_386\_PRINT:

MOV DX, OFFSET BELONG\_386

PRINTING:

CALL PRINT

AREA\_SIZE\_START:

MOV DX, OFFSET AREA\_SIZE\_INFO

CALL PRINT

MOV AX, ES:[3H]

MOV BX, 10H

MUL BX

CALL DEC\_WORD\_PRINT

MOV CX, 8

XOR SI, SI

MOV DX, OFFSET END\_LINE

CALL PRINT

LAST\_BYTES\_START:

MOV DL, ES:[SI + 8H]

MOV AH, 02H

INT 21H

INC SI

LOOP LAST\_BYTES\_START

MOV AX, ES:[3H]

MOV BX, ES

ADD BX, AX

INC BX

MOV ES, BX

POP AX

POP CX

CMP AL, 5AH

JE EXIT

MOV DX, OFFSET END\_LINE

CALL PRINT

JMP NEXT\_MCB

EXIT:

POP SI

POP ES

POP DX

POP CX

POP BX

POP AX

XOR AL, AL

MOV AH, 4CH

INT 21H

RET

LAB ENDS

END MAIN

**ИСХОДНЫЙ КОД ПРОГРАММЫ. LR3\_2.ASM**

LAB SEGMENT

ASSUME CS:LAB, DS:LAB, ES:NOTHING, SS:NOTHING

ORG 100H

MAIN: JMP BEGIN

;данные

AVAILABLE\_MEMORY db 'Available memory: $'

EXPENDED\_MEMORY db 'Expanded memory: $'

MCB\_NUM\_INFO db 'MCB number $'

AREA\_SIZE\_INFO db ' Area size: $'

END\_LINE db 0Dh, 0Ah, "$"

KBYTES db ' kbytes', 13,10,'$'

BLOCK db 0Dh, 0Ah, 'Block is $'

FREE db 'free$'

XMS db 'OS XMS UMB$'

DRIVER\_TOP db 'Excluded top memory of driver$'

DOS db 'MSDOS$'

OCCUP\_386 db '386MAX UMB$'

BLOCK\_386 db '386MAX$'

BELONG\_386 db '386MAX UMB$'

last db ?

;--------------------------------------------------------------------------------

PRINT PROC NEAR

PUSH AX

MOV AH, 09H

INT 21H

POP AX

RET

PRINT ENDP

;--------------------------------------------------------------------------------

PRINT\_KBYTES PROC NEAR

PUSH DX

MOV DX, OFFSET KBYTES

CALL PRINT

POP DX

RET

PRINT\_KBYTES ENDP

;--------------------------------------------------------------------------------

;ВЫВОДИТ СЛОВО(AX) В 10 С/С

DEC\_WORD\_PRINT PROC

PUSH AX

PUSH CX

PUSH DX

PUSH BX

MOV BX, 10

XOR CX, CX

;В ЦИКЛЕ ДЕЛИМ ЧИСЛО(AX) НА 10(BX) И ОСТАТКИ ОТ ДЕЛЕНИЯ(DX) ЗАНОСИМ В СТЕК

GET\_NUMBERS:

DIV BX

PUSH DX

XOR DX, DX

INC CX

CMP AX, 0

JNZ GET\_NUMBERS

;В ЦИКЛЕ ДОСТАЕМ ИЗ СТЕКА ЧИСЛА В 10 С/С И ВЫВОДИМ

WRITING:

POP DX

OR DL, 48 ;СДВИГ В ASCII ДО ЦИФР

MOV AH, 2

INT 21H

LOOP WRITING

POP BX

POP DX

POP CX

POP AX

RET

DEC\_WORD\_PRINT ENDP

;--------------------------------------------------------------------------------

