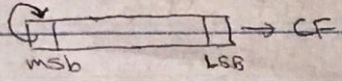
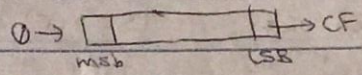
07/19 #RECAP

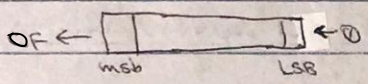
Shift Arithmetic Right:
SAR opnd, how many



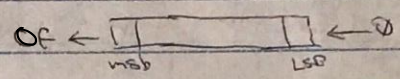
Shift Logical Right:
SHR



Shift Arithmetic Left:
SAL



= **Shift Logical Left**
SHL



Call Stack:

- Each methods = new stack frame
- Space for autovariable (lifespan of one function call)
- Diff. # of parameters → Diff. sizes of Stack
- Every stack frame requires return address
- Pointer to prev. frame (every stack has)
- Pointer to topmost frame is at %RBP or %EBP (frame pointer register)

Call Instruction:

Call Label ← Direct
call *operand ← Indirect (put this in → %RIP / %EIP)

Return Instruction

Ret

Function Invocation

PushQ	%RBP	Save old frame pointer	#Copy the value
movQ	%RSP, %RBP	make new frame	

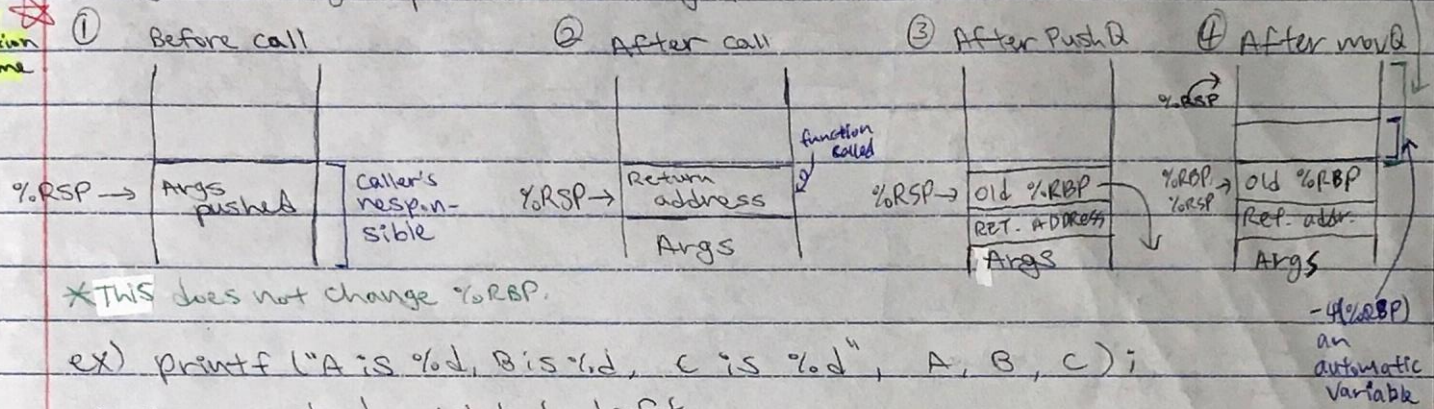
Function Exit

movQ	%RBP, %RSP	Restore previous stack pointer
PopQ	%RBP	Restore previous frame pointer
RET		Pop return address into %RIP

✡✡✡ very Important!

Construction & Deconstruction of stack frame

* Arguments get pushed in right to left order



* This does not change %RBP.

ex) printf ("A is %d, B is %d, C is %d", A, B, C);

Args are pushed right to left

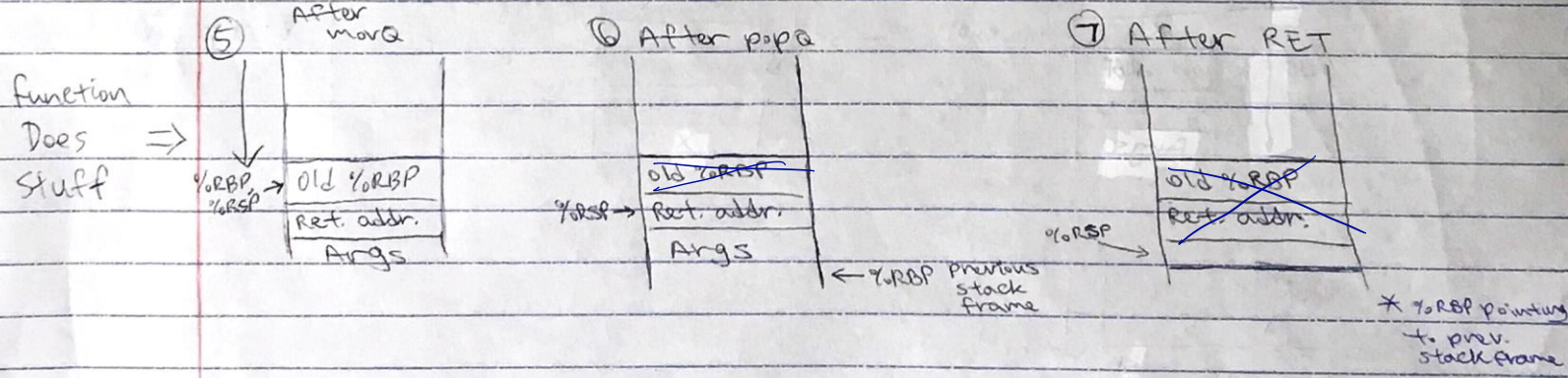
%RSP → fourth Pointer to "A is %d, B is %d, C is %d"

