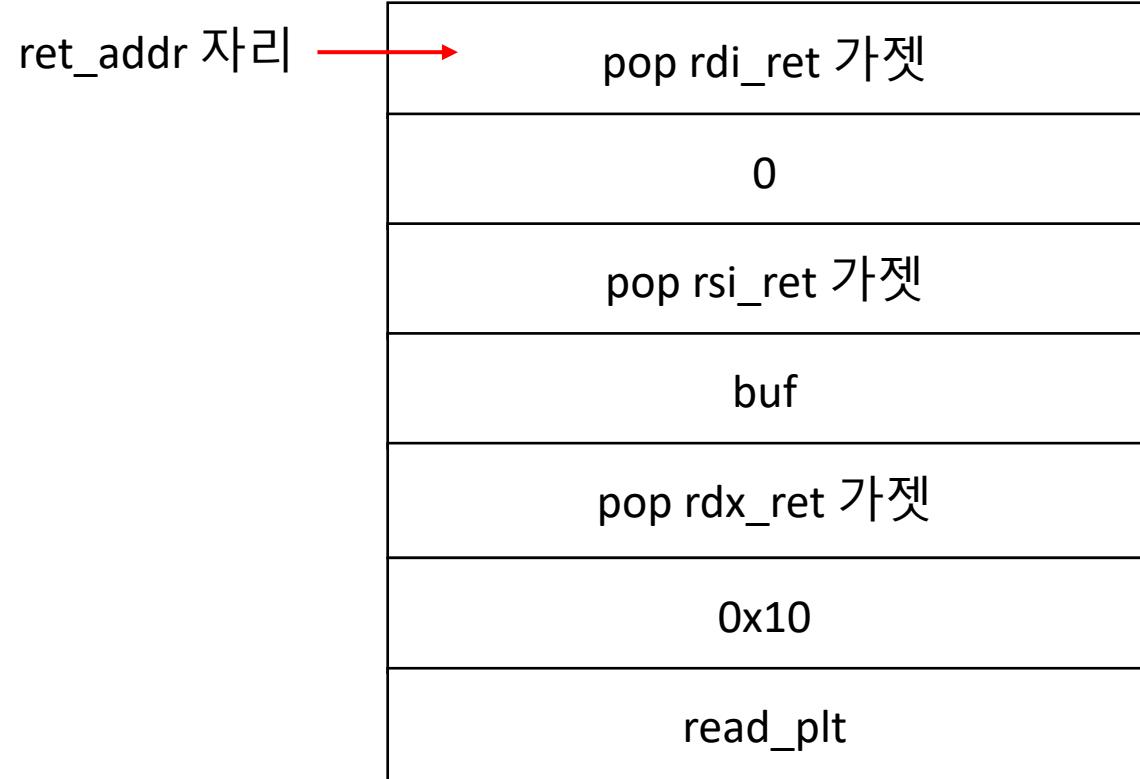


ROP\_2

# 복습

read(0,buf,0x10);



## 32bit vs 64bit (x86 vs x64)

- 32bit : 함수가 매개변수를 스택에서 참조함
- 64bit: 함수가 매개변수를 “레지스터”에서 참조함

# 64bit rop

rdi -> rsi -> rdx -> r10 -> r8 -> r9 ... 순서로 레지스터

ex)

read(0,buf,0x10);

rdi : 0

rsi : buf

rdx : 0x10

- 그렇다면, 가젯이 없다면 어떡할 것인가??

- 그렇다면, 가젯이 없다면 어떡할 것인가??

- 매개변수 3개인 함수 호출 시,
- pop rdi
- pop rsi
- pop rdx

# Return to CSU

- 실제로 함수가 시작될 때,  
`libc_csu_init -> main() -> libc_csu_fini`

```

.text:00000000000400600 loc_400600:           ; CODE XREF:    libc csu init+54↑j
.text:00000000000400600
.text:00000000000400603
.text:00000000000400606
.text:00000000000400609
.text:0000000000040060D
.text:00000000000400611
.text:00000000000400614
.text:00000000000400616
.text:00000000000400616 loc_400616:          ; CODE XREF:    libc csu init+34↑j
.text:00000000000400616
.text:0000000000040061A
.text:0000000000040061B
.text:0000000000040061C
.text:0000000000040061E
.text:00000000000400620
.text:00000000000400622
.text:00000000000400624

mov    rdx, r15
mov    rsi, r14
mov    edi, r13d
call   qword ptr [r12+rbx*8]
add    rbx, 1
cmp    rbp, rbx
jnz   short loc_400600

add    rsp, 8
pop    rbp
pop    rbp
pop    r12
pop    r13
pop    r14
pop    r15
retn

```

```

gdb-peda$ pd __libc_csu_init
Dump of assembler code for function __libc_csu_init:

```

```

0x000000000004005c0 <+0>:    push   r15
0x000000000004005c2 <+2>:    push   r14
0x000000000004005c4 <+4>:    mov    r15,rdx
0x000000000004005c7 <+7>:    push   r13
0x000000000004005c9 <+9>:    push   r12
0x000000000004005cb <+11>:   lea    r12,[rip+0x20083e]      # 0x600e10
0x000000000004005d2 <+18>:   push   rbp
0x000000000004005d3 <+19>:   lea    rbp,[rip+0x20083e]      # 0x600e18
0x000000000004005da <+26>:   push   rbx
0x000000000004005db <+27>:   mov    r13d,edi
0x000000000004005de <+30>:   mov    r14,rsi
0x000000000004005e1 <+33>:   sub    rbp,r12
0x000000000004005e4 <+36>:   sub    rsp,0x8
0x000000000004005e8 <+40>:   sar    rbp,0x3
0x000000000004005ec <+44>:   call   0x400438 <_init>
0x000000000004005f1 <+49>:   test   rbp,rbp
0x000000000004005f4 <+52>:   je    0x400616 <__libc_csu_init+86>
0x000000000004005f6 <+54>:   xor    ebx,ebx
0x000000000004005f8 <+56>:   nop    DWORD PTR [rax+rax*1+0x0]
0x00000000000400600 <+64>:   mov    rdx,r15
0x00000000000400603 <+67>:   mov    rsi,r14
0x00000000000400606 <+70>:   mov    edi,r13d
0x00000000000400609 <+73>:   call   QWORD PTR [r12+rbx*8]
0x0000000000040060d <+77>:   add    rbx,0x1
0x00000000000400611 <+81>:   cmp    rbp,rbx
0x00000000000400614 <+84>:   jne   0x400600 <__libc_csu_init+64>
0x00000000000400616 <+86>:   add    rsp,0x8
0x0000000000040061a <+90>:   pop    rbx
0x0000000000040060d <+77>:   add    rbx,0x1
0x00000000000400611 <+81>:   cmp    rbp,rbx
0x00000000000400614 <+84>:   jne   0x400600 <__libc_csu_init+64>
0x00000000000400616 <+86>:   add    rsp,0x8
0x0000000000040061a <+90>:   pop    rbx
0x0000000000040061b <+91>:   pop    rbp
0x0000000000040061c <+92>:   pop    r12
0x0000000000040061e <+94>:   pop    r13
0x00000000000400620 <+96>:   pop    r14
0x00000000000400622 <+98>:   pop    r15
0x00000000000400624 <+100>:  ret

```

```

0x00000000000400600 <+64>:   mov    rdx,r15
0x00000000000400603 <+67>:   mov    rsi,r14
0x00000000000400606 <+70>:   mov    edi,r13d
0x00000000000400609 <+73>:   call   QWORD PTR [r12+rbx*8]
0x0000000000040060d <+77>:   add    rbx,0x1
0x00000000000400611 <+81>:   cmp    rbp,rbx
0x00000000000400614 <+84>:   jne   0x400600 <__libc_csu_init+64>
0x00000000000400616 <+86>:   add    rsp,0x8
0x0000000000040061a <+90>:   pop    rbx
0x0000000000040061b <+91>:   pop    rbp
0x0000000000040061c <+92>:   pop    r12
0x0000000000040061e <+94>:   pop    r13
0x00000000000400620 <+96>:   pop    r14
0x00000000000400622 <+98>:   pop    r15
0x00000000000400624 <+100>:  ret

```

```
.text:0000000000400600 loc_400600: ; CODE XREF: libc csu init+54↑j
.text:0000000000400600
.text:0000000000400603
.text:0000000000400606
.text:0000000000400609
.text:000000000040060D
.text:0000000000400611
.text:0000000000400614
.text:0000000000400616
.text:0000000000400616 loc_400616: ; CODE XREF: libc csu init+34↑j
.text:0000000000400616
.text:000000000040061A
.text:000000000040061B
.text:000000000040061C
.text:000000000040061E
.text:0000000000400620
.text:0000000000400622
.text:0000000000400624
```

