Software Security
COSC 466/566
Spring 2023

Dr. Doowon Kim

THE UNIVERSITY OF
TENNESSEE

I

IT <IT@utk.edu>
To: Kim, Doowon

IT@utk.edu has shared a printer with you

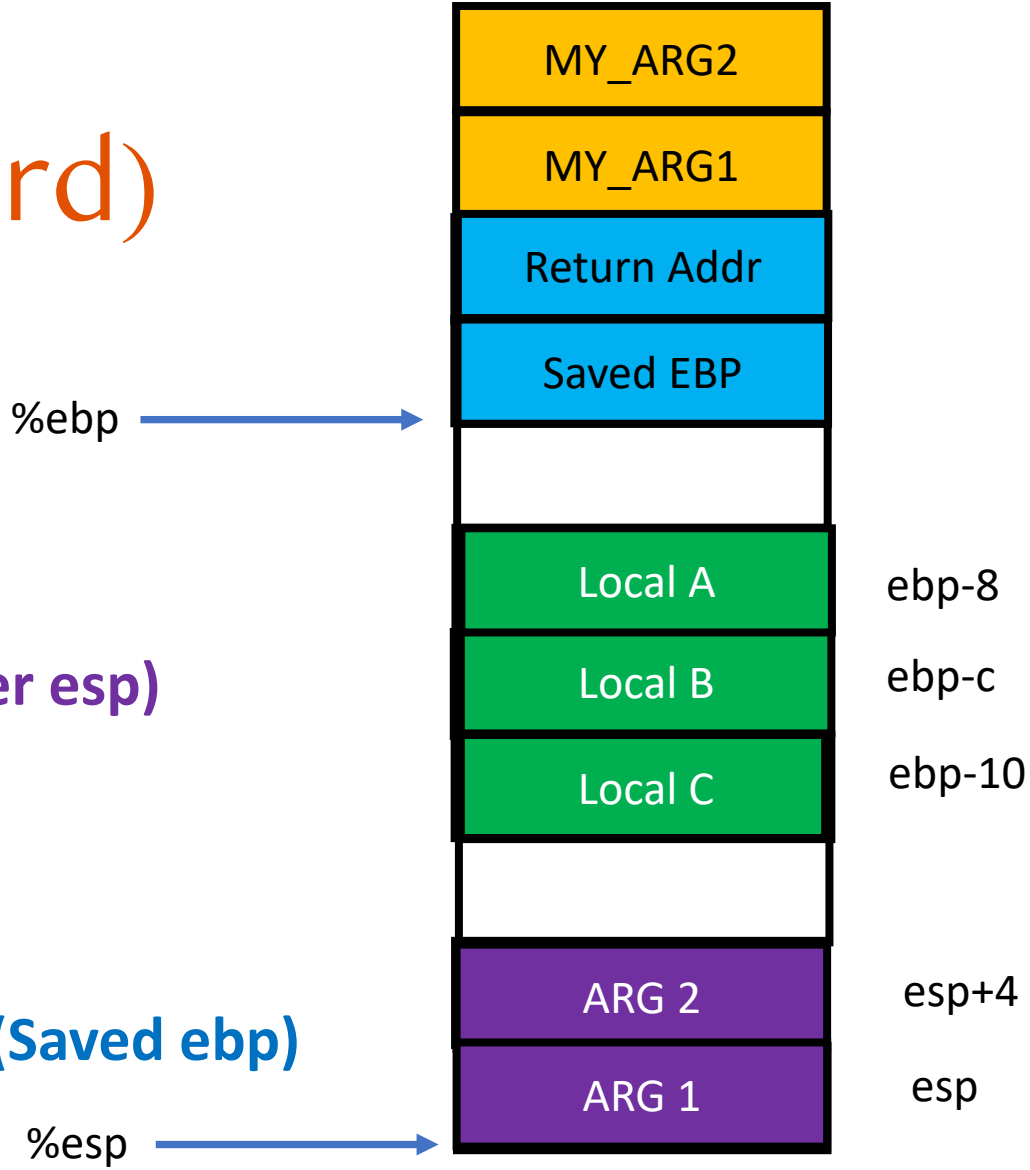
To accept or reject the HP LaserJet 400 M401n printer, click the button below and use the Google Cloud Print management page.

Add Printer

You are receiving this email at the account **dkim52@utk.edu** because a user has shared a printer with you through Google Cloud Print.

Stack (Grows Downward)

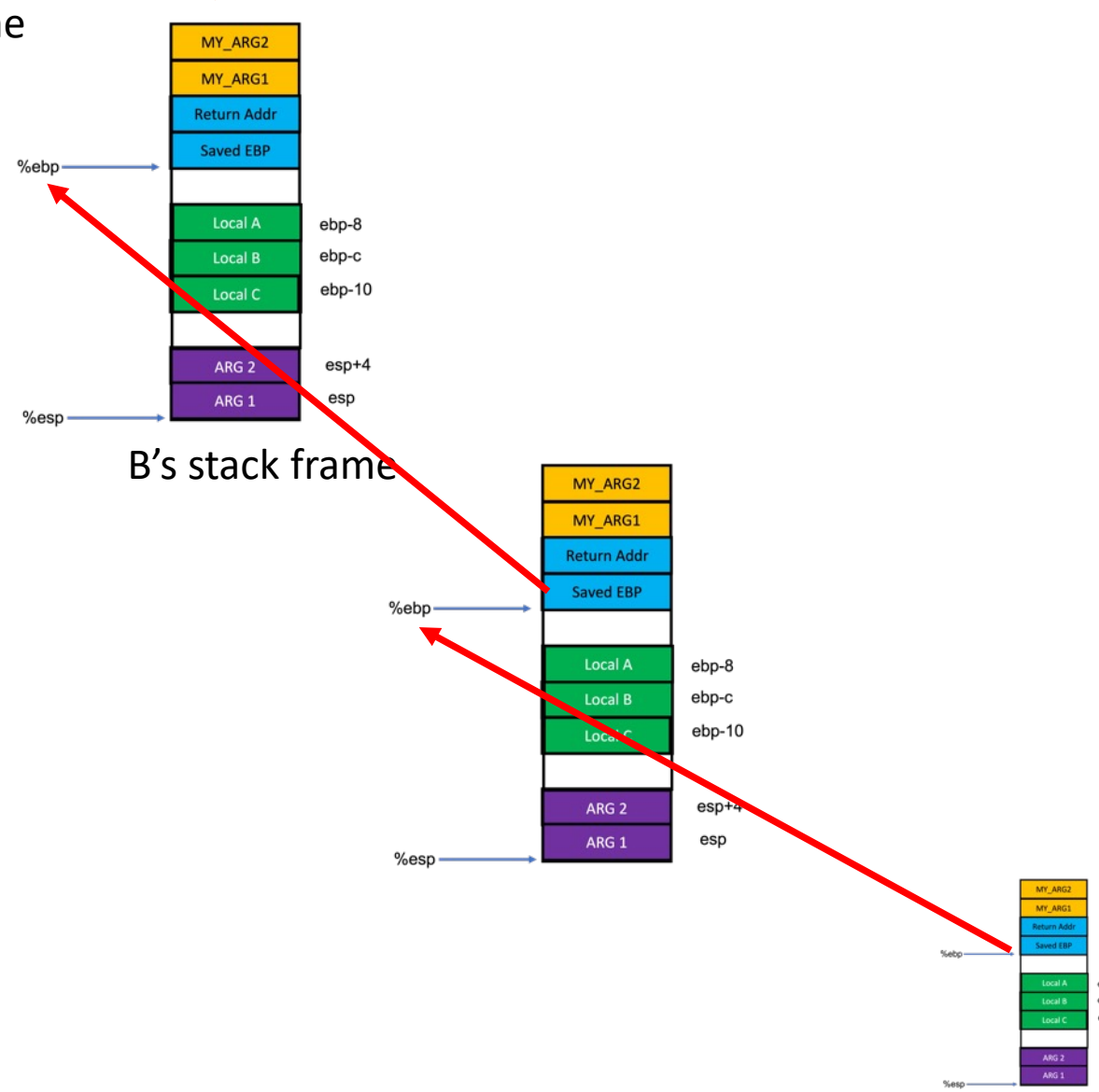
- Defines a variable scope of a function
  - **Local variables (negative index over ebp)**
  - **Arguments (positive index over ebp)**
  - **Function call arguments (positive index over esp)**
- Maintains nested function calls
  - **Return target (return address)**
  - **Local variables of the upper level function (Saved ebp)**
- Starts at `%ebp` (bottom), ends at `%esp` (top)

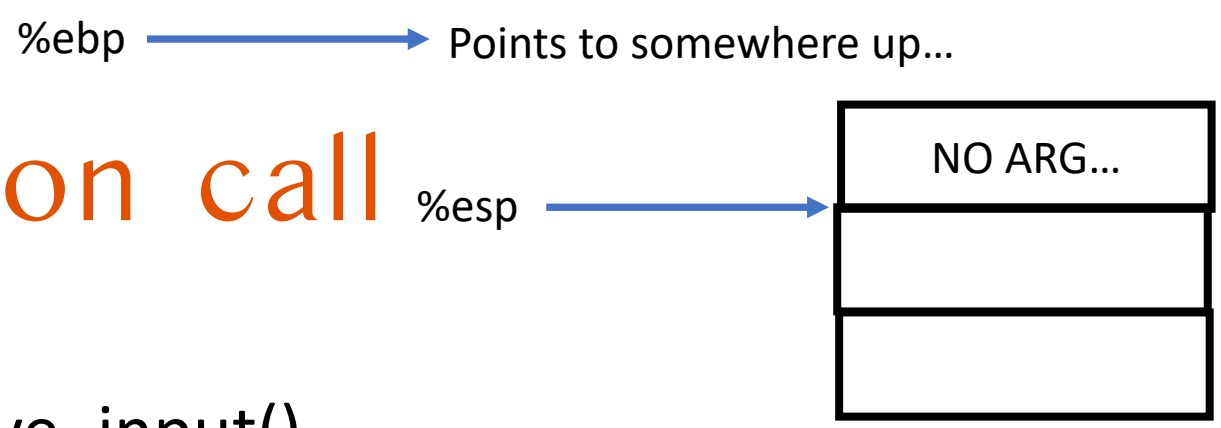


Stack (Grows Downward)

If a() calls b() and then b() calls c()….

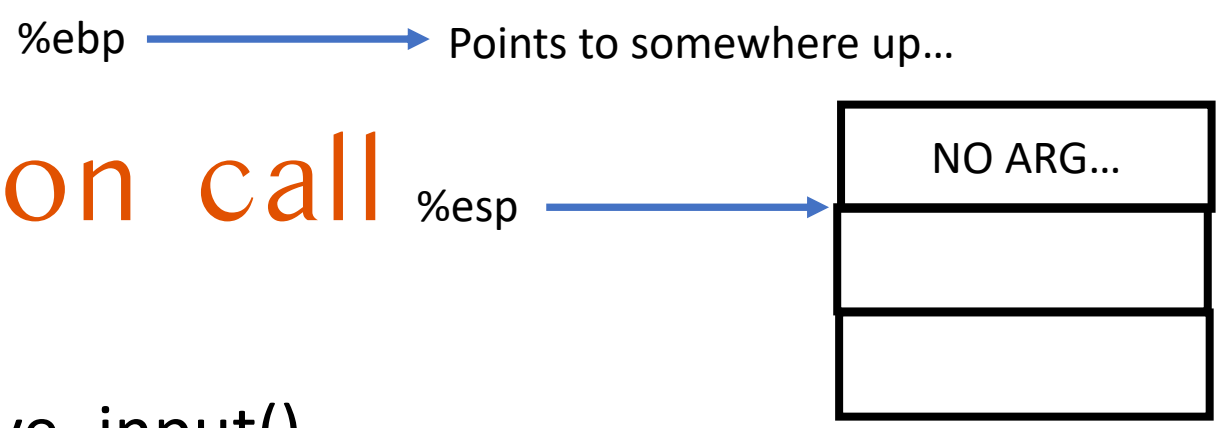
When it returns, we restore ebp!





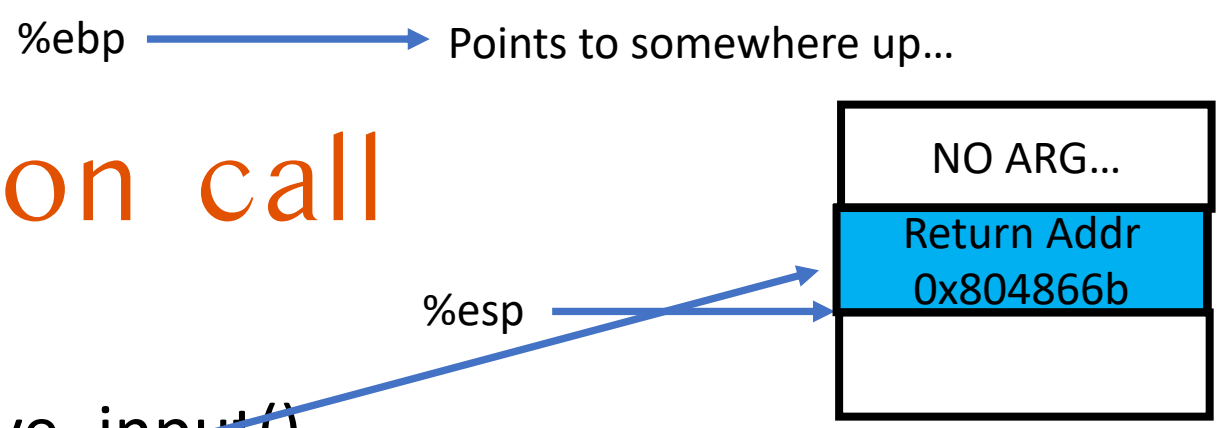
Example – Function call

- In bof-level0, main() calls receive_input()
  - `call   0x8048570 <receive_input>`
  - `0x0804866b <+11>:    xor    %eax,%eax`
- Head of receive_input
  - `0x08048570 <+0>:push   %ebp`
  - `0x08048571 <+1>:mov    %esp,%ebp`
  - `0x08048573 <+3>:push   %esi`
  - `0x08048574 <+4>:sub    $0x54,%esp`



Example – Function call

- In bof-level0, main() calls receive_input()
  - **`call 0x8048570 <receive_input>`**
  - `0x0804866b <+11>: xor %eax,%eax`
- Head of receive_input
  - `0x08048570 <+0>: push %ebp`
  - `0x08048571 <+1>: mov %esp,%ebp`
  - `0x08048573 <+3>: push %esi`
  - `0x08048574 <+4>: sub $0x54,%esp`

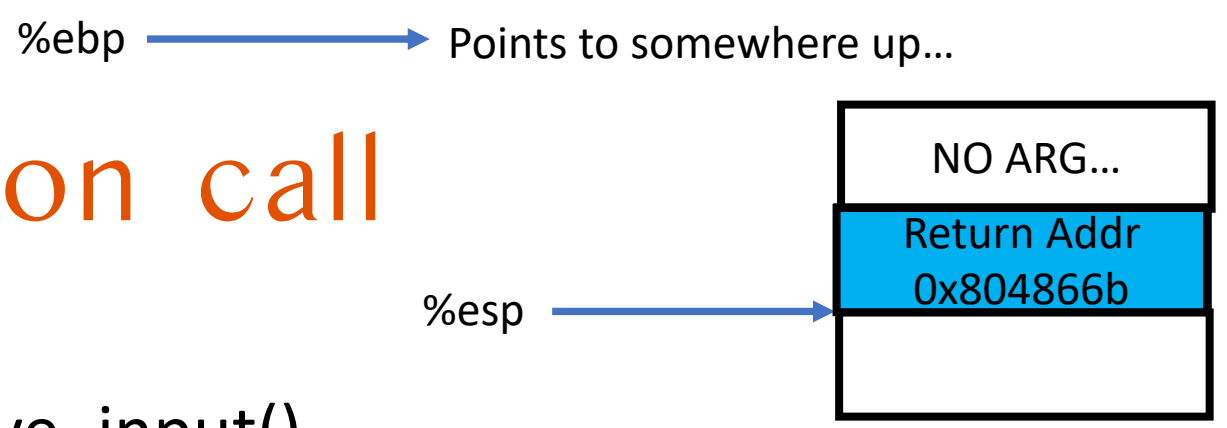


Example – Function call

- In bof-level0, main() calls receive_input()
  - **`call 0x8048570 <receive_input>`**
  - `0x0804866b <+11>: xor %eax,%eax`
- Head of receive_input
  - `0x08048570 <+0>: push %ebp`
  - `0x08048571 <+1>: mov %esp,%ebp`
  - `0x08048573 <+3>: push %esi`
  - `0x08048574 <+4>: sub $0x54,%esp`

Call: push the address to return to the stack, then jump!

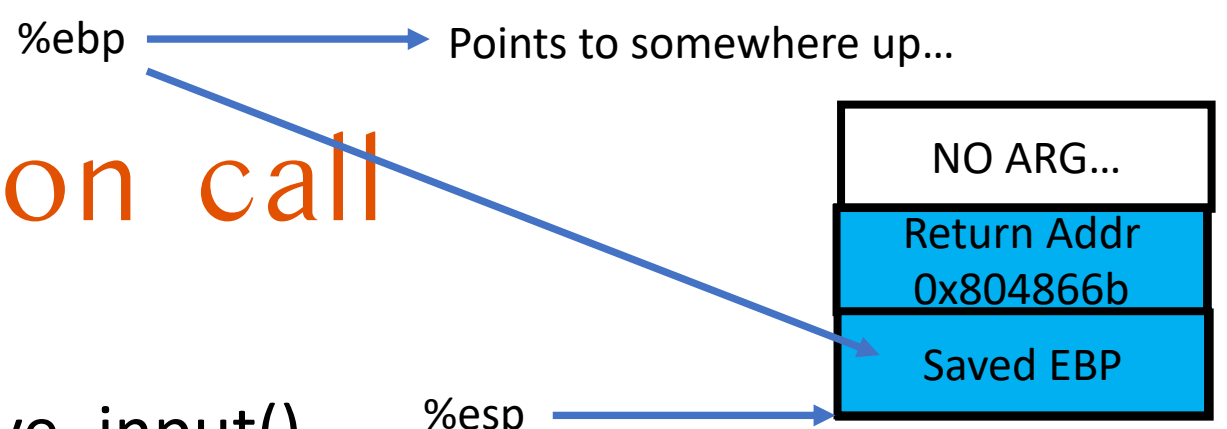
```
push %eip   -- points the next instruction
jmp 0x8048570 <receive_input>
```



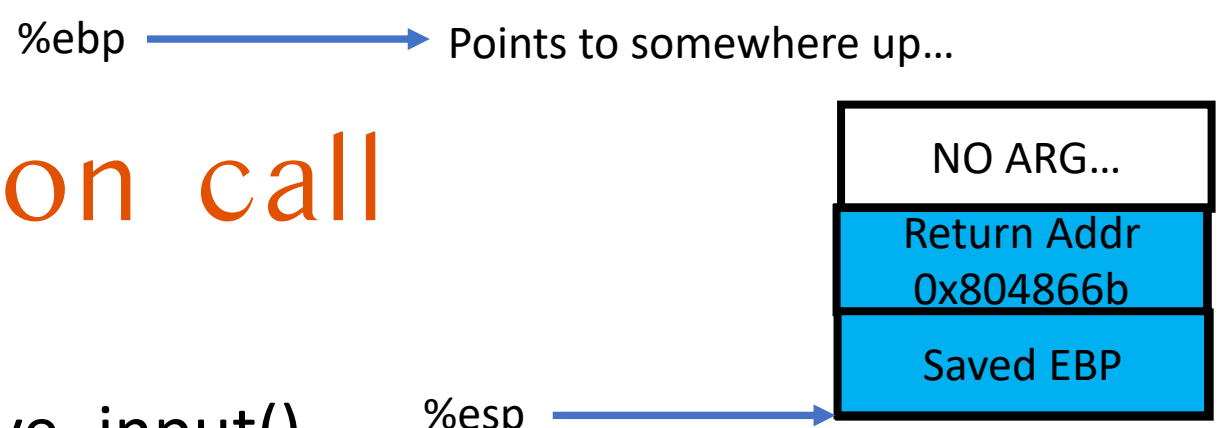
Example – Function call

- In bof-level0, main() calls receive_input()
  - `call   0x8048570 <receive_input>`
  - `0x0804866b <+11>:    xor    %eax,%eax`
- Head of receive_input
  - **`0x08048570 <+0>: push   %ebp`**
  - `0x08048571 <+1>: mov    %esp,%ebp`
  - `0x08048573 <+3>: push   %esi`
  - `0x08048574 <+4>: sub    $0x54,%esp`