;ВЫВОДИТ БАЙТ(AL) В 16 С/С

HEX\_BYTE\_PRINT PROC NEAR

PUSH AX

PUSH BX

PUSH DX

MOV AH, 0

MOV BL, 10H

DIV BL

MOV DX, AX ;В DL - ПЕРВАЯ ЦИФРА В DH - ВТОРАЯ

MOV AH, 02H

CMP DL, 0AH

JL PRINT\_1 ;ЕСЛИ В DL - ЦИФРА

ADD DL, 7 ;СДВИГ В ASCII С ЦИФР ДО БУКВ

PRINT\_1:

ADD DL, 48

INT 21H

MOV DL, DH

CMP DL, 0AH

JL PRINT\_2

ADD DL, 7

PRINT\_2:

ADD DL, 48

INT 21H;

POP DX

POP BX

POP AX

RET

HEX\_BYTE\_PRINT ENDP

;--------------------------------------------------------------------------------

;ВЫВОДИТ СЛОВО(AX) В 16 С/С

HEX\_WORD\_PRINT PROC NEAR

PUSH AX

PUSH DX

MOV DX,AX

MOV AL,DH

CALL HEX\_BYTE\_PRINT

MOV AL,DL

CALL HEX\_BYTE\_PRINT

POP DX

POP AX

RET

HEX\_WORD\_PRINT ENDP

;--------------------------------------------------------------------------------

FREE\_MEMORY PROC

PUSH AX

PUSH BX

PUSH CX

MOV BX, OFFSET PROGRAM\_END

ADD BX, 100H

MOV CL, 4

SHR BX, CL

ADD BX, 17

MOV AH, 4AH

INT 21H

POP CX

POP BX

POP AX

RET

FREE\_MEMORY ENDP

;--------------------------------------------------------------------------------

BEGIN:

PUSH AX

PUSH BX

PUSH CX

PUSH DX

PUSH ES

PUSH SI

call FREE\_MEMORY

PRINT\_AVAILABLE\_MEMORY:

XOR AX, AX

INT 12H

MOV DX, OFFSET AVAILABLE\_MEMORY

CALL PRINT

XOR DX, DX

CALL DEC\_WORD\_PRINT

CALL PRINT\_KBYTES

PRINT\_EXPENDED\_MEMORY:

MOV AL, 30H

OUT 70H, AL

IN AL, 71H

MOV BL, AL

MOV AL, 31H

OUT 70H, AL

IN AL, 71H

MOV AH, AL

MOV AL, BL

MOV DX, OFFSET EXPENDED\_MEMORY

CALL PRINT

XOR DX, DX

CALL DEC\_WORD\_PRINT

CALL PRINT\_KBYTES

PRINT\_MCB:

MOV AH, 52H

INT 21H

MOV AX, ES:[BX-2] ;АДРЕСС ПЕРВОГО MCB

MOV ES, AX

XOR CX, CX

NEXT\_MCB:

INC CX

MOV DX, OFFSET MCB\_NUM\_INFO

PUSH CX

CALL PRINT

MOV AX, CX

XOR DX, DX

CALL DEC\_WORD\_PRINT ;ВЫВОДИТ НОМЕР ТЕКУЩЕГО MCB

START:

MOV DX, OFFSET BLOCK

CALL PRINT

XOR AX, AX

MOV AL, ES:[0H]

PUSH AX

MOV AX, ES:[1H]

CMP AX, 0H

JE FREE\_PRINT

CMP AX, 6H

JE XMS\_PRINT

CMP AX, 7H

JE DRIVER\_TOP\_PRINT

CMP AX, 8H

JE DOS\_PRINT

CMP AX, 0FFFAH

JE OCCUP\_386\_PRINT

CMP AX, 0FFFDH

JE BLOCK\_386\_PRINT

CMP AX, 0FFFEH

JE BELONG\_386\_PRINT

XOR DX, DX

CALL HEX\_WORD\_PRINT

JMP AREA\_SIZE\_START

FREE\_PRINT:

MOV DX, OFFSET FREE

JMP PRINTING

XMS\_PRINT:

MOV DX, OFFSET XMS

JMP PRINTING

DRIVER\_TOP\_PRINT:

MOV DX, OFFSET DRIVER\_TOP

JMP PRINTING

DOS\_PRINT:

MOV DX, OFFSET DOS

JMP PRINTING

OCCUP\_386\_PRINT:

MOV DX, OFFSET OCCUP\_386

JMP PRINTING

BLOCK\_386\_PRINT:

MOV DX, OFFSET BLOCK\_386

JMP PRINTING

BELONG\_386\_PRINT:

MOV DX, OFFSET BELONG\_386

PRINTING:

CALL PRINT

AREA\_SIZE\_START:

MOV DX, OFFSET AREA\_SIZE\_INFO

CALL PRINT

MOV AX, ES:[3H]

MOV BX, 10H

MUL BX

CALL DEC\_WORD\_PRINT

MOV CX, 8

XOR SI, SI

MOV DX, OFFSET END\_LINE

CALL PRINT

LAST\_BYTES\_START:

MOV DL, ES:[SI + 8H]

MOV AH, 02H

INT 21H

INC SI

LOOP LAST\_BYTES\_START

MOV AX, ES:[3H]

MOV BX, ES

ADD BX, AX

INC BX

MOV ES, BX

POP AX

POP CX

CMP AL, 5AH

JE EXIT

MOV DX, OFFSET END\_LINE

CALL PRINT

JMP NEXT\_MCB

EXIT:

POP SI

POP ES

POP DX

POP CX

POP BX

POP AX

XOR AL, AL

MOV AH, 4CH

INT 21H

RET

PROGRAM\_END:

LAB ENDS

END MAIN

**ИСХОДНЫЙ КОД ПРОГРАММЫ. LR3\_3.ASM**

LAB SEGMENT

ASSUME CS:LAB, DS:LAB, ES:NOTHING, SS:NOTHING

ORG 100H

MAIN: JMP BEGIN

;данные

AVAILABLE\_MEMORY db 'Available memory: $'

EXPENDED\_MEMORY db 'Expanded memory: $'

MCB\_NUM\_INFO db 'MCB number $'

AREA\_SIZE\_INFO db ' Area size: $'

END\_LINE db 0Dh, 0Ah, "$"

KBYTES db ' kbytes', 13,10,'$'

BLOCK db 0Dh, 0Ah, 'Block is $'

FREE db 'free$'

XMS db 'OS XMS UMB$'

DRIVER\_TOP db 'Excluded top memory of driver$'

DOS db 'MSDOS$'

OCCUP\_386 db '386MAX UMB$'

BLOCK\_386 db '386MAX$'

BELONG\_386 db '386MAX UMB$'

MEMORY\_SUCCESS db 'Memory is allocated correctly',13,10,'$'

MEMORY\_FAIL db 'Memory is not allocated correctly',13,10,'$'

;--------------------------------------------------------------------------------

PRINT PROC NEAR

PUSH AX

MOV AH, 09H

INT 21H

POP AX

RET

PRINT ENDP

;--------------------------------------------------------------------------------

PRINT\_KBYTES PROC NEAR

PUSH DX

MOV DX, OFFSET KBYTES

CALL PRINT

POP DX

RET

PRINT\_KBYTES ENDP

;--------------------------------------------------------------------------------

;ВЫВОДИТ СЛОВО(AX) В 10 С/С

DEC\_WORD\_PRINT PROC

PUSH AX

PUSH CX

PUSH DX

PUSH BX

MOV BX, 10

XOR CX, CX

;В ЦИКЛЕ ДЕЛИМ ЧИСЛО(AX) НА 10(BX) И ОСТАТКИ ОТ ДЕЛЕНИЯ(DX) ЗАНОСИМ В СТЕК

GET\_NUMBERS:

DIV BX

PUSH DX

XOR DX, DX

INC CX

CMP AX, 0

JNZ GET\_NUMBERS

;В ЦИКЛЕ ДОСТАЕМ ИЗ СТЕКА ЧИСЛА В 10 С/С И ВЫВОДИМ

WRITING:

POP DX

OR DL, 48 ;СДВИГ В ASCII ДО ЦИФР

MOV AH, 2

INT 21H

LOOP WRITING

POP BX

POP DX

POP CX

POP AX

RET

DEC\_WORD\_PRINT ENDP

;--------------------------------------------------------------------------------

;ВЫВОДИТ БАЙТ(AL) В 16 С/С

HEX\_BYTE\_PRINT PROC NEAR

PUSH AX

PUSH BX

PUSH DX

MOV AH, 0

MOV BL, 10H

DIV BL

MOV DX, AX ;В DL - ПЕРВАЯ ЦИФРА В DH - ВТОРАЯ

MOV AH, 02H

CMP DL, 0AH

JL PRINT\_1 ;ЕСЛИ В DL - ЦИФРА

ADD DL, 7 ;СДВИГ В ASCII С ЦИФР ДО БУКВ

PRINT\_1:

ADD DL, 48

INT 21H

MOV DL, DH

CMP DL, 0AH

JL PRINT\_2

ADD DL, 7

PRINT\_2:

ADD DL, 48

INT 21H;

POP DX

POP BX

POP AX

RET

HEX\_BYTE\_PRINT ENDP

;--------------------------------------------------------------------------------

;ВЫВОДИТ СЛОВО(AX) В 16 С/С

HEX\_WORD\_PRINT PROC NEAR

PUSH AX

PUSH DX

MOV DX,AX

MOV AL,DH

CALL HEX\_BYTE\_PRINT

MOV AL,DL

CALL HEX\_BYTE\_PRINT

POP DX

POP AX

RET

HEX\_WORD\_PRINT ENDP

;--------------------------------------------------------------------------------

FREE\_MEMORY PROC

PUSH AX

PUSH BX

PUSH CX

MOV BX, OFFSET PROGRAM\_END

ADD BX, 100H

MOV CL, 4

SHR BX, CL

ADD BX, 17

MOV AH, 4AH

INT 21H

POP CX

POP BX

POP AX

RET

FREE\_MEMORY ENDP

;--------------------------------------------------------------------------------

ADD\_MEMORY PROC

PUSH AX

PUSH BX

PUSH DX

MOV BX, 1000H

MOV AH, 48H

INT 21H

JC MEM\_FAIL

MEM\_SUCCESS:

MOV DX,OFFSET MEMORY\_SUCCESS

CALL PRINT

JMP FUNC\_END

MEM\_FAIL:

MOV DX,OFFSET MEMORY\_FAIL

CALL PRINT

FUNC\_END:

POP DX

POP BX

POP AX

RETADD\_MEMORY ENDP

;--------------------------------------------------------------------------------

BEGIN:

PUSH AX

PUSH BX

PUSH CX

PUSH DX

PUSH ES

PUSH SI

call FREE\_MEMORY

call ADD\_MEMORY

PRINT\_AVAILABLE\_MEMORY:

XOR AX, AX

INT 12H

MOV DX, OFFSET AVAILABLE\_MEMORY

CALL PRINT

XOR DX, DX

CALL DEC\_WORD\_PRINT

CALL PRINT\_KBYTES

PRINT\_EXPENDED\_MEMORY:

MOV AL, 30H

OUT 70H, AL

IN AL, 71H

MOV BL, AL

MOV AL, 31H

OUT 70H, AL

IN AL, 71H

MOV AH, AL

MOV AL, BL

MOV DX, OFFSET EXPENDED\_MEMORY

CALL PRINT

XOR DX, DX

CALL DEC\_WORD\_PRINT

CALL PRINT\_KBYTES

PRINT\_MCB:

MOV AH, 52H

INT 21H

MOV AX, ES:[BX-2] ;АДРЕСС ПЕРВОГО MCB

MOV ES, AX

XOR CX, CX

NEXT\_MCB:

INC CX

MOV DX, OFFSET MCB\_NUM\_INFO

PUSH CX

CALL PRINT

MOV AX, CX

XOR DX, DX

CALL DEC\_WORD\_PRINT ;ВЫВОДИТ НОМЕР ТЕКУЩЕГО MCB

START:

MOV DX, OFFSET BLOCK

CALL PRINT

XOR AX, AX

MOV AL, ES:[0H]

PUSH AX

MOV AX, ES:[1H]

CMP AX, 0H

JE FREE\_PRINT

CMP AX, 6H

JE XMS\_PRINT

CMP AX, 7H

JE DRIVER\_TOP\_PRINT

CMP AX, 8H

JE DOS\_PRINT

CMP AX, 0FFFAH

JE OCCUP\_386\_PRINT

CMP AX, 0FFFDH

JE BLOCK\_386\_PRINT

CMP AX, 0FFFEH

JE BELONG\_386\_PRINT

XOR DX, DX

CALL HEX\_WORD\_PRINT

JMP AREA\_SIZE\_START

FREE\_PRINT:

MOV DX, OFFSET FREE

JMP PRINTING

XMS\_PRINT:

MOV DX, OFFSET XMS

JMP PRINTING

DRIVER\_TOP\_PRINT:

MOV DX, OFFSET DRIVER\_TOP

JMP PRINTING

DOS\_PRINT:

MOV DX, OFFSET DOS

JMP PRINTING

OCCUP\_386\_PRINT:

MOV DX, OFFSET OCCUP\_386

JMP PRINTING

BLOCK\_386\_PRINT:

MOV DX, OFFSET BLOCK\_386

JMP PRINTING

BELONG\_386\_PRINT:

MOV DX, OFFSET BELONG\_386

PRINTING:

CALL PRINT

AREA\_SIZE\_START:

MOV DX, OFFSET AREA\_SIZE\_INFO

CALL PRINT

MOV AX, ES:[3H]

MOV BX, 10H

MUL BX

CALL DEC\_WORD\_PRINT

MOV CX, 8

XOR SI, SI

MOV DX, OFFSET END\_LINE

CALL PRINT

LAST\_BYTES\_START:

MOV DL, ES:[SI + 8H]

MOV AH, 02H

INT 21H

INC SI

LOOP LAST\_BYTES\_START

MOV AX, ES:[3H]

MOV BX, ES

ADD BX, AX

INC BX

MOV ES, BX

POP AX

POP CX

CMP AL, 5AH

JE EXIT

MOV DX, OFFSET END\_LINE

CALL PRINT

JMP NEXT\_MCB

EXIT:

POP SI

POP ES

POP DX

POP CX

POP BX

POP AX

XOR AL, AL

MOV AH, 4CH

INT 21H

RET

PROGRAM\_END:

LAB ENDS

END MAIN

**ИСХОДНЫЙ КОД ПРОГРАММЫ. LR3\_4.ASM**

LAB SEGMENT

ASSUME CS:LAB, DS:LAB, ES:NOTHING, SS:NOTHING

ORG 100H

MAIN: JMP BEGIN

;данные

AVAILABLE\_MEMORY db 'Available memory: $'

EXPENDED\_MEMORY db 'Expanded memory: $'

MCB\_NUM\_INFO db 'MCB number $'

AREA\_SIZE\_INFO db ' Area size: $'

END\_LINE db 0Dh, 0Ah, "$"

KBYTES db ' kbytes', 13,10,'$'

BLOCK db 0Dh, 0Ah, 'Block is $'

FREE db 'free$'

XMS db 'OS XMS UMB$'

DRIVER\_TOP db 'Excluded top memory of driver$'

DOS db 'MSDOS$'

OCCUP\_386 db '386MAX UMB$'

BLOCK\_386 db '386MAX$'

BELONG\_386 db '386MAX UMB$'

MEMORY\_SUCCESS db 'Memory is allocated correctly',13,10,'$'

MEMORY\_FAIL db 'Memory is not allocated correctly',13,10,'$'