The assembly code shows two sections of instructions. The first section, starting at address 0x400600, contains a loop that iterates until rbp equals rbx. Inside the loop, it moves rdx to r15, rsi to r14, edi to r13d, calls a qword pointer at [r12+rbx\*8], adds 1 to rbx, compares rbp and rbx, and jumps back to the start if they are not equal. The second section, starting at address 0x400616, contains a series of pop and add instructions that restore the registers r12, r13, r14, r15, rbp, and rbx from the stack, followed by a ret instruction.

```
mov    rdx, r15  
mov    rsi, r14  
mov    edi, r13d  
call   qword ptr [r12+rbx*8]  
add    rbx, 1  
cmp    rbp, rbx  
jnz    short loc_400600
```

csu1

ret\_addr 자리

csu2

AAAAAAA

rbx

rbp

r12

r13

r14

r15

csu1

; CODE ;

csu2

```
add    rsp, 8  
pop    rbx  
pop    rbp  
pop    r12  
pop    r13  
pop    r14  
pop    r15  
retn
```

```
mov    rdx, r15  
mov    rsi, r14  
mov    edi, r13d  
call   qword ptr [r12+rbx*8]  
add    rbx, 1  
cmp    rbp, rbx  
jnz    short loc_400600
```

csu1

ret\_addr 자리

csu2

AAAAAAA

rbx

rbp

r12

r13 ( => edi)

r14 ( => rsi)

r15 ( => rdx)

csu1

; CODE

csu2

```
add    rsp, 8  
pop    rbx  
pop    rbp  
pop    r12  
pop    r13  
pop    r14  
pop    r15  
retn
```

```
mov    rdx, r15  
mov    rsi, r14  
mov    edi, r13d  
call   qword ptr [r12+rbx*8]  
add    rbx, 1  
cmp    rbp, rbx  
jnz    short loc_400600
```

csu1

ret\_addr 자리

csu2

AAAAAAA

rbx ( 0 )

rbp

호출할 함수 주소

r13 ( => edi)

r14 ( => rsi)

r15 ( => rdx)

csu1

; CODE ;

csu2

```
add    rsp, 8  
pop    rbx  
pop    rbp  
pop    r12  
pop    r13  
pop    r14  
pop    r15  
retn
```

```
mov    rdx, r15  
mov    rsi, r14  
mov    edi, r13d  
call   qword ptr [r12+rbx*8]  
add    rbx, 1  
cmp    rbp, rbx  
jnz    short loc_400600
```

csu1

ret\_addr 자리

csu2

AAAAAAA

rbx ( 1 )

rbp ( 1 )

호출할 함수 주소

r13 ( => edi)

r14 ( => rsi)

r15 ( => rdx)

csu1

; CODE ;

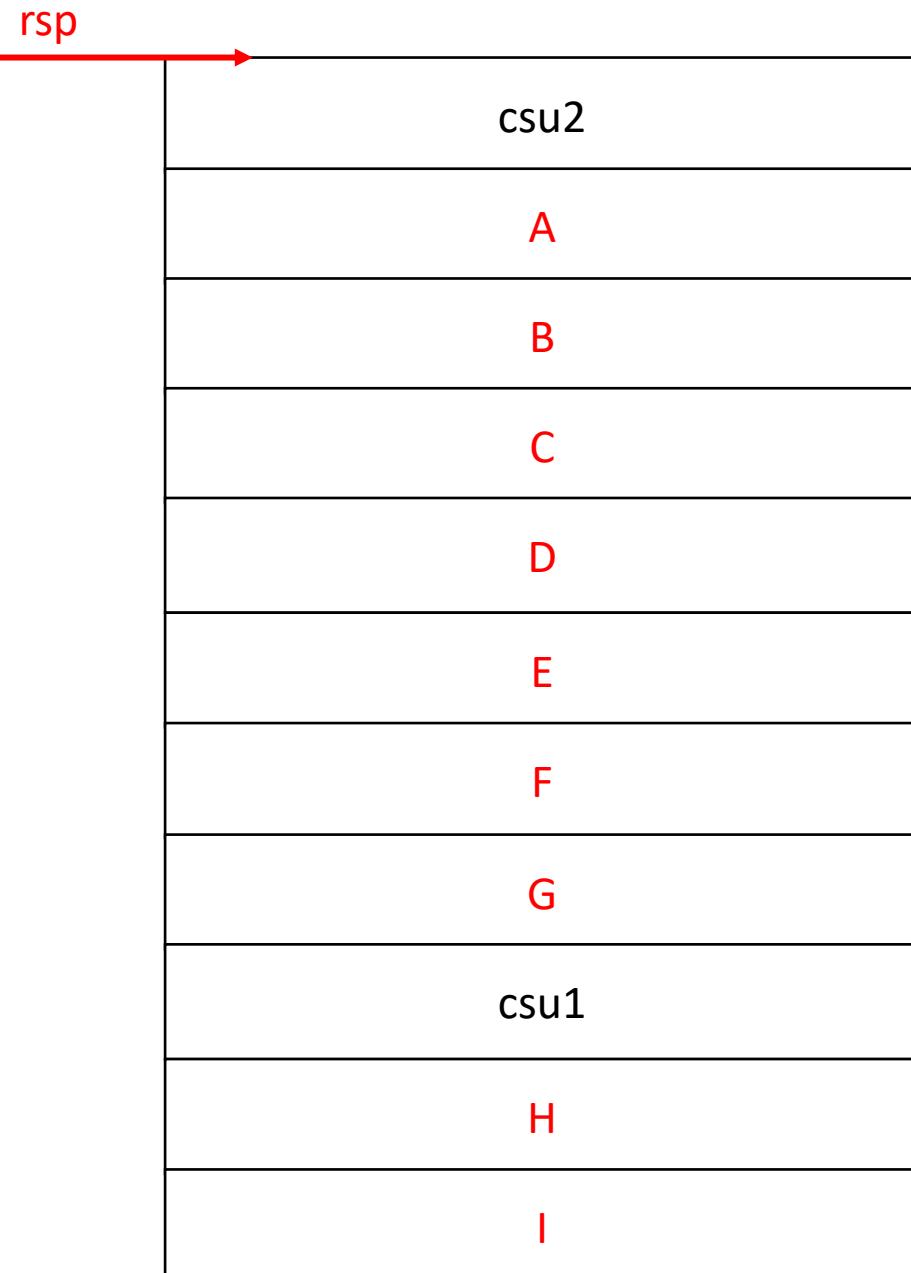
csu2

```
add    rsp, 8  
pop    rbx  
pop    rbp  
pop    r12  
pop    r13  
pop    r14  
pop    r15  
retn
```

```
mov     rdx, r15          ← csu1
mov     rsi, r14
mov     edi, r13d
call    qword ptr [r12+rbx*8]
add    rbx, 1
cmp    rbp, rbx
jnz    short loc_400600
```

```
; CODE : ← csu2
add    rsp, 8
pop    rbx
pop    rbp
pop    r12
pop    r13
pop    r14
pop    r15
ret
```

