0xf12	0xe4

Fill in the following table with the corresponding form, translation, and value for each of the given operands. Revision Corrected translation

Operand	Form	Translation	Value
%rax	Register	% rax	0xf08
0x4(%rax)	Memory	M[% rax +4] or M[0xf0c]	0x14
0x4(%rax, %r8, 4)	Memory	M[0xf2c]	null
0xf02(,%rsi,2)	Memory	M[0xf02 + % rsi *2] M[0xf12]	0xe4

```
0000000000116d <main>:
--> 1171: 55                    push   %rbp
    1172: 48 89 e5              mov    %rsp,%rbp
    1175: 48 83 ec 10           sub    $0x10,%rsp
    1179: e8 cb ff ff ff        callq  1149 <assign>
    117e: e8 da ff ff ff        callq  115d <adder>
    1183: 89 45 fc              mov    %eax,-0x4(%rbp)
    1186: 8b 45 fc              mov    -0x4(%rbp),%eax
    1189: 89 c6                 mov    %eax,%esi
```

Register	Value
%eax	379
%edi	1
%rsp	0xe68
%rbp	0x1578
%rip	0x1171

"Stack top"

Address	Stack value
0xe68	

Revision ~~Added~~ Answered underline question

sub $8, %rsp

mov %rbp, (%rsp)

Modify the above contents to execute the instruction `push %rbp`. What two steps does `push %rbp` do?

2) What are the contents of registers and the stack after executing `mov %rsp, %rbp`? Please show the before and after state (cross out old values).

```
0000000001149 <assign>:
 114d: 55                              push   %rbp
 114e: 48 89 e5                        mov    %rsp,%rbp
 1151: c7 45 fc 28 00 00 00            movl   $0x28,-0x4(%rbp)
 1158: 8b 45 fc                        mov    -0x4(%rbp),%eax
 115b: 5d                              pop    %rbp
--> 115c: c3                           retq
```

Register	Value
%eax	0x28
%edi	1
%rsp	~~0xe48~~ 0xeJ0
%rbp	0xe60
%rip	~~115c~~ 0x117e

"Stack top"

Revision

Missing stack top

Address	Stack value	
0xe36	0x28	←
0xe40	0xe60	←
0xe48	0x117e	
0xeJ0		
0xeJ8		
0xe60	0x1578	
0xe68		

7) Skipping ahead, what are the state of registers and the stack when executing `retq` in adder? Please show the before and after state (cross out old values).