Smith College
Computer Science

CSC231 — Assembly

Week #12 — Spring 2017

Dominique Thiébaut
dthiebaut@smith.edu

Passing Parameters by **Reference**: an Example

An Example

```
# example.py
from __future__ import print_function

def incrementAll( array ):
    for i in range( len( array ) ):
        array[ i ] += 1


Table = [1, 2, 3, 4]
print( str( Table ) )

incrementAll( Table )

print( str( Table ) )
```

```
python example.py
[1, 2, 3, 4]
[2, 3, 4, 5]
```

An Example

```python
# example.py
from __future__ import print_function

def incrementAll( array ):
    for i in range( len( array ) ):
        array[ i ] += 1


Table = [1, 2, 3, 4]
print( str( Table ) )

incrementAll( Table )

print( str( Table ) )
```

```
             section .data
Table        dd      1,2,3,4

             section .text
incrementAll:
             push    ebp
             mov     ebp, esp
             push    ebx
             push    ecx
             mov     ecx, 4
             mov     ebx, dword[ebp+8]
.for:        inc     dword[ebx]
             add     ebx, 4
             loop    .for
             pop     ecx
             pop     ebx
             pop     ebp
             ret     4


_Start:
             mov     eax, Table
             push    eax
             call    incrementAll
```

Rule for Writing Functions:

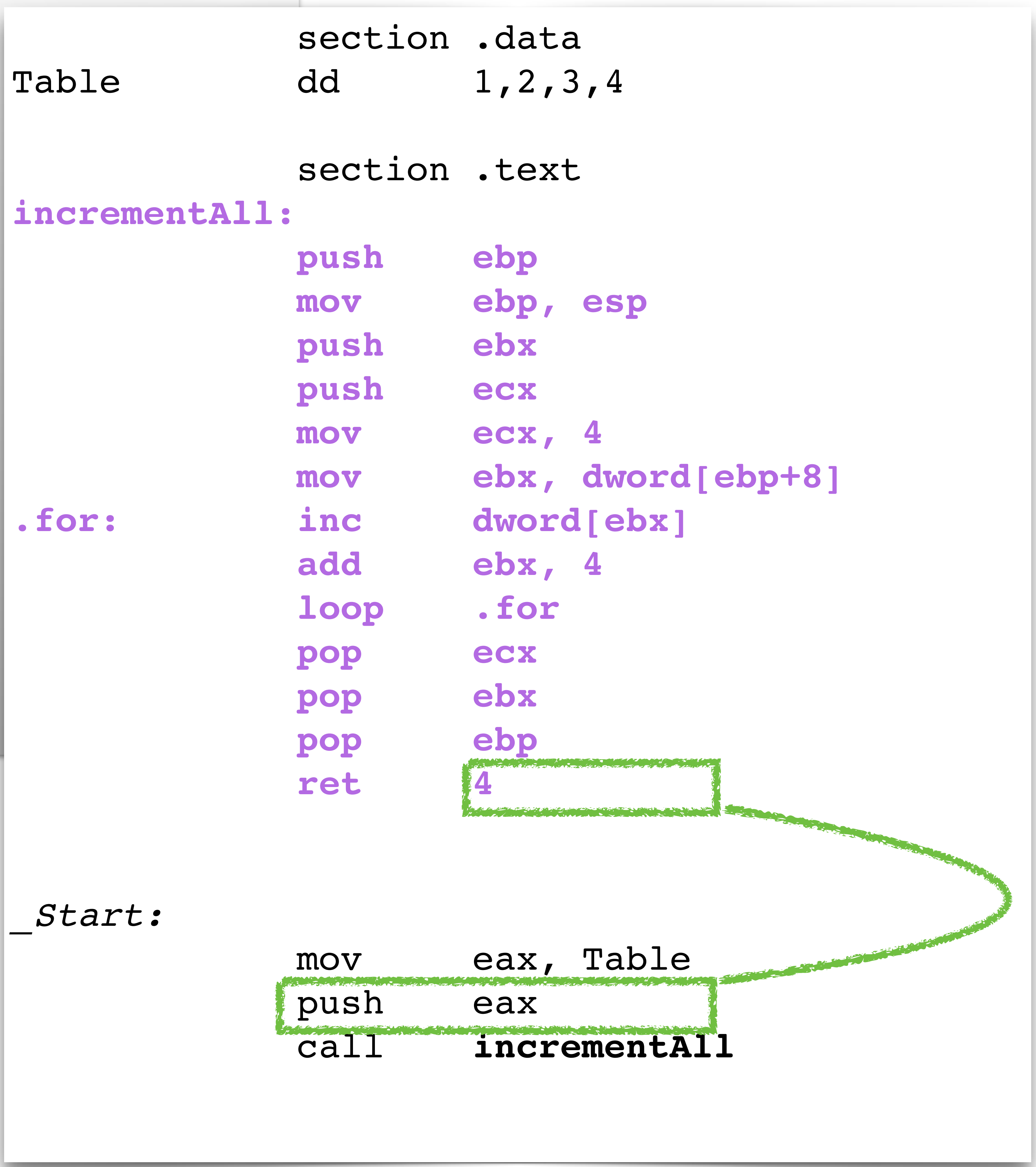
Pushing and *popping* operations into/from the stack must always cancel each other out!

(a + 3 (b^2 + (c+1) ^(d-1)))

Same as with parentheses

An Example

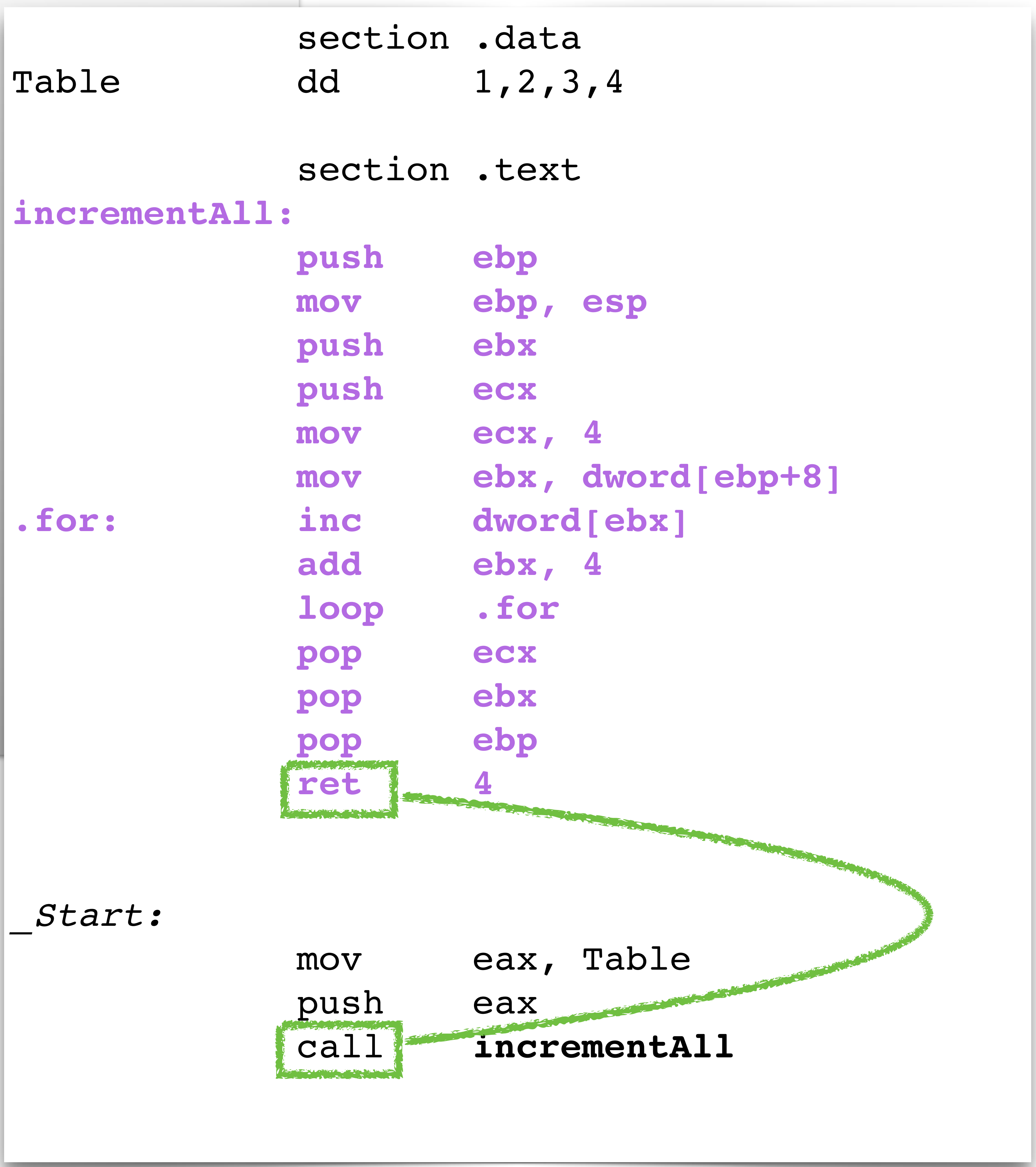
```python
# example.py
from __future__ import print_function

def incrementAll( array ):
    for i in range( len( array ) ):
        array[ i ] += 1


Table = [1, 2, 3, 4]
print( str( Table ) )

incrementAll( Table )

print( str( Table ) )
```



```nasm
             section .data
Table        dd      1,2,3,4

             section .text
incrementAll:
             push    ebp
             mov     ebp, esp
             push    ebx
             push    ecx
             mov     ecx, 4
             mov     ebx, dword[ebp+8]
.for:        inc     dword[ebx]
             add     ebx, 4
             loop    .for
             pop     ecx
             pop     ebx
             pop     ebp
             ret     4


_Start:
             mov     eax, Table
             push    eax
             call    incrementAll
```

An Example

```
# example.py
from __future__ import print_function

def incrementAll( array ):
    for i in range( len( array ) ):
        array[ i ] += 1


Table = [1, 2, 3, 4]
print( str( Table ) )

incrementAll( Table )

print( str( Table ) )
```



```
                section .data
Table           dd      1,2,3,4

                section .text
incrementAll:
                push    ebp
                mov     ebp, esp
                push    ebx
                push    ecx
                mov     ecx, 4
                mov     ebx, dword[ebp+8]
.for:           inc     dword[ebx]
                add     ebx, 4
                loop    .for
                pop     ecx
                pop     ebx
                pop     ebp
                ret     4


_Start:
                mov     eax, Table
                push    eax
                call    incrementAll
```

An Example

```python
# example.py
from __future__ import print_function

def incrementAll( array ):
    for i in range( len( array ) ):
        array[ i ] += 1


Table = [1, 2, 3, 4]
print( str( Table ) )

incrementAll( Table )

print( str( Table ) )
```

```nasm
                section .data
Table           dd      1,2,3,4

                section .text
incrementAll:
                push    ebp
                mov     ebp, esp
                push    ebx
                push    ecx
                mov     ecx, 4
                mov     ebx, dword[ebp+8]
.for:           inc     dword[ebx]
                add     ebx, 4
                loop    .for
                pop     ecx
                pop     ebx
                pop     ebp
                ret     4


_Start:
                mov     eax, Table
                push    eax
                call    incrementAll
```

An Example

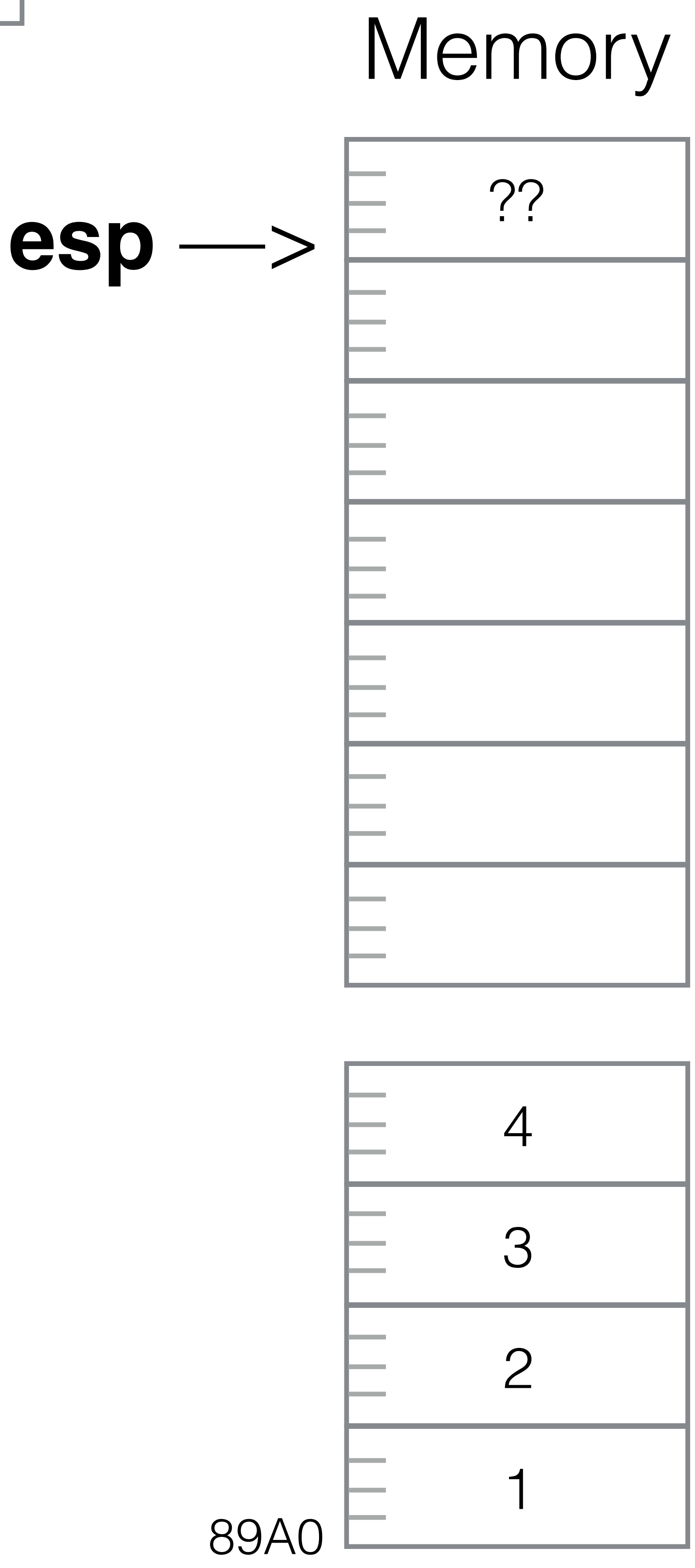
```python
# example.py
from __future__ import print_function

def incrementAll( array ):
    for i in range( len( array ) ):
        array[ i ] += 1


Table = [1, 2, 3, 4]
print( str( Table ) )

incrementAll( Table )

print( str( Table ) )
```

```
                section .data
Table           dd      1,2,3,4

                section .text
incrementAll:
                push    ebp
                mov     ebp, esp
                push    ebx
                push    ecx
                mov     ecx, 4
                mov     ebx, dword[ebp+8]
.for:           inc     dword[ebx]
                add     ebx, 4
                loop    .for
                pop     ecx
                pop     ebx
                pop     ebp
                ret     4


_Start:
                mov     eax, Table
                push    eax
                call    incrementAll
```