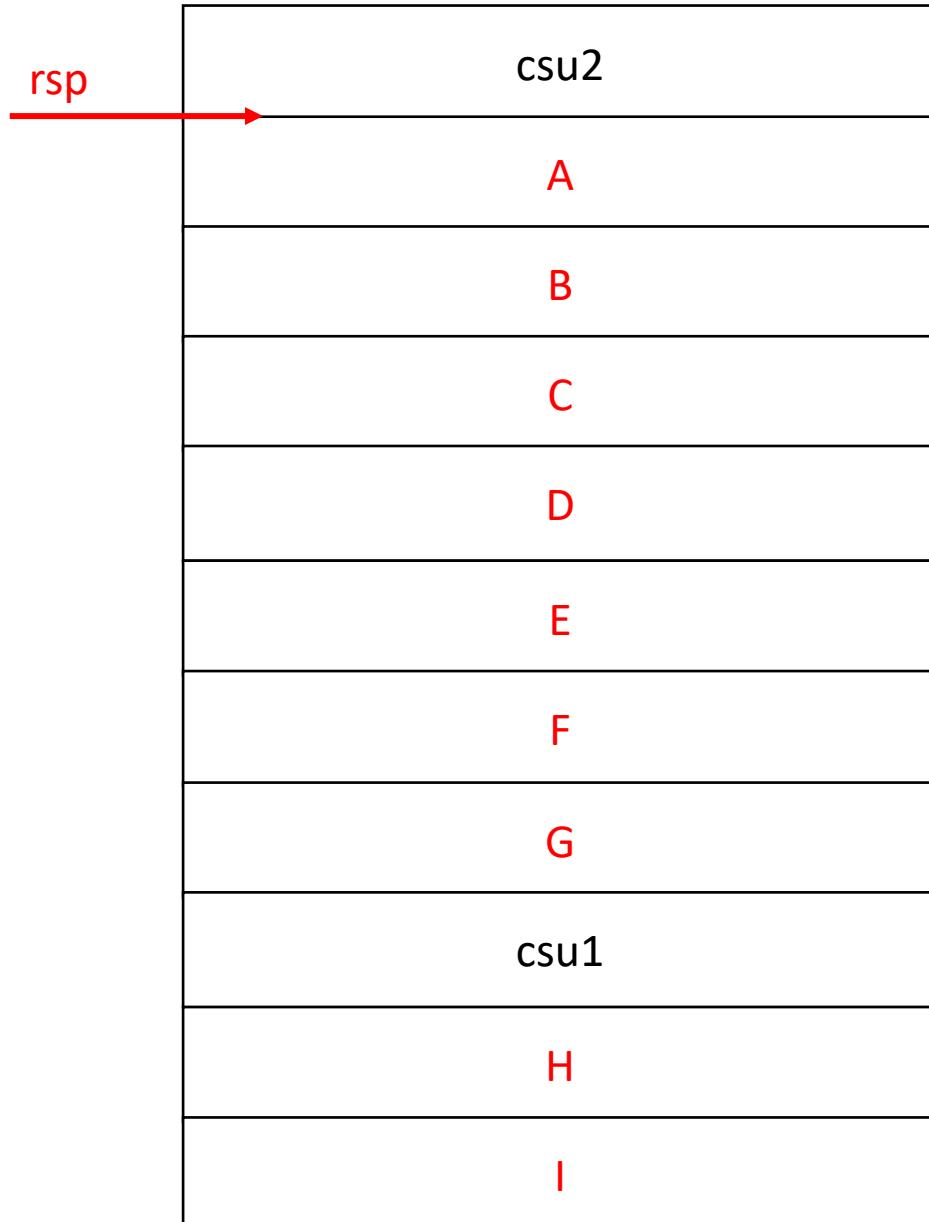
rbx:	rdx:
rbp:	rsi:
r12:	edi:
r13:	
r14:	
r15:	



```
mov     rdx, r15          ← csu1
mov     rsi, r14
mov     edi, r13d
call    qword ptr [r12+rbx*8]
add    rbx, 1
cmp    rbp, rbx
jnz    short loc_400600
```

```
add    rsp, 8   ← rip ; CODE ← csu2
pop    rbx
pop    rbp
pop    r12
pop    r13
pop    r14
pop    r15
ret
```

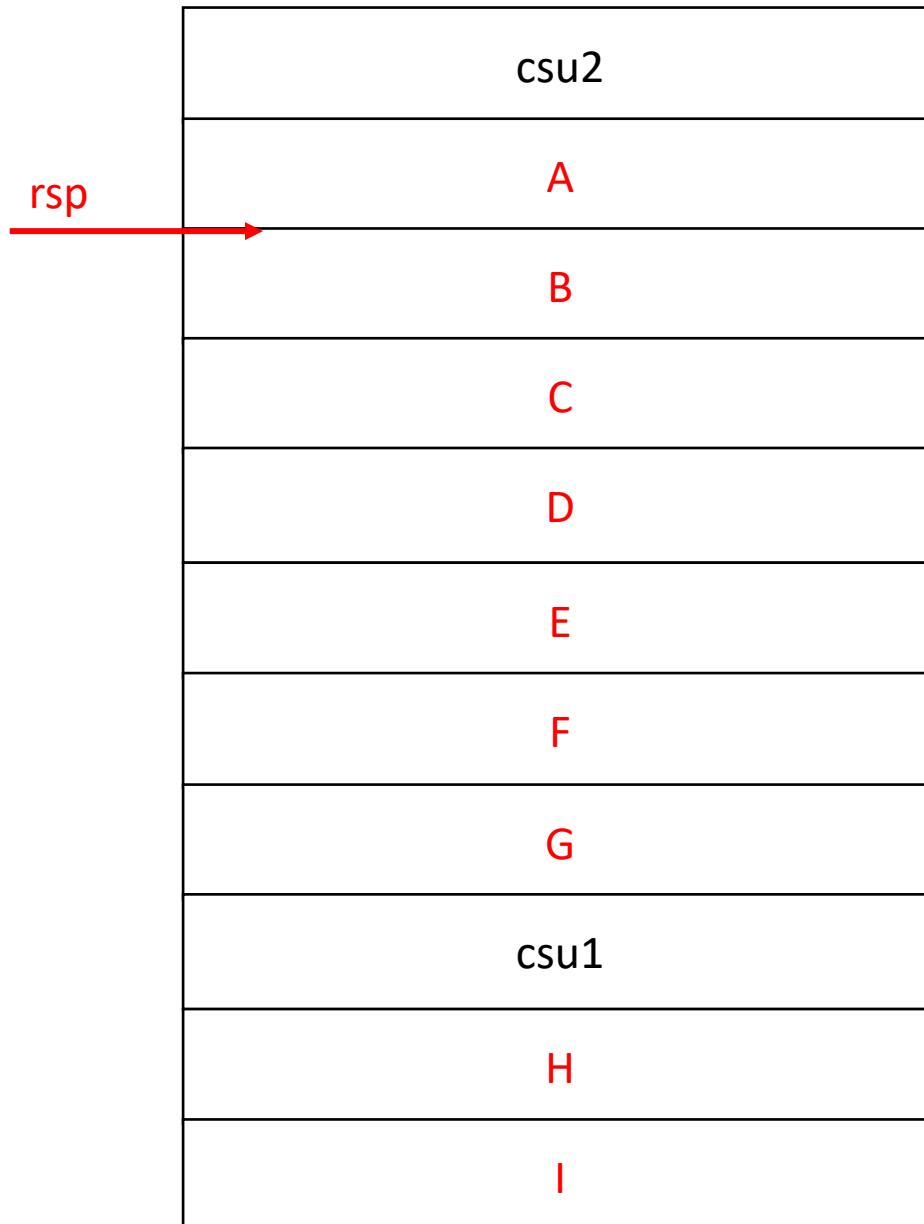
rbx:	rdx:
rbp:	rsi:
r12:	edi:
r13:	
r14:	
r15:	



```
mov     rdx, r15          ← csu1
mov     rsi, r14
mov     edi, r13d
call    qword ptr [r12+rbx*8]
add    rbx, 1
cmp    rbp, rbx
jnz    short loc_400600
```

```
; CODE :          ← csu2
add    rsp, 8             ← rip
pop    rbx
pop    rbp
pop    r12
pop    r13
pop    r14
pop    r15
ret
```

rbx:	rdx:
rbp:	rsi:
r12:	edi:
r13:	
r14:	
r15:	

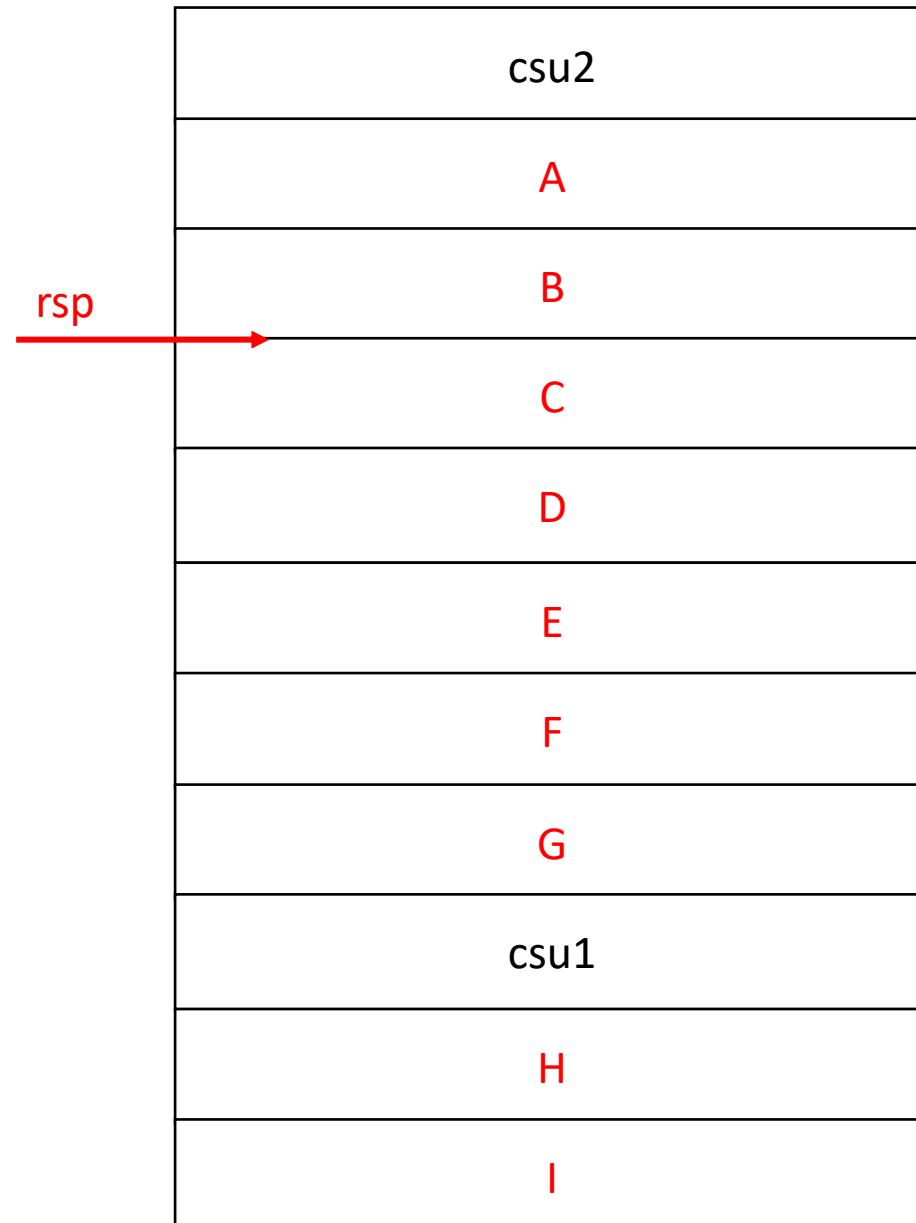


```
mov     rdx, r15          ← csu1
mov     rsi, r14
mov     edi, r13d
call    qword ptr [r12+rbx*8]
add    rbx, 1
cmp    rbp, rbx
jnz    short loc_400600
```

```
; CODE :          ← csu2
add    rsp, 8             ← rip
pop    rbx
pop    rbp
pop    r12
pop    r13
pop    r14
pop    r15
ret
```