Example – Function call



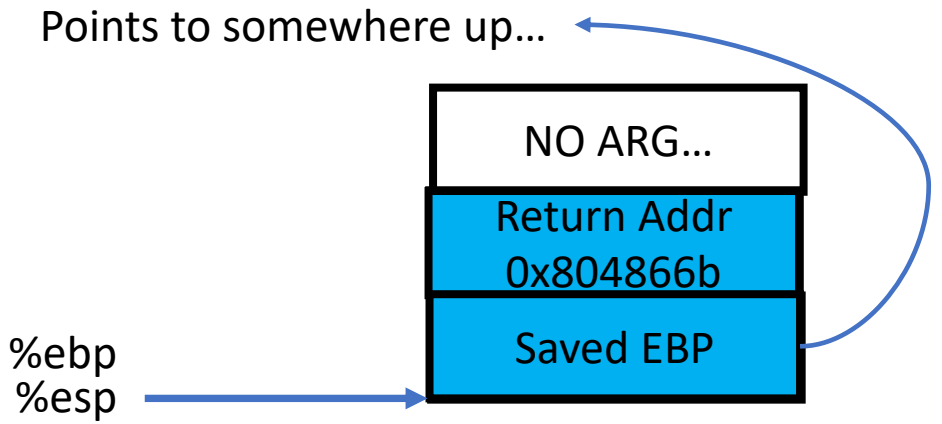
- In bof-level0, main() calls receive_input()
  - `call  0x8048570 <receive_input>`
  - `0x0804866b <+11>:  xor  %eax,%eax`
- Head of receive_input
  - **`0x08048570 <+0>: push  %ebp`**
  - `0x08048571 <+1>: mov  %esp,%ebp`
  - `0x08048573 <+3>: push  %esi`
  - `0x08048574 <+4>: sub  $0x54,%esp`



Example – Function call

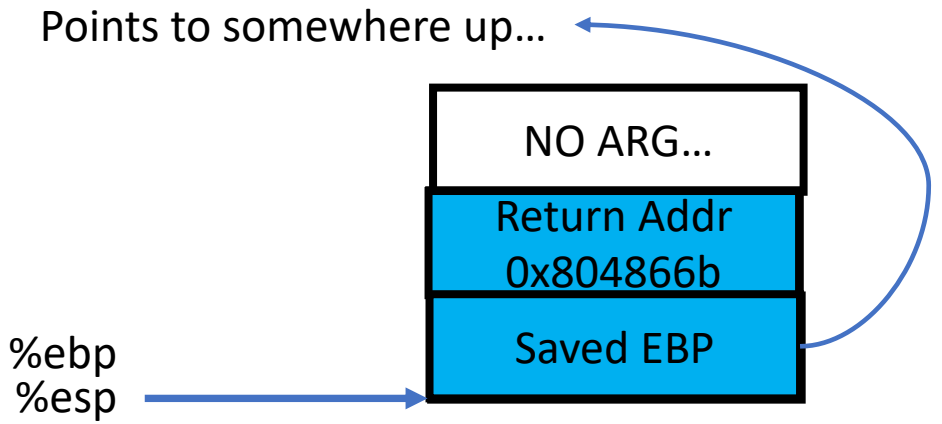
- In bof-level0, main() calls receive_input()
  - `call  0x8048570 <receive_input>`
  - `0x0804866b <+11>:  xor  %eax,%eax`
- Head of receive_input
  - `0x08048570 <+0>: push  %ebp`
  - **`0x08048571 <+1>: mov  %esp,%ebp`**
  - `0x08048573 <+3>: push  %esi`
  - `0x08048574 <+4>: sub  $0x54,%esp`

Example – Function call



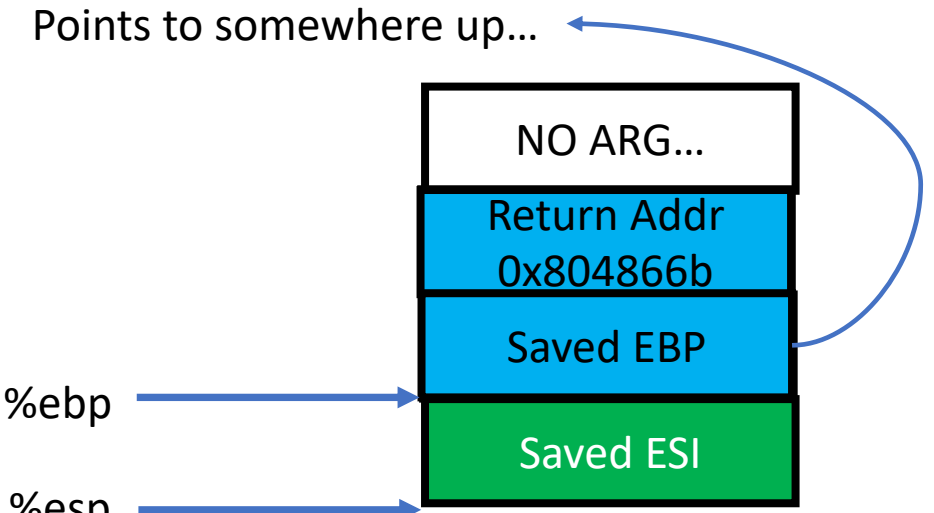
- In bof-level0, main() calls receive_input()
  - `call  0x8048570 <receive_input>`
  - `0x0804866b <+11>:  xor  %eax,%eax`
- Head of receive_input
  - `0x08048570 <+0>: push  %ebp`
  - **`0x08048571 <+1>: mov  %esp,%ebp`**
  - `0x08048573 <+3>: push  %esi`
  - `0x08048574 <+4>: sub  $0x54,%esp`

Example – Function call



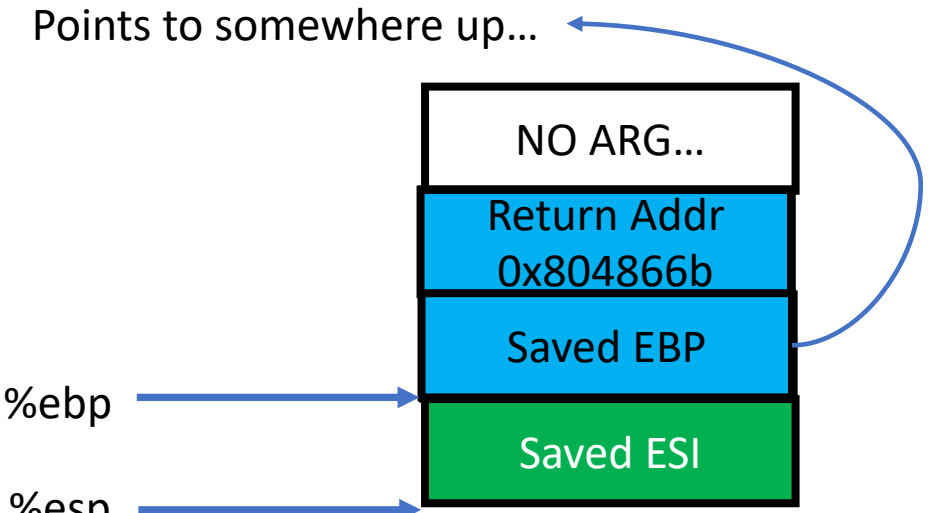
- In bof-level0, main() calls receive_input()
  - `call  0x8048570 <receive_input>`
  - `0x0804866b <+11>:  xor  %eax,%eax`
- Head of receive_input
  - `0x08048570 <+0>: push  %ebp`
  - `0x08048571 <+1>: mov  %esp,%ebp`
  - **`0x08048573 <+3>: push  %esi`**
  - `0x08048574 <+4>: sub  $0x54,%esp`

Example – Function call



- In bof-level0, main() calls receive_input()
  - `call  0x8048570 <receive_input>`
  - `0x0804866b <+11>:  xor  %eax,%eax`
- Head of receive_input
  - `0x08048570 <+0>: push  %ebp`
  - `0x08048571 <+1>: mov  %esp,%ebp`
  - **`0x08048573 <+3>: push  %esi`**
  - `0x08048574 <+4>: sub  $0x54,%esp`

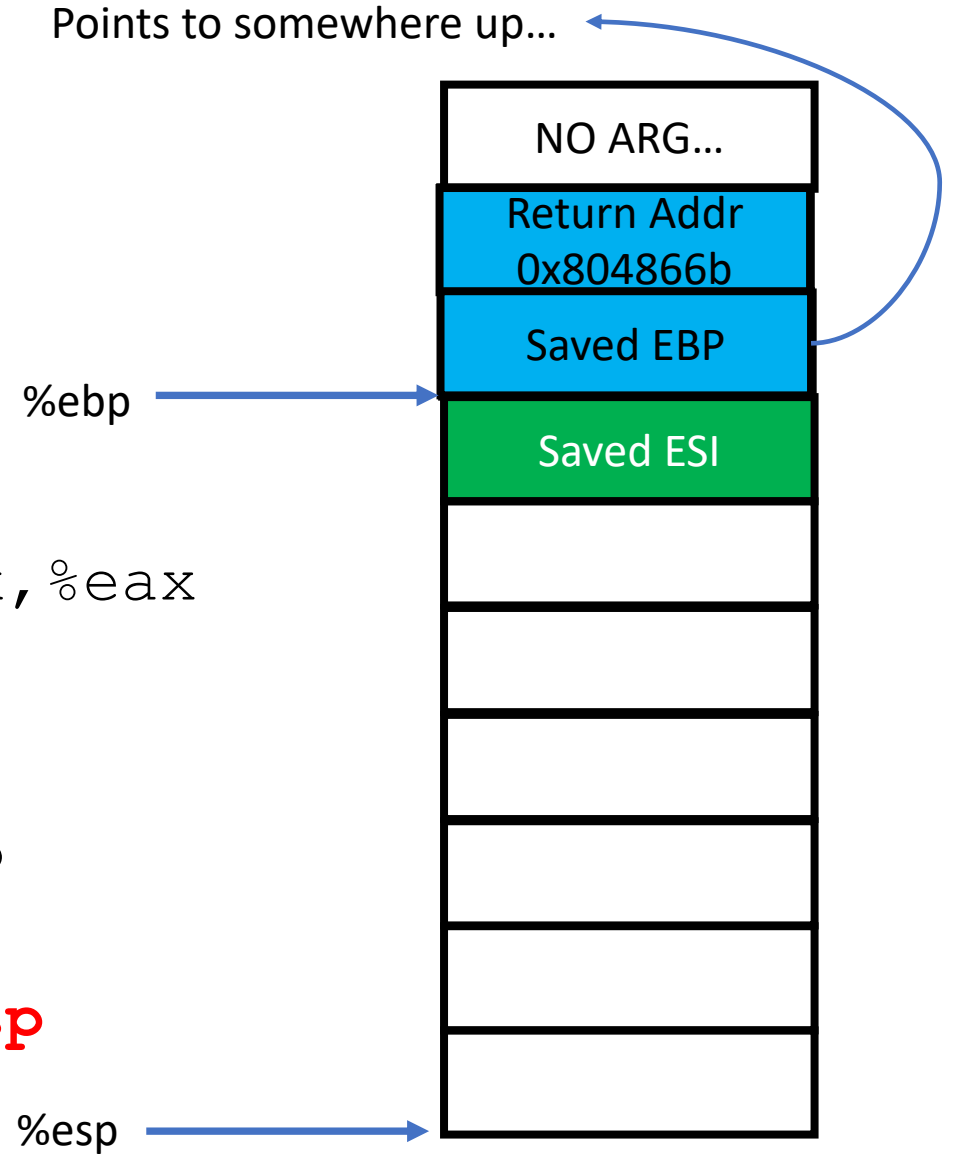
Example – Function call



- In bof-level0, main() calls receive_input()
  - `call  0x8048570 <receive_input>`
  - `0x0804866b <+11>:  xor  %eax,%eax`
- Head of receive_input
  - `0x08048570 <+0>: push  %ebp`
  - `0x08048571 <+1>: mov  %esp,%ebp`
  - `0x08048573 <+3>: push  %esi`
  - **`0x08048574 <+4>: sub  $0x54,%esp`**

Example – Function call

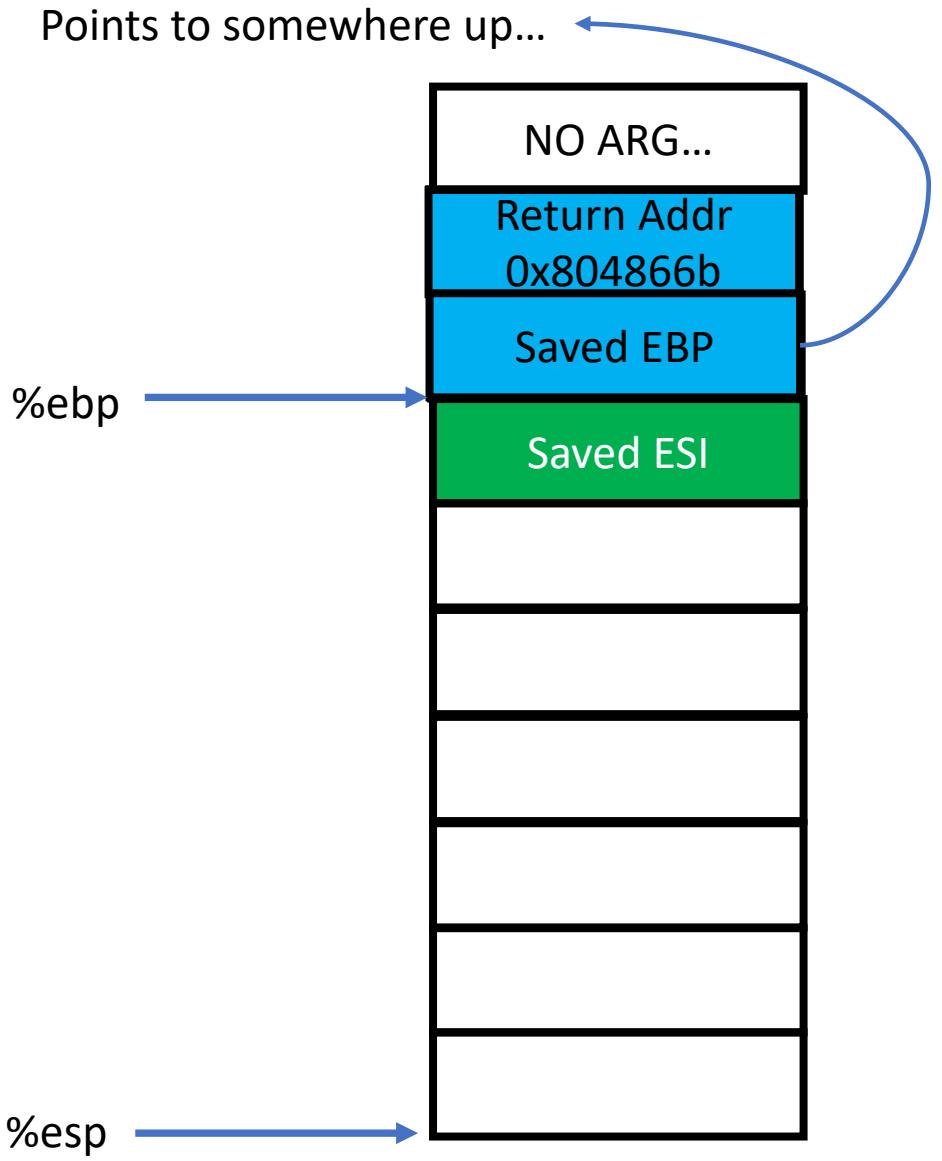
- In bof-level0, main() calls receive_input()
  - `call  0x8048570 <receive_input>`
  - `0x0804866b <+11>:  xor  %eax,%eax`
- Head of receive_input
  - `0x08048570 <+0>: push  %ebp`
  - `0x08048571 <+1>: mov  %esp,%ebp`
  - `0x08048573 <+3>: push  %esi`
  - **`0x08048574 <+4>: sub  $0x54,%esp`**



Example – Function call

- Call printf?

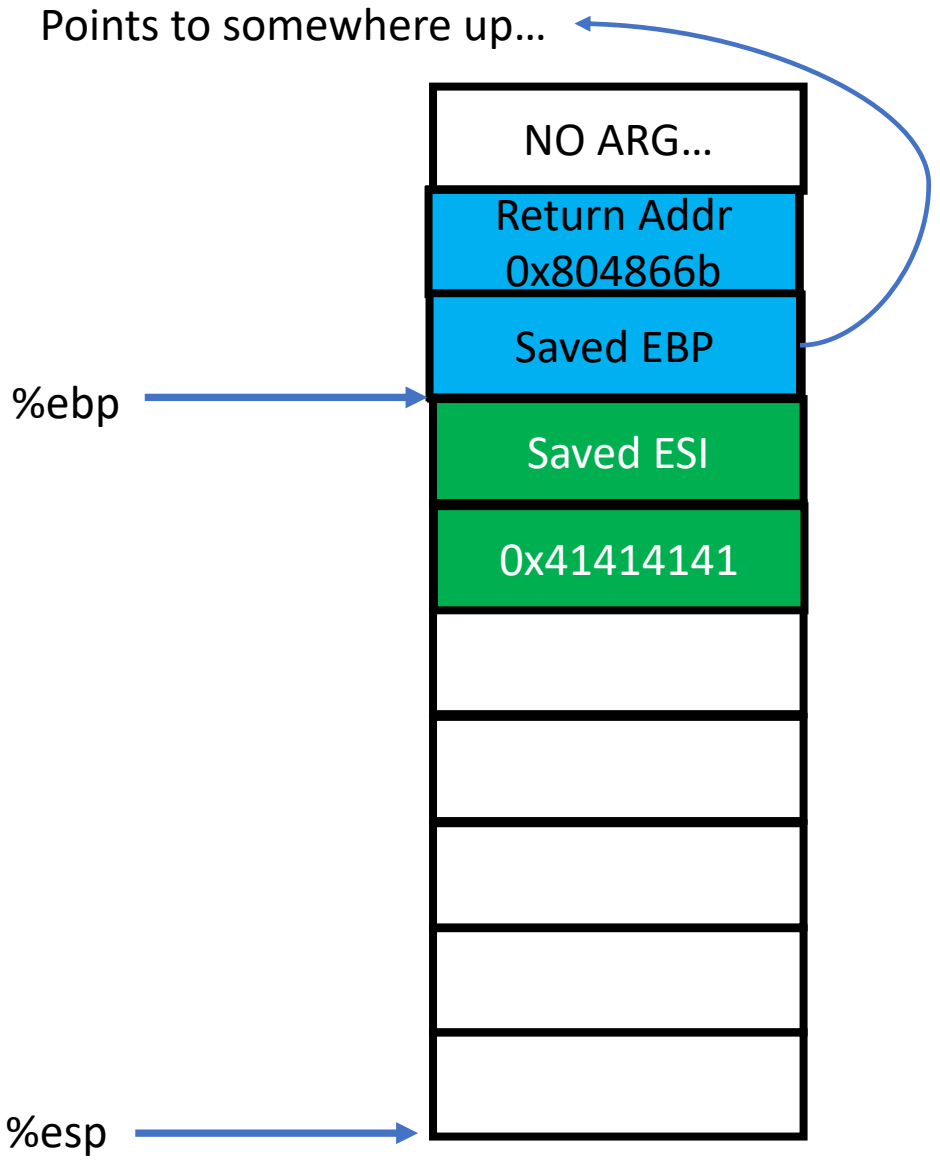
```
movl    $0x41414141,-0x8(%ebp)
movl    $0x42424242,-0xc(%ebp)
lea     0x8048727,%eax
mov     -0x8(%ebp),%ecx
mov     -0xc(%ebp),%edx
mov     %eax,(%esp)
mov     %ecx,0x4(%esp)
mov     %edx,0x8(%esp)
call    0x8048370 <printf@plt>
```



Example – Function call

- Call printf?