Single-Step Execution

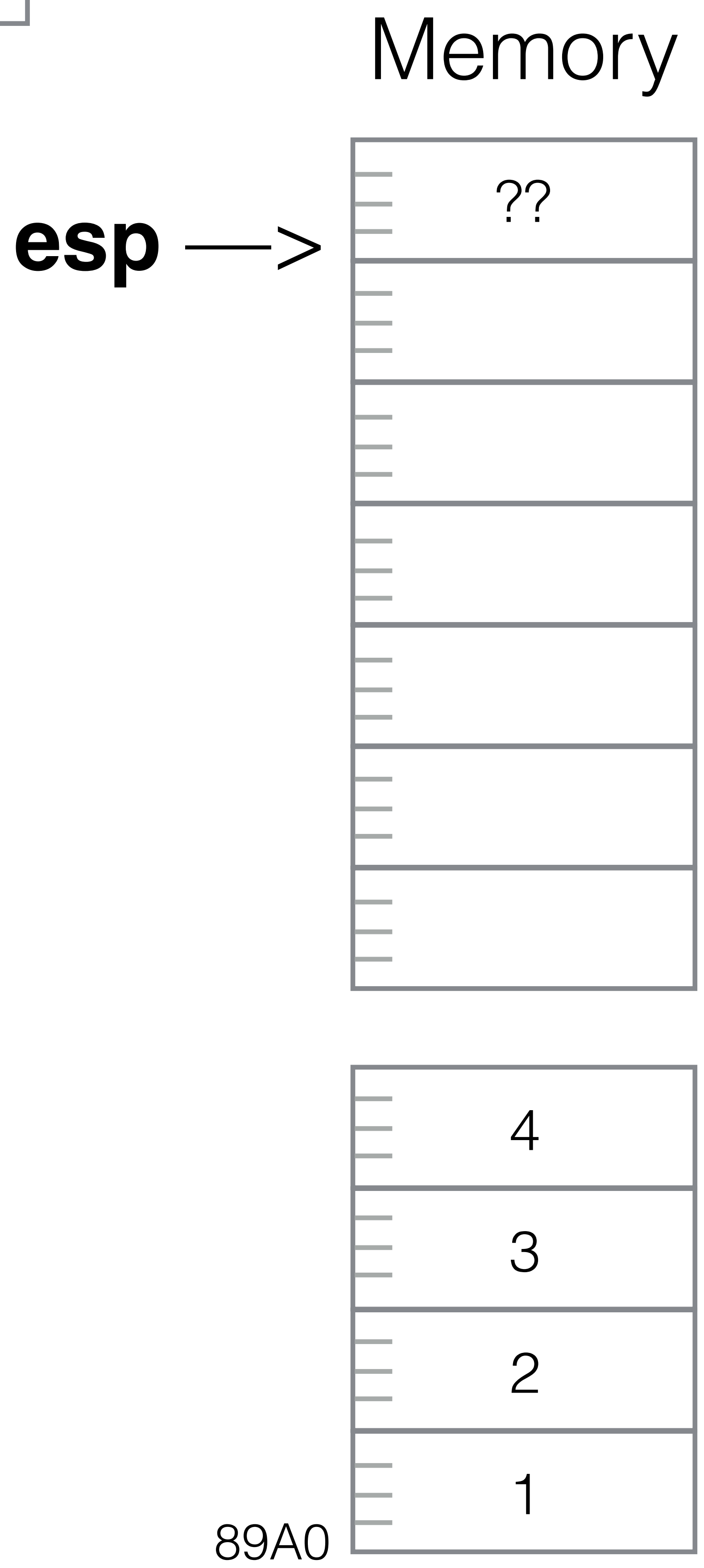
eax ?? **ebx** ?? **ecx** ??

```
                    section .data
Table               dd      1,2,3,4

                    section .text
incrementAll:
                    push    ebp
                    mov     ebp, esp
                    push    ebx
                    push    ecx
                    mov     ecx, 4
                    mov     ebx, dword[ebp+8]
.for:               inc     dword[ebx]
                    add     ebx, 4
                    loop    .for
                    pop     ecx
                    pop     ebx
                    pop     ebp
                    ret     4


_Start:
                    mov     eax, Table           <— eip
                    push    eax
                    call    incrementAll
                    xxx
```



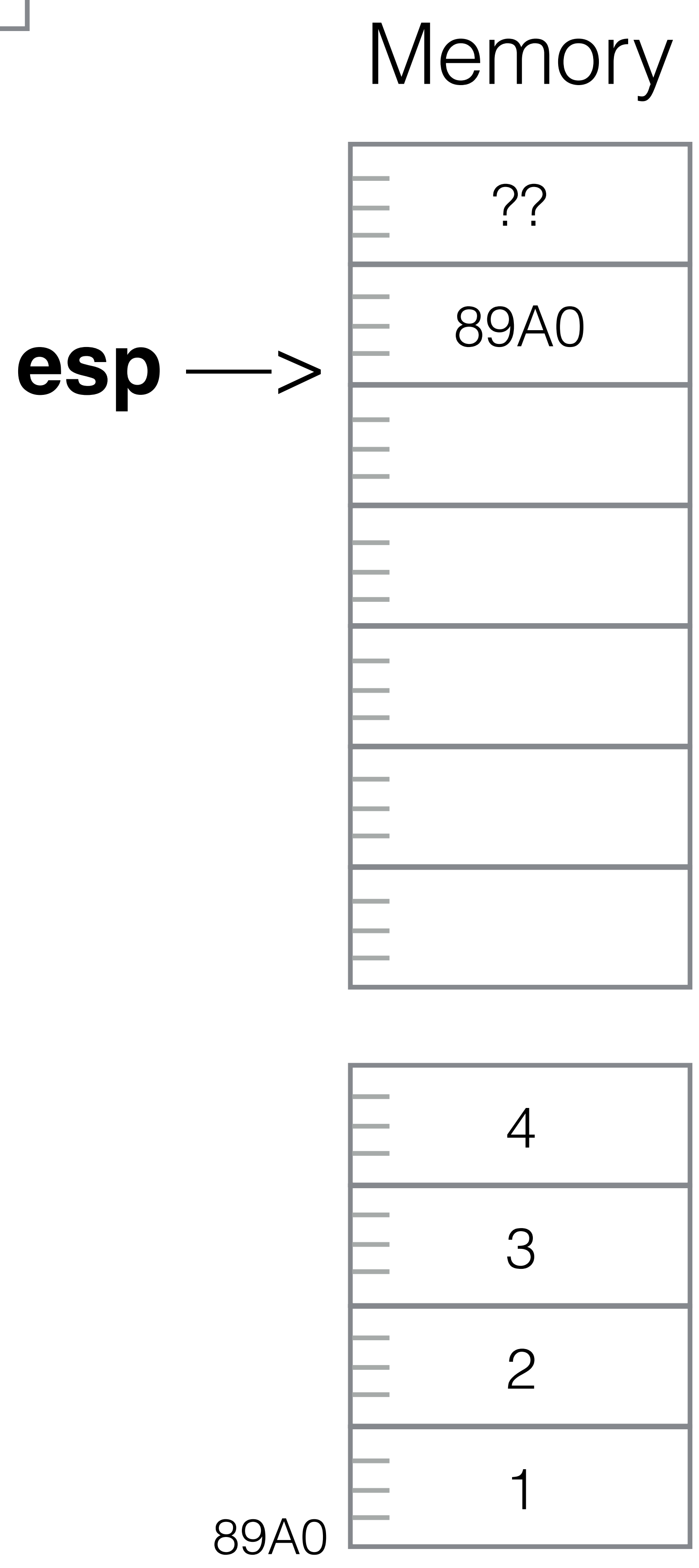
eax 89A0 **ebx** ?? **ecx** ??

```
                section .data
Table           dd      1,2,3,4

                section .text
incrementAll:
                push    ebp
                mov     ebp, esp
                push    ebx
                push    ecx
                mov     ecx, 4
                mov     ebx, dword[ebp+8]
.for:           inc     dword[ebx]
                add     ebx, 4
                loop    .for
                pop     ecx
                pop     ebx
                pop     ebp
                ret     4


_Start:
                mov     eax, Table
                push    eax             <— eip
                call    incrementAll
                xxx
```



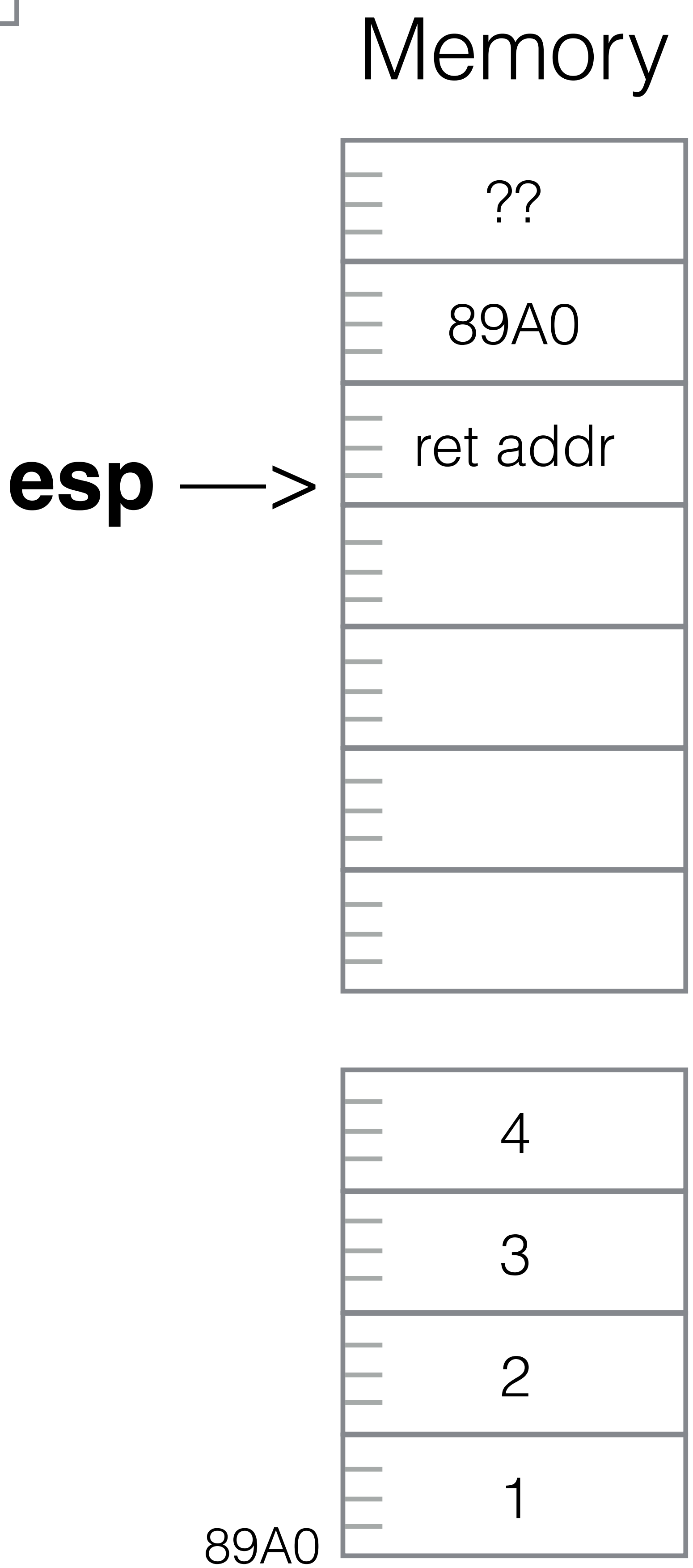
eax 89A0 **ebx** ?? **ecx** ??

```
                section .data
Table           dd      1,2,3,4

                section .text
incrementAll:
                push    ebp
                mov     ebp, esp
                push    ebx
                push    ecx
                mov     ecx, 4
                mov     ebx, dword[ebp+8]
.for:           inc     dword[ebx]
                add     ebx, 4
                loop    .for
                pop     ecx
                pop     ebx
                pop     ebp
                ret     4


_Start:
                mov     eax, Table
                push    eax
                call    incrementAll     <— eip
                xxx
```



eax 89A0 **ebx** ?? **ecx** ??

```
             section .data
Table        dd      1,2,3,4

             section .text
incrementAll:
             push    ebp                  <— eip
             mov     ebp, esp
             push    ebx
             push    ecx
             mov     ecx, 4
             mov     ebx, dword[ebp+8]
.for:        inc     dword[ebx]
             add     ebx, 4
             loop    .for
             pop     ecx
             pop     ebx
             pop     ebp
             ret     4


_Start:
             mov     eax, Table
             push    eax
             call    incrementAll
             xxx
```



Memory

??