;--------------------------------------------------------------------------------

PRINT PROC NEAR

PUSH AX

MOV AH, 09H

INT 21H

POP AX

RET

PRINT ENDP

;--------------------------------------------------------------------------------

PRINT\_KBYTES PROC NEAR

PUSH DX

MOV DX, OFFSET KBYTES

CALL PRINT

POP DX

RET

PRINT\_KBYTES ENDP

;--------------------------------------------------------------------------------

;ВЫВОДИТ СЛОВО(AX) В 10 С/С

DEC\_WORD\_PRINT PROC

PUSH AX

PUSH CX

PUSH DX

PUSH BX

MOV BX, 10

XOR CX, CX

;В ЦИКЛЕ ДЕЛИМ ЧИСЛО(AX) НА 10(BX) И ОСТАТКИ ОТ ДЕЛЕНИЯ(DX) ЗАНОСИМ В СТЕК

GET\_NUMBERS:

DIV BX

PUSH DX

XOR DX, DX

INC CX

CMP AX, 0

JNZ GET\_NUMBERS

;В ЦИКЛЕ ДОСТАЕМ ИЗ СТЕКА ЧИСЛА В 10 С/С И ВЫВОДИМ

WRITING:

POP DX

OR DL, 48 ;СДВИГ В ASCII ДО ЦИФР

MOV AH, 2

INT 21H

LOOP WRITING

POP BX

POP DX

POP CX

POP AX

RET

DEC\_WORD\_PRINT ENDP

;--------------------------------------------------------------------------------

;ВЫВОДИТ БАЙТ(AL) В 16 С/С

HEX\_BYTE\_PRINT PROC NEAR

PUSH AX

PUSH BX

PUSH DX

MOV AH, 0

MOV BL, 10H

DIV BL

MOV DX, AX ;В DL - ПЕРВАЯ ЦИФРА В DH - ВТОРАЯ

MOV AH, 02H

CMP DL, 0AH

JL PRINT\_1 ;ЕСЛИ В DL - ЦИФРА

ADD DL, 7 ;СДВИГ В ASCII С ЦИФР ДО БУКВ

PRINT\_1:

ADD DL, 48

INT 21H

MOV DL, DH

CMP DL, 0AH

JL PRINT\_2

ADD DL, 7

PRINT\_2:

ADD DL, 48

INT 21H;

POP DX

POP BX

POP AX

RET

HEX\_BYTE\_PRINT ENDP

;--------------------------------------------------------------------------------

;ВЫВОДИТ СЛОВО(AX) В 16 С/С

HEX\_WORD\_PRINT PROC NEAR

PUSH AX

PUSH DX

MOV DX,AX

MOV AL,DH

CALL HEX\_BYTE\_PRINT

MOV AL,DL

CALL HEX\_BYTE\_PRINT

POP DX

POP AX

RET

HEX\_WORD\_PRINT ENDP

;--------------------------------------------------------------------------------

FREE\_MEMORY PROC

PUSH AX

PUSH BX

PUSH CX

MOV BX, OFFSET PROGRAM\_END

ADD BX, 100H

MOV CL, 4

SHR BX, CL

ADD BX, 17

MOV AH, 4AH

INT 21H

POP CX

POP BX

POP AX

RET

FREE\_MEMORY ENDP

;--------------------------------------------------------------------------------

ADD\_MEMORY PROC

PUSH AX

PUSH BX

PUSH DX

MOV BX, 1000H

MOV AH, 48H

INT 21H

JC MEM\_FAIL

MEM\_SUCCESS:

MOV DX,OFFSET MEMORY\_SUCCESS

CALL PRINT

JMP FUNC\_END

MEM\_FAIL:

MOV DX,OFFSET MEMORY\_FAIL

CALL PRINT

FUNC\_END:

POP DX

POP BX

POP AX

RETADD\_MEMORY ENDP

;--------------------------------------------------------------------------------

BEGIN:

PUSH AX

PUSH BX

PUSH CX

PUSH DX

PUSH ES

PUSH SI

call ADD\_MEMORY

call FREE\_MEMORY

PRINT\_AVAILABLE\_MEMORY:

XOR AX, AX

INT 12H

MOV DX, OFFSET AVAILABLE\_MEMORY

CALL PRINT

XOR DX, DX

CALL DEC\_WORD\_PRINT

CALL PRINT\_KBYTES

PRINT\_EXPENDED\_MEMORY:

MOV AL, 30H

OUT 70H, AL

IN AL, 71H

MOV BL, AL

MOV AL, 31H

OUT 70H, AL

IN AL, 71H

MOV AH, AL

MOV AL, BL

MOV DX, OFFSET EXPENDED\_MEMORY

CALL PRINT

XOR DX, DX

CALL DEC\_WORD\_PRINT

CALL PRINT\_KBYTES

PRINT\_MCB:

MOV AH, 52H

INT 21H

MOV AX, ES:[BX-2] ;АДРЕСС ПЕРВОГО MCB

MOV ES, AX

XOR CX, CX

NEXT\_MCB:

INC CX

MOV DX, OFFSET MCB\_NUM\_INFO

PUSH CX

CALL PRINT

MOV AX, CX

XOR DX, DX

CALL DEC\_WORD\_PRINT ;ВЫВОДИТ НОМЕР ТЕКУЩЕГО MCB

START:

MOV DX, OFFSET BLOCK

CALL PRINT

XOR AX, AX

MOV AL, ES:[0H]

PUSH AX

MOV AX, ES:[1H]

CMP AX, 0H

JE FREE\_PRINT

CMP AX, 6H

JE XMS\_PRINT

CMP AX, 7H

JE DRIVER\_TOP\_PRINT

CMP AX, 8H

JE DOS\_PRINT

CMP AX, 0FFFAH

JE OCCUP\_386\_PRINT

CMP AX, 0FFFDH

JE BLOCK\_386\_PRINT

CMP AX, 0FFFEH

JE BELONG\_386\_PRINT

XOR DX, DX

CALL HEX\_WORD\_PRINT

JMP AREA\_SIZE\_START

FREE\_PRINT:

MOV DX, OFFSET FREE

JMP PRINTING

XMS\_PRINT:

MOV DX, OFFSET XMS

JMP PRINTING

DRIVER\_TOP\_PRINT:

MOV DX, OFFSET DRIVER\_TOP

JMP PRINTING

DOS\_PRINT:

MOV DX, OFFSET DOS

JMP PRINTING

OCCUP\_386\_PRINT:

MOV DX, OFFSET OCCUP\_386

JMP PRINTING

BLOCK\_386\_PRINT:

MOV DX, OFFSET BLOCK\_386

JMP PRINTING

BELONG\_386\_PRINT:

MOV DX, OFFSET BELONG\_386

PRINTING:

CALL PRINT

AREA\_SIZE\_START:

MOV DX, OFFSET AREA\_SIZE\_INFO

CALL PRINT

MOV AX, ES:[3H]

MOV BX, 10H

MUL BX

CALL DEC\_WORD\_PRINT

MOV CX, 8

XOR SI, SI

MOV DX, OFFSET END\_LINE

CALL PRINT

LAST\_BYTES\_START:

MOV DL, ES:[SI + 8H]

MOV AH, 02H

INT 21H

INC SI

LOOP LAST\_BYTES\_START

MOV AX, ES:[3H]

MOV BX, ES

ADD BX, AX

INC BX

MOV ES, BX

POP AX

POP CX

CMP AL, 5AH

JE EXIT

MOV DX, OFFSET END\_LINE

CALL PRINT

JMP NEXT\_MCB

EXIT:

POP SI

POP ES

POP DX

POP CX

POP BX

POP AX

XOR AL, AL

MOV AH, 4CH

INT 21H

RET

PROGRAM\_END:

LAB ENDS

END MAIN