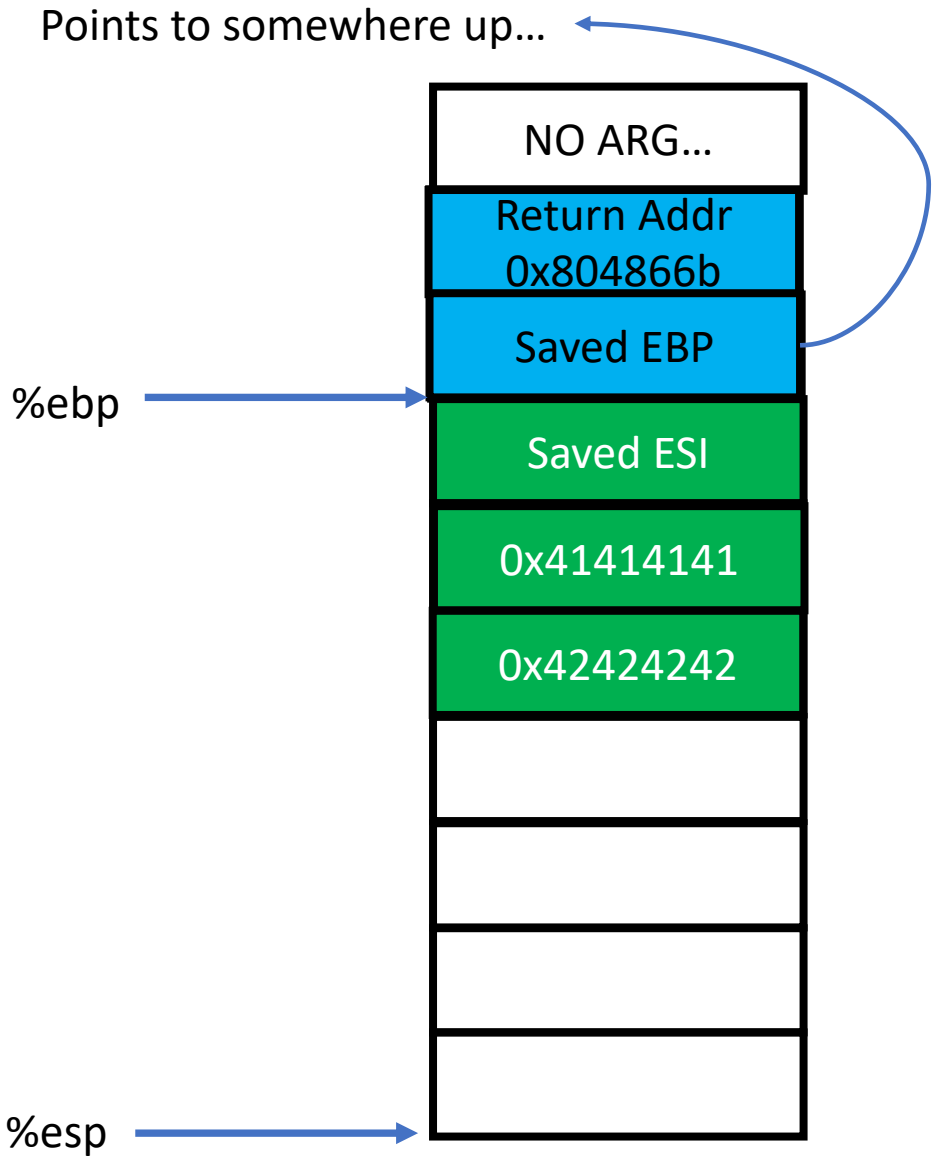
```
movl    $0x41414141,-0x8(%ebp)
movl    $0x42424242,-0xc(%ebp)
lea     0x8048727,%eax
mov     -0x8(%ebp),%ecx
mov     -0xc(%ebp),%edx
mov     %eax,(%esp)
mov     %ecx,0x4(%esp)
mov     %edx,0x8(%esp)
call    0x8048370 <printf@plt>
```



Example – Function call

- Call printf?

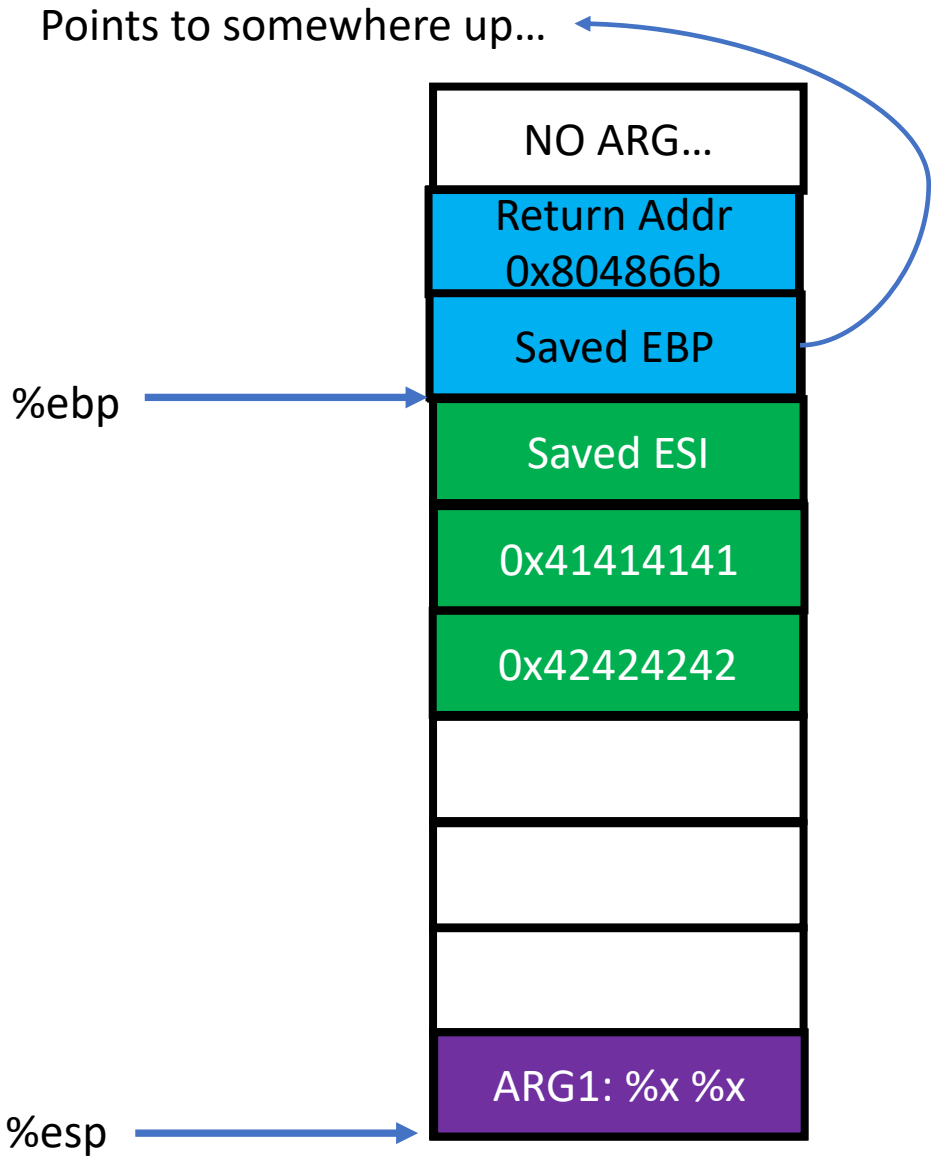
```
movl    $0x41414141,-0x8(%ebp)
movl    $0x42424242,-0xc(%ebp)
lea     0x8048727,%eax
mov     -0x8(%ebp),%ecx
mov     -0xc(%ebp),%edx
mov     %eax,(%esp)
mov     %ecx,0x4(%esp)
mov     %edx,0x8(%esp)
call    0x8048370 <printf@plt>
```



Example – Function call

- Call printf?

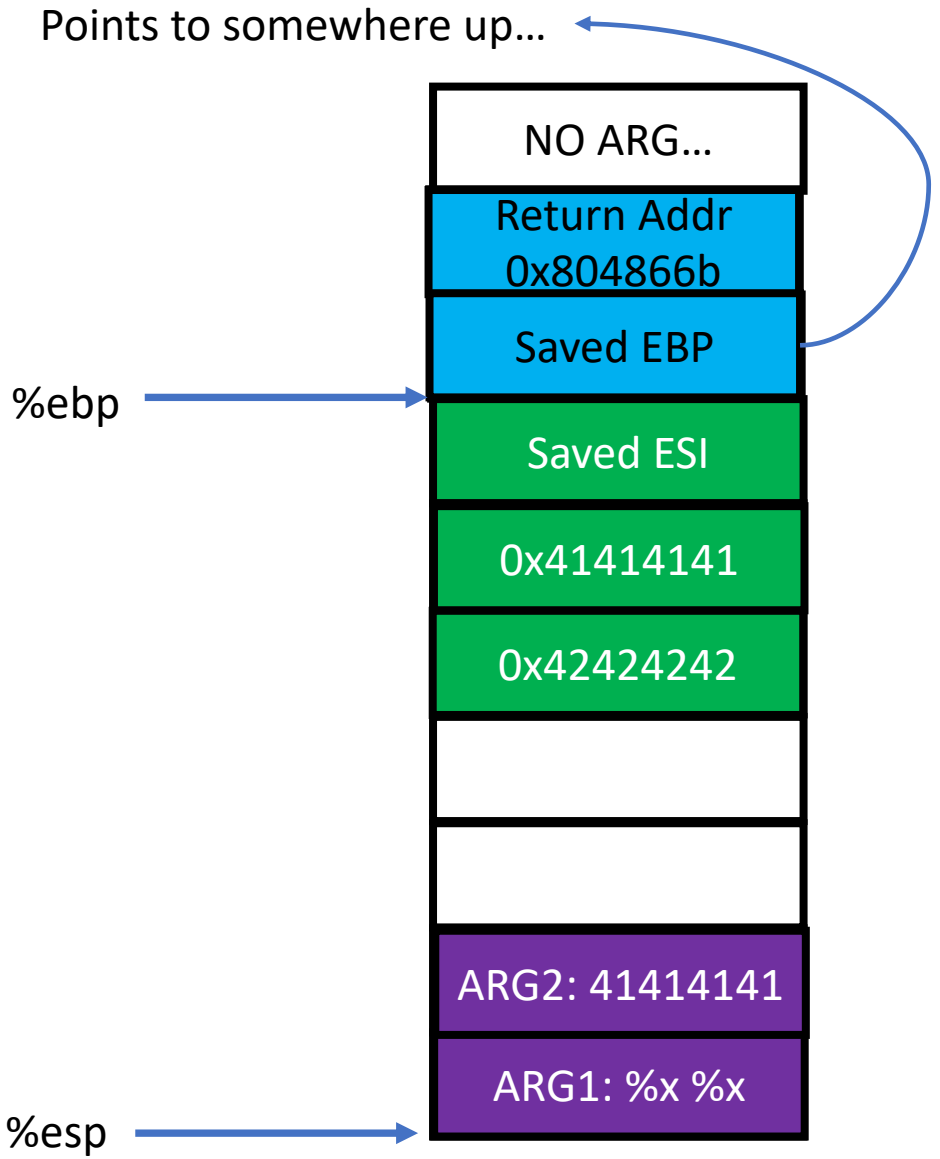
```
movl    $0x41414141,-0x8(%ebp)
movl    $0x42424242,-0xc(%ebp)
lea     0x8048727,%eax
mov     -0x8(%ebp),%ecx
mov     -0xc(%ebp),%edx
mov     %eax,(%esp)
mov     %ecx,0x4(%esp)
mov     %edx,0x8(%esp)
call    0x8048370 <printf@plt>
```



Example – Function call

- Call printf?

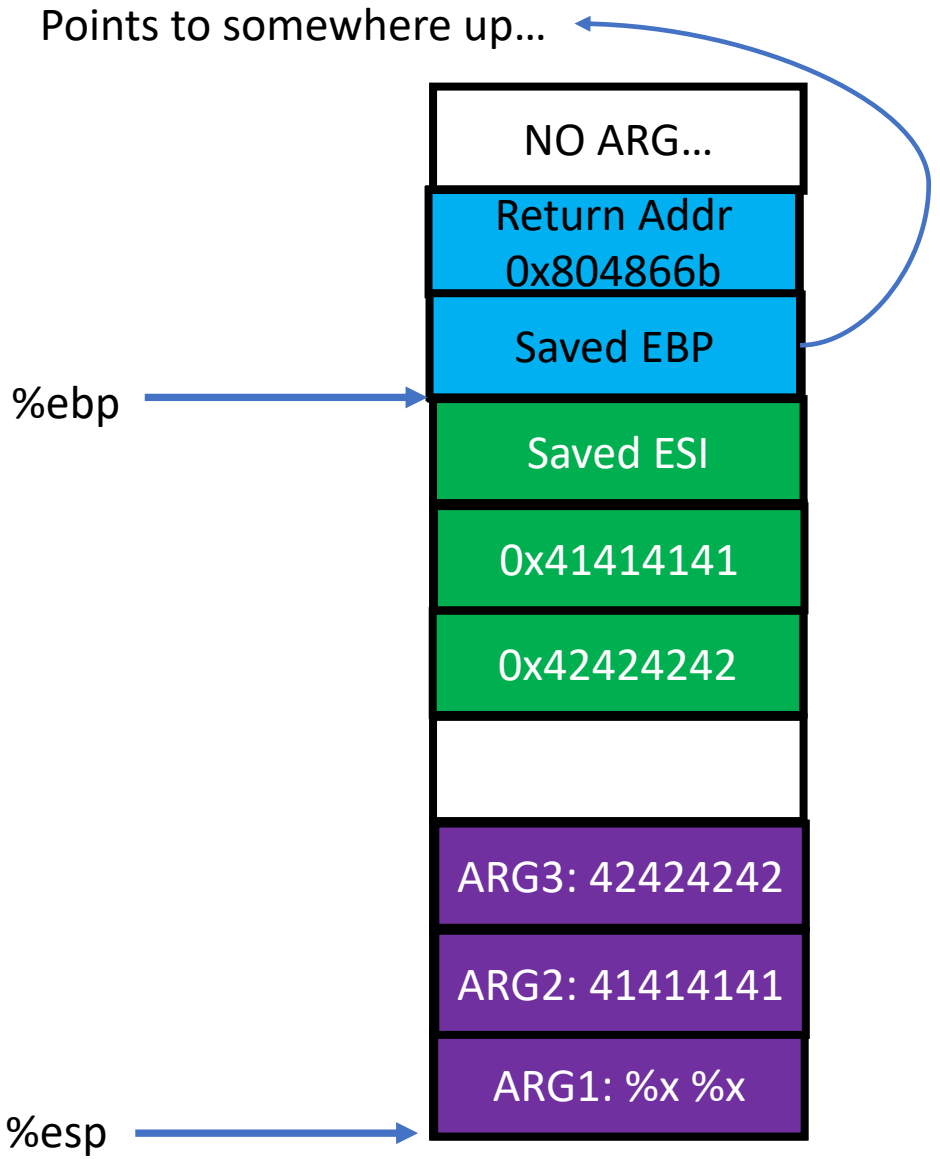
```
movl    $0x41414141,-0x8(%ebp)
movl    $0x42424242,-0xc(%ebp)
lea     0x8048727,%eax
mov     -0x8(%ebp),%ecx
mov     -0xc(%ebp),%edx
mov     %eax,(%esp)
mov     %ecx,0x4(%esp)
mov     %edx,0x8(%esp)
call    0x8048370 <printf@plt>
```



Example – Function call

- Call printf?

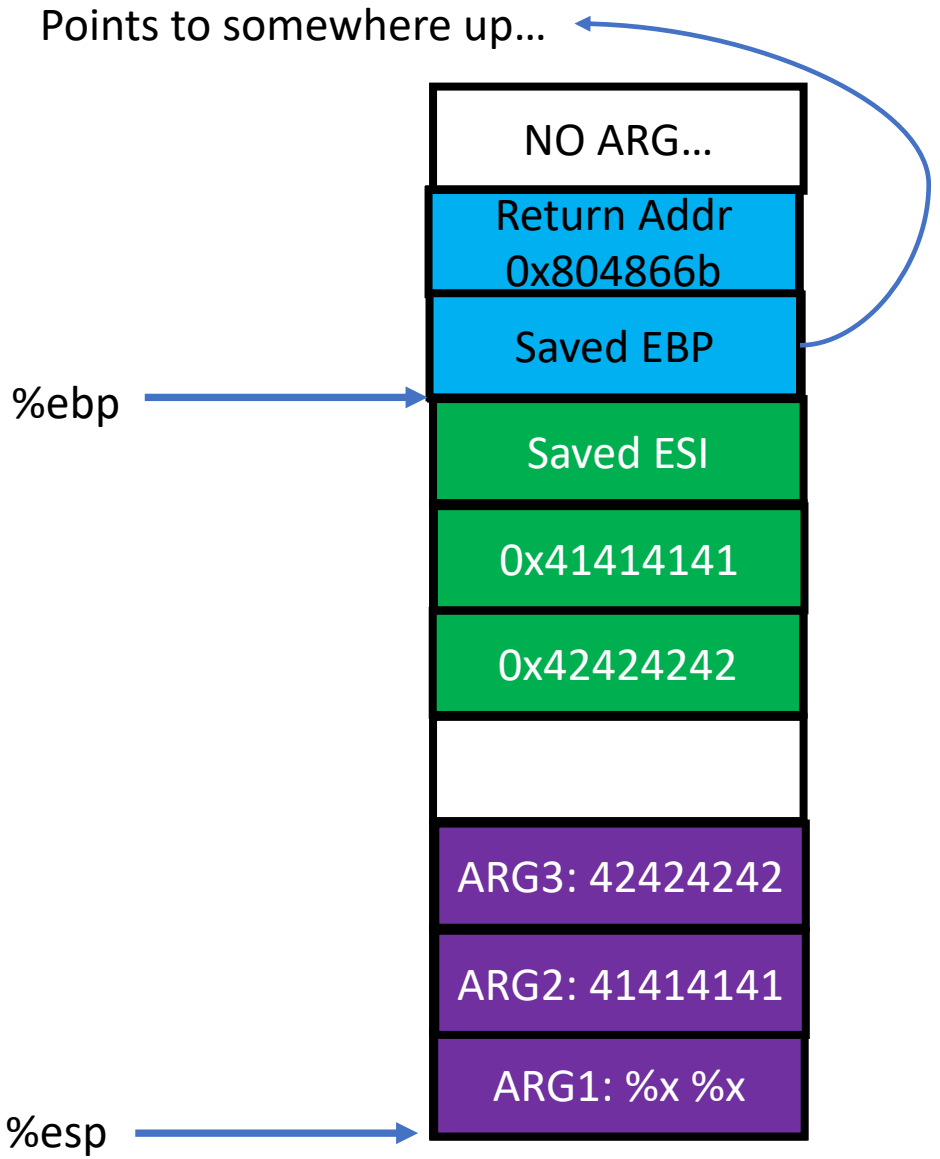
```
movl    $0x41414141,-0x8(%ebp)
movl    $0x42424242,-0xc(%ebp)
lea     0x8048727,%eax
mov     -0x8(%ebp),%ecx
mov     -0xc(%ebp),%edx
mov     %eax,(%esp)
mov     %ecx,0x4(%esp)
mov     %edx,0x8(%esp)
call    0x8048370 <printf@plt>
```



Example – Function call

- Call printf?