89A0

esp —> ret addr

4

3

2

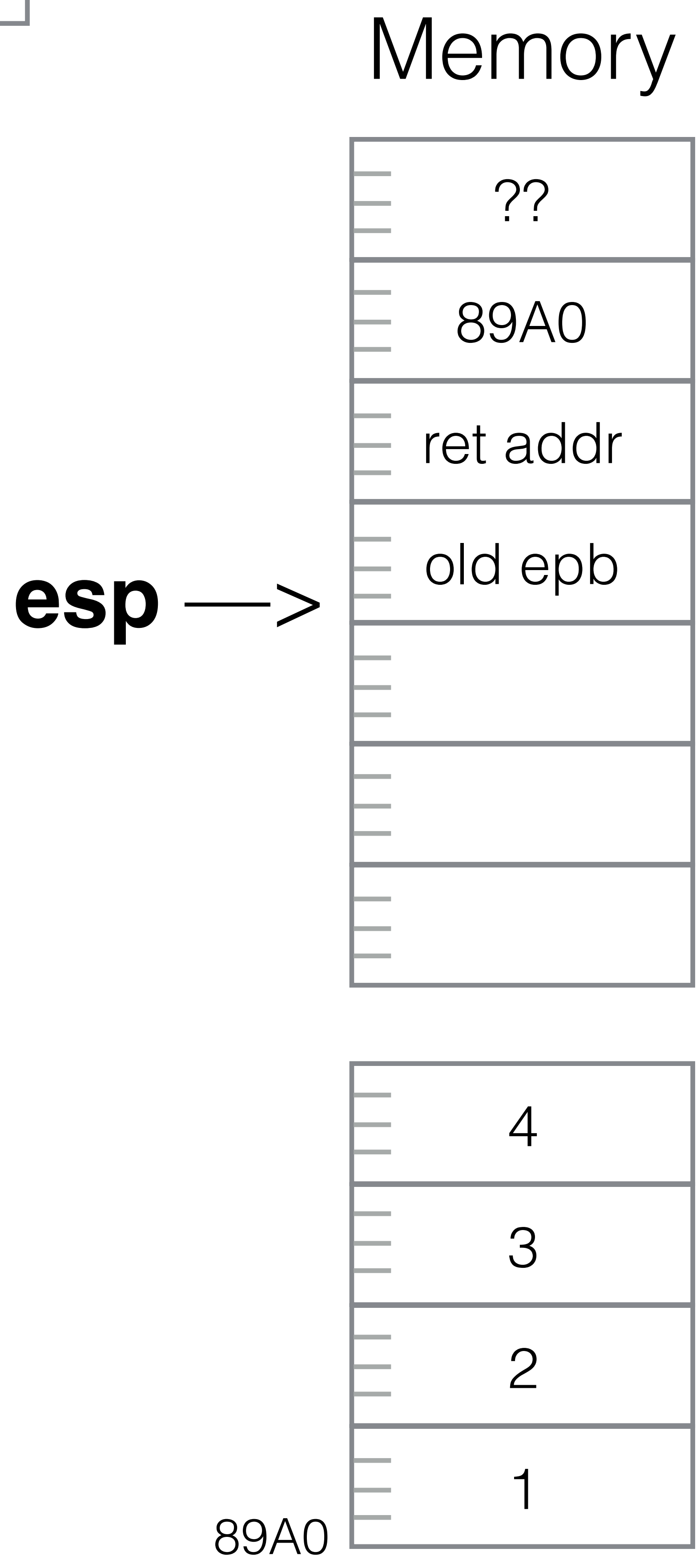
1

89A0

eax 89A0 **ebx** ?? **ecx** ??

```
              section .data
Table         dd      1,2,3,4

              section .text
incrementAll:
              push    ebp
              mov     ebp, esp
              push    ebx
              push    ecx
              mov     ecx, 4
              mov     ebx, dword[ebp+8]
.for:         inc     dword[ebx]
              add     ebx, 4
              loop    .for
              pop     ecx
              pop     ebx
              pop     ebp
              ret     4


_Start:
              mov     eax, Table
              push    eax
              call    incrementAll
              xxx
```

<— **eip**



Memory

??

89A0

ret addr

esp —> old epb

4

3

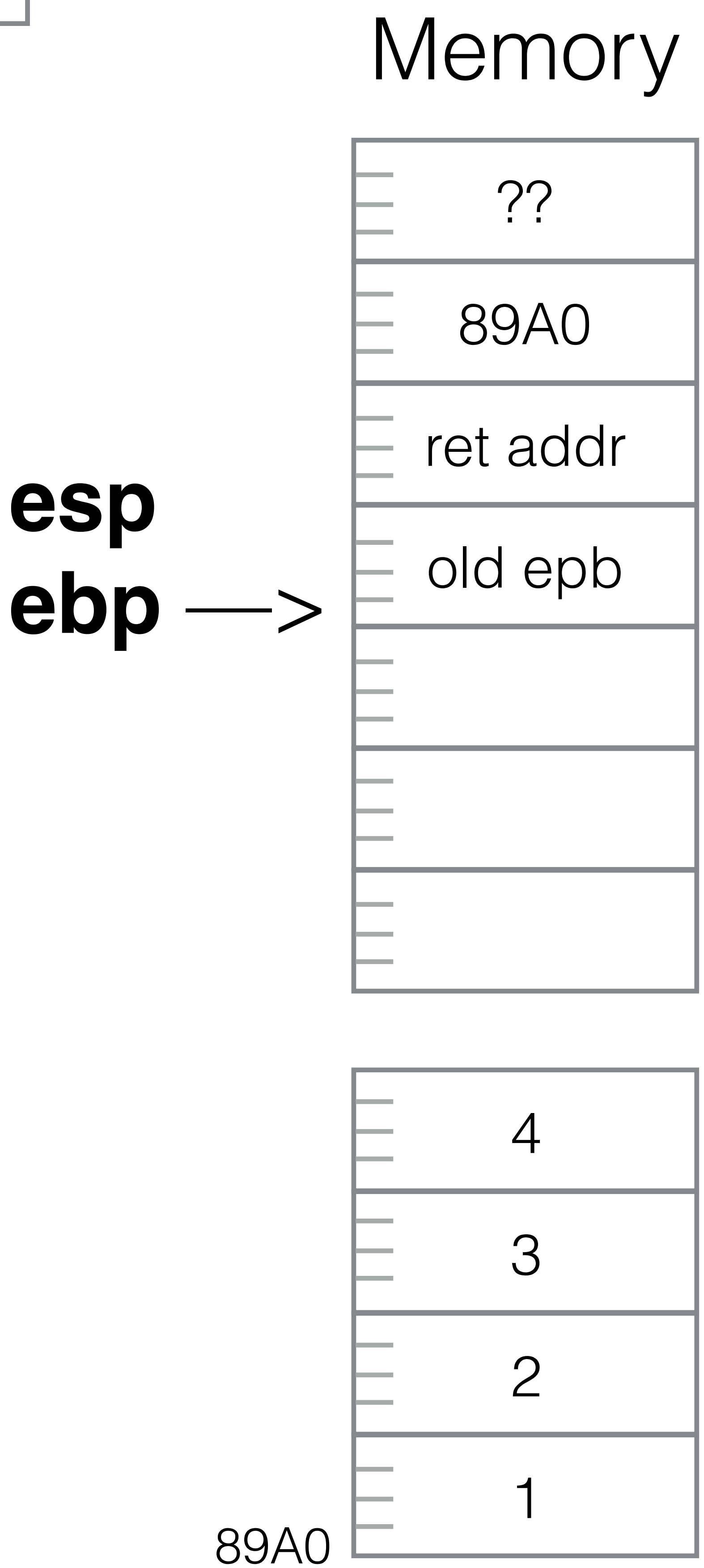
2

1

89A0

eax 89A0 **ebx** ?? **ecx** ??

```
                section .data
Table           dd      1,2,3,4

                section .text
incrementAll:
                push    ebp
                mov     ebp, esp
                push    ebx              <— eip
                push    ecx
                mov     ecx, 4
                mov     ebx, dword[ebp+8]
.for:           inc     dword[ebx]
                add     ebx, 4
                loop    .for
                pop     ecx
                pop     ebx
                pop     ebp
                ret     4


_Start:
                mov     eax, Table
                push    eax
                call    incrementAll
                xxx
```



Memory

esp

ebp —>

??

89A0

ret addr

old epb

4

3

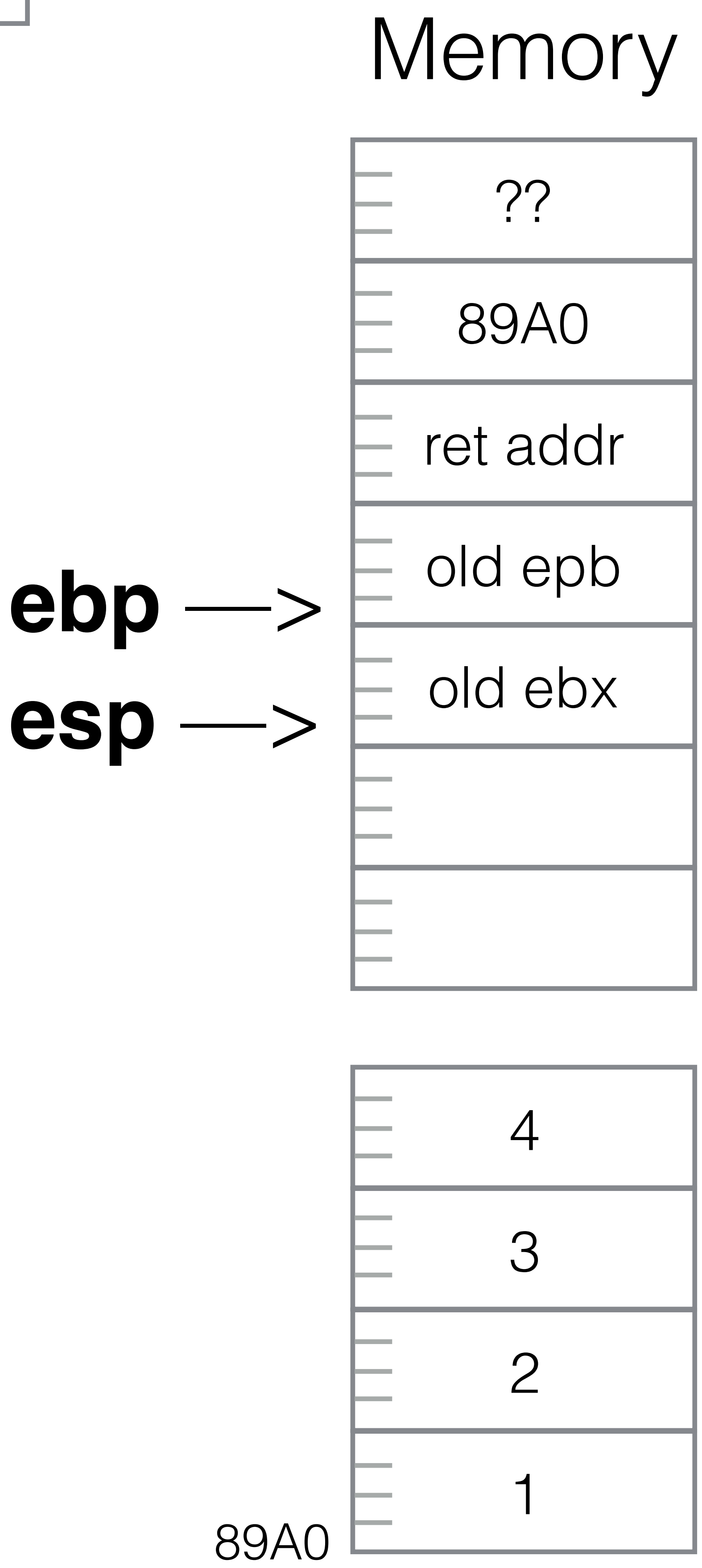
2

1

89A0

eax 89A0 **ebx** ?? **ecx** ??

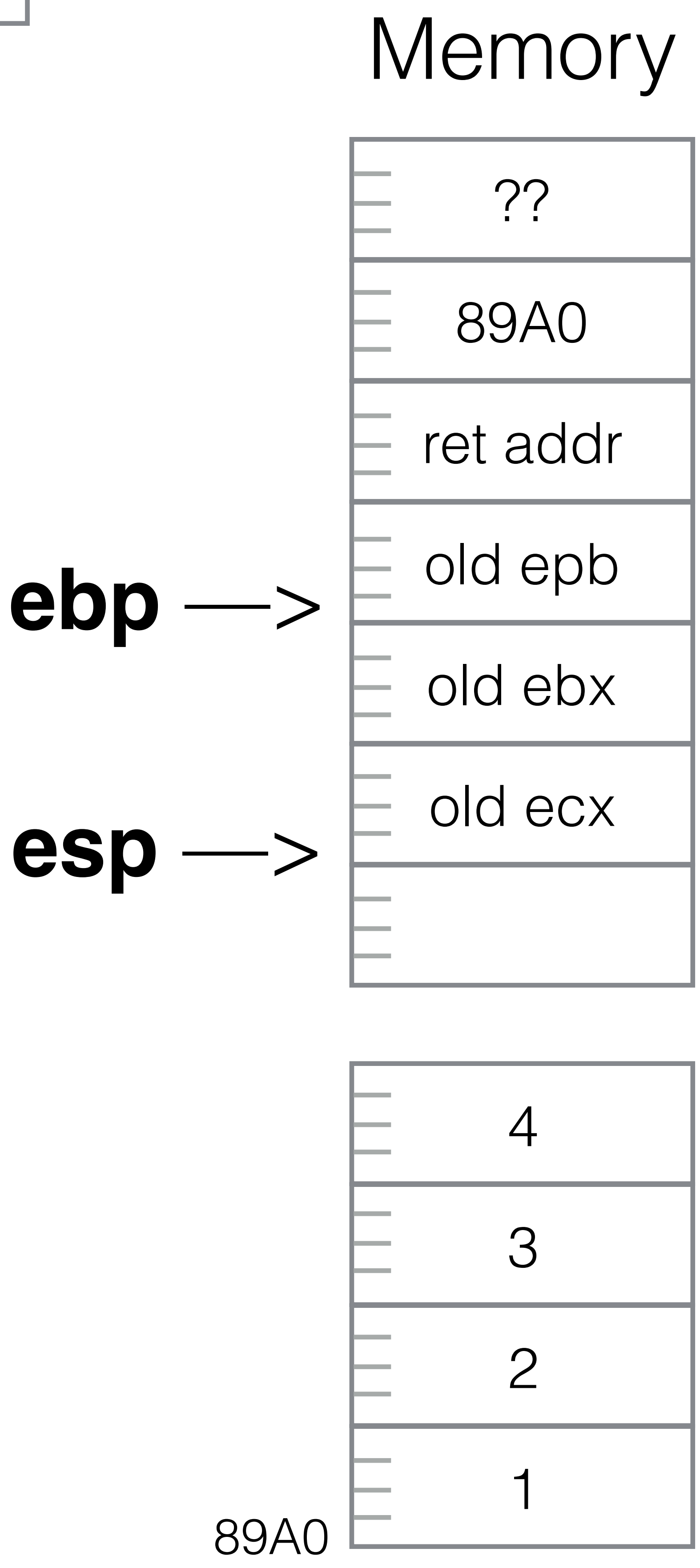
```
                 section .data
Table            dd        1,2,3,4

                 section .text
incrementAll:
                 push      ebp
                 mov       ebp, esp
                 push      ebx
                 push      ecx                  <— eip
                 mov       ecx, 4
                 mov       ebx, dword[ebp+8]
.for:            inc       dword[ebx]
                 add       ebx, 4
                 loop      .for
                 pop       ecx
                 pop       ebx
                 pop       ebp
                 ret       4


_Start:
                 mov       eax, Table
                 push      eax
                 call      incrementAll
                 xxx
```



Memory

eax 89A0 **ebx** ?? **ecx** ??

```
             section .data
Table        dd      1,2,3,4

             section .text
incrementAll:
             push    ebp
             mov     ebp, esp
             push    ebx
             push    ecx
             mov     ecx, 4               <— eip
             mov     ebx, dword[ebp+8]
.for:        inc     dword[ebx]
             add     ebx, 4
             loop    .for
             pop     ecx
             pop     ebx
             pop     ebp
             ret     4


_Start:
             mov     eax, Table
             push    eax
             call    incrementAll
             xxx
```



Memory

??