rbx: B  
rbp:  
r12:  
r13:  
r14:  
r15:

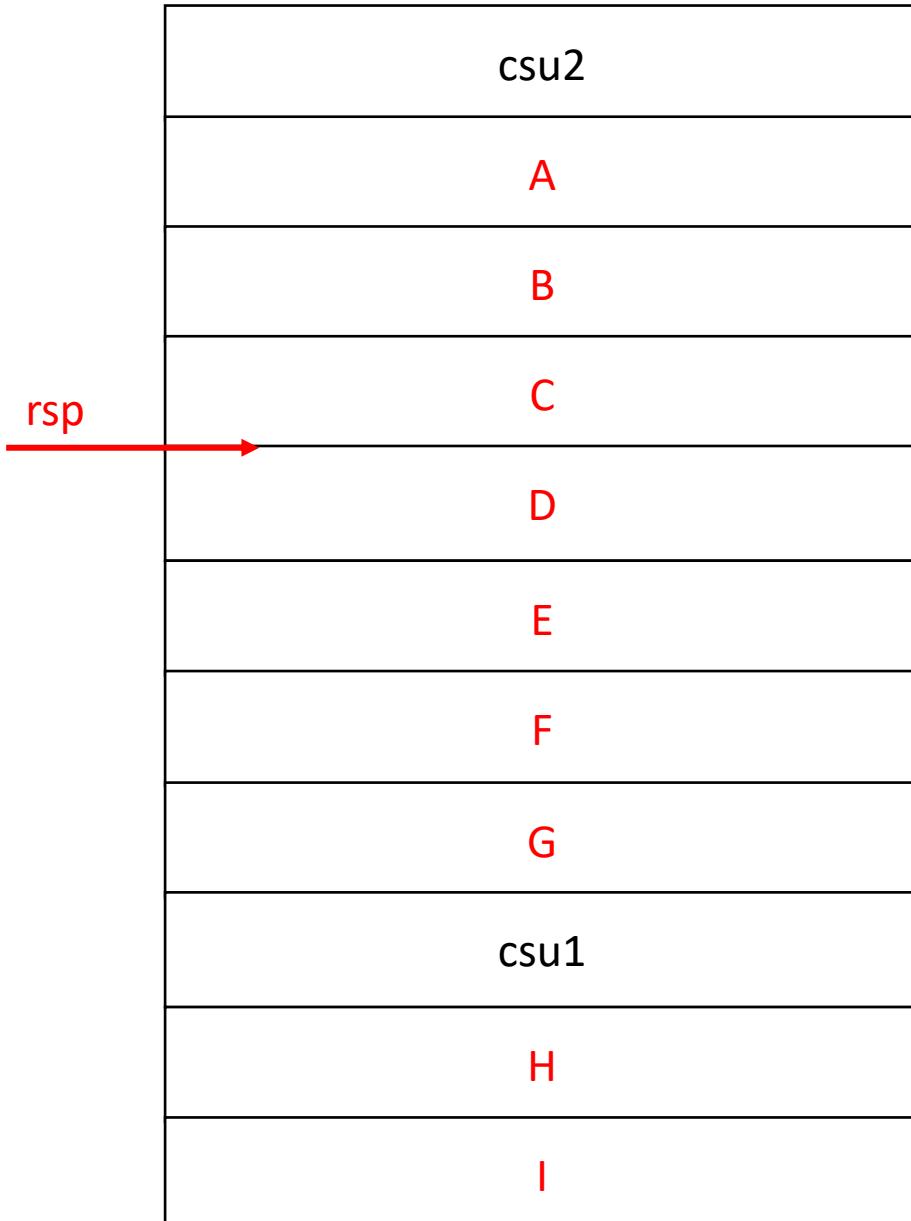
rdx:  
rsi:  
edi:



```
mov     rdx, r15          ← csu1
mov     rsi, r14
mov     edi, r13d
call    qword ptr [r12+rbx*8]
add    rbx, 1
cmp    rbp, rbx
jnz    short loc_400600
```

```
; CODE :          ← csu2
add    rsp, 8
pop    rbx
pop    rbp
pop    r12
pop    r13
pop    r14
pop    r15
ret
```

rbx:	B	rdx:	
rbp:	C	rsi:	
r12:		edi:	
r13:			
r14:			
r15:			



```
mov     rdx, r15  
mov     rsi, r14  
mov     edi, r13d  
call    qword ptr [r12+rbx*8]  
add    rbx, 1  
cmp    rbp, rbx  
jnz    short loc_400600
```

csu1

```
; CODE:  
add    rsp, 8  
pop    rbx  
pop    rbp  
pop    r12  
pop    r13  
pop    r14  
pop    r15  
retn
```

csu2

rip

rbx:	B	rdx:
rbp:	C	rsi:
r12:	D	edi:
r13:		
r14:		
r15:		