third " A

second " B

first " C

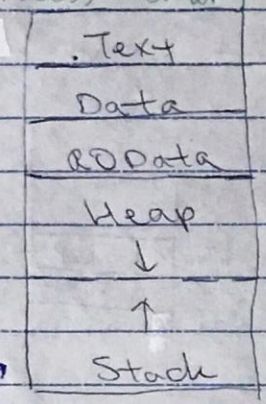
* 0(%RBP) is %RBP for previous stack frame



* same length Call Instruction: push the address of next instruction / line
- take the target and push it to the instruction pointer
- Call Stack exists in memory.
- %RSP points to top of Stack
- OS and Program determine initial location of Stack (i.e. initial value of %RSP)
- Address stack layout randomization makes different Stack location for different program invocation. → for security (3rd party cannot examine)

* C code → Assembly ... consistent output

Process Memory



← machine instruction, ~~fixed size, can't change~~

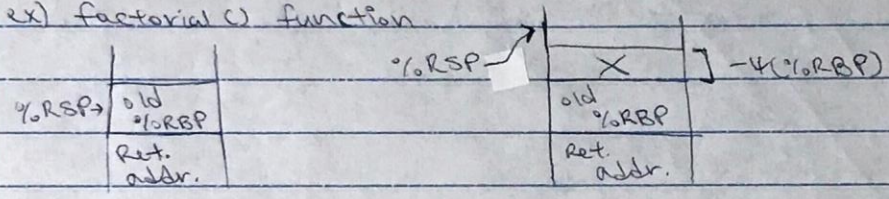
← fixed size, can change (ex. global, local variables, static)

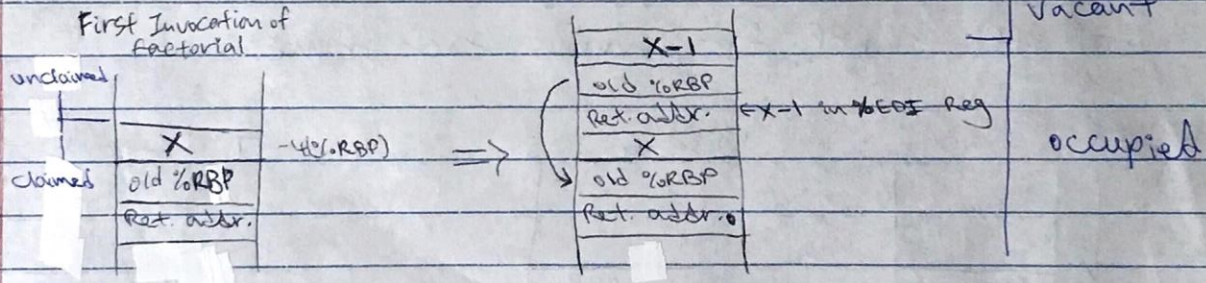
← readonly data (ex. constant, string)

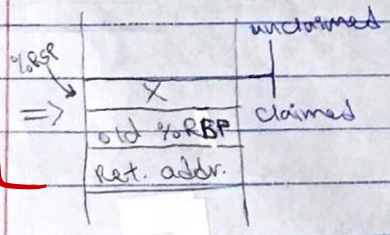
ASLR: This stack location will be diff.

know how to convert assembly file → c code

ex) factorial() function







1st Small Arg:	2nd Small Arg:
%EDI / %RDI	%EDI / %RSI

not static

* Auto variable exists as long as Stack frame exists.
* Static variable does not change!

Very Good Visual Explanation — Call Stack

http://duartes.org/gustavo/blog/post/epilogues-canaries-buffer-overflows/

Dynamic Memory:

```
#include <STDLIB.H>
void * malloc (size_T);   // this many bytes
void free (void *); //deallocation
```

* Anything you allocate, must be deallocated!