89A0

ret addr

ebp —> old epb

old ebx

esp —> old ecx

4

3

2

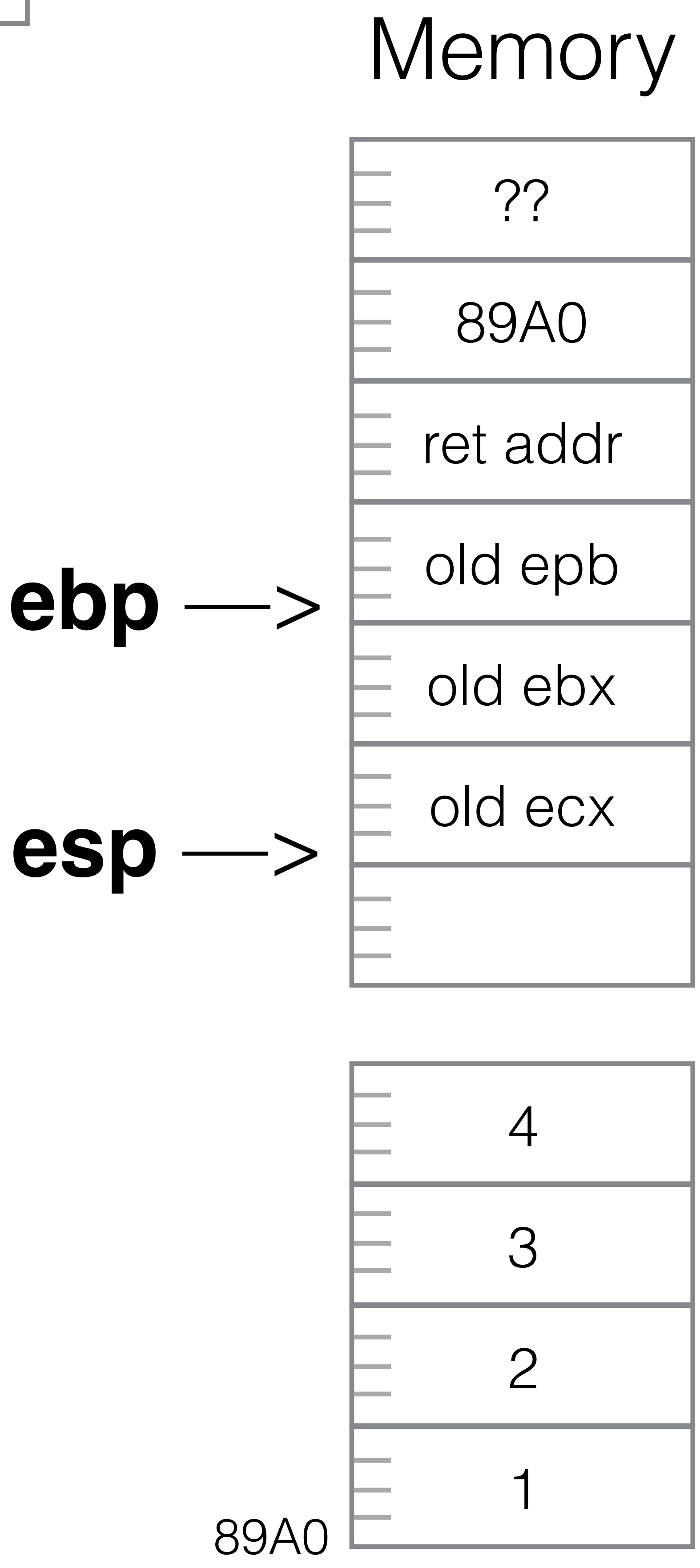
1

89A0

eax 89A0 **ebx** ?? **ecx** 4

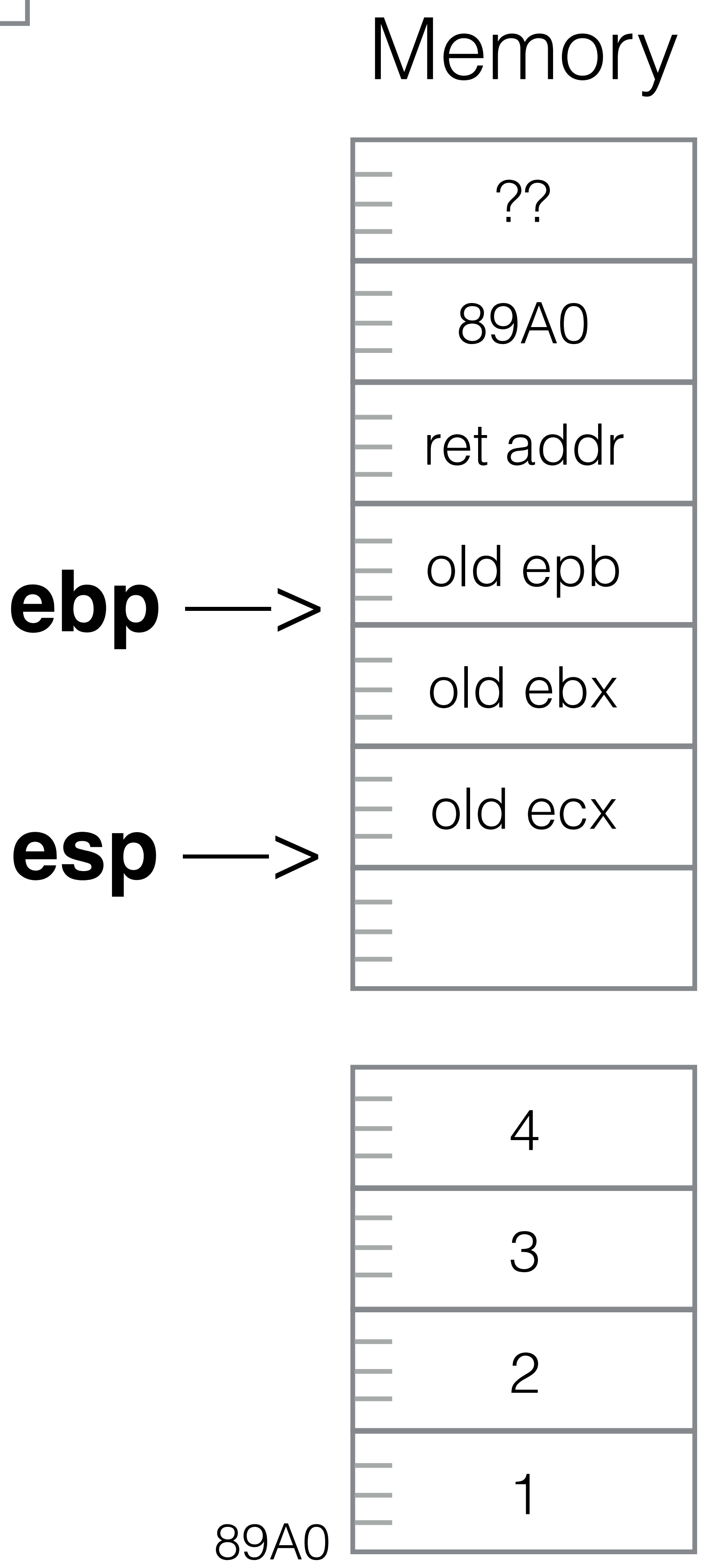
```
                    section .data
Table               dd          1,2,3,4

                    section .text
incrementAll:
                    push        ebp
                    mov         ebp, esp
                    push        ebx
                    push        ecx
                    mov         ecx, 4
                    mov         ebx, dword[ebp+8]
.for:               inc         dword[ebx]
                    add         ebx, 4
                    loop        .for
                    pop         ecx
                    pop         ebx
                    pop         ebp
                    ret         4


_Start:
                    mov         eax, Table
                    push        eax
                    call        incrementAll
                    xxx
```

<— **eip**



eax 89A0 **ebx** 89A0 **ecx** 4

```
                section .data
Table           dd      1,2,3,4

                section .text
incrementAll:
                push    ebp
                mov     ebp, esp
                push    ebx
                push    ecx
                mov     ecx, 4
                mov     ebx, dword[ebp+8]
.for:           inc     dword[ebx]            <— eip
                add     ebx, 4
                loop    .for
                pop     ecx
                pop     ebx
                pop     ebp
                ret     4


_Start:
                mov     eax, Table
                push    eax
                call    incrementAll
                xxx
```



Memory

??

89A0

ret addr

ebp —> old epb

old ebx

esp —> old ecx

4

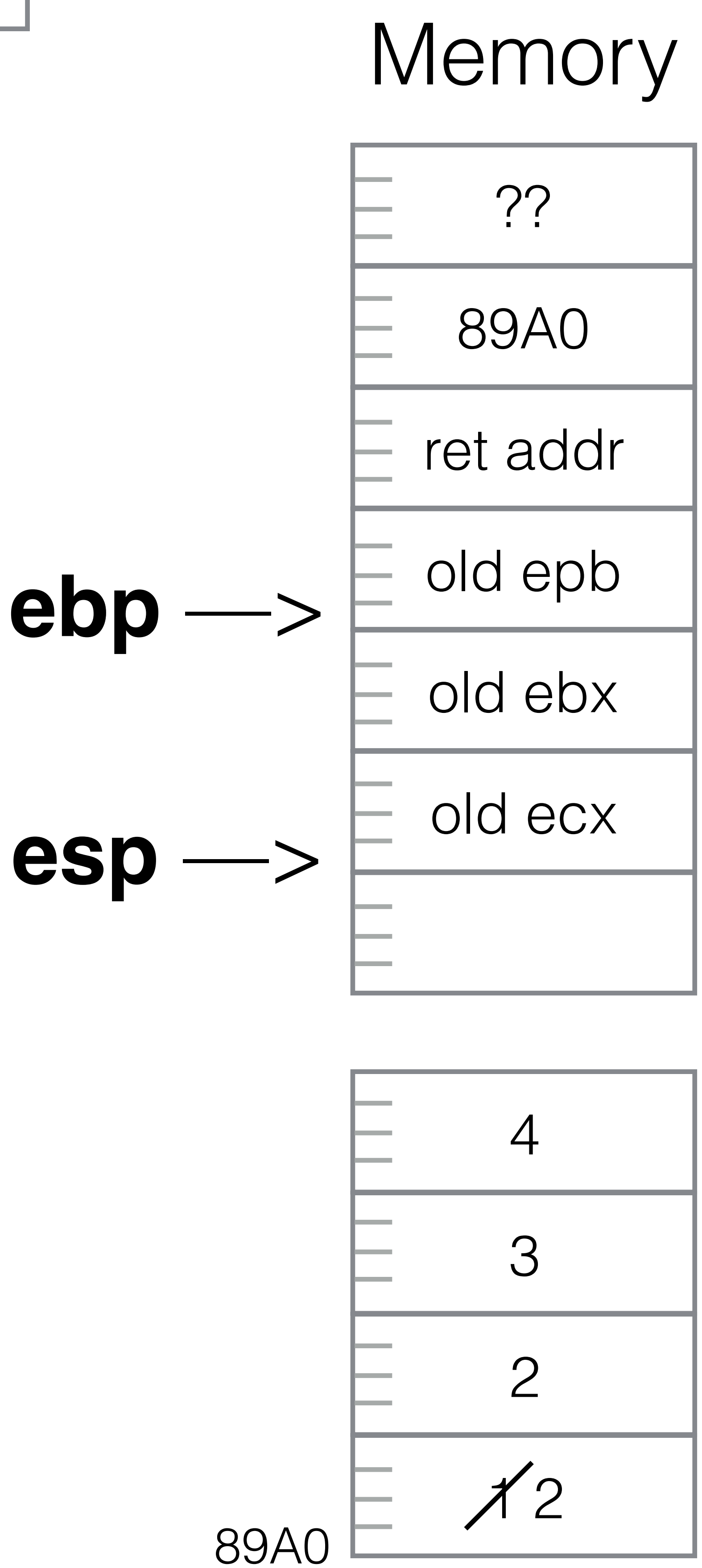
3

2

89A0 1

eax 89A0 **ebx** 89A0 **ecx** 4

```
             section .data
Table        dd      1,2,3,4

             section .text
incrementAll:
             push    ebp
             mov     ebp, esp
             push    ebx
             push    ecx
             mov     ecx, 4
             mov     ebx, dword[ebp+8]
.for:        inc     dword[ebx]
             add     ebx, 4              <— eip
             loop    .for
             pop     ecx
             pop     ebx
             pop     ebp
             ret     4


_Start:
             mov     eax, Table
             push    eax
             call    incrementAll
             xxx
```



Memory

??

