Стек

Стек — это область памяти, специально выделяемая для временного хранения данных.

Для работы со стеком предназначены три регистра:

- **ss** — сегментный регистр стека;
- **sp/esp** — регистр указателя стека;
- **bp/ebp** — регистр указателя базы кадра стека.

В каждый момент времени доступен только один стек, адрес сегмента которого содержится в регистре ss. Этот стек называется *текущим*. Для того чтобы обратиться к другому стеку, необходимо загрузить в регистр ss другой адрес. Регистр ss автоматически используется процессором для выполнения всех команд, работающих со стеком.

Особенности работы со стеком:

- запись и чтение данных в стеке осуществляется в соответствии с принципом **LIFO** (Last In First Out — “последним пришел, первым ушел”);
- по мере записи данных в стек последний растет в сторону младших адресов. Эта особенность заложена в алгоритм команд работы со стеком;
- при использовании регистров **esp/sp** и **ebp/bp** для адресации памяти ассемблер автоматически считает, что содержащиеся в нем значения представляют собой смещения относительно сегментного регистра **ss**.

Регистр **esp/sp** всегда указывает на вершину стека, то есть содержит смещение, по которому в стек был занесен последний элемент. Команды работы со стеком неявно изменяют этот регистр так, чтобы он указывал всегда на последний записанный в стек элемент. Если стек пуст, то значение **esp** равно адресу последнего байта сегмента, выделенного под стек.

При занесении элемента в стек процессор уменьшает значение регистра **esp**, а затем записывает элемент по адресу новой вершины.

При извлечении данных из стека процессор копирует элемент, расположенный по адресу вершины, а затем увеличивает значение регистра указателя стека **esp**. Таким образом, получается, что стек растет вниз, в сторону уменьшения адресов.

Для доступа к элементам внутри стека используется регистр **ebp**.

Например, при входе в подпрограмму передача параметров осуществляется путем записи их в стек. Если подпрограмма тоже активно работает со стеком, то доступ к этим параметрам затруднен. Поэтому после записи данных в стек нужно сохранить адрес вершины стека в регистре **ebp**. Значение в **ebp** в дальнейшем можно использовать для доступа к переданным параметрам.

Стек используется

1. при вызове подпрограмм;
2. для временного сохранения значений регистров;
3. для локальных переменных

Команды для работы со стеком

1. Запись слова в стек

PUSH operand onto the stack

PUSH источник
R16, M16, I16

Алгоритм работы:

sp:= sp-2
ss:[sp] := источник

Выполнение команды не влияет на флаги

Описание:

Уменьшает указатель вершины стека на 2 и записывает значение источника по адресу новой вершины стека. Размер записываемых значений — слово. В стек можно записывать непосредственные значения.

Пример,
push ax
push bx

2. Извлечение слова из стека
POP operand from the stack

POP приемник
R16, M16
~~CS~~

Алгоритм работы:
приемник:=ss:[sp];
sp:=sp+2;
Выполнение команды не влияет на флаги
Описание:
Замещает прежнее значение слова в памяти или в регистре значением из вершины стека, которая адресуется регистрами ss:sp. После чего значение указателя вершины стека увеличивается на 2. Нельзя в качестве приемника использовать сегментный регистр cs.
Пример,

```
push ax
push bx
...
pop bx
pop ax
```

3. Размещение всех регистров общего назначения в стеке
PUSH All general registers onto stack

PUSHA

Алгоритм работы:
Temp:=sp
Push ax
Push cx
Push dx
Push bx
Push temp
Push bp
Push si
Push di.
Описание:
Записывает в стек восемь РОН, уменьшая sp на 16. В стек помещается содержимое sp по состоянию до выполнения команды.
выполнение команды не влияет на флаги
Пример,

```
.386
Pusha
```

4. Извлечение всех регистров общего назначения из стека
POP All general registers from the stack

POPA

Алгоритм работы:
di := Pop
si := Pop
bp := Pop
Pop (пропуск значения, соответствующего sp)

bx := Pop
dx := Pop
cx := Pop
ax := Pop
Описание:
Содержимое **di** восстанавливается первым. Содержимое sp извлекается, но не восстанавливается. Значение указателя стека **sp** увеличивается на 16. Выполнение команды не влияет на флаги. Эту команду можно использовать в процедурах и программах обработки прерываний.

```
.386
    pusha
...
    popa
```

5. Размещение регистра флагов в стеке
PUSH Flags register onto the stack

PUSHF

Алгоритм работы:
sp:= sp-2
ss:[sp] := flags
Выполнение команды не влияет на флаги
Описание:
Значение указателя вершины стека уменьшается на 2 и по новому адресу записывается значение регистра флагов. Команда pushf может использоваться для получения содержимого регистра флагов. Для этого регистр флагов записывается в стек и затем извлекается в обычный регистр.
Команда используется в программах обработки прерываний для сохранения состояния процессора.
Пример,

```
.386
pushf
pop   ax
```

6. Извлечение регистра флагов из стека
POP Flags register from the stack

POPF

Алгоритм работы:
Flags:= ss:[sp]
sp:=sp+2
Описание:
Считывает слово из вершины стека и записывает его значение в регистр флагов.
Значение указателя вершины стека увеличивается на 2
Содержимое регистра флагов:

14	1312	11	10	09	08	07	06	04	02	00
NT	IOPL	OF	DF	IF	TF	SF	ZF	AF	PF	CF
r	R	r	R	r	R	r	r	r	r	r

NT- флаг вложенности задачи
IOPL – уровень привилегий ввода вывода.
Из-за того, что регистр **eflags**/**flags** непосредственно недоступен, команда **popf** является одной из немногих возможностей влияния на его содержимое.

Пример,

```
;установить флаг трассировки
        .386
        pushf
        pop ax
        or ax,100h
        push    ax
        popf
```