csu2

A

B

C

D

E

F

G

csu1

H

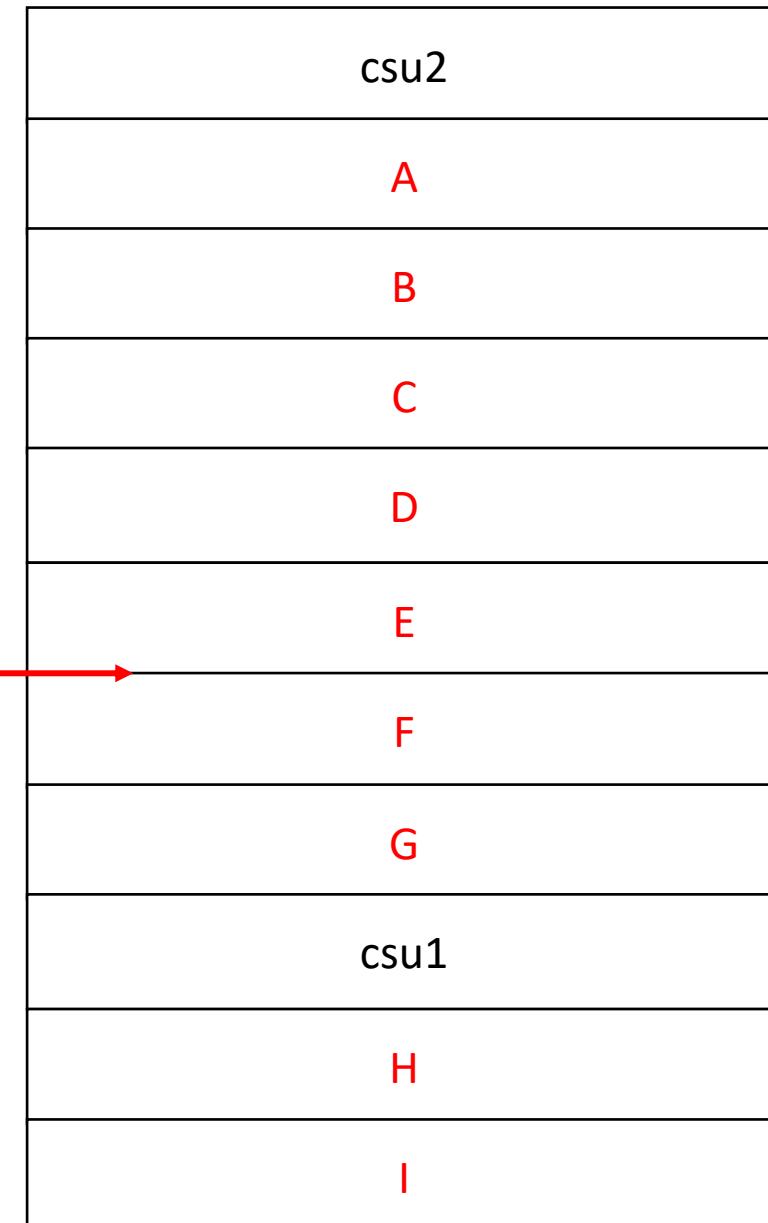
I

rsp

```
mov     rdx, r15          ← csu1
mov     rsi, r14
mov     edi, r13d
call    qword ptr [r12+rbx*8]
add    rbx, 1
cmp    rbp, rbx
jnz    short loc_400600
```

```
; CODE : ← csu2
add    rsp, 8
pop    rbx
pop    rbp
pop    r12
pop    r13
pop    r14
pop    r15
ret
```

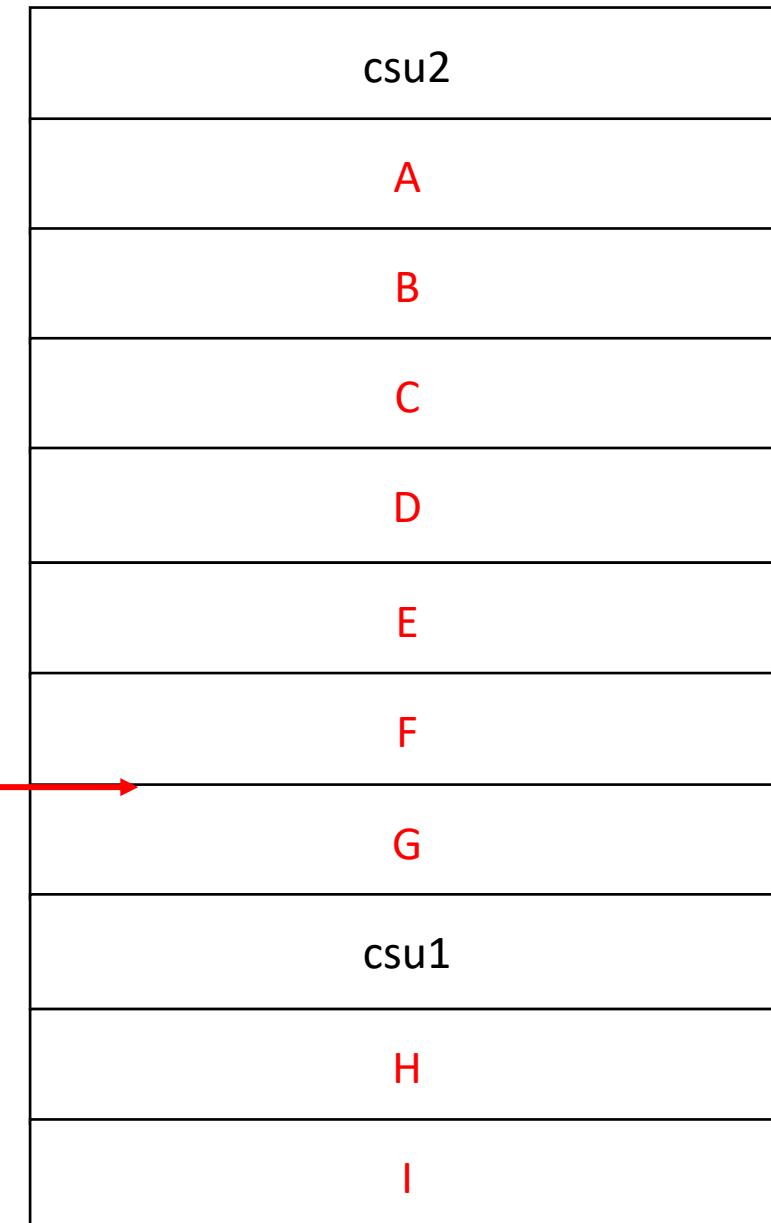
rbx:	B	rdx:	
rbp:	C	rsi:	
r12:	D	edi:	
r13:	E		
r14:			
r15:			



```
mov     rdx, r15          ← csu1
mov     rsi, r14
mov     edi, r13d
call    qword ptr [r12+rbx*8]
add    rbx, 1
cmp    rbp, rbx
jnz    short loc_400600
```

```
; CODE : ← csu2
add    rsp, 8
pop    rbx
pop    rbp
pop    r12
pop    r13
pop    r14
pop    r15
ret
```

rbx:	B	rdx:	
rbp:	C	rsi:	
r12:	D	edi:	
r13:	E		
r14:	F		
r15:			



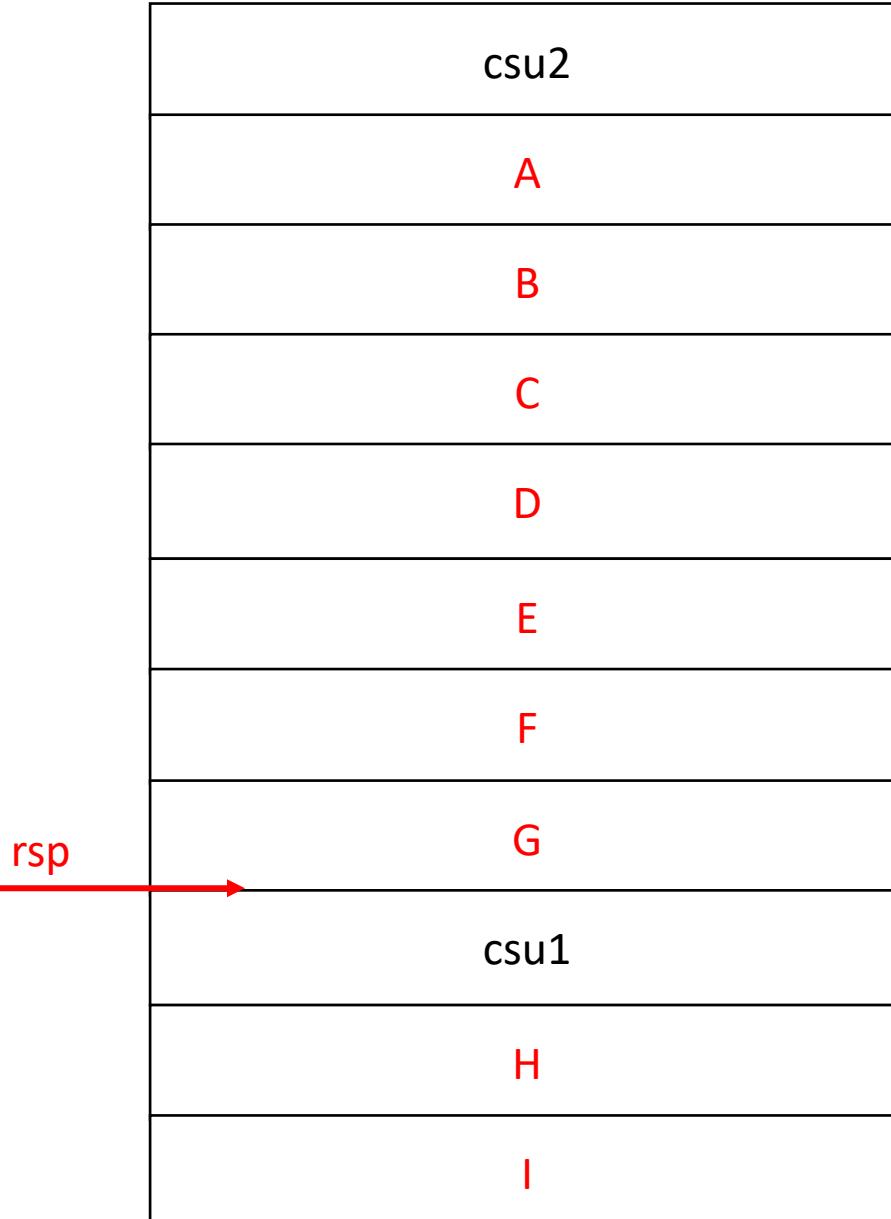
```
mov     rdx, r15  
mov     rsi, r14  
mov     edi, r13d  
call    qword ptr [r12+rbx*8]  
add    rbx, 1  
cmp    rbp, rbx  
jnz    short loc_400600
```