89A0

ret addr

ebp —> old epb

old ebx

esp —> old ecx

4

3

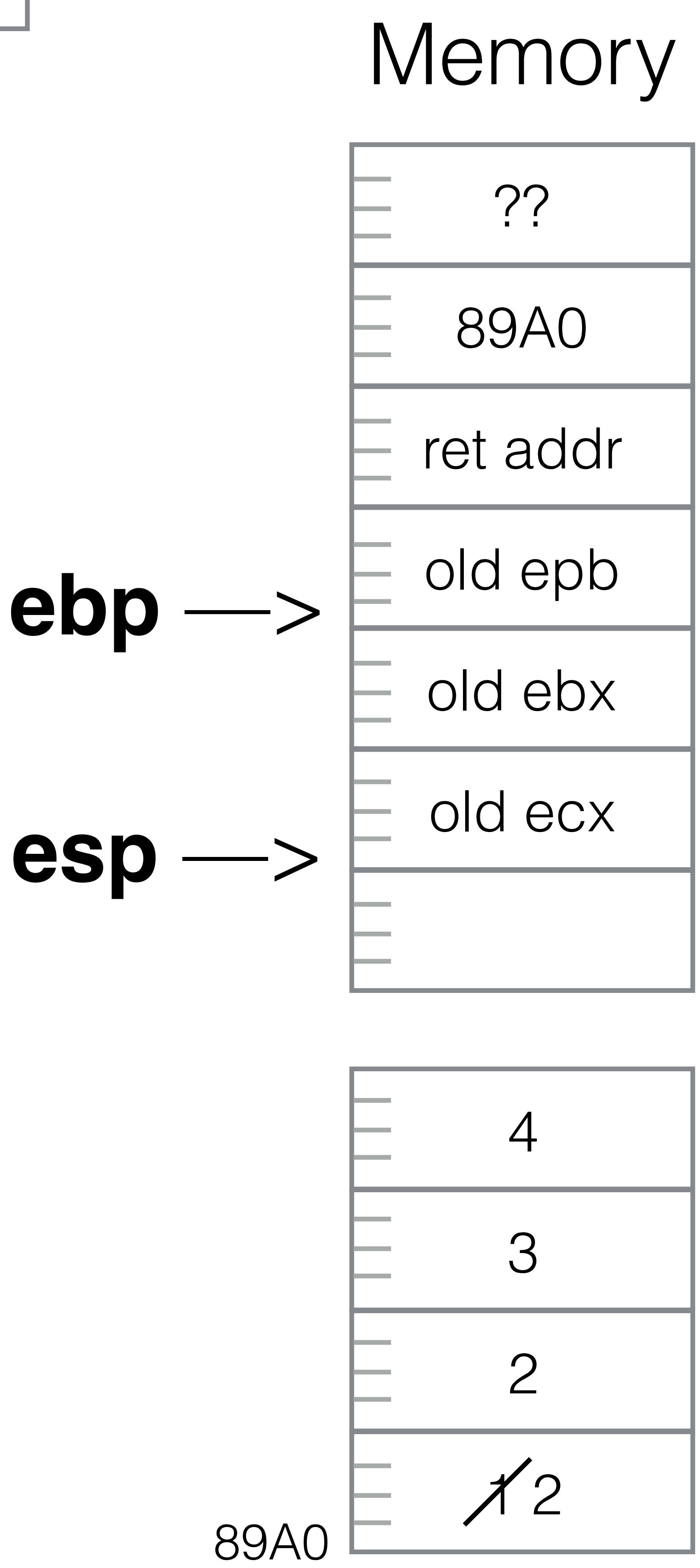
2

~~1~~ 2

89A0

eax 89A0 **ebx** 89A4 **ecx** 4

```
             section .data
Table        dd      1,2,3,4

             section .text
incrementAll:
             push    ebp
             mov     ebp, esp
             push    ebx
             push    ecx
             mov     ecx, 4
             mov     ebx, dword[ebp+8]
.for:        inc     dword[ebx]
             add     ebx, 4
             loop    .for             <— eip
             pop     ecx
             pop     ebx
             pop     ebp
             ret     4


_Start:
             mov     eax, Table
             push    eax
             call    incrementAll
             xxx
```



Memory

??

89A0

ret addr

ebp —> old epb

old ebx

esp —> old ecx

4

3

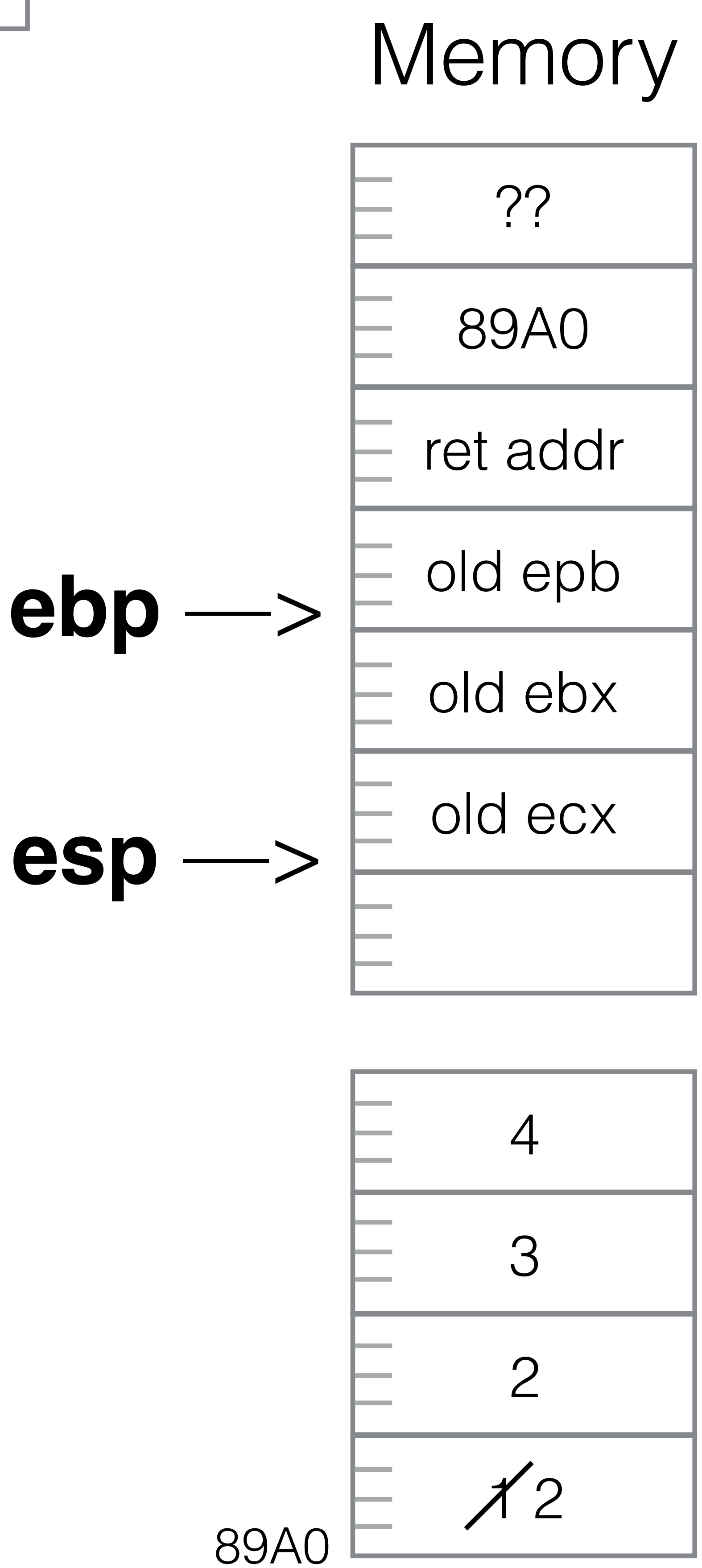
2

~~1~~2

89A0

eax 89A0 **ebx** 89A4 **ecx** ~~4~~ 3

```
             section .data
Table        dd      1,2,3,4

             section .text
incrementAll:
             push    ebp
             mov     ebp, esp
             push    ebx
             push    ecx
             mov     ecx, 4
             mov     ebx, dword[ebp+8]
.for:        inc     dword[ebx]           <— eip
             add     ebx, 4
             loop    .for
             pop     ecx
             pop     ebx
             pop     ebp
             ret     4


_Start:
             mov     eax, Table
             push    eax
             call    incrementAll
             xxx
```



Memory

??

89A0

ret addr

ebp —> old epb

old ebx

esp —> old ecx

4

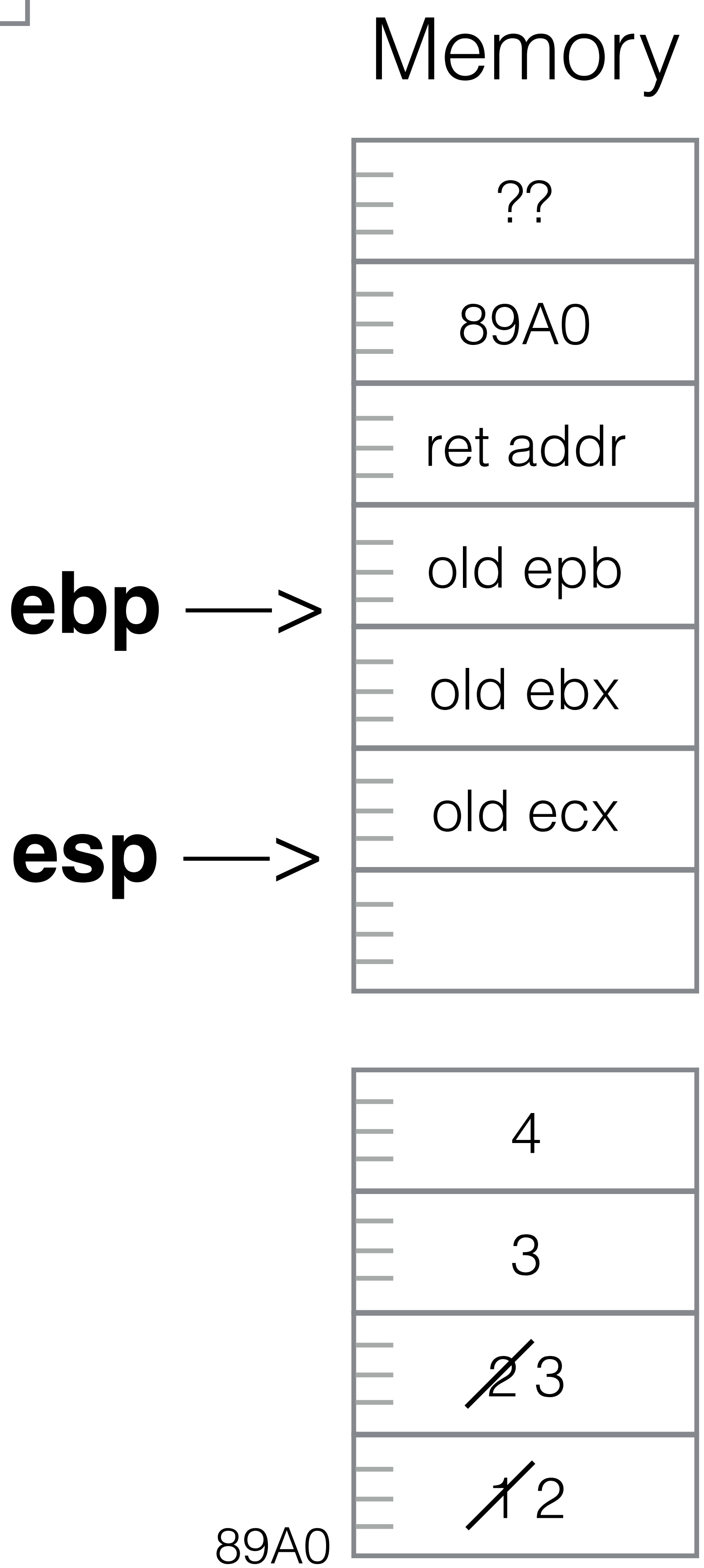
3

2

89A0 ~~1~~ 2

eax 89A0 **ebx** 89A4 **ecx** ~~4~~3

```
             section .data
Table        dd      1,2,3,4

             section .text
incrementAll:
             push    ebp
             mov     ebp, esp
             push    ebx
             push    ecx
             mov     ecx, 4
             mov     ebx, dword[ebp+8]
.for:        inc     dword[ebx]
             add     ebx, 4                <— eip
             loop    .for
             pop     ecx
             pop     ebx
             pop     ebp
             ret     4


_Start:
             mov     eax, Table
             push    eax
             call    incrementAll
             xxx
```



Memory

??