* Never dereference a pointer.
* Do not use a pointer that you freed.
  - Memory leak
  - Dangling pointer
* Never free anything more than once.
* Never exceed the size of what you asked for.
* Never free any pointer that you did not dynamically allocate
* Don't do pointer arithmetic (ex. don't free p+1 or p-1)
* Initialize chunk before using

know this

*

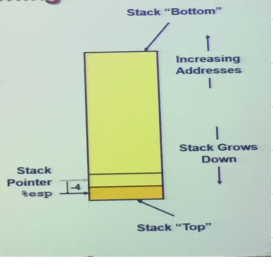
```
void * realloc (void *, size_T newsize);
```
- free previous pointer and create memory size of newsize

```
void * calloc (size_T NELEM, size_T ELEMSIZE);
// multiplies two sizes to get one final size of allocation
```

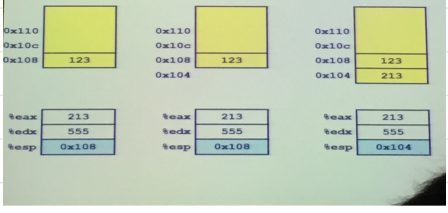
IA32 Stack Pushing

ushing

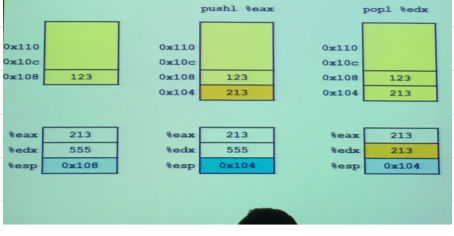
- pushl *Src*
- Fetch operand at *Src*
- Decrement %esp by 4
- Write operand at address given by %esp



Stack Operation Examples



Stack Operation Examples



Procedure Control Flow

- Use stack to support procedure call and return

rocedure call:

`call label` Push return address on stack; Jump to `label`

eturn address value

- Address of instruction beyond `call`
- Example from disassembly

```
804854e: e8 3d 06 00 00     call   8048b90 <main>
8048553: 50                 pushl  %eax
```

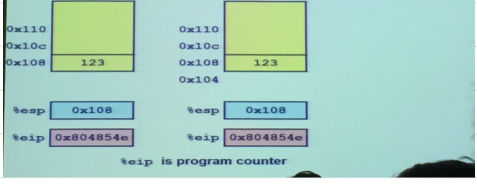
  - Return address = 0x8048553

rocedure return:

- `ret` Pop address from stack; Jump to address

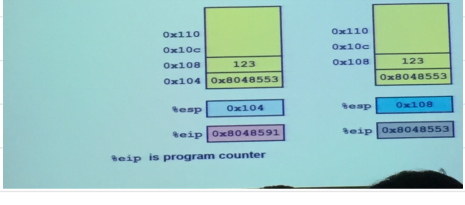
Procedure Call Example

```
804854e:  e8 3d 06 00 00      call   8048b90 <main>
8048553:  50                  pushl  %eax
```



Procedure Return Example

```
8048591:  c3                  ret
```



Stack-Based Languages

anguages that Support Recursion

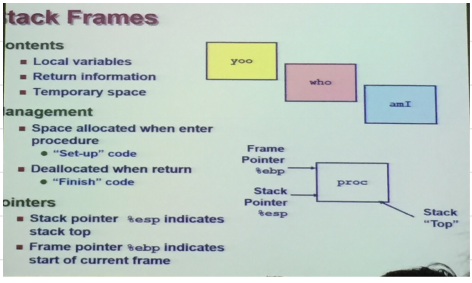
- e.g., C, Pascal, Java
- Code must be "*Reentrant*"
  - Multiple simultaneous instantiations of single procedure
- Need some place to store state of each instantiation
  - Arguments
  - Local variables
  - Return pointer

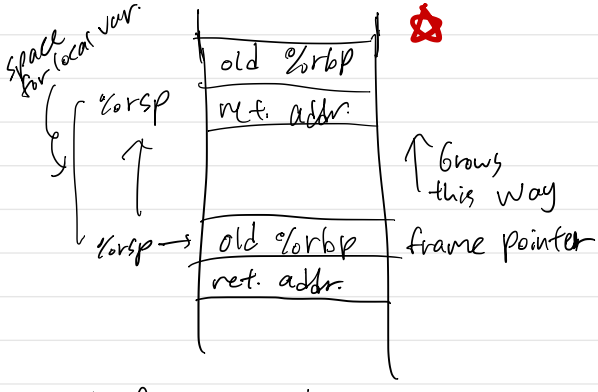
tack Discipline

- State for given procedure needed for limited time
  - From when called to when return
- Callee returns before caller does

tack Allocated in *Frames*

- state for single procedure instantiation





- we depend on callee or caller function to restore.
- caller, things at the frame pointer is not changed
- Diff. functions = diff. stack frame pointer

[5 points EXTRA CREDIT]. List some words that can be spelled with only the sixteen hexadecimal digits (0-9, A-F) that fit into 32 or fewer bits.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
o L s g

- 0xBABEFACE
- 0xFEE15BAD
- 0x18BEEF
- 0xDEADFACE
- 0x8BADF00D

- working with hex addr. in stack
- working with size of union & struct
- Pass by val vs Pass by ref.

mid-term