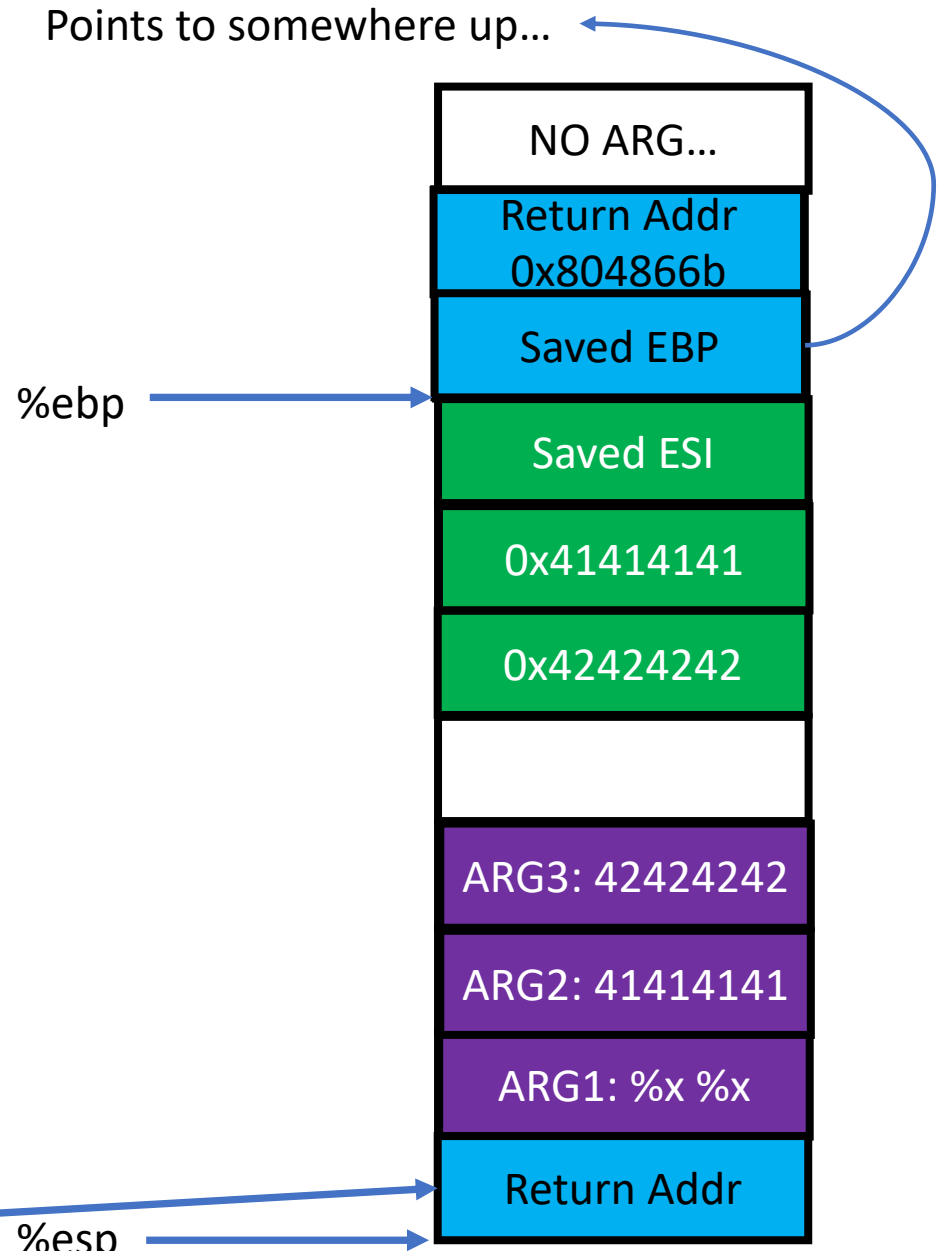
```
movl    $0x41414141,-0x8(%ebp)
movl    $0x42424242,-0xc(%ebp)
lea     0x8048727,%eax
mov     -0x8(%ebp),%ecx
mov     -0xc(%ebp),%edx
mov     %eax,(%esp)
mov     %ecx,0x4(%esp)
mov     %edx,0x8(%esp)
call    0x8048370 <printf@plt>
lea     0x8048765,%ecx
```



Example – Function call

- Call printf?

```
movl    $0x41414141,-0x8(%ebp)
movl    $0x42424242,-0xc(%ebp)
lea     0x8048727,%eax
mov     -0x8(%ebp),%ecx
mov     -0xc(%ebp),%edx
mov     %eax,(%esp)
mov     %ecx,0x4(%esp)
mov     %edx,0x8(%esp)
call    0x8048370 <printf@plt>
lea     0x8048765,%ecx
```

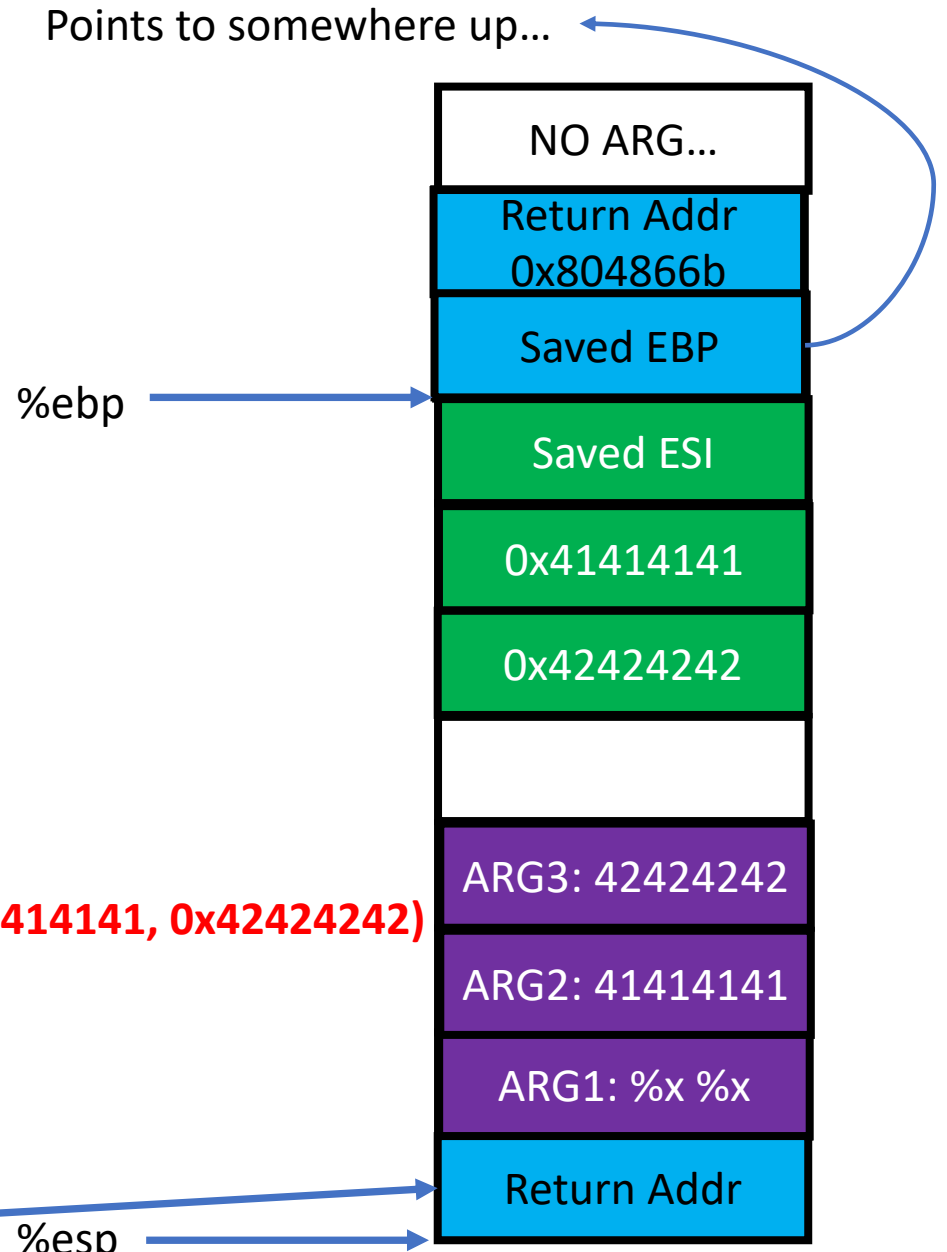


Example – Function call

- Call printf?

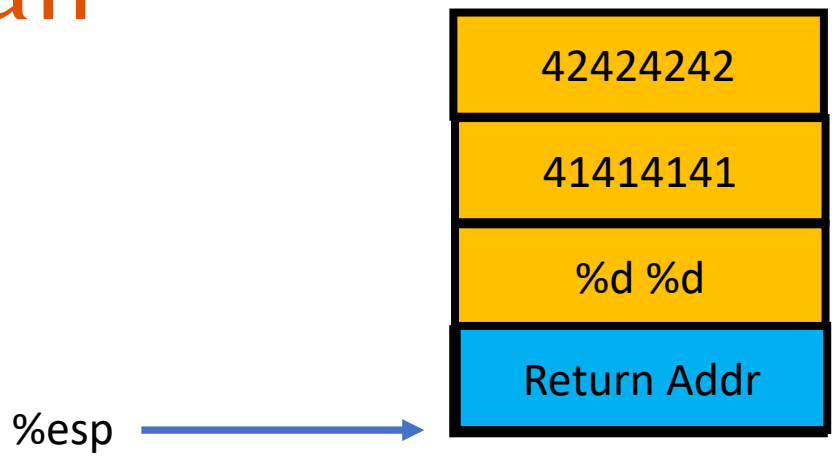
```
movl    $0x41414141,-0x8(%ebp)
movl    $0x42424242,-0xc(%ebp)
lea     0x8048727,%eax
mov     -0x8(%ebp),%ecx
mov     -0xc(%ebp),%edx
mov     %eax,(%esp)
mov     %ecx,0x4(%esp)
mov     %edx,0x8(%esp)
call    0x8048370 <printf@plt>
lea     0x8048765,%ecx
```

Calls printf("%x %x", 0x41414141, 0x42424242)



Example – Function call

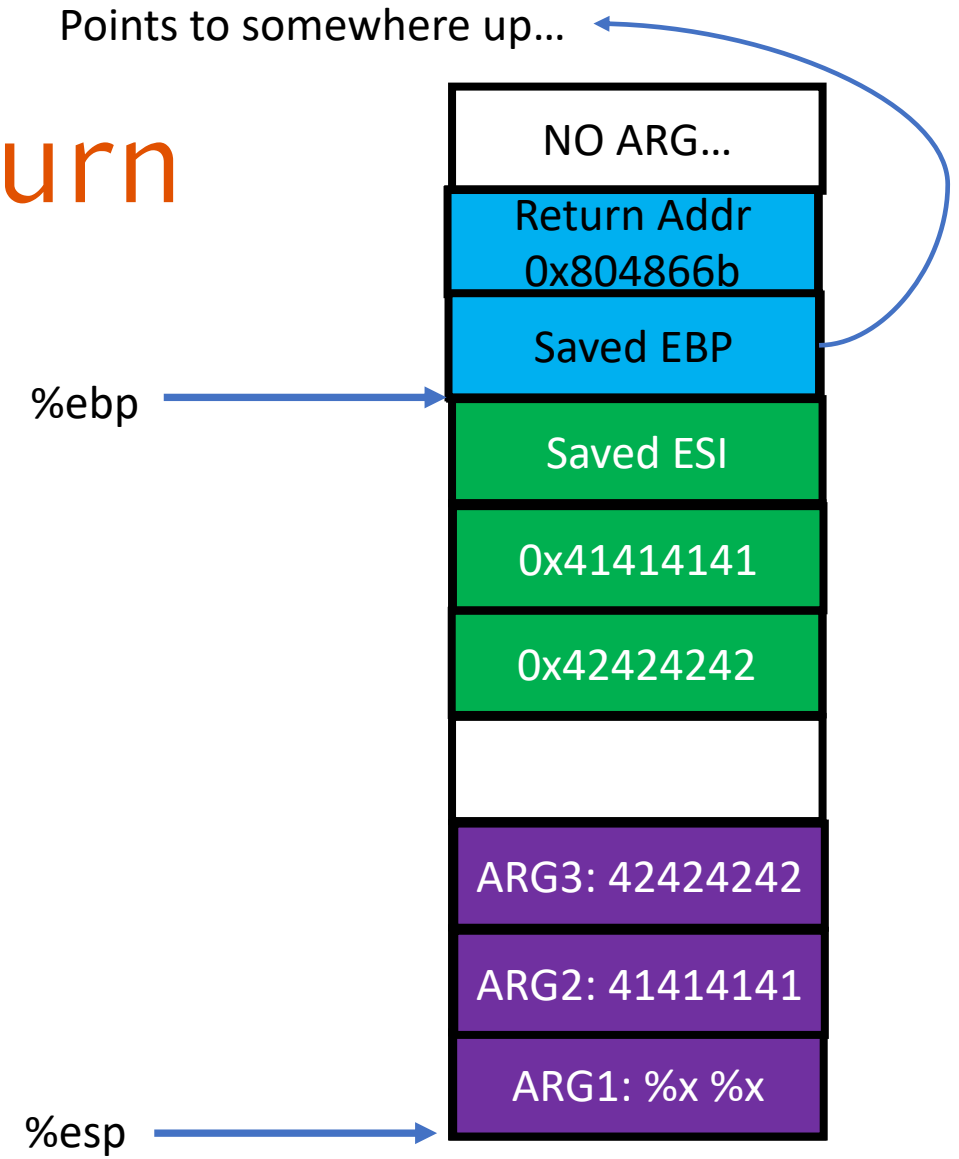
- What printf will see?



Example – Function Return

- Function return of receive_input

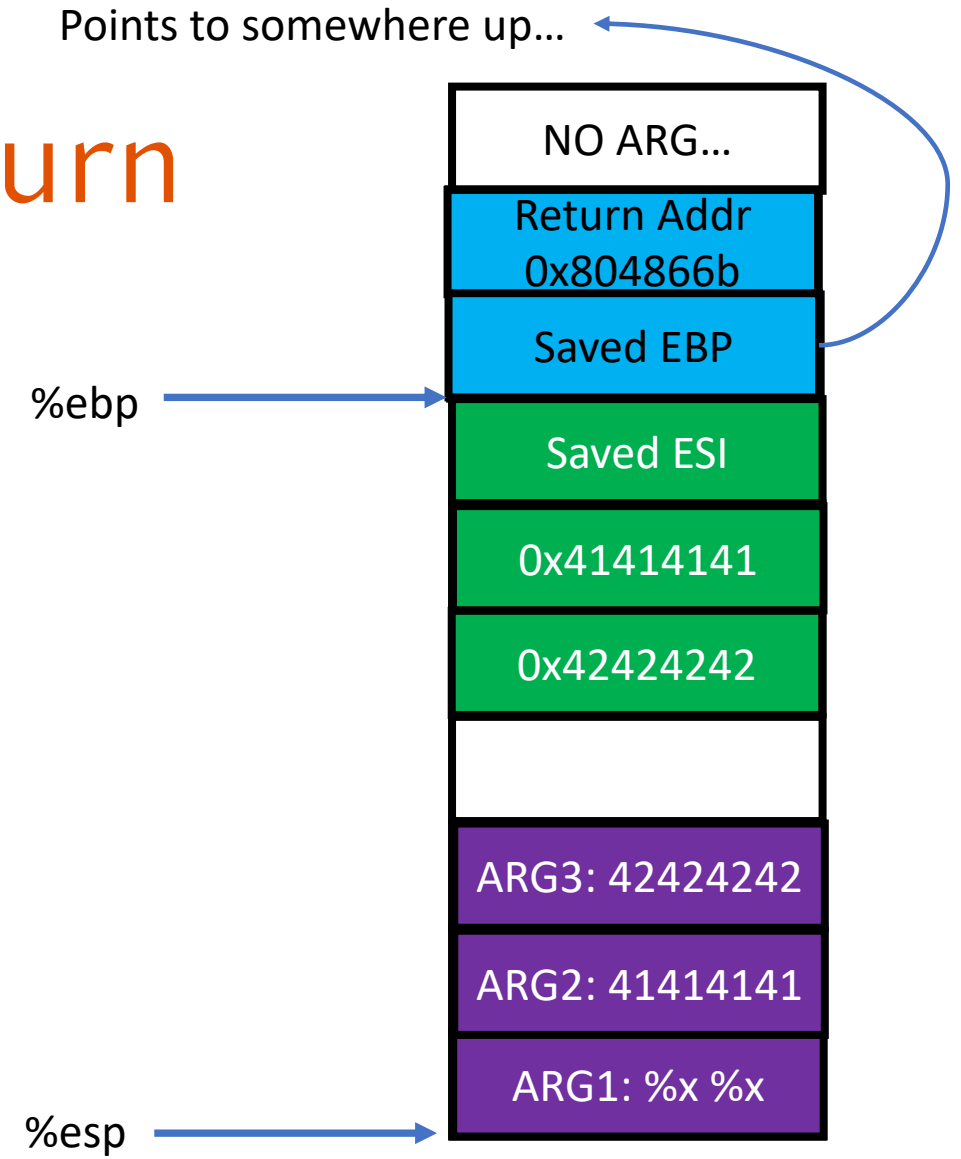
```
add     $0x54,%esp
pop     %esi
pop     %ebp
ret
```



Example – Function Return

- Function return of receive_input

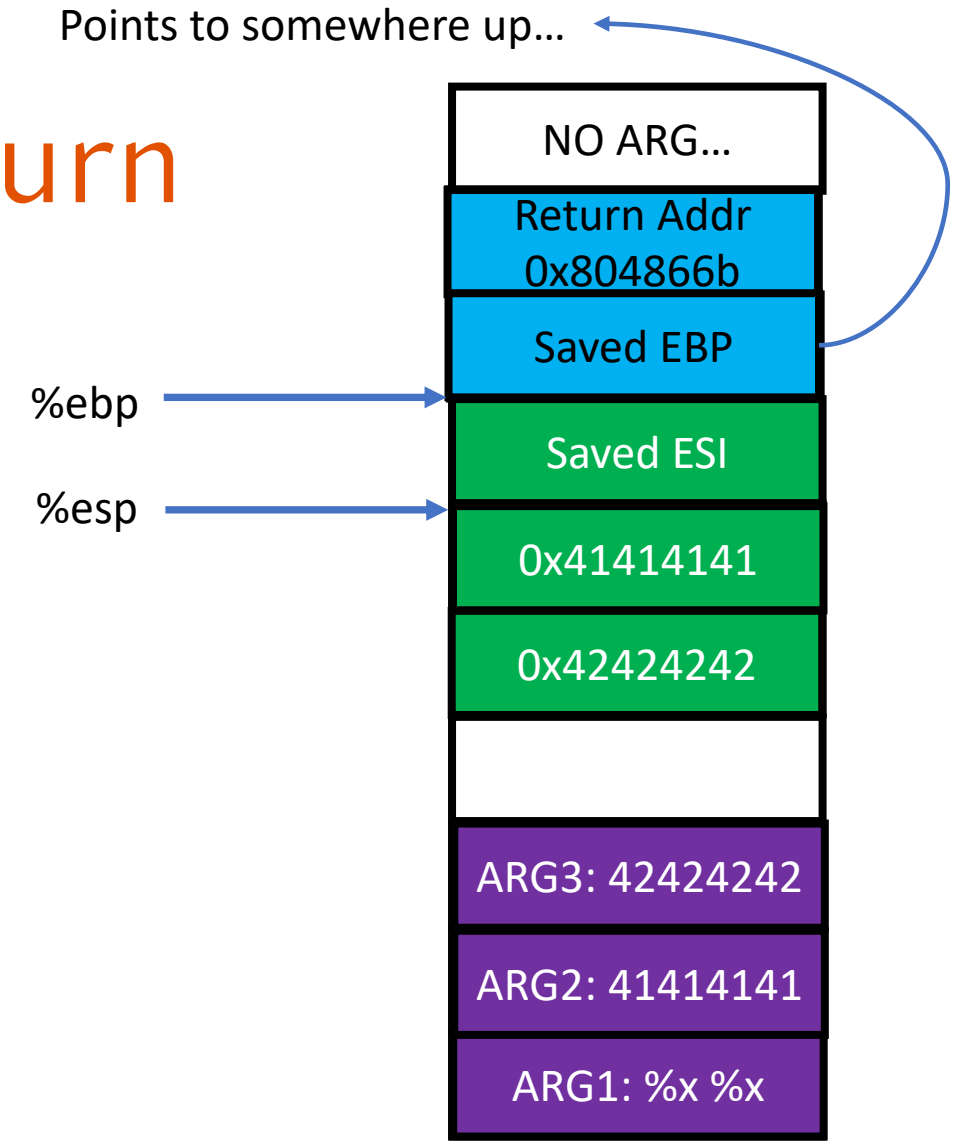
```
add     $0x54,%esp
pop     %esi
pop     %ebp
ret
```