89A0

ret addr

ebp —> old epb

old ebx

esp —> old ecx

4

3

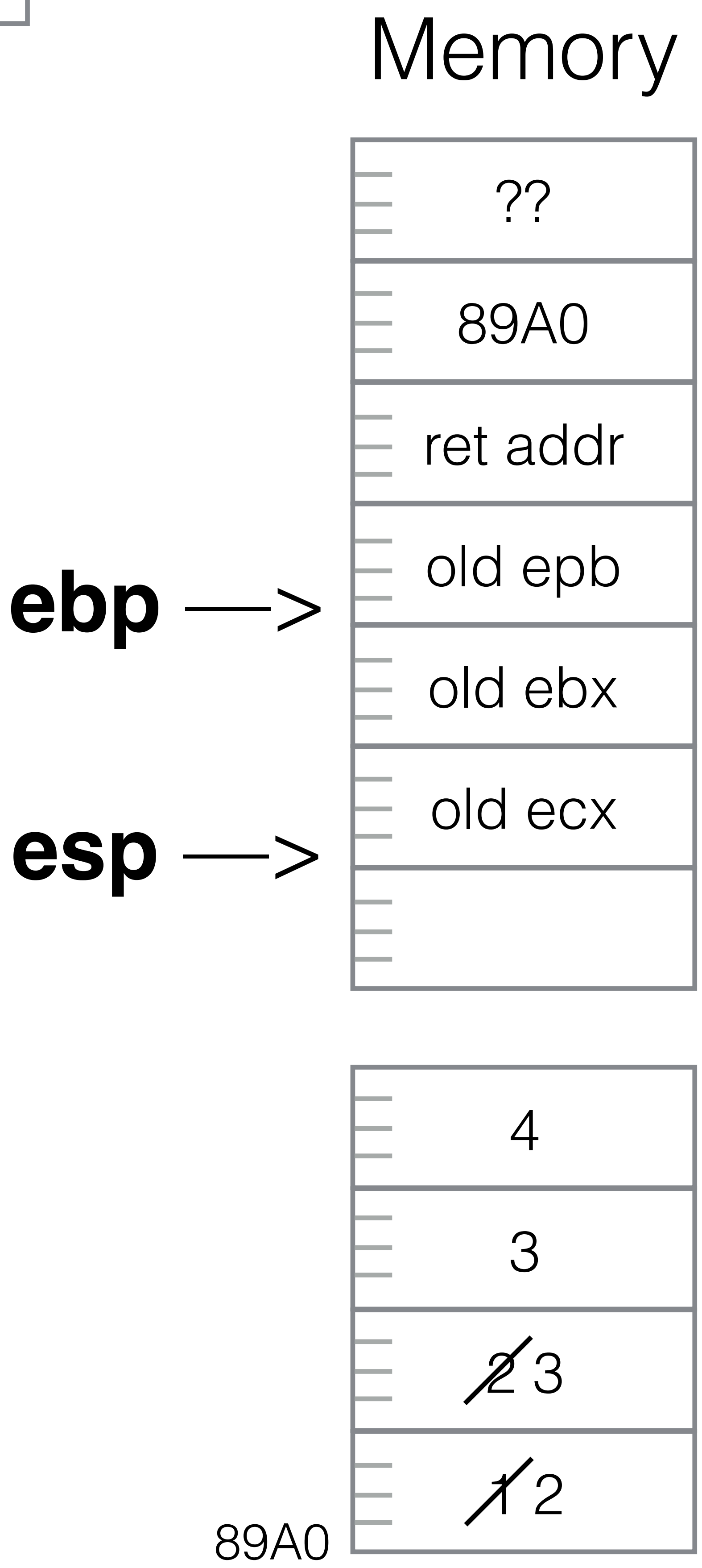
~~2~~3

~~1~~2

89A0

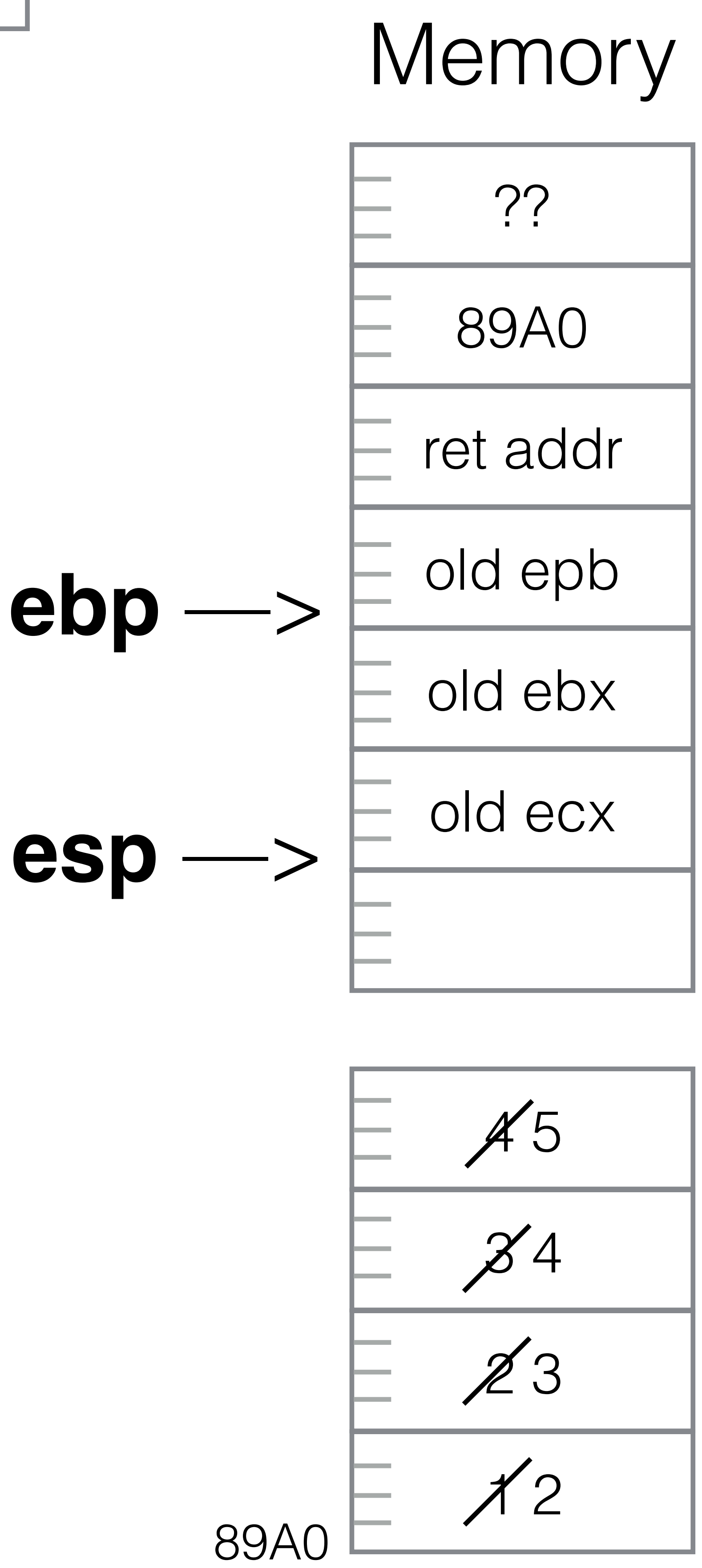
eax 89A0 **ebx** 89A8 **ecx** ~~4~~3

```
                section .data
Table           dd      1,2,3,4

                section .text
incrementAll:
                push    ebp
                mov     ebp, esp
                push    ebx
                push    ecx
                mov     ecx, 4
                mov     ebx, dword[ebp+8]
.for:           inc     dword[ebx]
                add     ebx, 4
                loop    .for                  <— eip
                pop     ecx
                pop     ebx
                pop     ebp
                ret     4


_Start:
                mov     eax, Table
                push    eax
                call    incrementAll
                xxx
```



eax 89A0 **ebx** 89B2 **ecx** ~~4~~ ~~3~~ ~~2~~ ~~1~~ 0

```
                section .data
Table           dd      1,2,3,4

                section .text
incrementAll:
                push    ebp
                mov     ebp, esp
                push    ebx
                push    ecx
                mov     ecx, 4
                mov     ebx, dword[ebp+8]
.for:           inc     dword[ebx]
                add     ebx, 4
                loop    .for
                pop     ecx                  <— eip
                pop     ebx
                pop     ebp
                ret     4


_Start:
                mov     eax, Table
                push    eax
                call    incrementAll
                xxx
```



Memory

??

89A0

ret addr

old epb

old ebx

old ecx

ebp —>

esp —>

~~4~~ 5

~~3~~ 4

~~2~~ 3

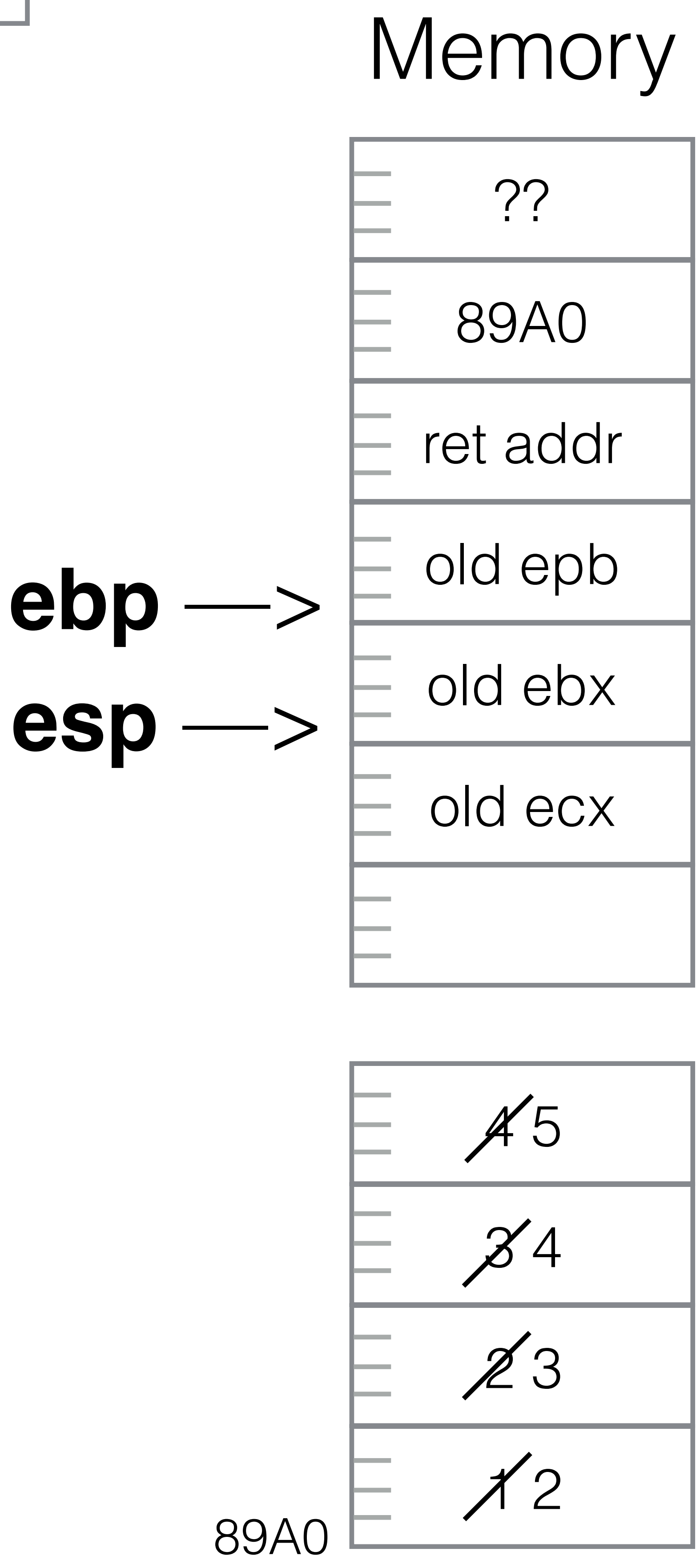
~~1~~ 2

89A0

eax 89A0 **ebx** 89B2 **ecx** ??

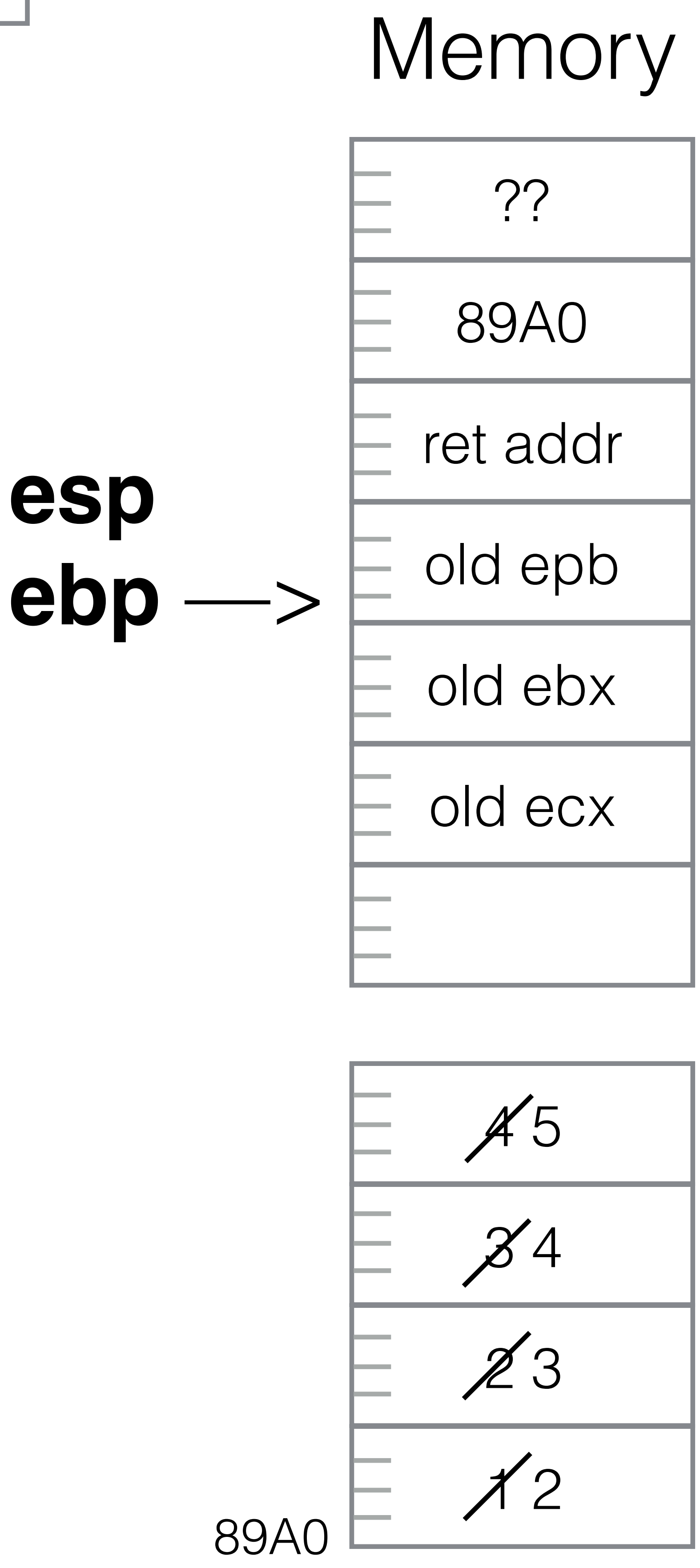
```
             section .data
Table        dd      1,2,3,4

             section .text
incrementAll:
             push    ebp
             mov     ebp, esp
             push    ebx
             push    ecx
             mov     ecx, 4
             mov     ebx, dword[ebp+8]
.for:        inc     dword[ebx]
             add     ebx, 4
             loop    .for
             pop     ecx
             pop     ebx
             pop     ebp
             ret     4


_Start:
             mov     eax, Table
             push    eax
             call    incrementAll
             xxx
```

<— **eip**



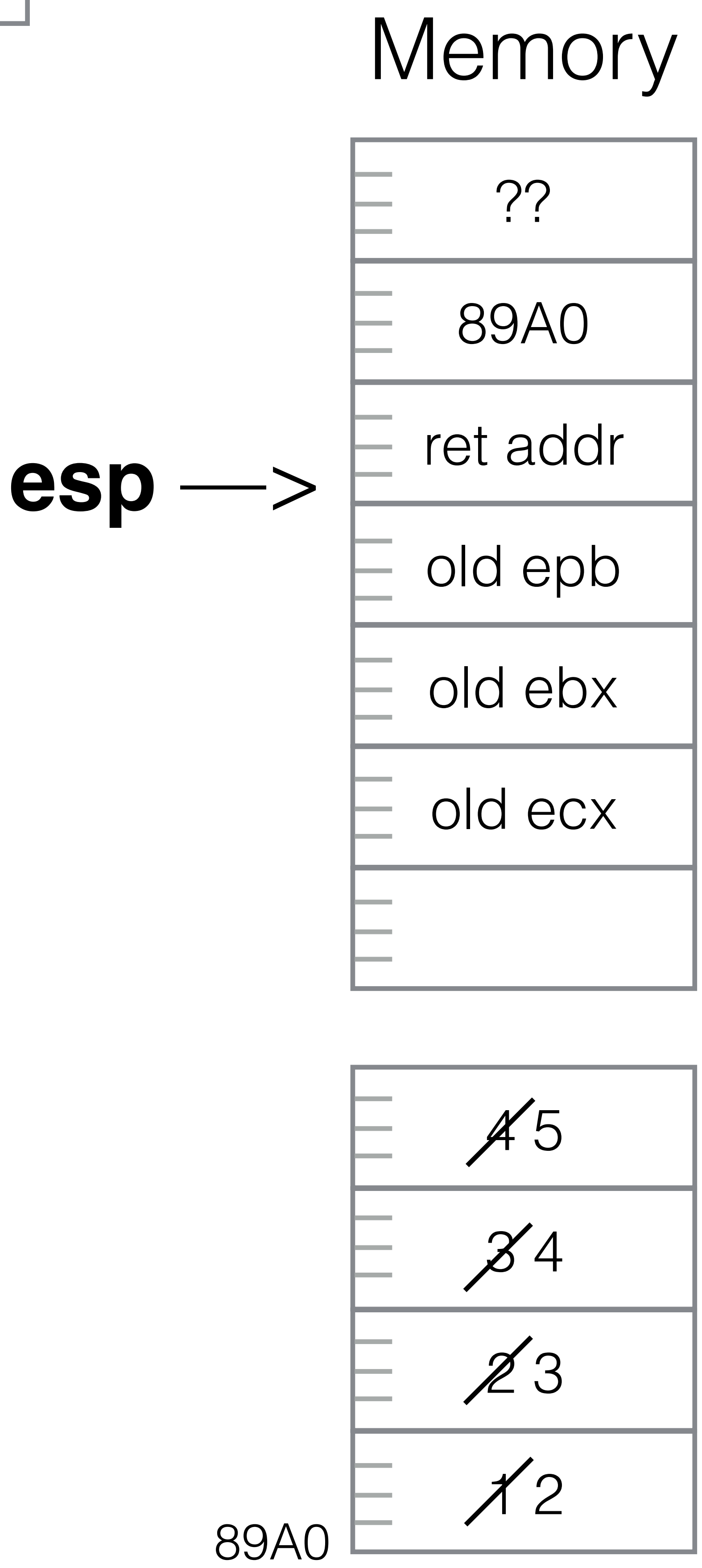
eax 89A0 **ebx** ?? **ecx** ??

```
                section .data
Table           dd      1,2,3,4

                section .text
incrementAll:
                push    ebp
                mov     ebp, esp
                push    ebx
                push    ecx
                mov     ecx, 4
                mov     ebx, dword[ebp+8]
.for:           inc     dword[ebx]
                add     ebx, 4
                loop    .for
                pop     ecx
                pop     ebx
                pop     ebp             <— eip
                ret     4


_Start:
                mov     eax, Table
                push    eax
                call    incrementAll
                xxx
```



eax 89A0 ebx ?? ecx ??

