**Федеральное государственное образовательное учреждение   
высшего образования**

**«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИМЕНИ И.С. ТУРГЕНЕВА»**

**Институт приборостроения, автоматизации и информационных технологий**

**Кафедра информационных систем и цифровых технологий**

Отчет к лабораторной работе № 7

«Использование стека.  
Проверка баланса расстановки скобок в строке»

Выполнила:

Студентка группы 21ПГ

Банных Мария Алексеевна

Приняла:

доцент кафедры   
информационных систем   
и цифровых технологий  
Конюхова Оксана Владимировна  
  
доцент кафедры   
информационных систем   
и цифровых технологий  
Амелина Ольга Викторовна

Орёл 2022 год

Ответы на 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.

Виды:

* видеопамять,
* буфер FIFO,
* стек.

1. *Определение стека. Организация стека.*

***Стек*** – это однонаправленная очередь, данные в которую помещаются и извлекаются в строго 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.

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)

1. *Команды работы со стеком.*

Для работы со стеком существуют две основные операции:

1) добавление элемента в вершину стека (PUSH);

2) 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.

1. *Какие регистры используют при работе со стеком? Каково их назначение?*

Под стек выделяется отдельный сегмент – сегмент стека, начальный адрес которого помещается в соответствующий сегментный регистр – SS(Stack Segment). Адресация стека обеспечивается специальным регистром – указателем стека SP, в который предварительно помещается наибольший адрес области основной памяти, отведённой под стек (дно стека).

Для доступа к элементам не в вершине, а внутри стека используется регистр *BP* – указатель базы кадра стека. Например, при входе в процедуру выполняется передача нужных параметров путем записи их в стек. Если процедура также использует стек, то доступ к этим параметрам становится проблематичным. Выход заключается в том, чтобы после записи параметров в регистр *BP* записать адрес вершины стека *SP*. Значение регистра *SP* в дальнейшем будет изменяться, однако в регистре *BP* хранится адрес, используя который, можно получить доступ к переданным параметрам.

1. *Как изменяется содержимое указателя вершины стека при включении нового элемента в стек и извлечении элемента из стека? Почему?*

При помещении элементов в стек адрес вершины стека (содержимое регистра *SP*) уменьшается (смещается в сторону меньших адресов), а при извлечении элементов из стека – увеличивается (смещается в сторону больших адресов).

Практическая часть:

.model small

steck segment stack "stack"

dw 20 dup('$')

steck ends

data segment

skobki db "(()){}[][](){}[]()(("

result db (?)

data ends

kod segment

assume SS: steck, DS: data, CS: kod

begin:

mov ax, data

mov ds, ax

xor ax, ax

lea si, skobki ;загружаем адрес начала строки в регистр si

elements:

mov al, [si] ;в регистр al помещаем элемент строки

inc si

cmp al, '(' ;сравниваем элемент со скобками

je open ;если открывающаяся, переходим в open

cmp al, '{'

je open

cmp al, '['

je open

cmp al, ')'

je close ;если закрывающаяся, переходим в close

cmp al, '}'

je close

cmp al, ']'

je close

jmp test5

open:

inc cx ;счётчик += 1

push ax ;помещаем скобку в стек

jmp elements ;переходим к следующему элементу строки

close:

cmp cx, 0 ;проверка, пусто ли в стеке

jne test1 ;если нет, то переход в test1

mov result, 3 ;иначе лишние закрывающиеся скобки

jmp elements ;переходим к следующему элементу строки

test1:

dec cx ;уменьшение счётчика на 1

pop bx ;извлечение скобки из стека

cmp bl, '(' ;проверка на совпадение открывающейся и закрывающейся скобок

jne test2 ;если не эта скобка, проверяем другие

cmp al, ')' ;если закрывающаяся скобка

je elements ;проверяем следующий элемент

test2:

cmp bl, '{'

jne test3

cmp al, '}'

je elements

test3:

cmp bl, '['

jne test4

cmp al, ']'

je elements

test4:

mov result, 1 ;несоответствие скобок

cmp cx, 0 ;если стек пуст

je check ;переход в check

test5:

mov result, 2 ;если нет, то не все скобки закрыты

check:

cmp result, 1 ;если ошибка в несоответствии скобок

je searching ;то переход в searching

jmp exit ;иначе на выход

searching:

cmp bl, '(' ;ищем, какой скобки не хватает

je locate1

cmp bl, '{'

je locate2

cmp bl, '['

je locate3

cmp bl, ')'

je locate4

cmp bl, '}'

je locate5

cmp bl, ']'

je locate6

jmp exit

locate1:

mov dl, ')' ;записываем нужную скобку в регистр bl

jmp exit

locate2:

mov dl, '}'

jmp exit

locate3:

mov dl, ']'

jmp exit

locate4:

mov dl, '('

jmp exit

locate5:

mov dl, '{'

jmp exit

locate6:

mov dl, '['

exit:

xor ax, ax

mov al, result ;в регистр al помещаем кд результата проверки

mov ah, 4ch

int 21h

kod ends

end begin

ends