csu1

```
; CODE:  
add    rsp, 8  
pop    rbx  
pop    rbp  
pop    r12  
pop    r13  
pop    r14  
pop    r15  
ret
```

csu2

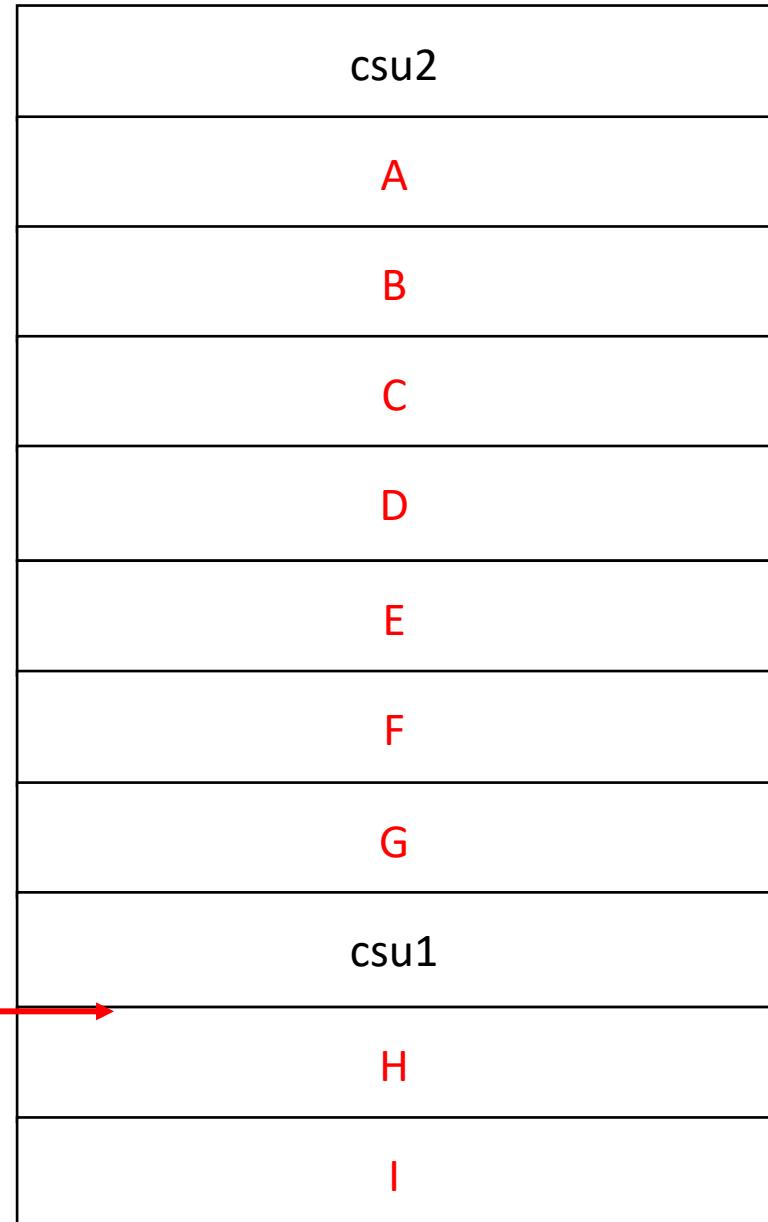
rbx: B                    rdx:  
rbp: C                    rsi:  
r12: D                    edi:  
r13: E  
r14: F  
r15: G



```
rip  
mov rdx, r15 ←  
mov rsi, r14 ← csu1  
mov edi, r13d  
call qword ptr [r12+rbx*8]  
add rbx, 1  
cmp rbp, rbx  
jnz short loc_400600
```

```
; CODE : csu2  
add rsp, 8 ←  
pop rbx  
pop rbp  
pop r12  
pop r13  
pop r14  
pop r15  
retn
```

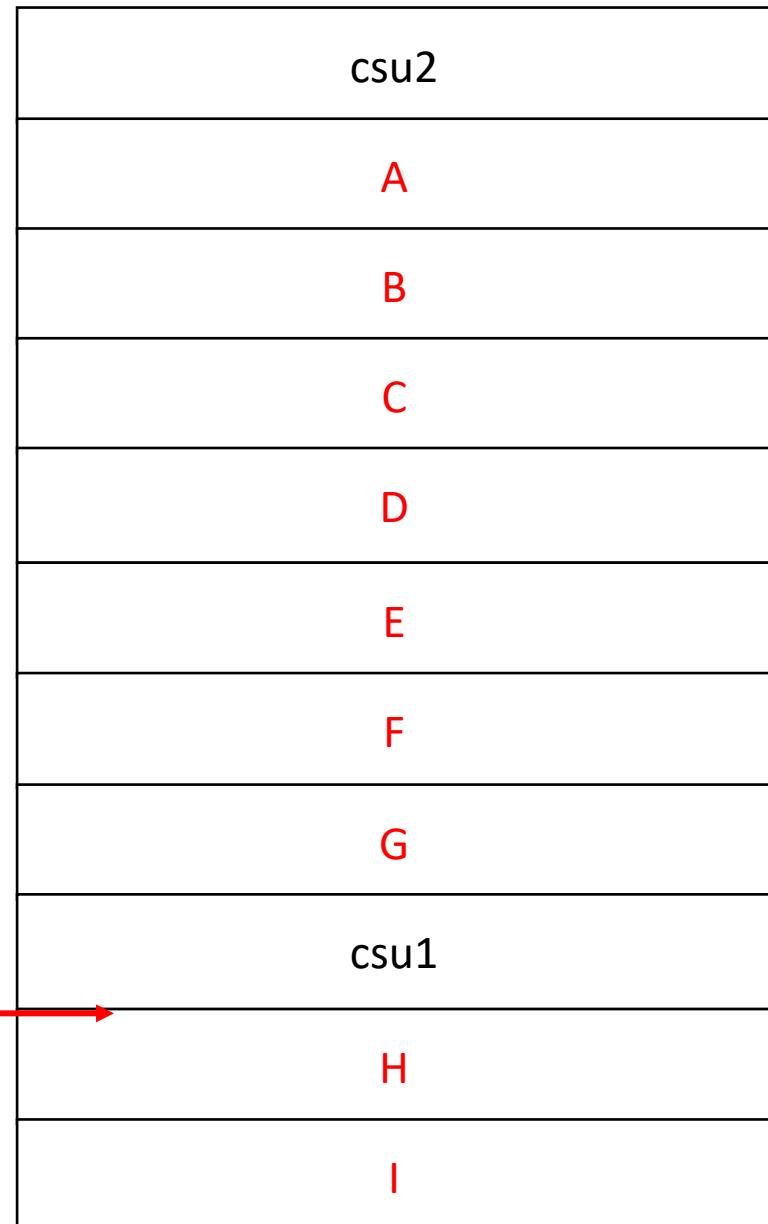
rbx: B	rdx:
rbp: C	rsi:
r12: D	edi:
r13: E	
r14: F	
r15: G	



```
mov     rdx, r15    rip ← csu1
mov     rsi, r14
mov     edi, r13d
call    qword ptr [r12+rbx*8]
add    rbx, 1
cmp    rbp, rbx
jnz    short loc_400600
```

```
; CODE : csu2
add    rsp, 8
pop    rbx
pop    rbp
pop    r12
pop    r13
pop    r14
pop    r15
ret
```