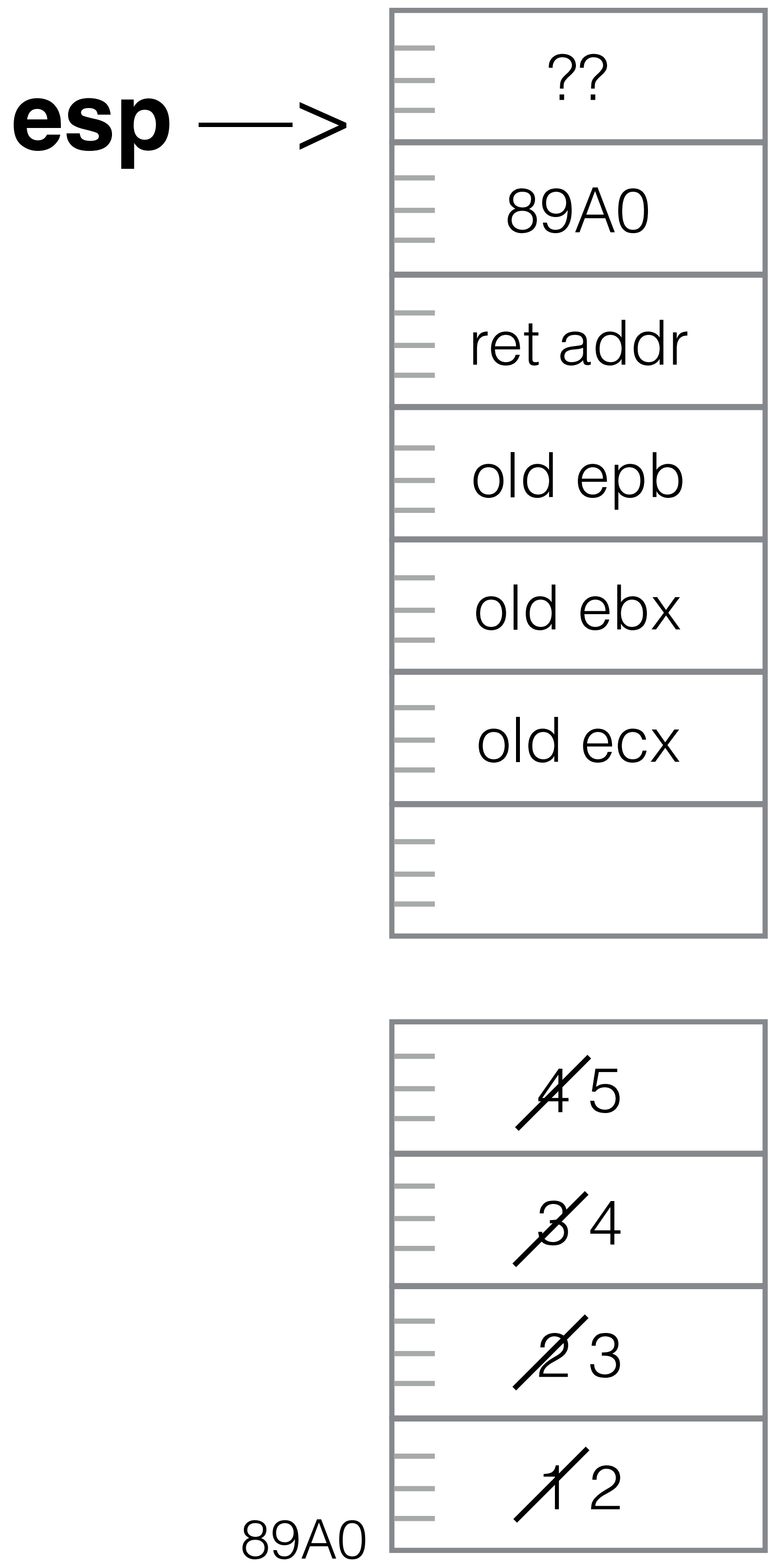
```
              section .data
Table         dd      1,2,3,4

              section .text
incrementAll:
              push    ebp
              mov     ebp, esp
              push    ebx
              push    ecx
              mov     ecx, 4
              mov     ebx, dword[ebp+8]
.for:         inc     dword[ebx]
              add     ebx, 4
              loop    .for
              pop     ecx
              pop     ebx
              pop     ebp
              ret     4                      <— eip


_Start:
              mov     eax, Table
              push    eax
              call    incrementAll
              xxx
```



eax 89A0 **ebx** ?? **ecx** ??

```
                section .data
Table           dd      1,2,3,4

                section .text
incrementAll:
                push    ebp
                mov     ebp, esp
                push    ebx
                push    ecx
                mov     ecx, 4
                mov     ebx, dword[ebp+8]
.for:           inc     dword[ebx]
                add     ebx, 4
                loop    .for
                pop     ecx
                pop     ebx
                pop     ebp
                ret     4


_Start:
                mov     eax, Table
                push    eax
                call    incrementAll
                xxx                              <— eip
```

Memory

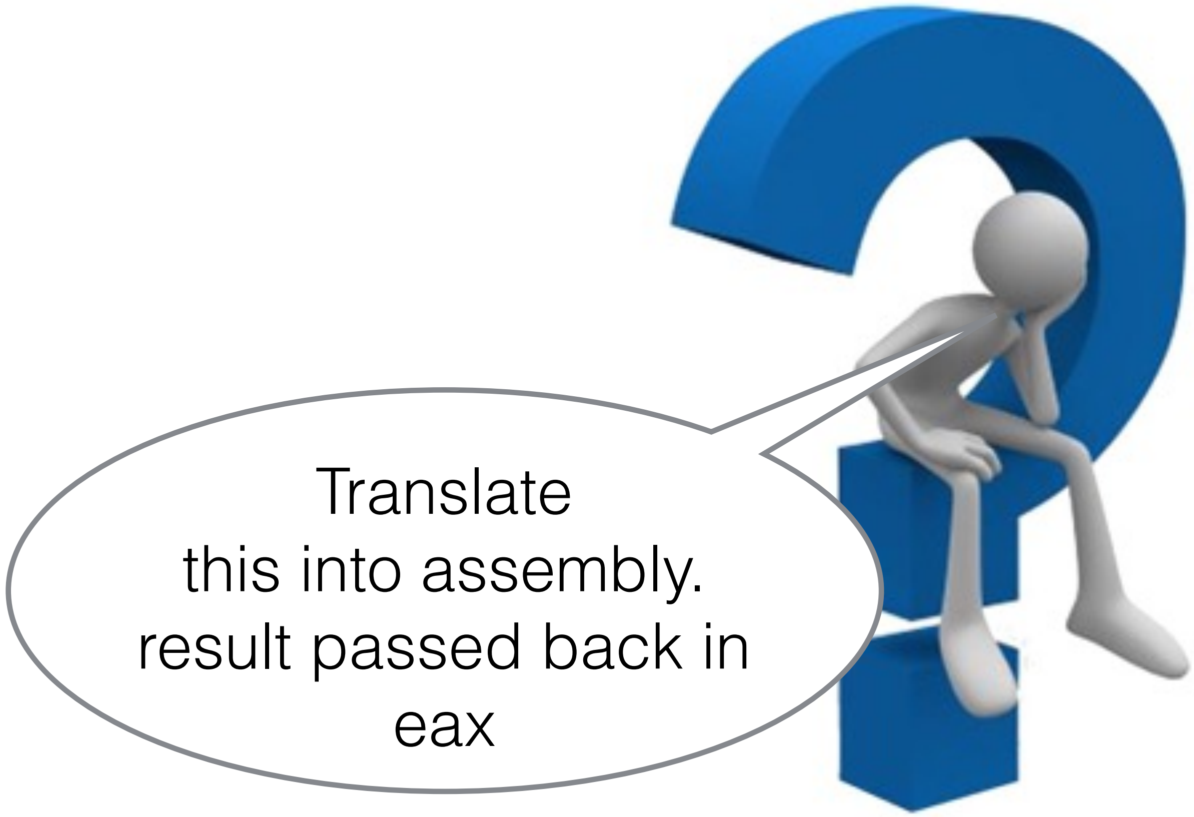


- Passing through **registers** ✓
- Passing through the **stack** ✓
  - Passing by **Value** ✓
  - Passing by **Reference** ✓

Exercise

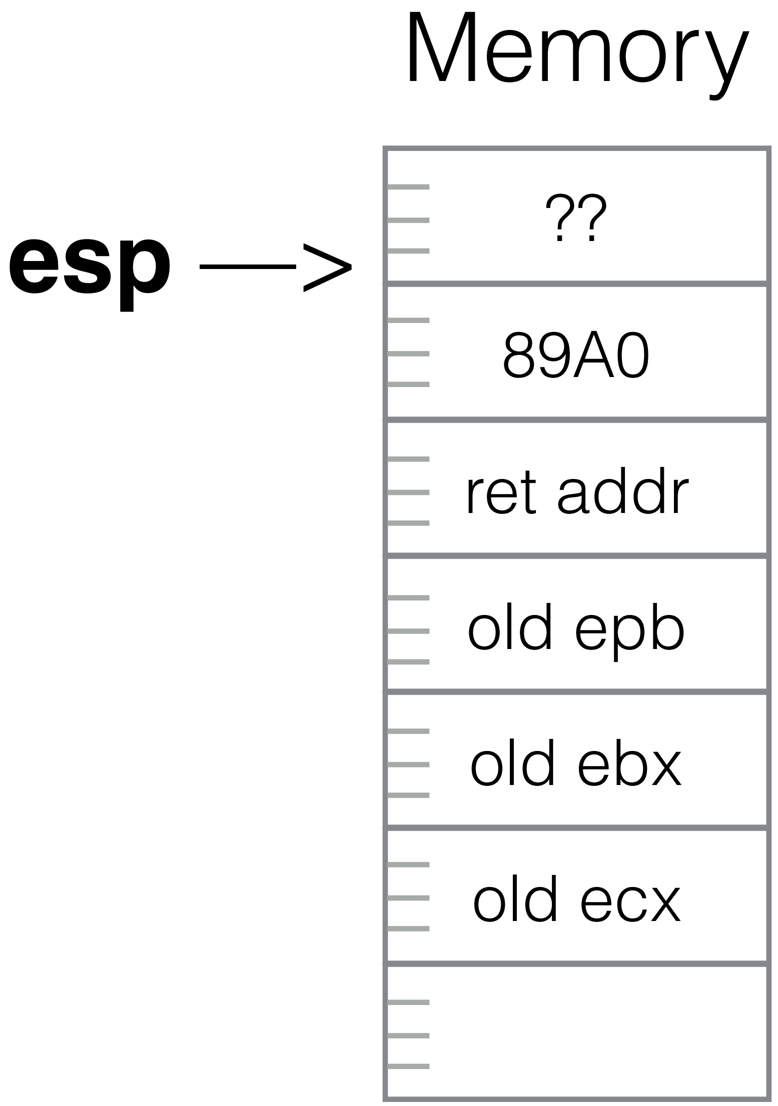
```
void func( int x, int y, int z, int t, int q ) {
   return( x+y+2*z-t-3*q );
}
```

```
int a, b, c, d;
a = 3;
b = 5;
c = -10000;
d = -1;
a = func( a, b, c+d, 2*d, 1 );
```



Translate
this into assembly.
result passed back in
eax

Question 1: What about the data "below" esp? Can it be used?



Memory

esp —>

??

89A0

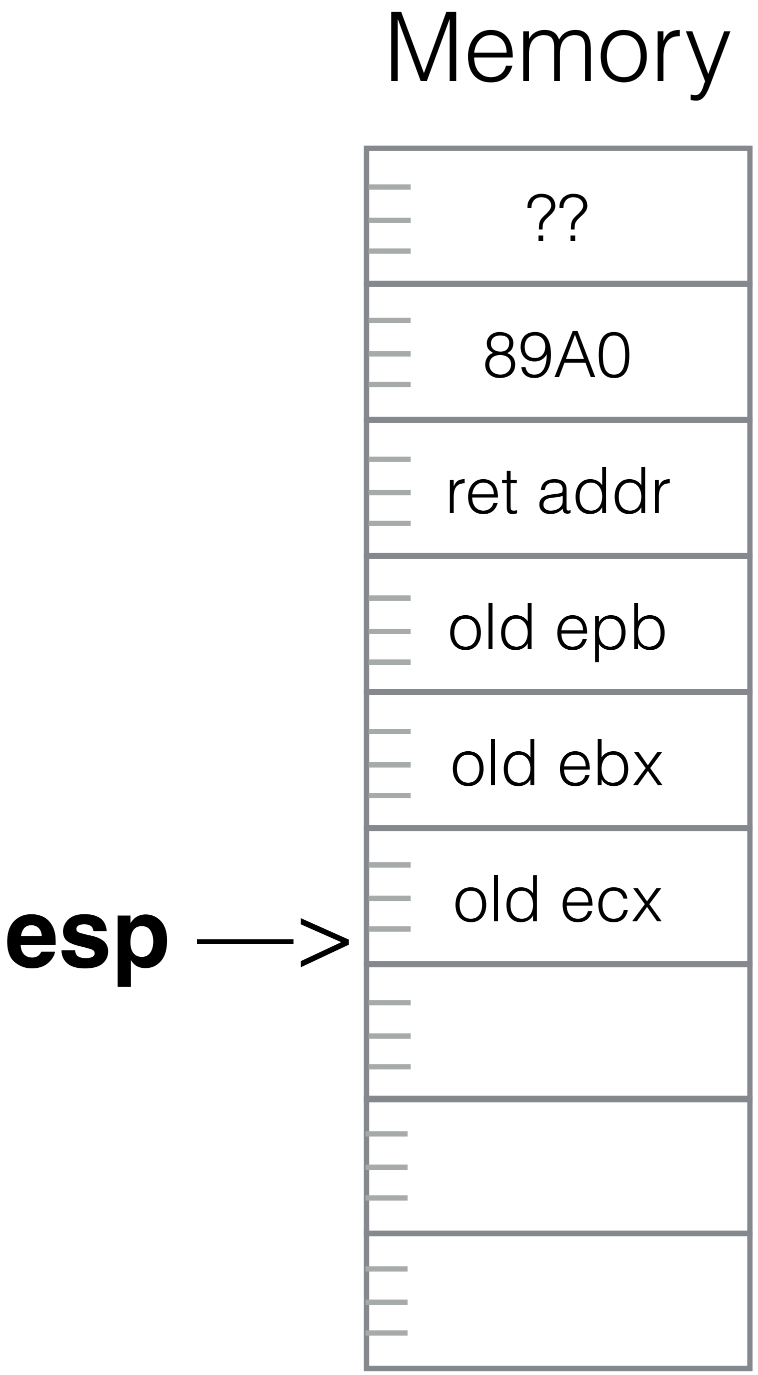
ret addr

old epb

old ebx

old ecx

Question 2: What about local variables?



