МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ   
РОССИЙСКОЙ ФЕДЕРАЦИИ   
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем

Отчет   
по лабораторной работе №7  
на тему: «Использование стека. Проверка баланса расстановки скобок в строке»  
по дисциплине «Архитектура ЭВМ и систем»

Выполнили: Марочкин М.А. Шифр: 170584  
 Яшин М. О. Шифр: 170133  
 Шорин В.Д. Шифр: 171406  
 Панин М.С. Шифр: 179586  
 Евдокимов Н.А. Шифр: 170576  
ИПАИТ  
Направление: 09.03.04 «Программная инженерия»  
Группа: 71-ПГ  
Проверил(а):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Отметка о зачете:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 Дата «\_\_\_\_» \_\_\_\_\_\_ 2017 г.

Орел, 2017г.

**Цель работы.**

Цели лабораторной работы:

1. Изучение принципов функционирования памяти и микропроцессора компьютера при выполнении операций со стеком и строками.

2. Приобретение навыков использования команд для работы со стеком и строками.

3. Получение представления об особенностях обработки данных, команд и режимах доступа к данным при использовании стека.

**Контрольные вопросы:**

1. Память с последовательным доступом. Виды памяти с последовательным доступом.

Память с последовательным доступом строится либо с использованием продвижения данных по цепочке элементов (по подобию с регистрами сдвига), либо с хранением данных в адресном ЗУ при соответствующем управлении адресом доступа.

Основными представителями такого вида памяти являются видеопамять, буфер FIFO, стек.

2. Определение стека. Организация стека.

Стек – это однонаправленная очередь, в которую данные помещаются и из которой извлекаются строго в определённом порядке. Стековая память обеспечивает такой режим работы, когда информация записывается и считывается по принципу «последним записан – первым считан» (LIFO – Last Input First Output). Такая память используется для временного хранения данных, например, для запоминания и восстановления регистров процессора (контекста) при обработке подпрограмм и прерываний.

Когда слово A заносится в стек, оно располагается в первой свободной ячейке. Каждое следующее записываемое слово перемещает всё содержимое стека на одну ячейку вверх и занимает освободившуюся ячейку. Запись очередного слова после H приводит к переполнению стека, поскольку он рассчитан на 7 слов, и потере кода A.

Считывание информации из стека осуществляется в обратном порядке. Доступ к произвольному коду в стеке формально недопустим до извлечения всех данных, записанных позже.

3. Команды работы со стеком.

Занесение информации в стек называется включением, считывание информации из стека – извлечением. Для работы со стеком существуют две основные операции: добавление элемента в вершину стека (PUSH) и извлечение элемента из вершины стека (POP). Команда PUSH имеет один операнд, который может быть непосредственным значением, 2-байтовым регистром или адресом ячейки памяти. При записи в стек данного сначала производится уменьшение на 2 содержимого указателя стека SP (стек оперирует словами), которое затем используется в качестве адреса ячейки, куда и производится запись. Команда POP также имеет один операнд, который может быть 2-байтовым регистром. При считывании слова из стека в качестве адреса этого данного берётся текущее содержимое указателя стека, а после извлечения данного содержимое SP увеличивается на 2.

4. Какие регистры используются при работе со стеком? Каково их назначение.

Под стек выделяется отдельный сегмент – сегмент стека, начальный адрес которого помещается в соответствующий сегментный регистр – SS. Адресация стека обеспечивается специальным регистром – указателем стека SP, в который предварительно помещается наибольший адрес области основной памяти, отведённой под стек (дно стека). Адрес последнего включённого в стек элемента называется вершиной стека (TOS – Top Of Stack).

**Текст программы.**

s\_s segment stack "stack"

dw 20 dup ($)

s\_s ends

d\_s segment

string db '5+{35]{45/[3+6]})(','$'

rez db 0

d\_s ends

assume ss: s\_s, ds: d\_s, cs: c\_s

c\_s segment

begin:

mov ax, d\_s

mov ds, ax

xor ax, ax

xor cx, cx

lea si, string

newsymbol:

lodsb

inc cx

cmp al, '$'

je endstring

cmp al, '('

je instack

cmp al, '['

je instack

cmp al, '{'

je instack

cmp al, ')'

je outstack

cmp al, ']'

je outstack

cmp al, '}'

je outstack

jmp newsymbol

instack:

inc dx

push ax

jmp newsymbol

outstack:

cmp dx, 0

jne go\_on

mov rez, 3

jmp endstring

go\_on:

dec dx

pop bx

cmp bl, '('

jne nexttry1

cmp al, ')'

je newsymbol

jmp er1

nexttry1:

cmp bl, '['

jne nexttry2

cmp al, ']'

je newsymbol

jmp er1

nexttry2:

cmp bl, '{'

jne er1

cmp al, '}'

je newsymbol

er1:

mov rez, 1

cmp dx, 0

je exit

jmp cleanstack

endstring:

cmp dx, 0

je exit

mov rez, 2

cleanstack:

pop ax

loop cleanstack

cmp result, 1

je skobka0

jmp exit

skobka0:

cmp bl, '('

je skobka1

cmp bl, '['

je skobka2

cmp bl, '{'

je skobka3

jmp exit

skobka1:

mov dl, ')'

jmp exit

skobka2:

mov dl, ']'

jmp exit

skobka3:

mov dl, '}'

jmp exit

exit:

mov cl, rez

mov ah, 4ch

int 21h

c\_s ends

end begin