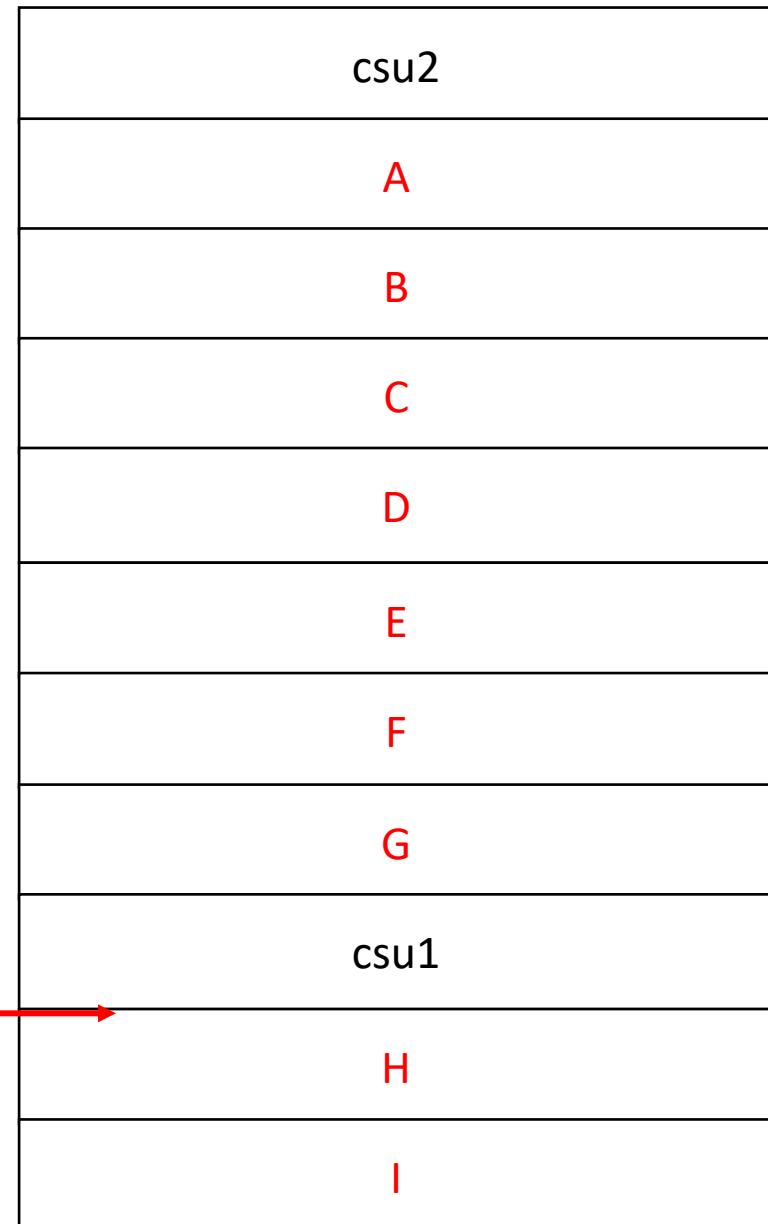
rbx: B	rdx: G
rbp: C	rsi:
r12: D	edi:
r13: E	
r14: F	
r15: G	



```
mov    rdx, r15          rip ← csu1
mov    rsi, r14
mov    edi, r13d
call   qword ptr [r12+rbx*8]
add    rbx, 1
cmp    rbp, rbx
jnz    short loc_400600
```

```
; CODE : csu2
add    rsp, 8
pop    rbx
pop    rbp
pop    r12
pop    r13
pop    r14
pop    r15
ret
```

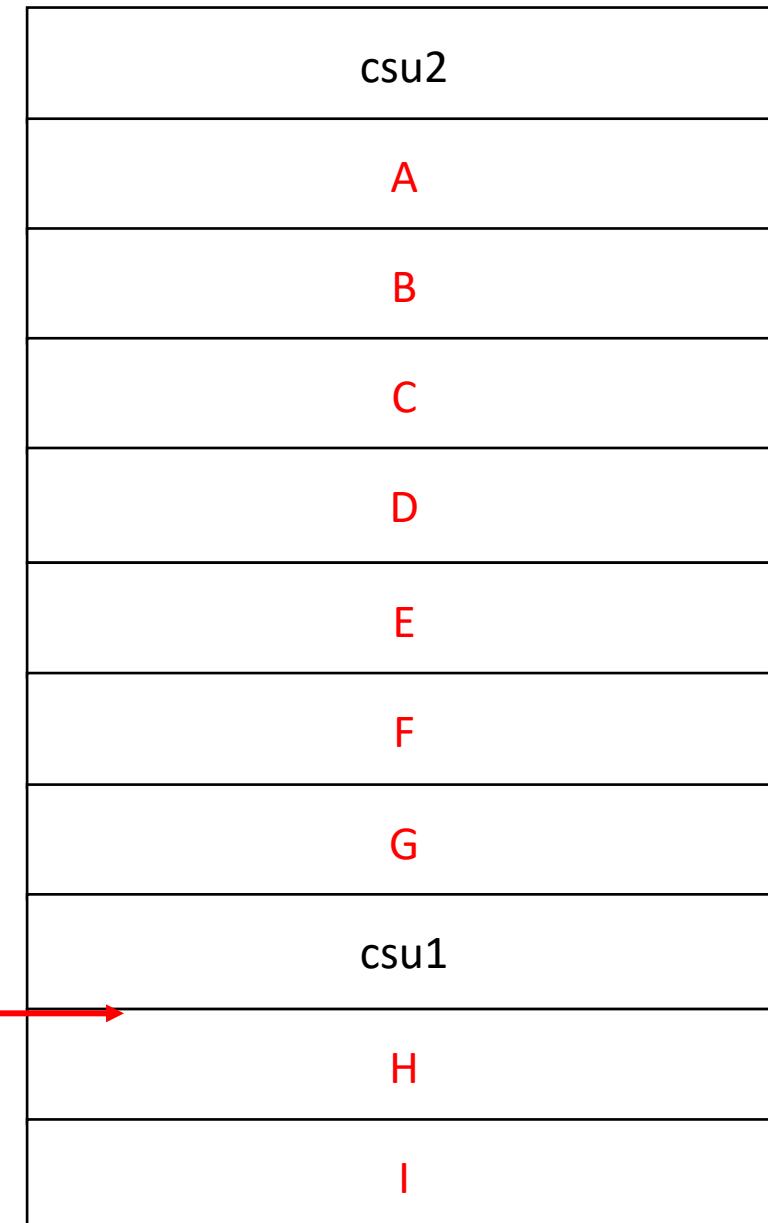
rbx: B	rdx: G
rbp: C	rsi: F
r12: D	edi:
r13: E	
r14: F	
r15: G	



```
mov     rdx, r15          ← csu1
mov     rsi, r14
mov     edi, r13d          ← rip
call    qword ptr [r12+rbx*8]
add    rbx, 1
cmp    rbp, rbx
jnz    short loc_400600
```

```
; CODE :          ← csu2
add    rsp, 8
pop    rbx
pop    rbp
pop    r12
pop    r13
pop    r14
pop    r15
ret
```

rbx: B	rdx: G
rbp: C	rsi: F
r12: D	edi: E
r13: E	
r14: F	
r15: G	



```
mov     rdx, r15          ← csu1
mov     rsi, r14
mov     edi, r13d
call    qword ptr [r12+rbx*8]   rip
add    rbx, 1
cmp    rbp, rbx
jnz    short loc_400600
```

csu2

A

B

• CODE •  
CALL ( D + B \* 8)  
인자 : (E , F , G)

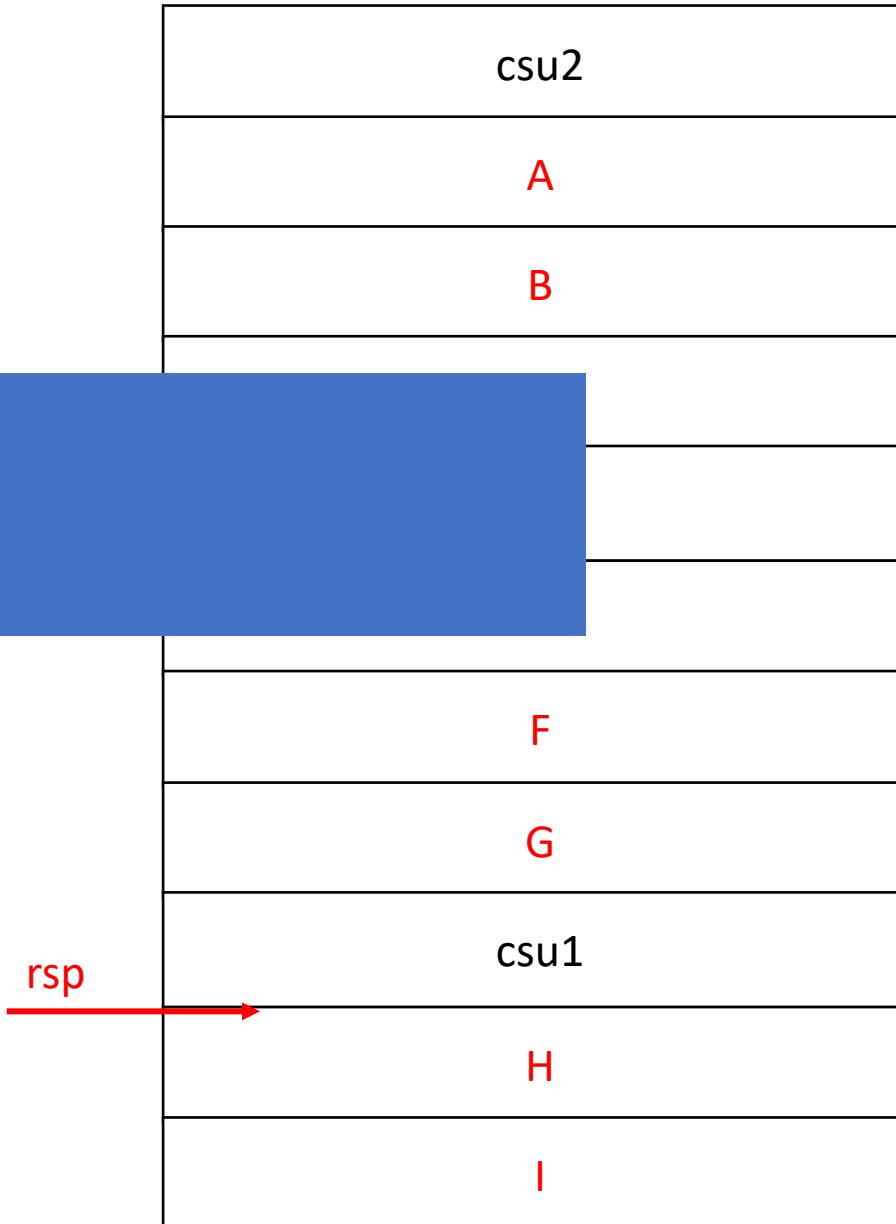
```
add
pop
pop
pop
pop
pop
pop
pop
ret
```