Memory

??

89A0

ret addr

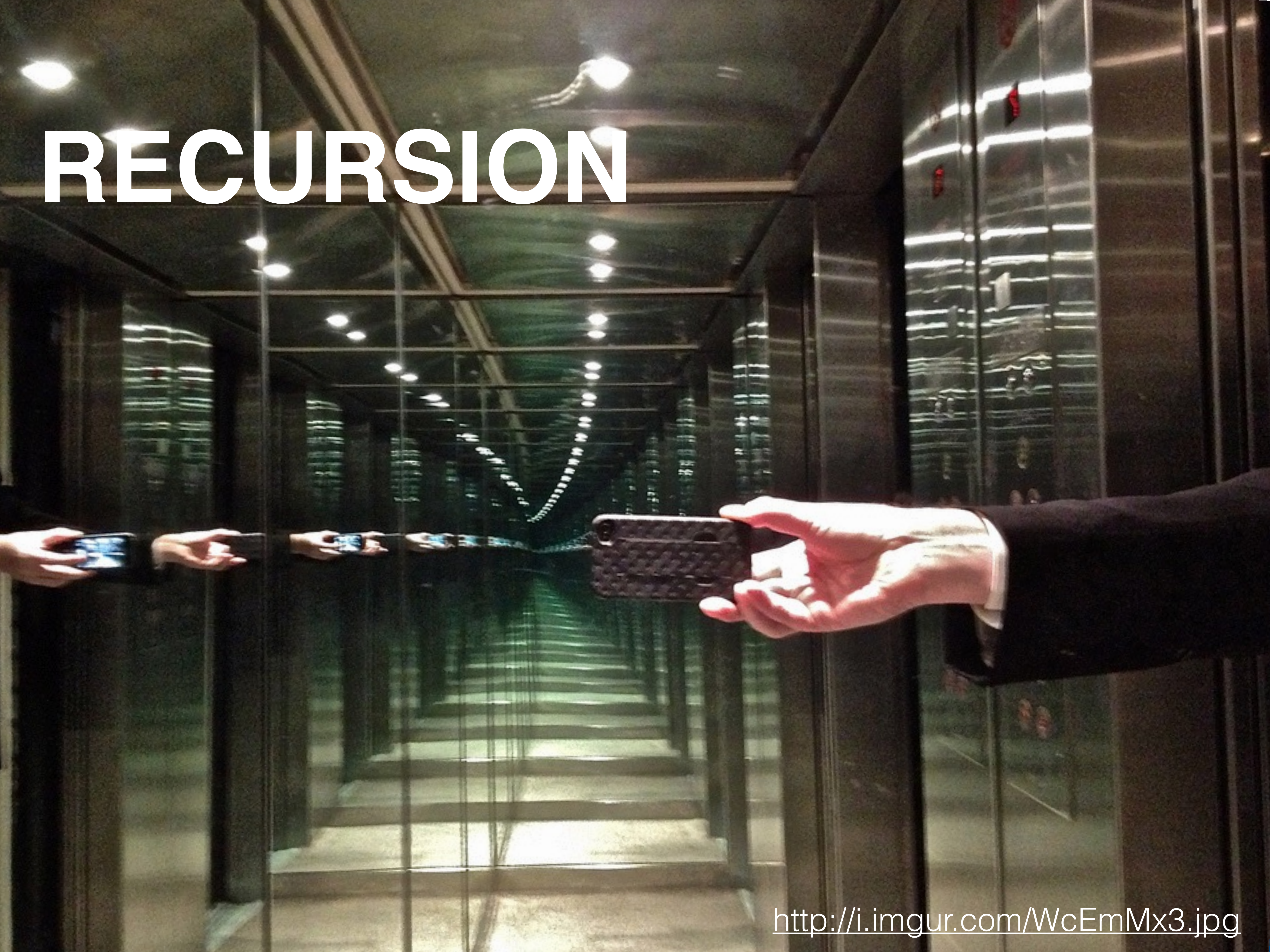
old epb

old ebx

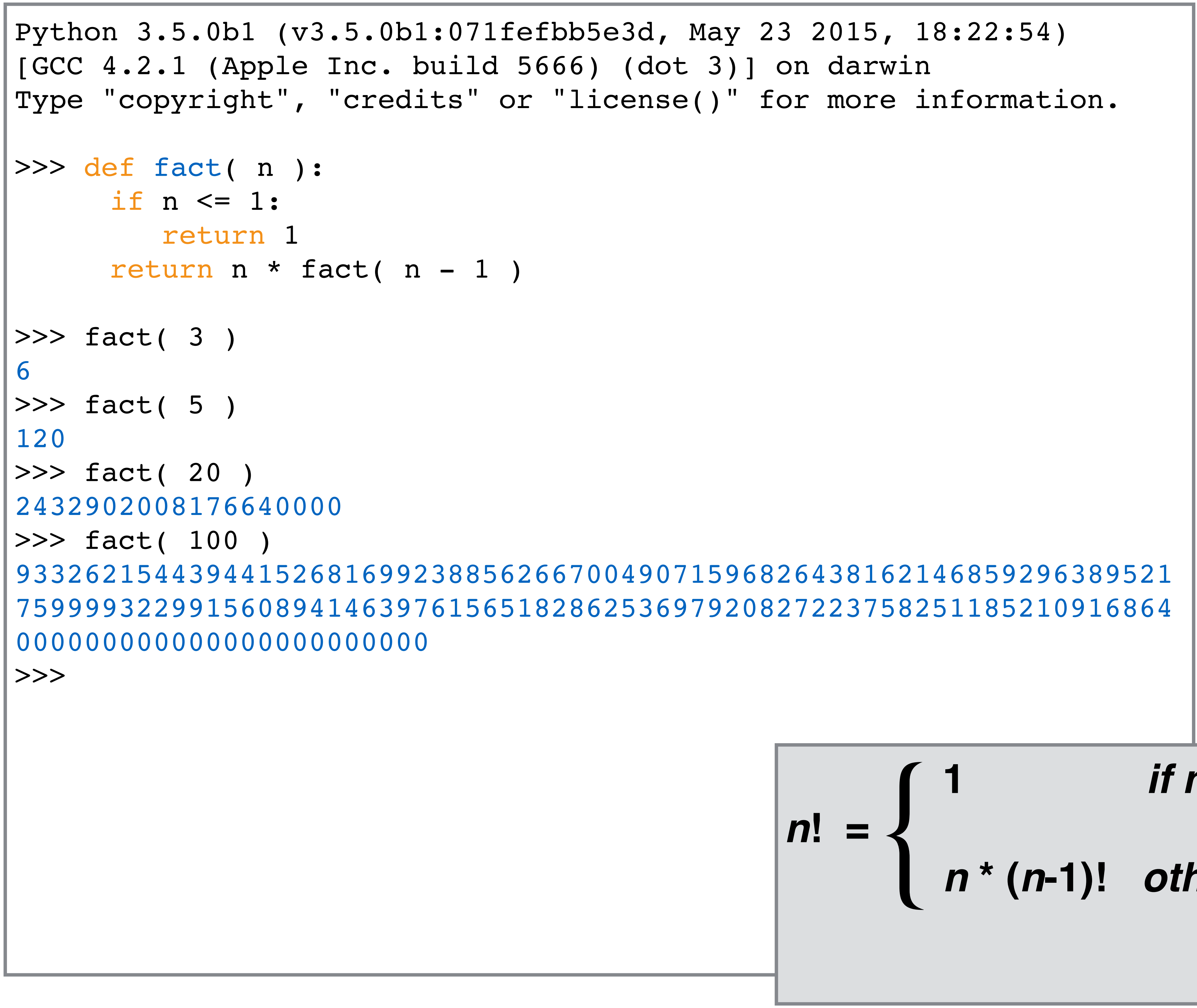
old ecx

esp —>

RECURSION



http://i.imgur.com/WcEmMx3.jpg

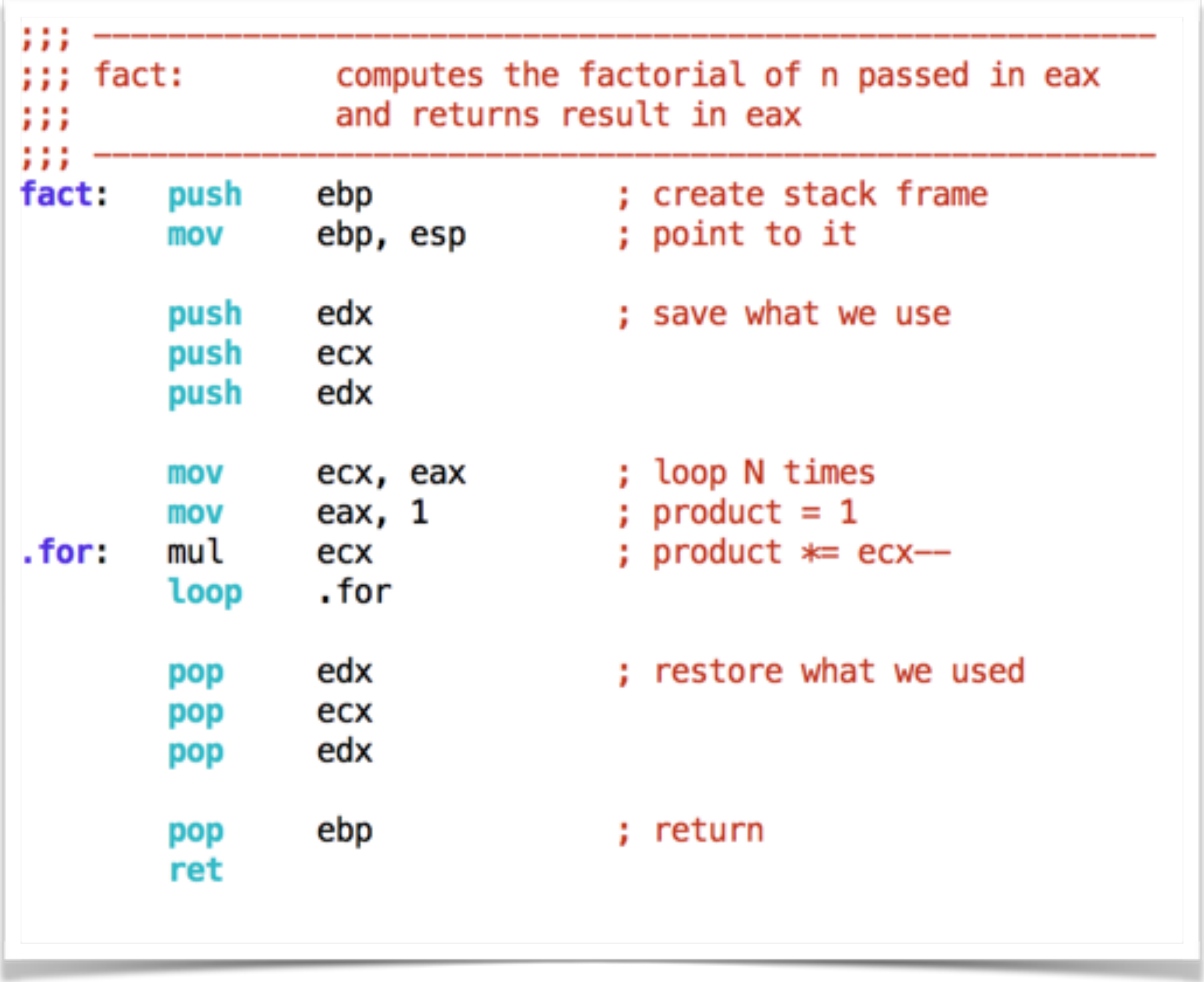


```
Python 3.5.0b1 (v3.5.0b1:071fefbb5e3d, May 23 2015, 18:22:54)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.

>>> def fact( n ):
        if n <= 1:
            return 1
        return n * fact( n - 1 )

>>> fact( 3 )
6
>>> fact( 5 )
120
>>> fact( 20 )
2432902008176640000
>>> fact( 100 )
9332621544394415268169923885626670049071596826438162146859296389521
7599993229915608941463976156518286253697920827223758251185210916864
000000000000000000000000
>>>
```

$$n! = \begin{cases} 1 & \textit{if } n<=1 \\ n * (n\text{-}1)! & \textit{otherwise} \end{cases}$$

Write the function **fact(n)** and call it from **main()**, *in Assembly*

Compare to Non-Recursive Version



```
;;; ---------------------------------------------------------
;;; fact:        computes the factorial of n passed in eax
;;;              and returns result in eax
;;; ---------------------------------------------------------
fact:   push    ebp             ; create stack frame
        mov     ebp, esp        ; point to it

        push    edx             ; save what we use
        push    ecx
        push    edx

        mov     ecx, eax        ; loop N times
        mov     eax, 1          ; product = 1
.for:   mul     ecx             ; product *= ecx--
        loop    .for

        pop     edx             ; restore what we used
        pop     ecx
        pop     edx

        pop     ebp             ; return
        ret
```

Question 1

- Compare the execution time of the recursive version of ***factorial()*** to its non-recursive version.

Question 2

- If the maximum stack size given to a program is 8 GBytes, how many terms could ***fact()*** compute, at most, if we didn't care about multiplication overflow?
  Note: We can get the default stack size linux uses with

```
        ulimit -s
```

Towers of Hanoi… in Assembly

- In Python first
- In Assembly next

https://media-cdn.tripadvisor.com/media/photo-s/0f/00/ee/18/ulun-danu-bratan-temple.jpg