Example – Function Return

- Function return of receive_input

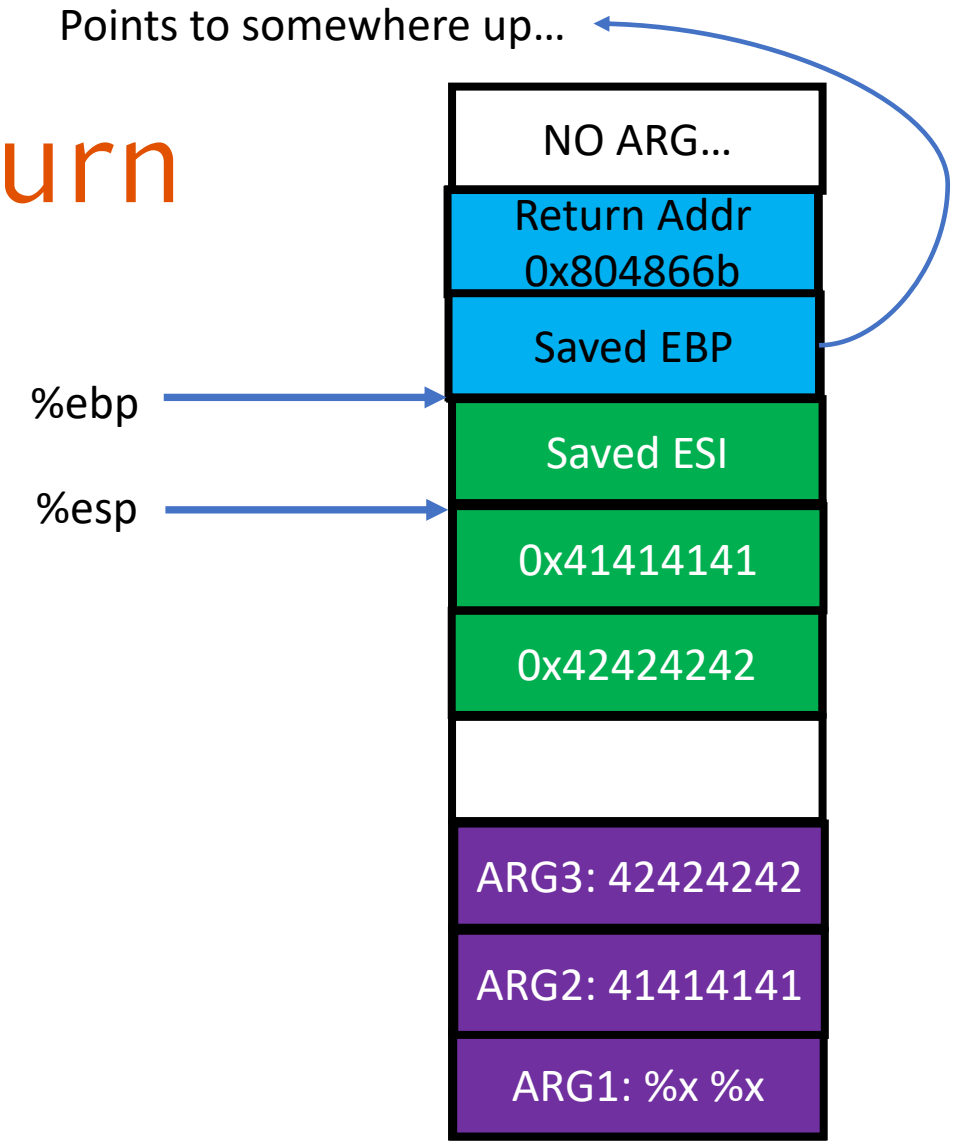
```
add     $0x54,%esp
pop     %esi
pop     %ebp
ret
```



Example – Function Return

- Function return of receive_input

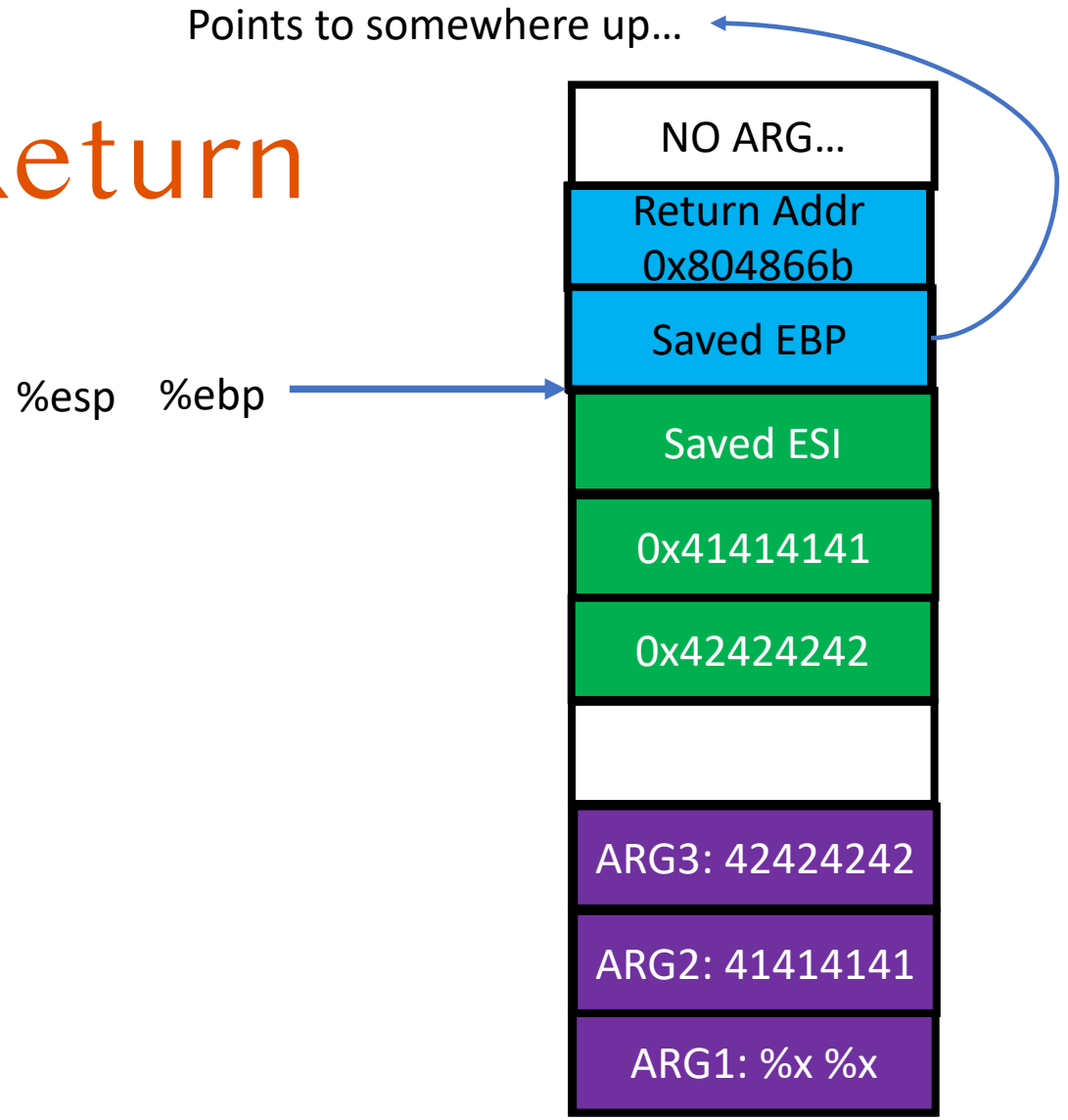
```
add     $0x54,%esp
pop     %esi
pop     %ebp
ret
```



Example – Function Return

- Function return of receive_input

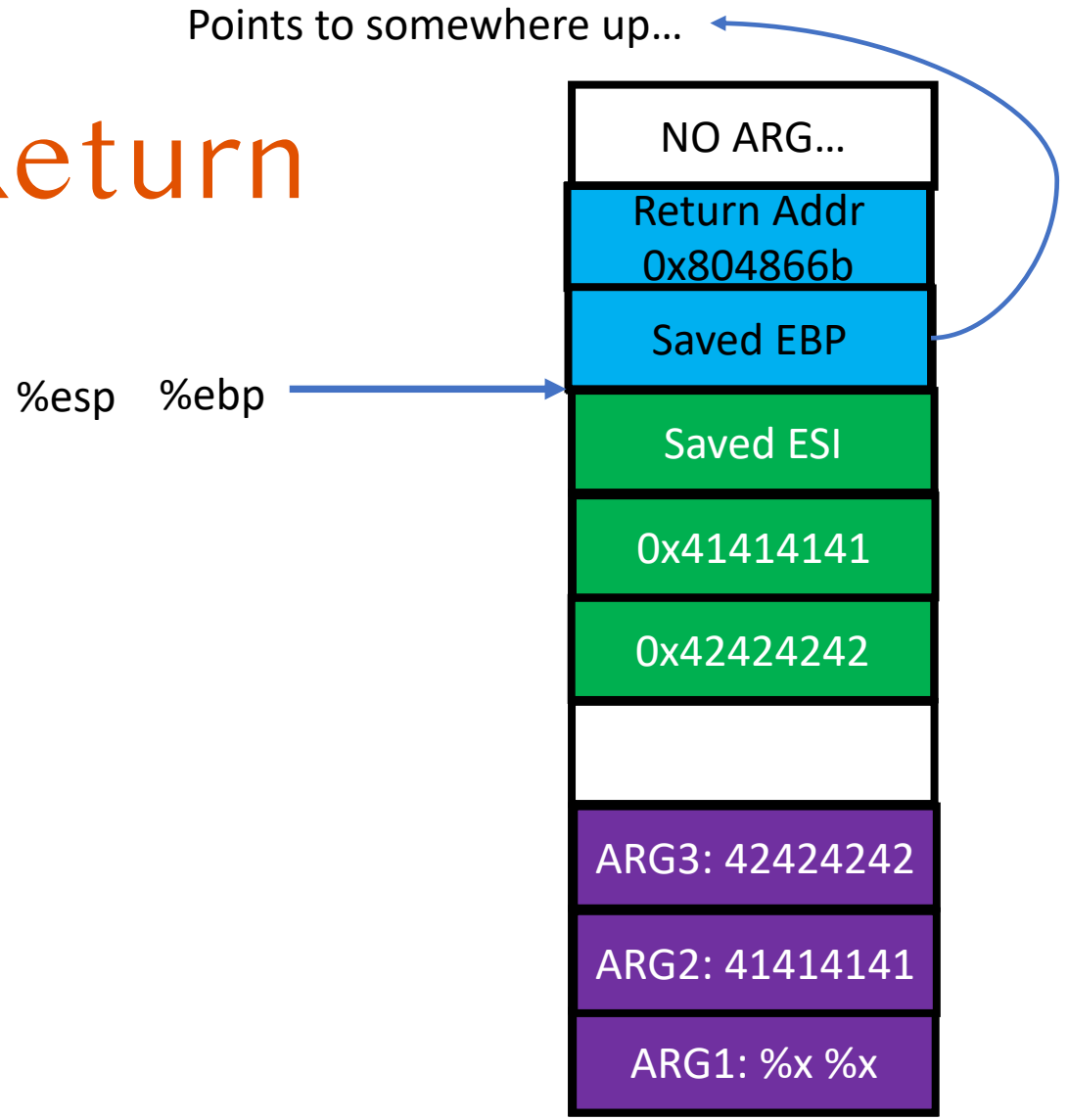
```
add     $0x54,%esp
pop     %esi
pop     %ebp
ret
```



Example – Function Return

- Function return of receive_input

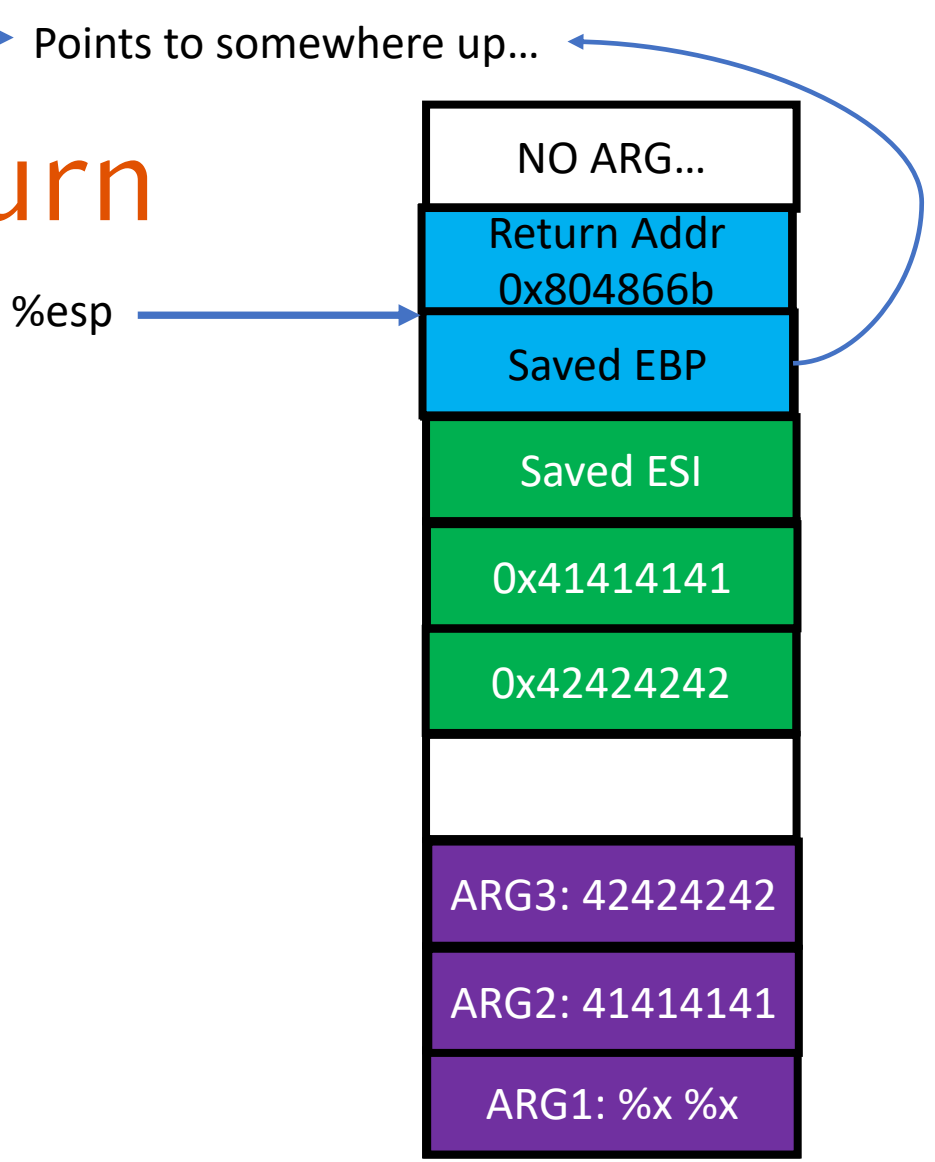
```
add     $0x54,%esp
pop     %esi
pop     %ebp
ret
```



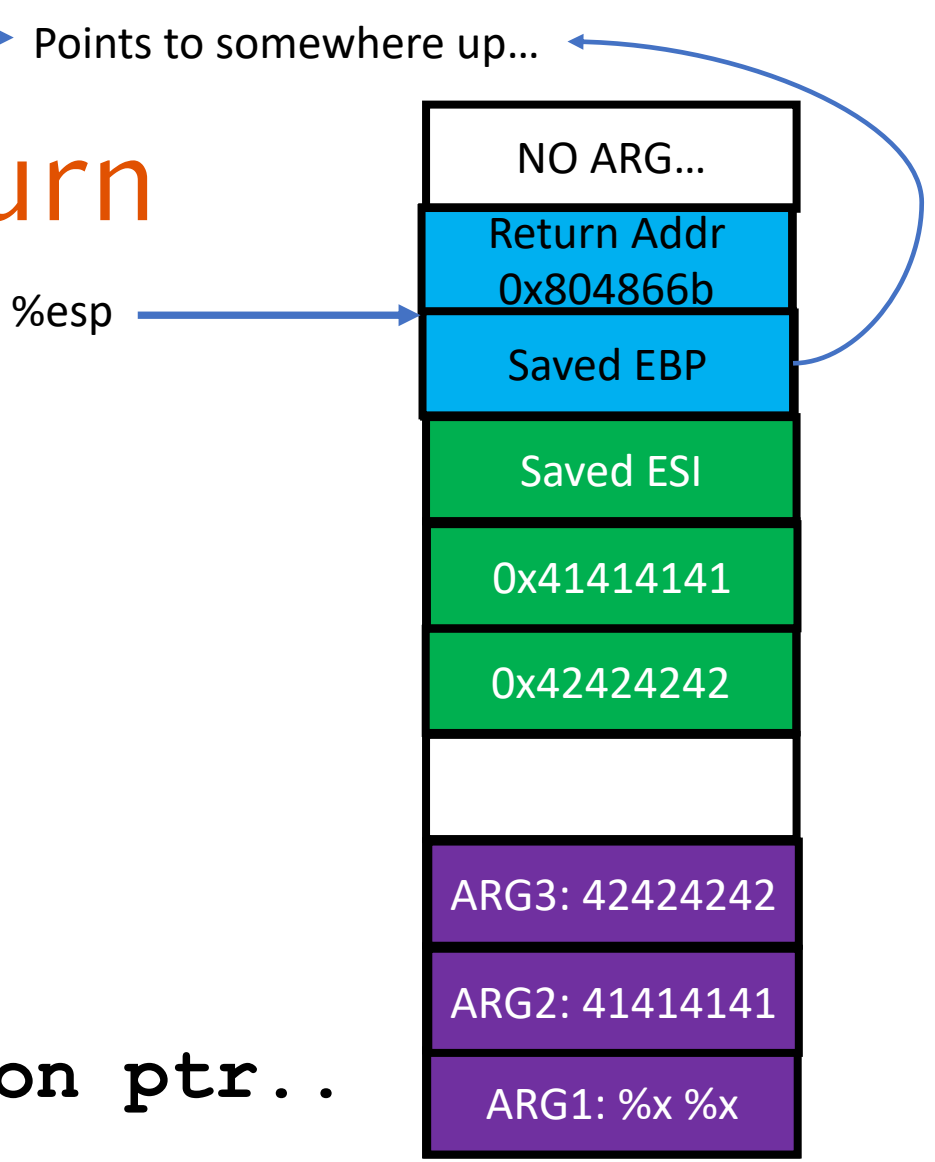
Example – Function Return

- Function return of receive_input

```
add     $0x54,%esp
pop     %esi
pop     %ebp
ret
```



Example – Function Return



- Function return of receive_input

```
add     $0x54,%esp
pop     %esi
pop     %ebp
ret
```

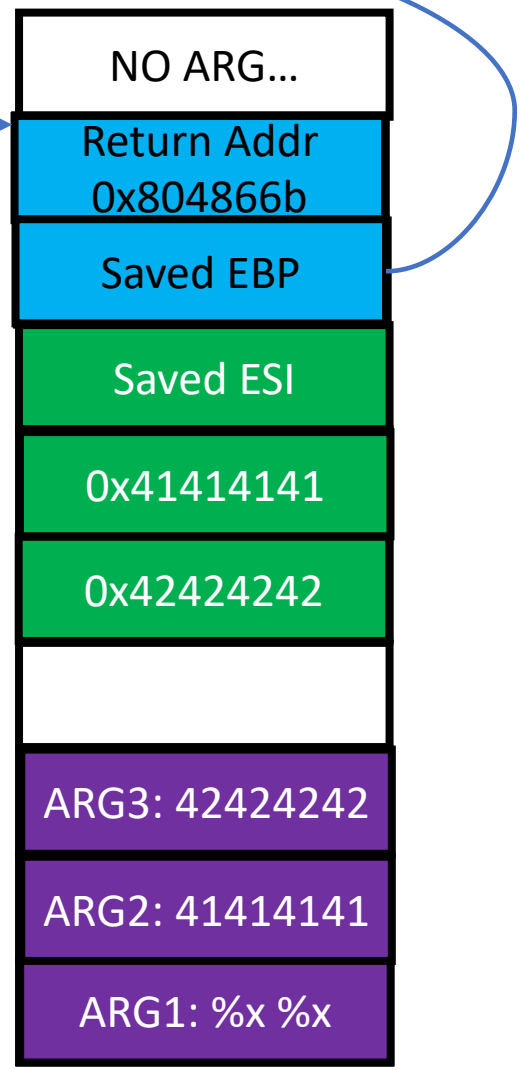
ret: pop %eip, change instruction ptr..