r14

r15

rbx: B  
rbp: C  
r12: D  
r13: E  
r14: F  
r15: G

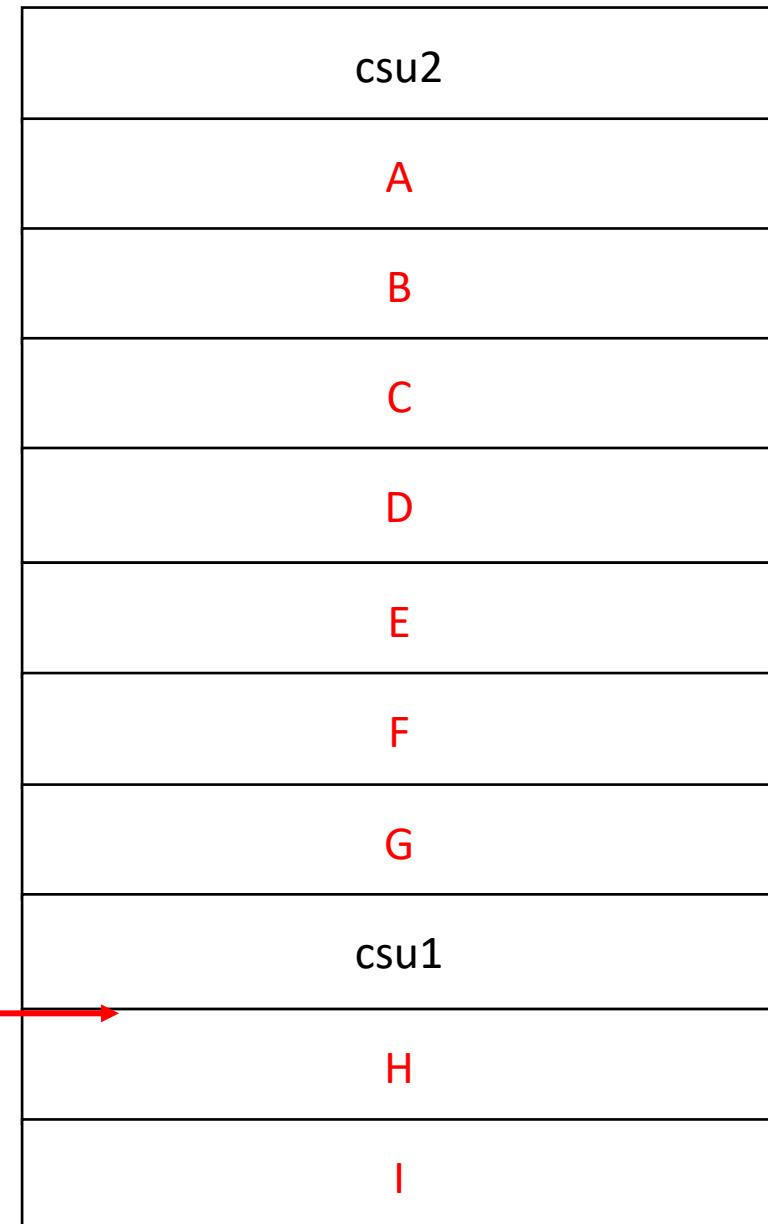
rdx: G  
rsi: F  
edi: E



```
mov     rdx, r15          ← csu1
mov     rsi, r14
mov     edi, r13d
call    qword ptr [r12+rbx*8]
add    rbx, 1              rip
cmp    rbp, rbx
jnz    short loc_400600
```

```
; CODE : csu2
add    rsp, 8
pop    rbx
pop    rbp
pop    r12
pop    r13
pop    r14
pop    r15
ret
```

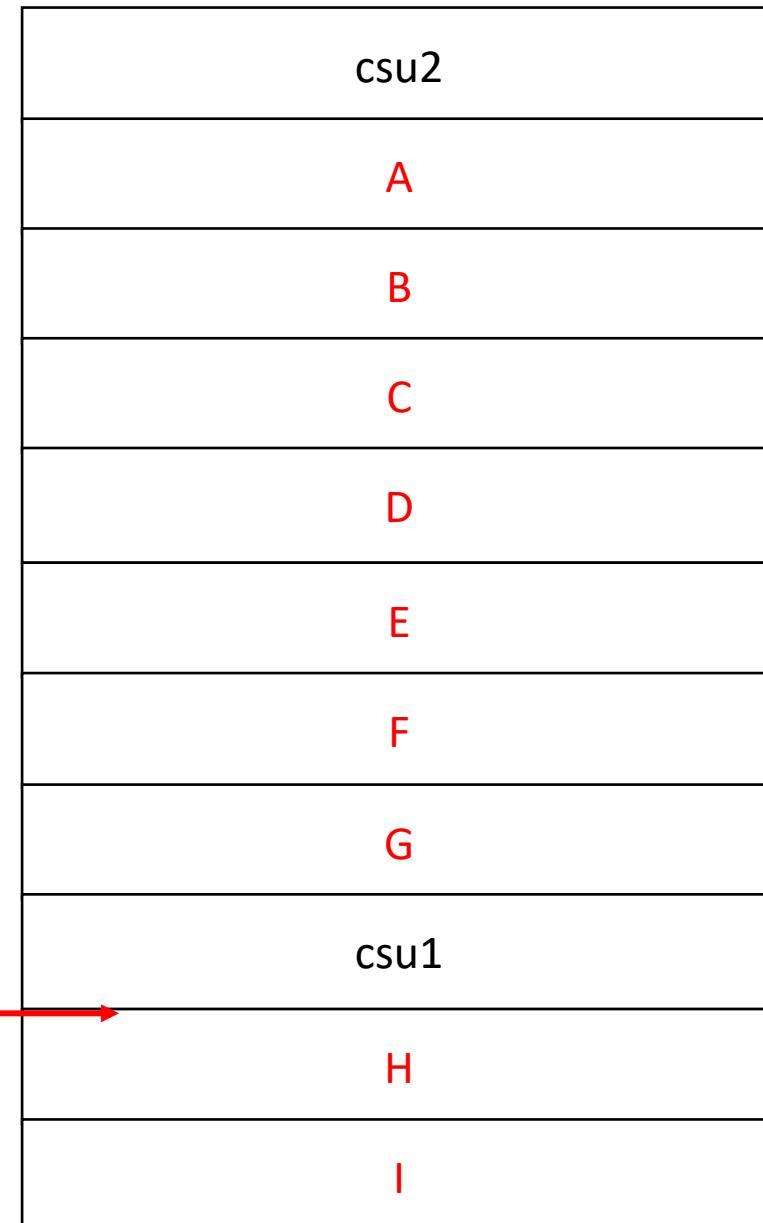
rbx: B +1	rdx: G
rbp: C	rsi: F
r12: D	edi: E
r13: E	
r14: F	
r15: G	



```
mov     rdx, r15          ← csu1
mov     rsi, r14
mov     edi, r13d
call    qword ptr [r12+rbx*8]
add    rbx, 1
cmp    rbp, rbx           rip
jnz    short loc_400600
```

```
; CODE : ← csu2
add    rsp, 8
pop    rbx
pop    rbp
pop    r12
pop    r13
pop    r14
pop    r15
ret
```

rbx: B +1	rdx: G
rbp: C	rsi: F
r12: D	edi: E
r13: E	
r14: F	
r15: G	



	csu2
	0
	0
	1
	호출할 함수 주소
	매개변수 1
	매개변수 2
	매개변수 3
	csu1
다시 위의 행위 반복	아무거나 7개
csu1	main 함수

# 과제

디스어셈블러 Ghidra 개인적으로 익혀보기  
(다른 디스어셈블러 쓰는 사람은 해당 X)

HackCTF – RTC

+ 방학에 시간 많으니, wargame 사이트 하나 꾸준히 풀어보세요