Example – Function Return

- Function return of receive_input

```
add     $0x54,%esp
pop     %esi
pop     %ebp
ret
```

```
call   0x8048570 <receive_input>
0x0804866b <+11>:       xor    %eax,%eax
```



Buffer Overflow

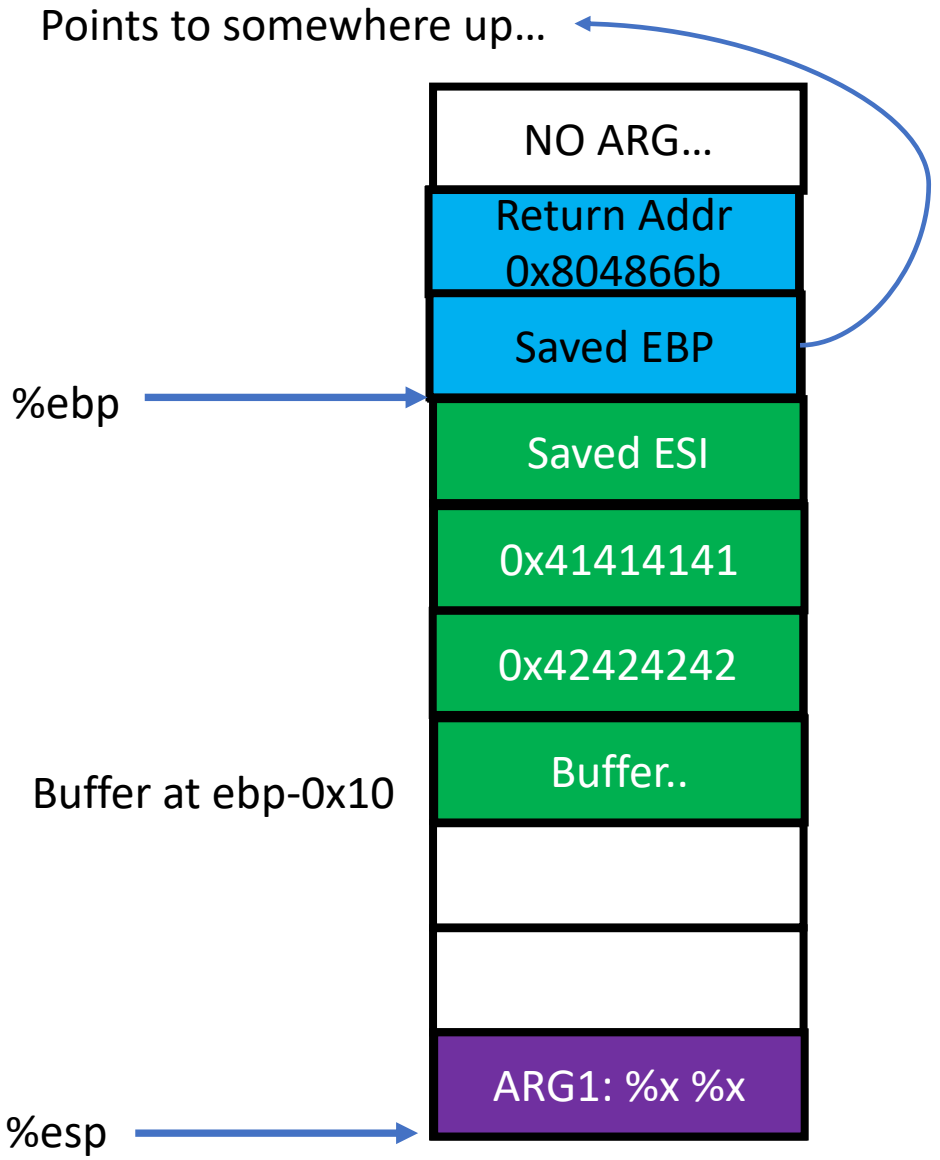
- Example

```
void receive_input() {
    int a = 0x41414141, b = 0x42424242;
    char buf[20];

    printf("Values in two local variables are:\n"
           "a = 0x%08x and b = 0x%08x\n", a, b);

    printf("Can you change these values to:\n"
           "a = 0x48474645 and b = 0x44434241?\n");

    printf("Type YES if you agree with this... "
           "(a fake message, you may overflow the input buffer).\n");
    fgets(buf, 128, stdin);
```

- char buf[20];
- fgets(buf, 128, stdin);

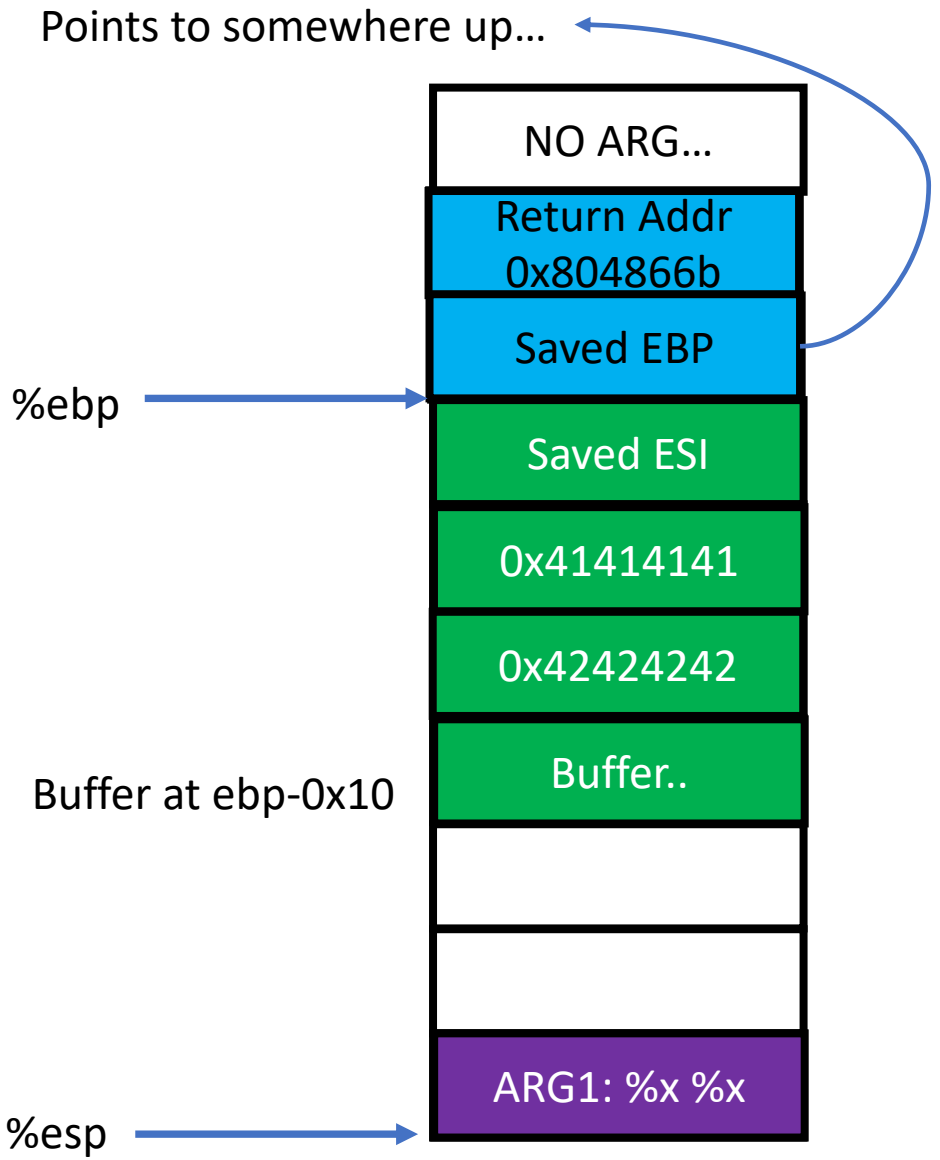
Buffer Overflow

- Overwrite values in stack by overflowing
  - A local variable buffer
- Suppose we have char buffer[4];
  - -0x8(%ebp) stores 0x41414141
  - -0xc(%ebp) stroes 0x42424242
  - **-0x10(%ebp) is a buffer, size 4 byte.**
- Program gets input from you via
  - `fgets(buffer, 128, stdin);`
  - Read **128** bytes..
- What if you type “1111aaaabbbb”?



Buffer Overflow

- 4 byte buffer…
- What if you type “1111aaaabbbb”?



ASCII CODE

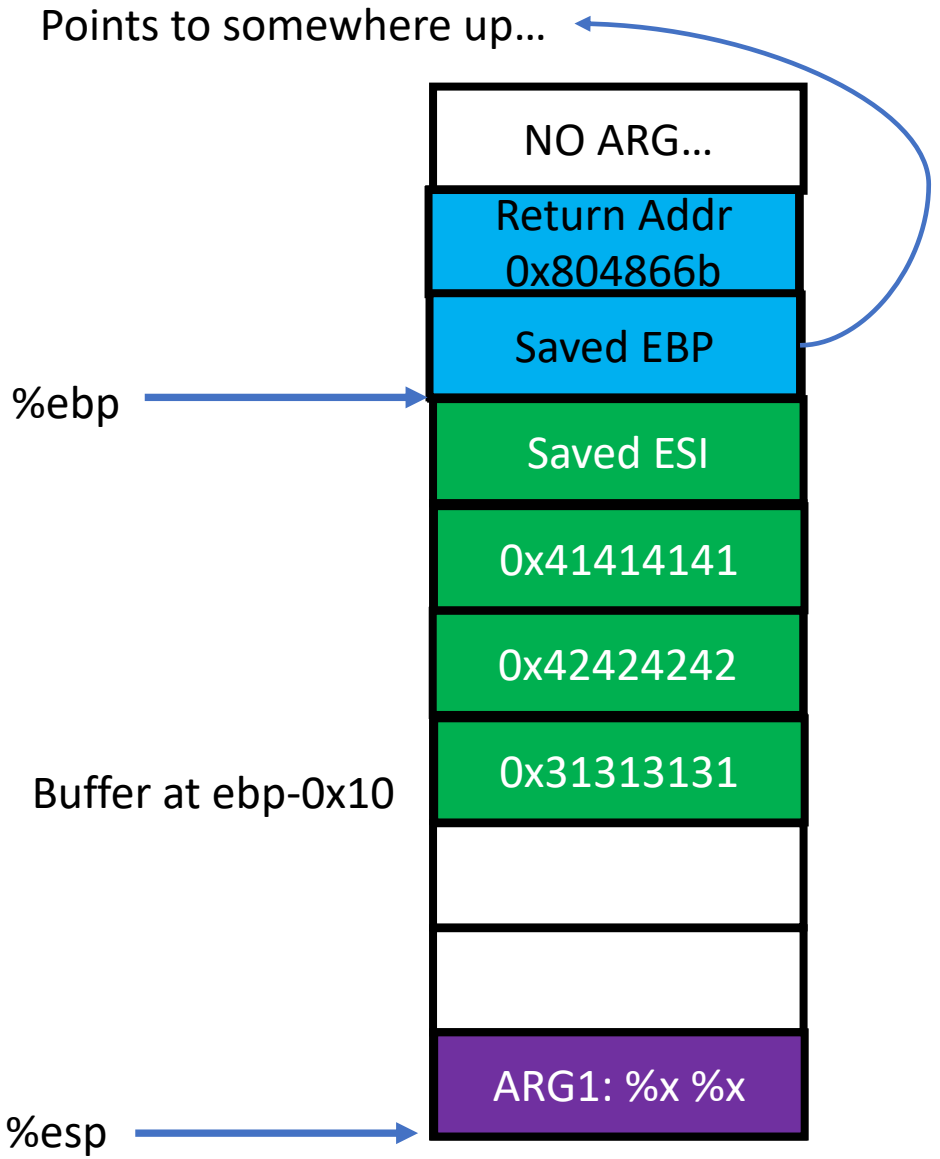
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

- Image from: https://cs.smu.ca/~porter/csc/ref/ascii.html and www.LookupTables.com

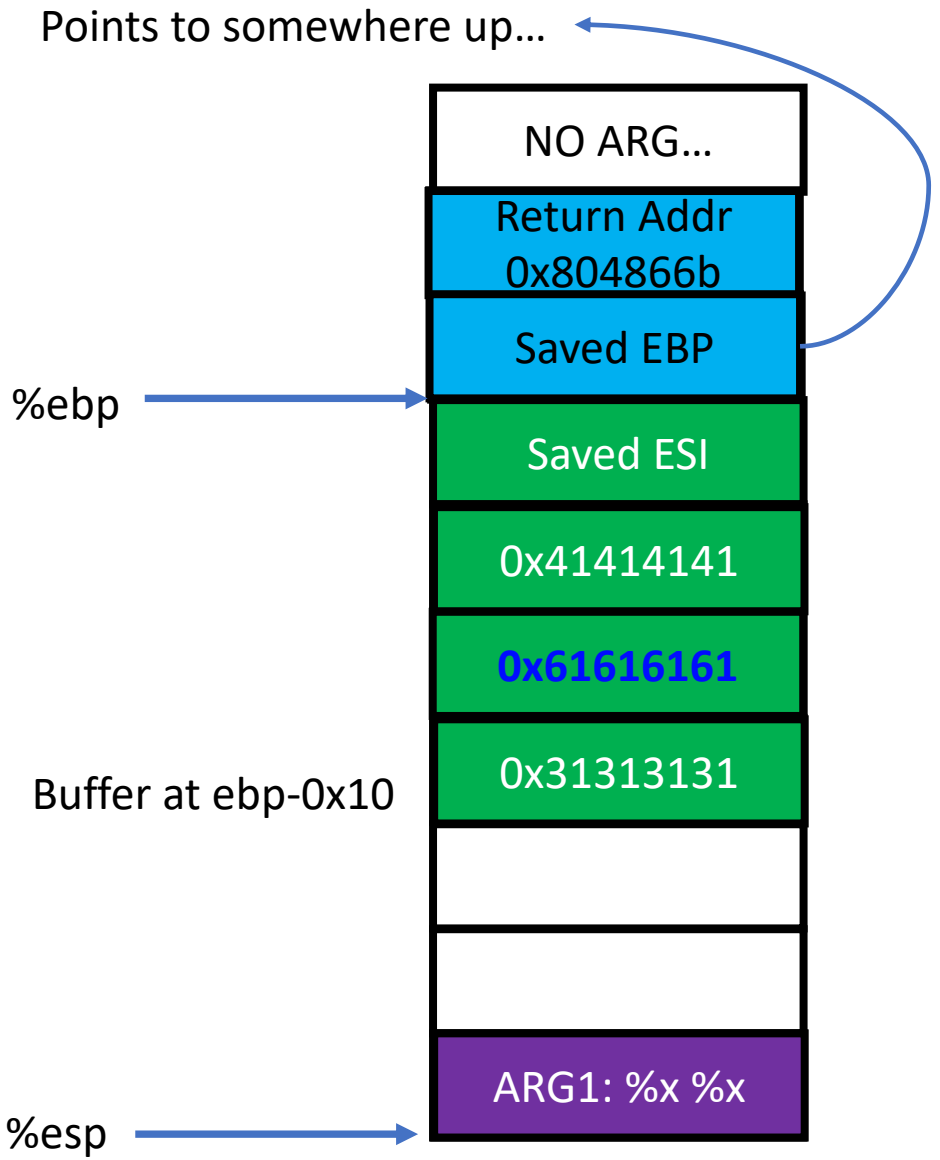
Buffer Overflow

- 4 byte buffer…
- What if you type “1111aaaabbbb”?



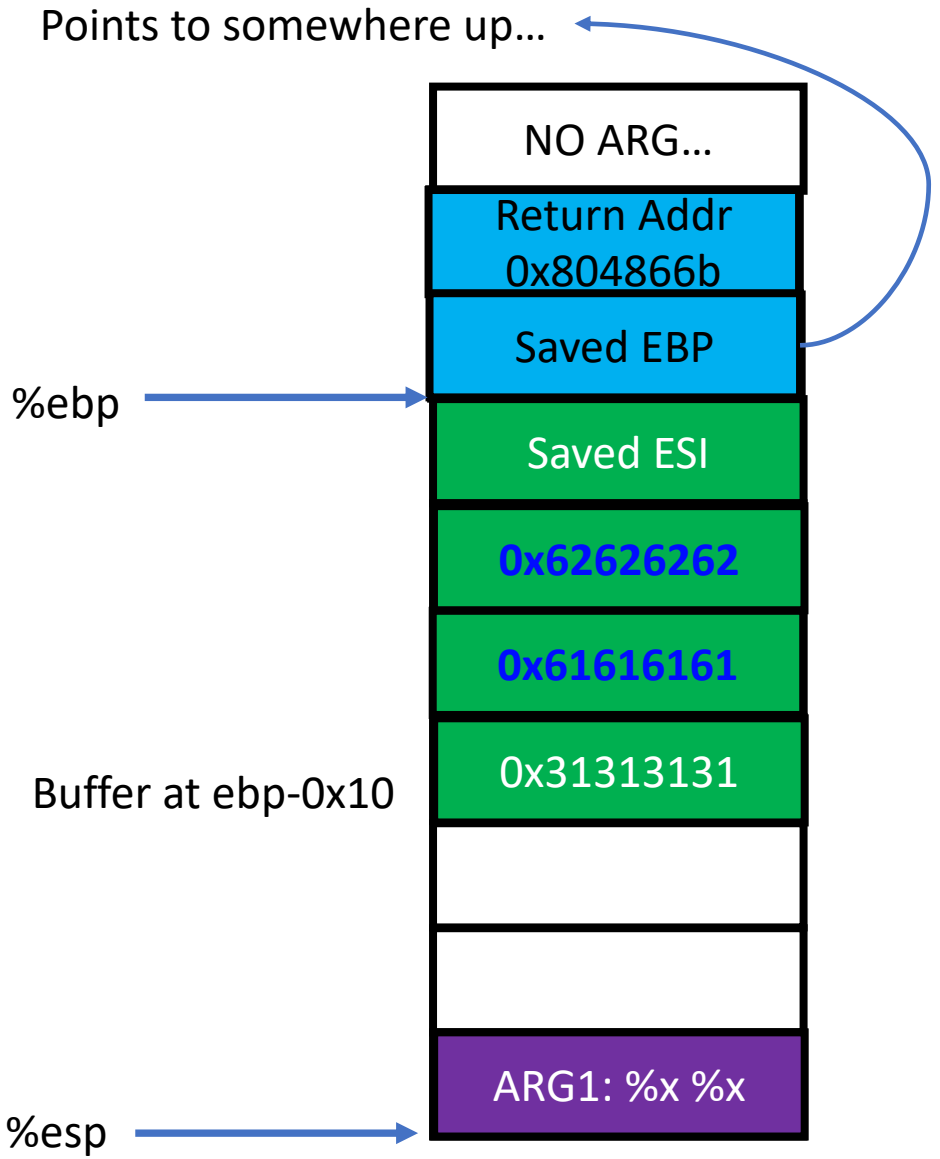
Buffer Overflow

- 4 byte buffer…
- What if you type “1111aaaabbbb”?



Buffer Overflow

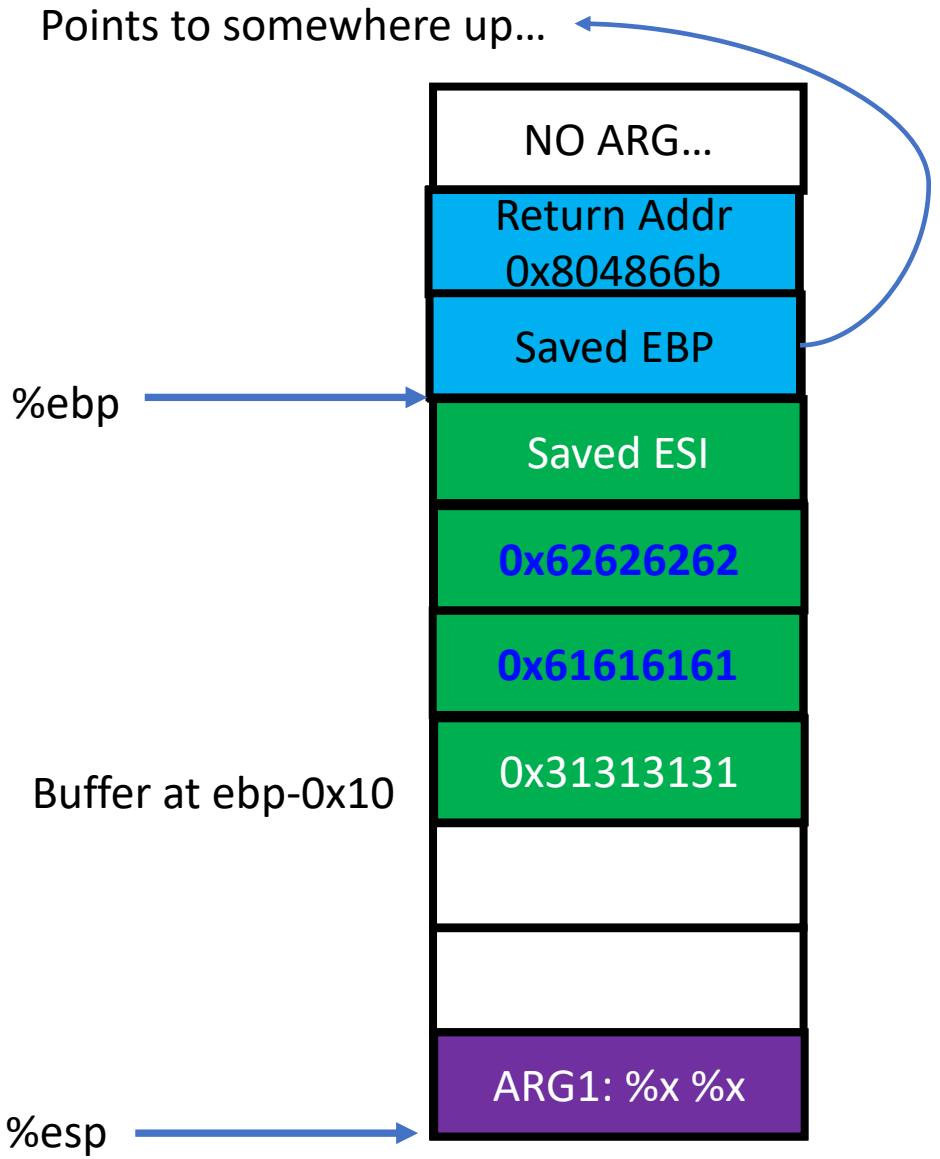
- 4 byte buffer…
- What if you type “1111aaaabbbb”?



Buffer Overflow

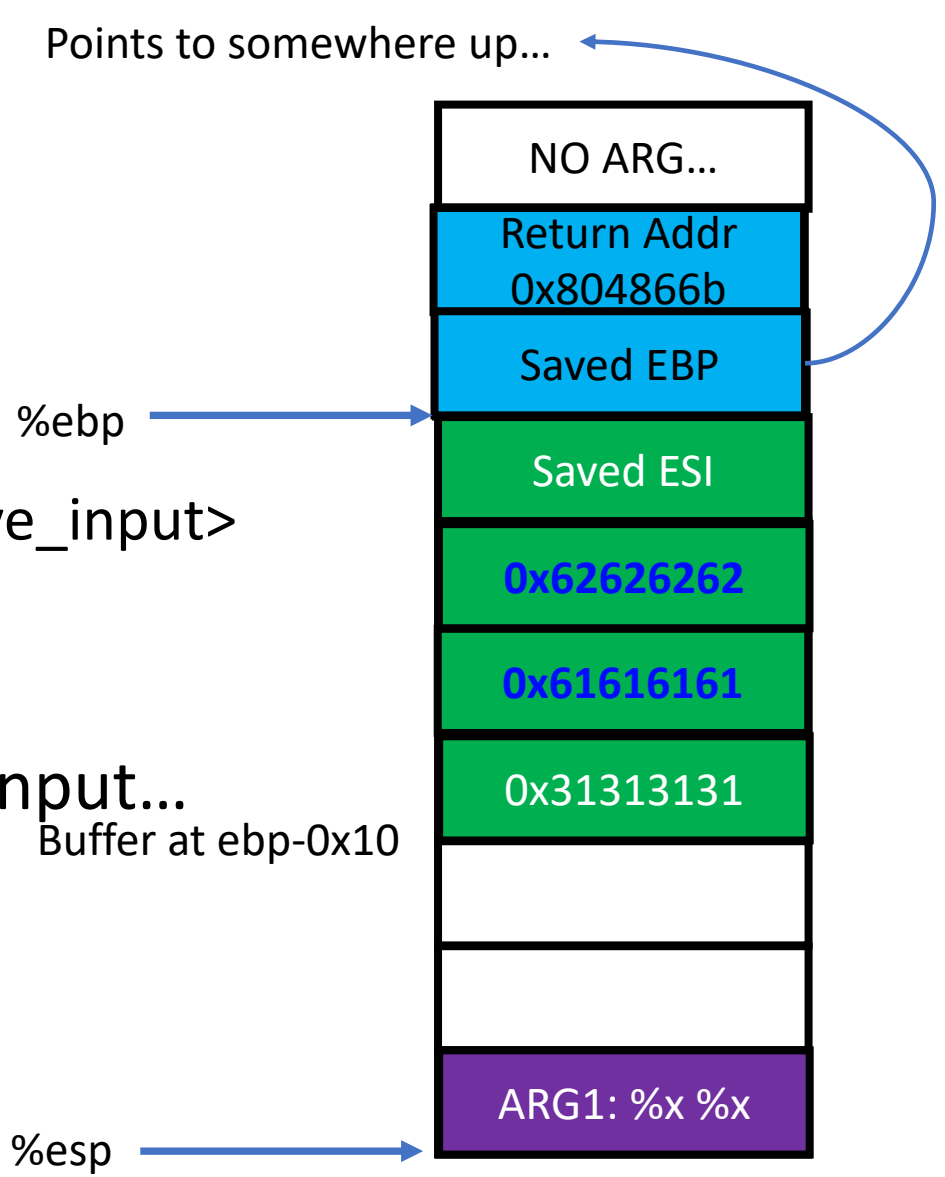
- 4 byte buffer…
- What if you type “1111aaaabbbb”?

- You can change variables!



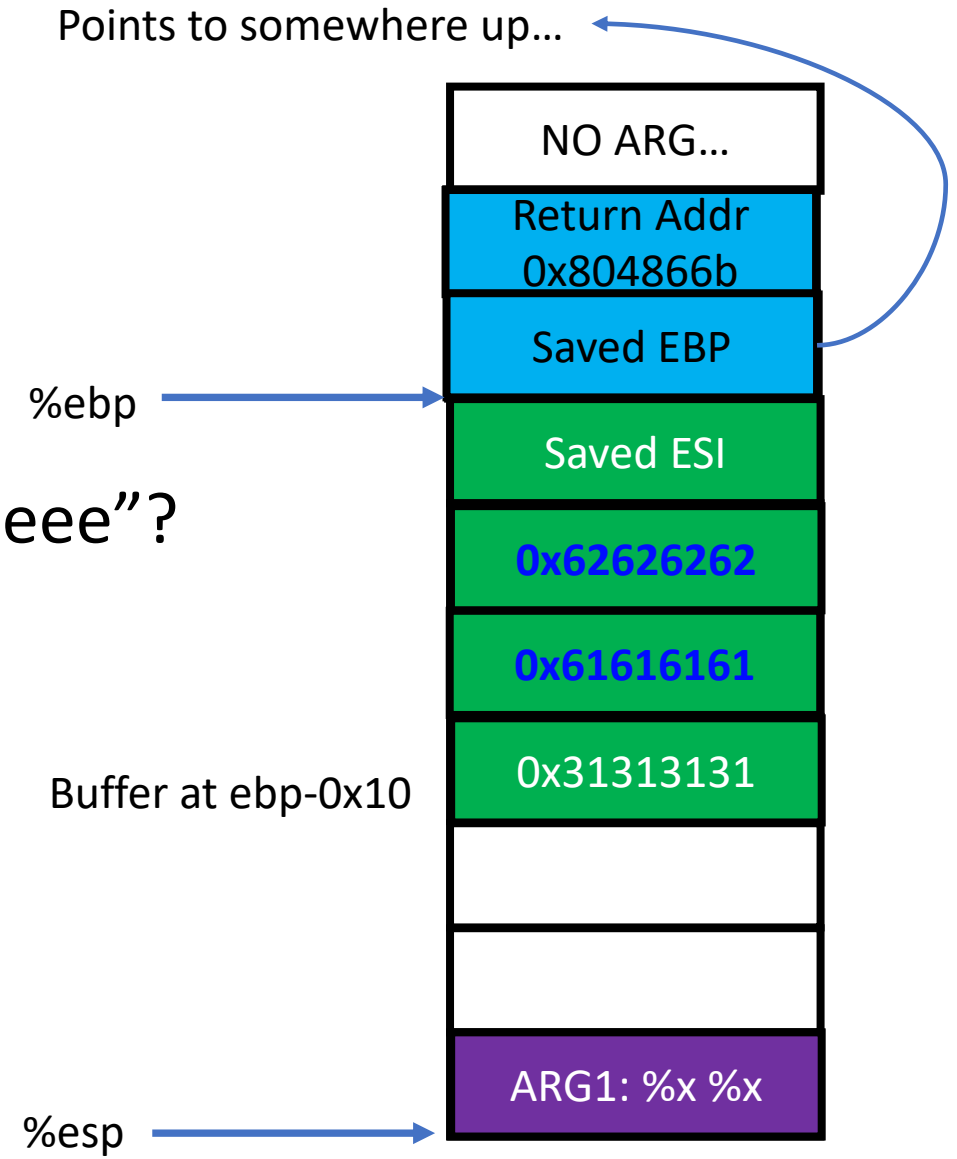
Return Address

- Store the execution points after the call
  - 0x08048666 <+6>: call 0x8048570 <receive_input>
  - 0x0804866b <+11>: xor %eax,%eax
- Should store 0x804866b on calling receive_input…



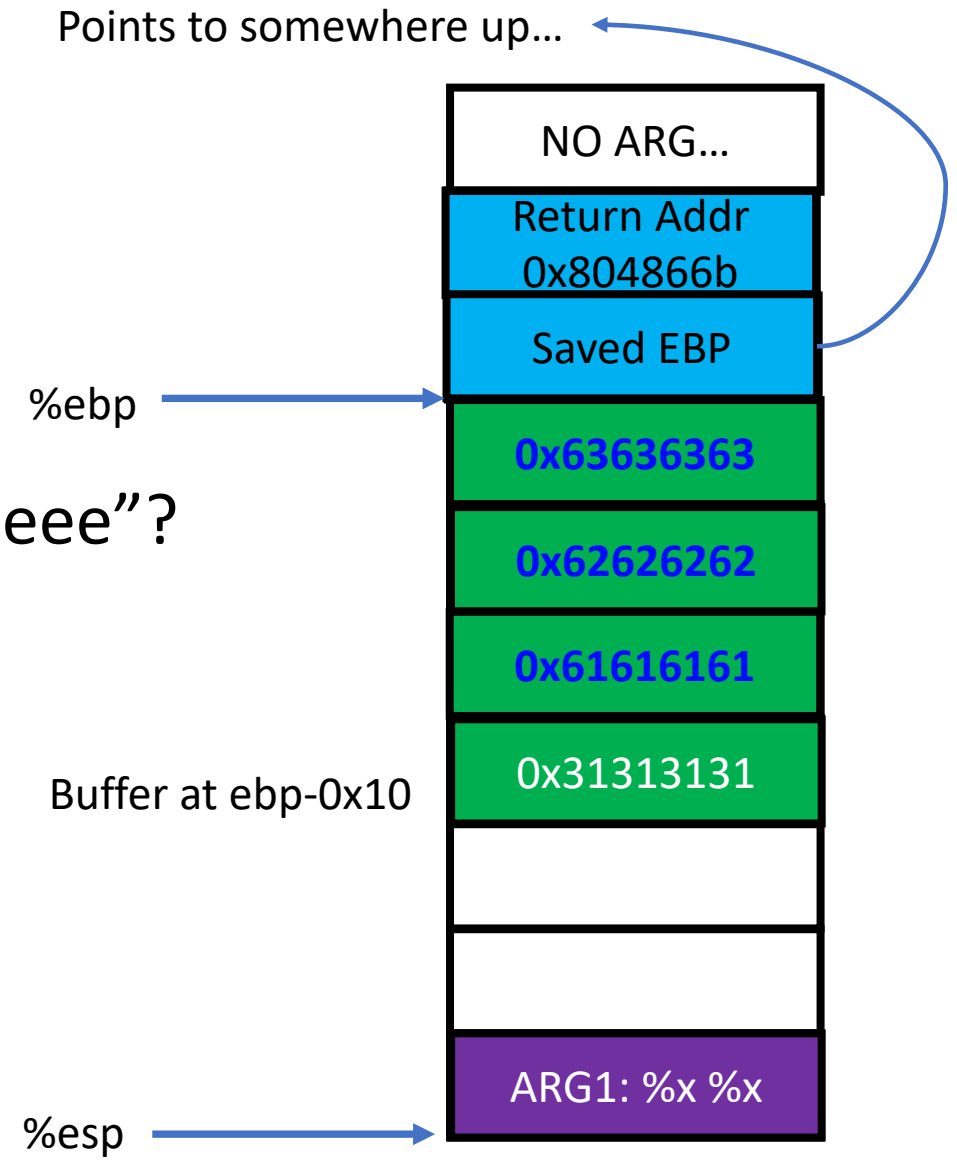
Buffer Overflow

- 4 byte buffer…
- What if you type “1111aaaabbbbccccddddeeee”?



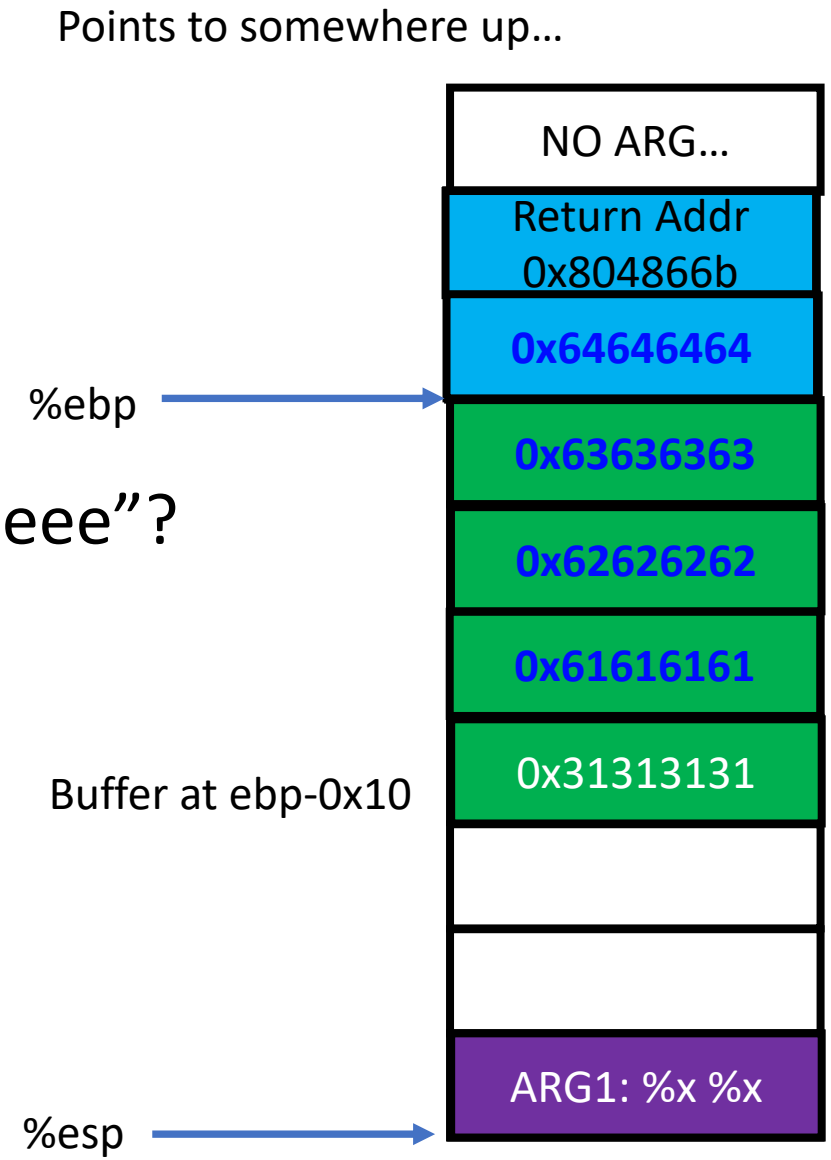
Buffer Overflow

- 4 byte buffer…
- What if you type “1111aaaabbbbccccddddeeee”?



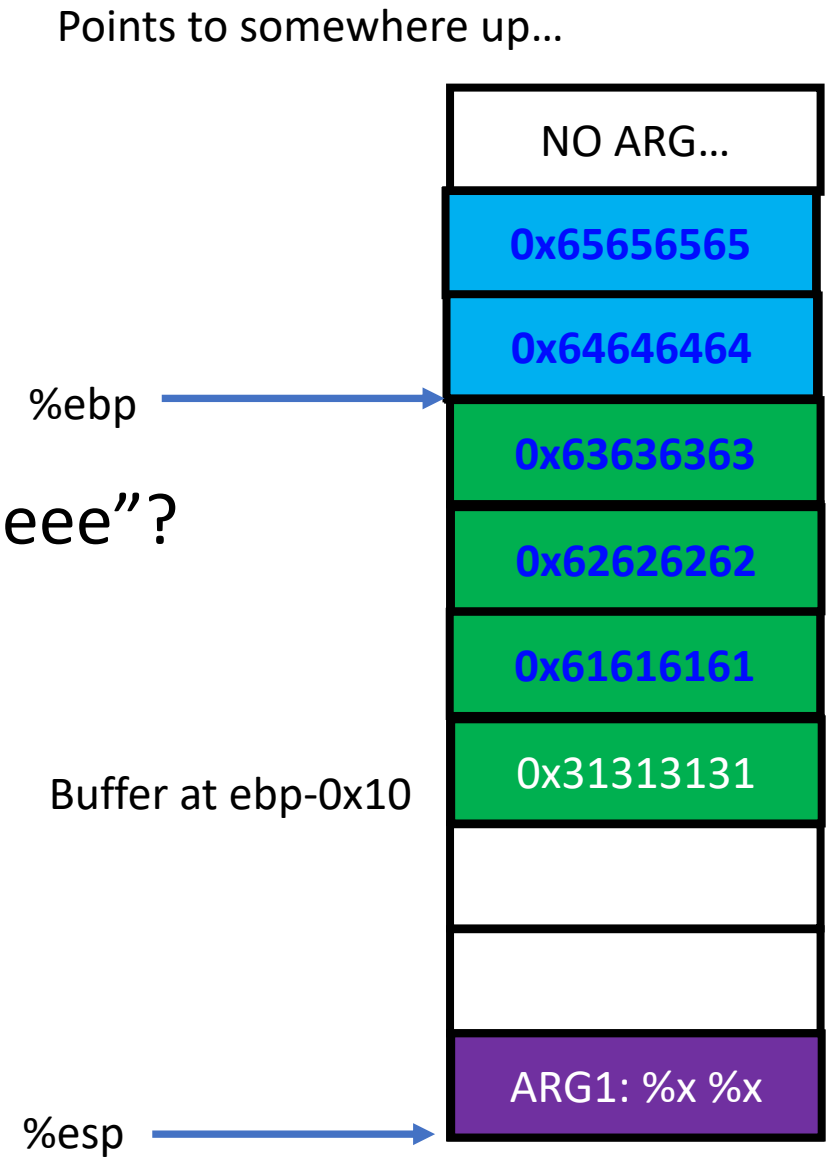
Buffer Overflow

- 4 byte buffer…
- What if you type “1111aaaabbbbccccddddeeee”?



Buffer Overflow

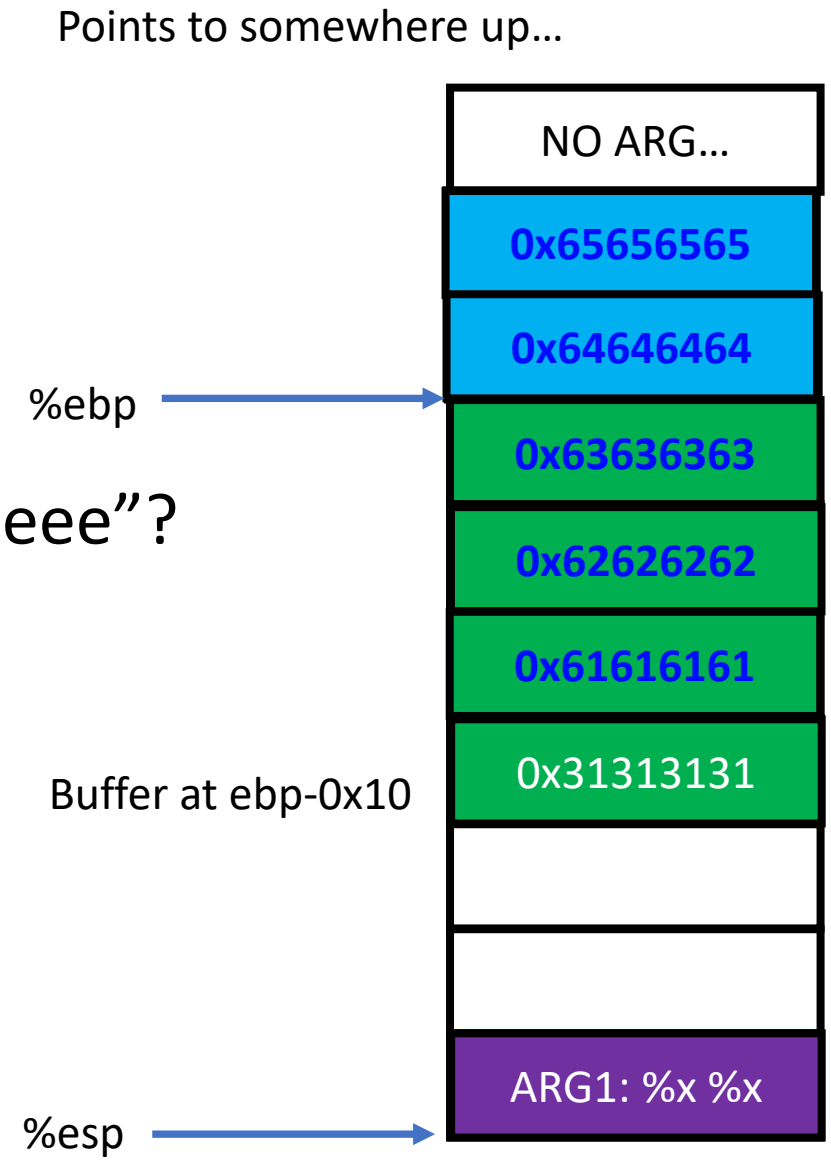
- 4 byte buffer…
- What if you type “1111aaaabbbbccccddddeeee”?



Buffer Overflow

- 4 byte buffer…
- What if you type “1111aaaabbbbccccddddeeee”?

- Overwrites the return address!

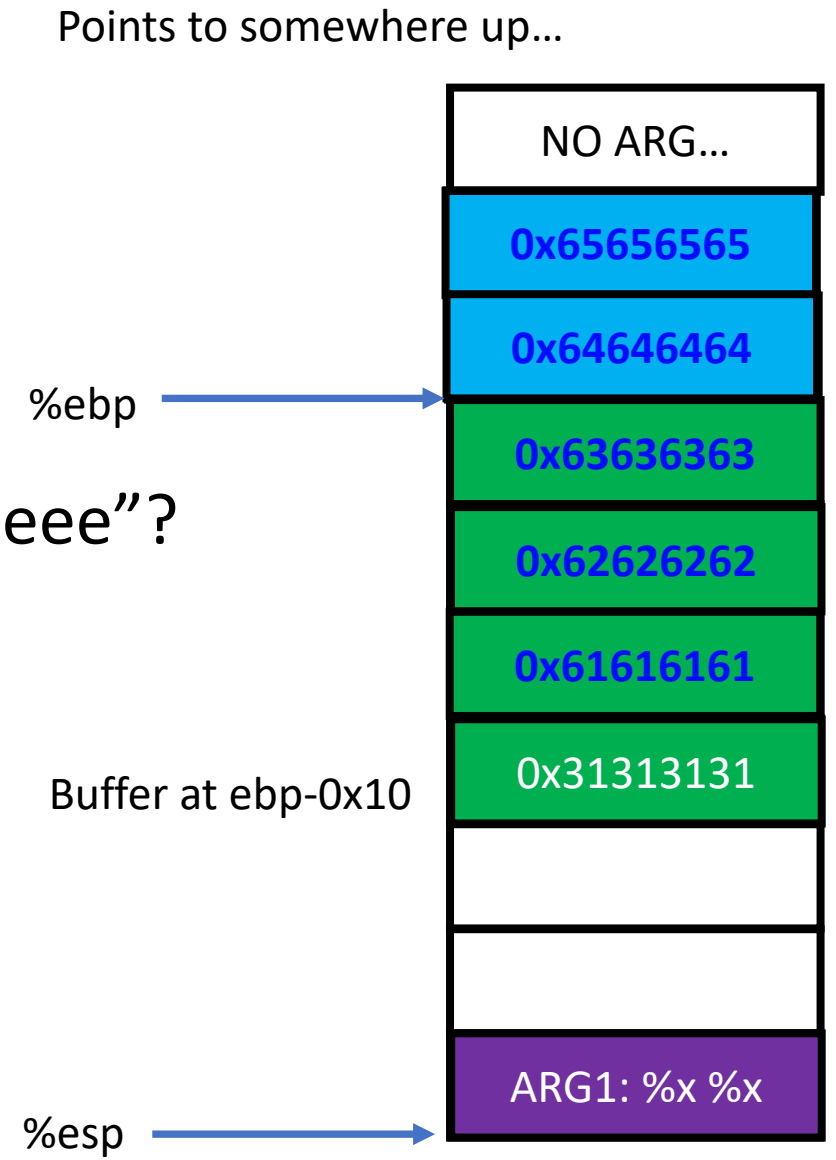


Buffer Overflow

- 4 byte buffer…
- What if you type “1111aaaabbbbccccddddeeee”?

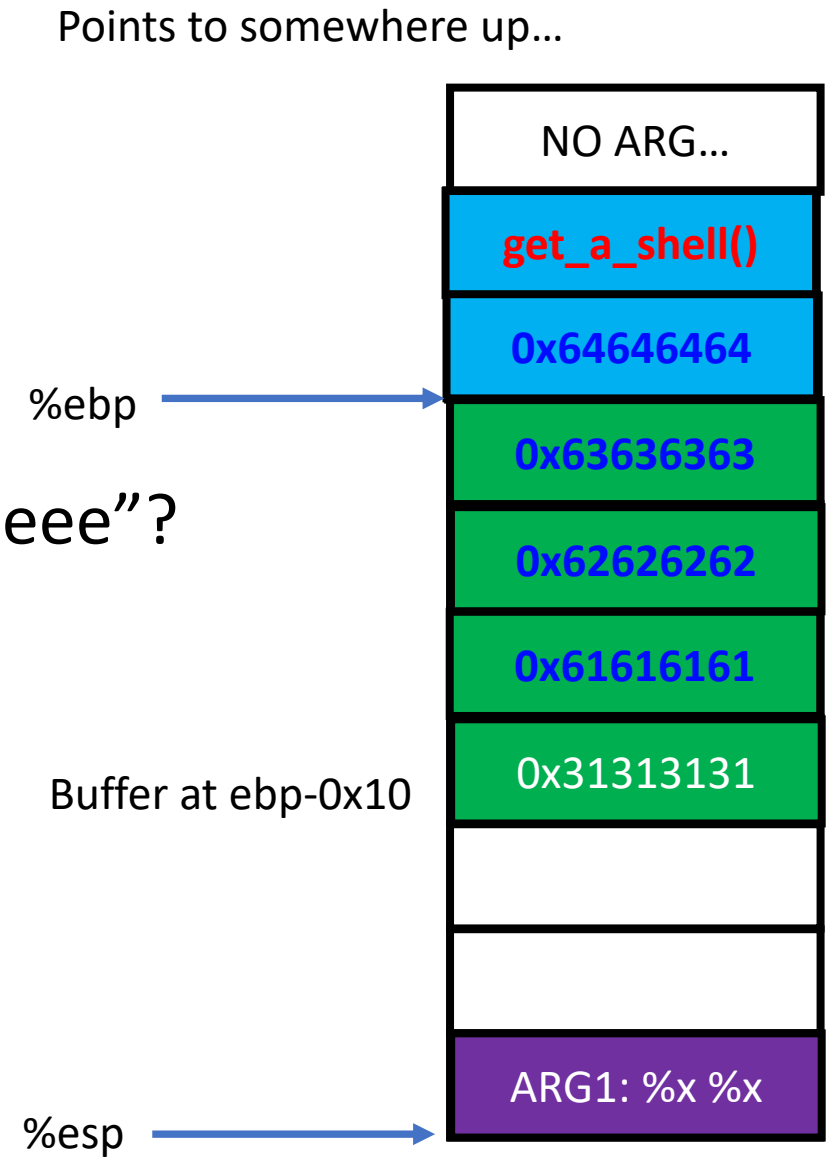
- Overwrites the return address!

- Can you set that as the address of
  - get_a_shell()?



Buffer Overflow

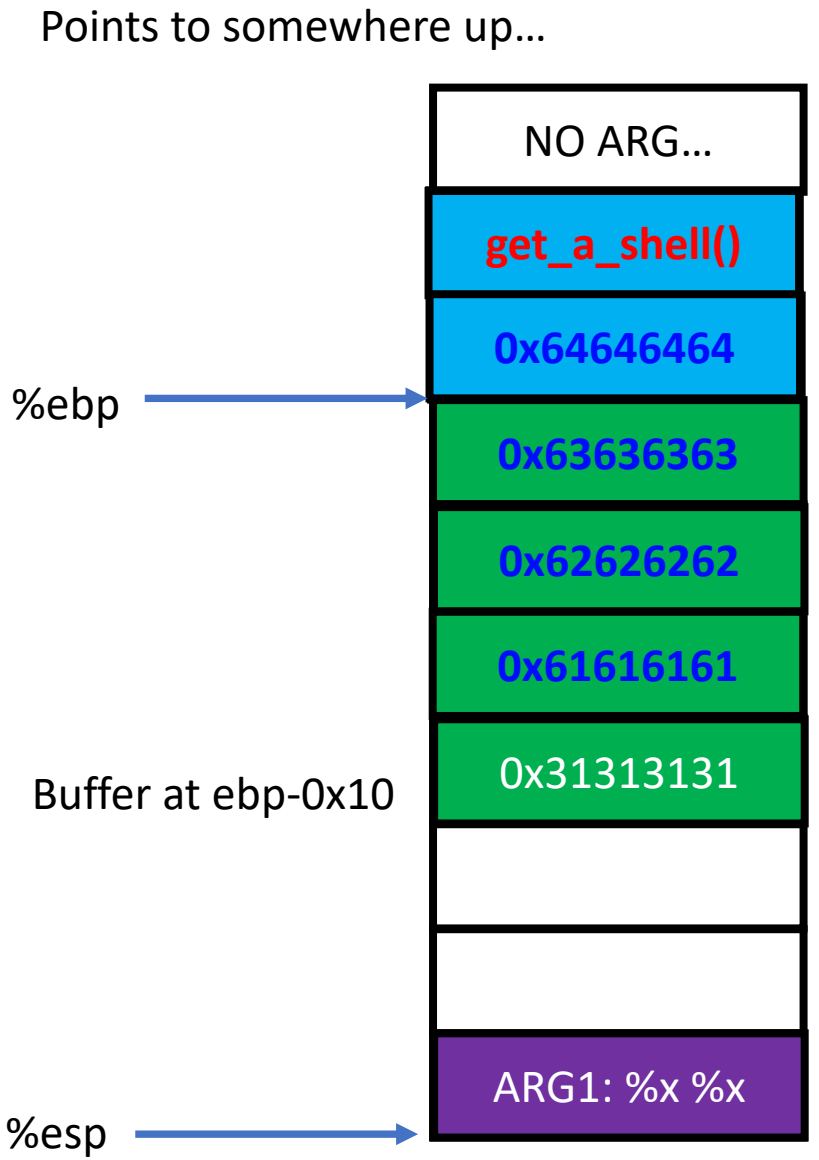
- 4 byte buffer…
- What if you type “1111aaaabbbbccccddddeeee”?
- Overwrites the return address!
- Can you set that as the address of
  - get_a_shell()?



Buffer Overflow

- Function return of receive_input

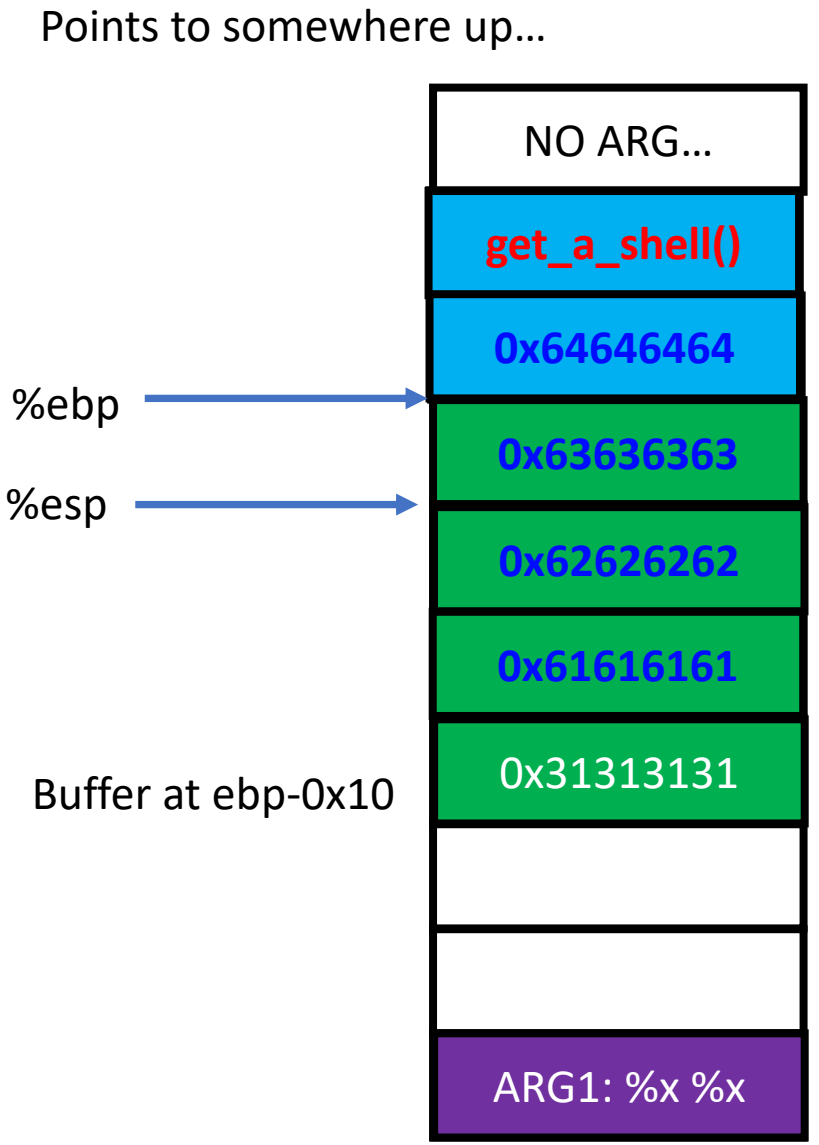
```
add     $0x54,%esp
pop     %esi
pop     %ebp
ret
```



Buffer Overflow

- Function return of receive_input

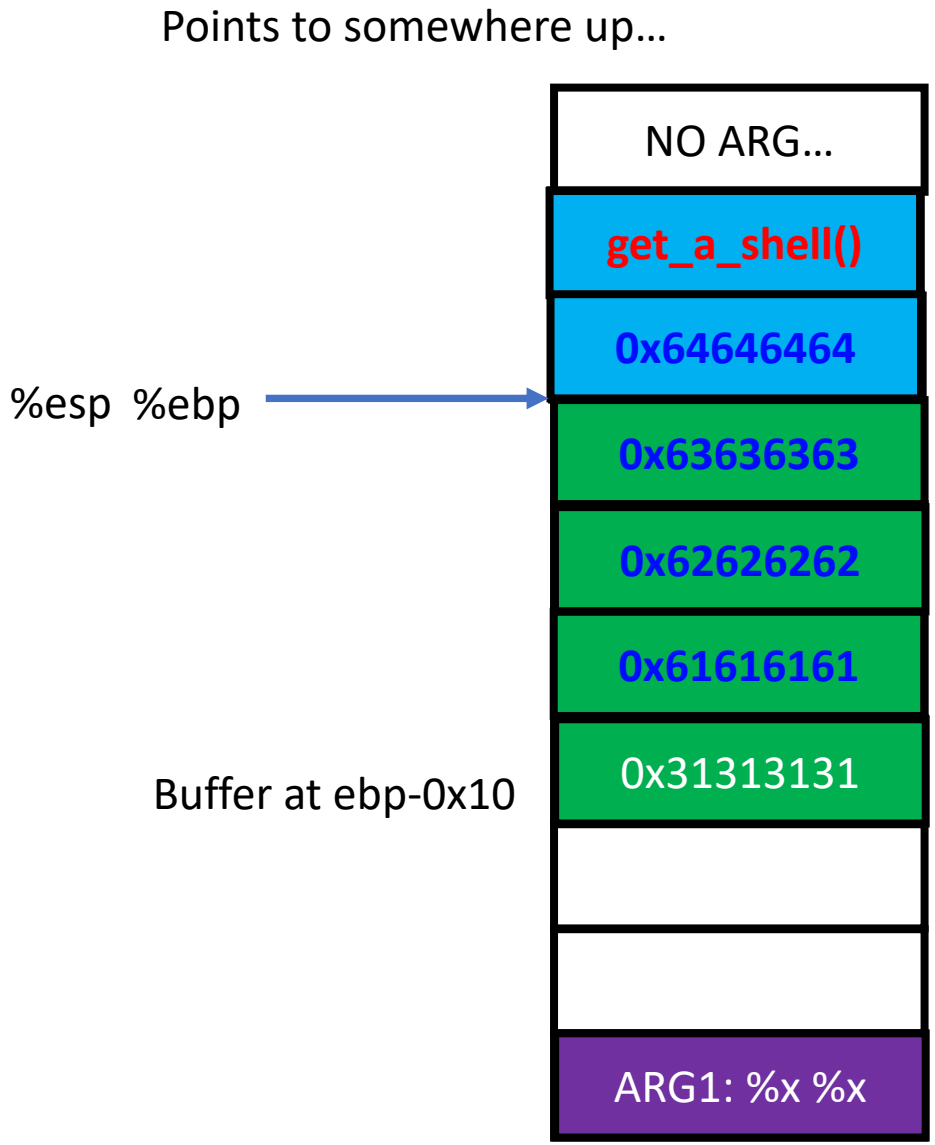
```
add     $0x54,%esp
pop     %esi
pop     %ebp
ret
```



Buffer Overflow

- Function return of receive_input

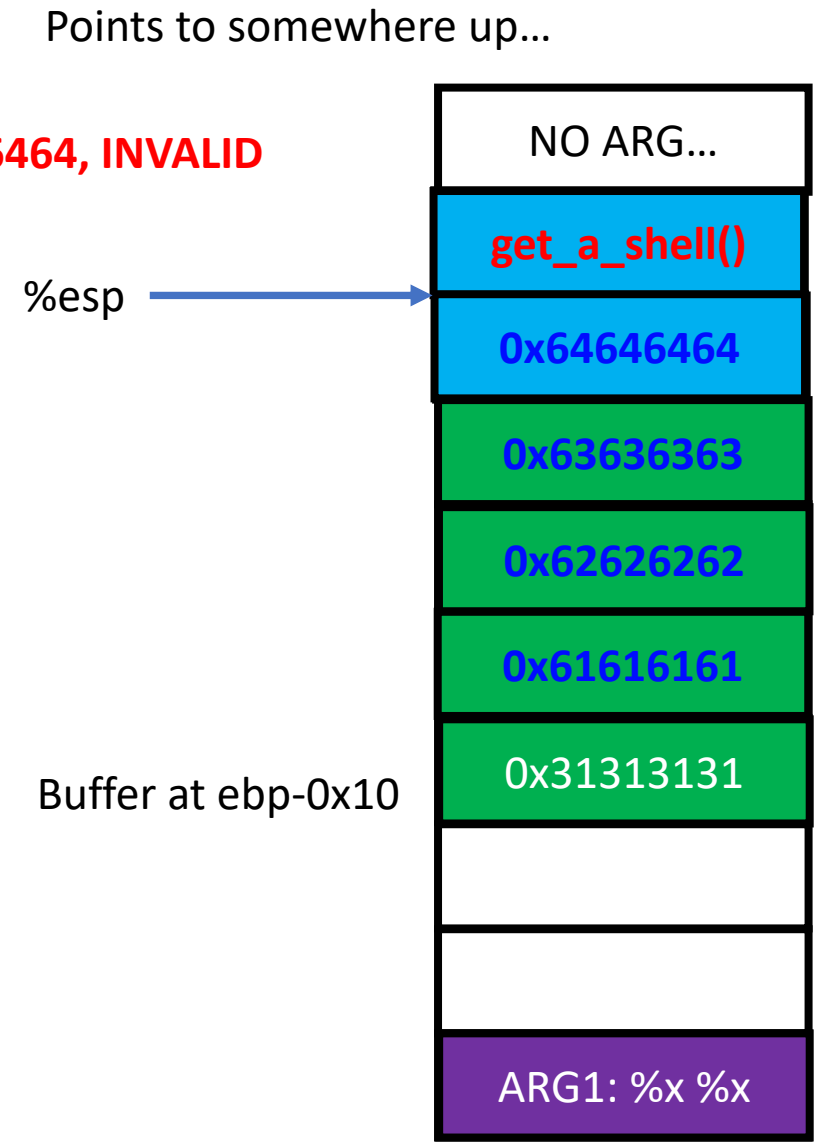
```
add     $0x54,%esp
pop     %esi
pop     %ebp
ret
```



Buffer Overflow

- Function return of receive_input

```
add     $0x54,%esp
pop     %esi
pop     %ebp
ret
```



Buffer Overflow

- Function return of receive_input

```
add     $0x54,%esp
pop     %esi
pop     %ebp
ret
```

Run get_a_shell()!