4.4

Case(i)

(When I wrote this down, I kind of like not realizing the way to find return address using ebp + 1, so I use this method when I'm doing case1, however, when I realize I can use ebp+1 to find the address, there is no time for me to go back and revise, I hope my credits won't be reduced because of the method I use)

I print out the stacktrace of receiver's pid to locate the return address of case (i), I set up a initial pointer at the prptr->prstkptr, which is the same address as esp since it returned,

```
...creating a shell
into callback
into receive
into send
the adr of ptr is 0xEFD8F50
the adr of ptr after is 0xEFD8FC8
the content of saveptr is 1067223
```

When I got this kprintf I created in receive test, I know I get into receive callback function.

which is

```
the adr of ptr is 0xEFD8F50
the adr of ptr after is 0xEFD8FC8
the content of saveptr is 1067223
sp EFD8F50 fp EFD8F58 proc->prstkbase EFD8FFC
DATA (0EFD8F50) 00000000 (0)
DATA (0EFD8F54) 00000000 (0)
```

As the graph above shows, the pointer of prstkptr is EFD8F50

On the other hand, I follow the return address of stacktrace, the first return address is

```
FP   (0EFD8F58) 0EFD8F74 (251498356)
RET  0xEFD8F5C
DATA (0EFD8F60) 00121000 (1183744)
DATA (0EFD8F64) 00121268 (1184360)
DATA (0EFD8F68) 00000000 (0)
```

Which is for the return address of stacktrace(),

```
FP   (0EFD8F74) 0EFD8FA4 (251498404)
RET  0xEFD8F78
DATA (0EFD8F7C) 001212B4 (1184436)
DATA (0EFD8F80) 00121268 (1184360)
DATA (0EFD8F84) 0010276E (1058670)
```

0XEFD8F78 is the return address for resched().

```
FP   (0EFD8FA4) 0EFD8FC4 (251498436)
RET  0xEFD8FA8
DATA (0EFD8FAC) 00121218 (1184280)
DATA (0EFD8FB0) 0EFC9000 (251432960)
DATA (0EFD8FB4) 001212B0 (1184432)
```

0XEFD8FA8 is the return address for dispath();

```
FP   (0EFD8FC4) 0EFD8FF4 (251498484)
RET  0xEFD8FC8
DATA (0EFD8FCC) 00002710 (10000)
DATA (0EFD8FD0) 00000000 (0)
```

0x EFD8FC8 is the return address of sleepms() I implemented in the receiver testcase.
Then I move my created pointer from current stack pointer to sleepms(), which is 0x EFD8FC8 – 0x EFD8F50 = 120 bytes
Since int pointer is 4 bytes, which means if I want to move the pointer, which is 120/4 = 30, I should +30 to the int pointer, which moves the pointer to the return address of sleepms().
On the other hand, I define a new global pointer, and copy the content of the original return address, which is savedptr.

```
unsigned int* ret;
asm("movl %%ebp, %0\n\t"
    :"=r"(ret)
    :
    :
);
```

In xruncb_uh, I use the assembly code to move my newly created pointer ret to ebp, and since address of eip is 4 bytes above ebp, I use ret+1 as the address of eip, because1*4 = 4, which you can see in xruncb_uh, it's 0x EFD8FCC, and since I stored the savedptr, I copy the content in savedptr and put that value in return address of xruncb_uh to return to the original return address, which back to receive.
I use arm assembly to point the int pointer I created to ebp. As the graph shows,

```
the adr of ptr afterrrrr in cb is 0xEFD8FDC
```

```
FP   (0EFD8FD8) 0EFD8FF4 (251498484)
RET  0xEFD8FDC
```

The address of ptr I created and the return address are the same. Those address are the same,

```
into xruncb_uh
```

```
The callback message received is get1
```

And it did get into xruncb_uh() and mycallback().

```
DATA (0EFD8FF0) 00000000 (0)

FP   (0EFD8FF4) 0EFD8FFC (251498492)
RET  0xEFD8FF8
STACKMAGIC (should be A0AAAA9): A0AAAA9
the content of saveptr in cb is 1067236
the adr of ptr in cb is 0xEFD8FD8
the adr of ptr afterrrrr in cb is 0xEFD8FDC
the message received is get1
```

It gets back to kprintf" the message received is get1" in my testreceive(), means it is successfully implemented.

Case(ii)

I write the separate cases for (i) and (ii) in send.c.

I made my revision in clkhandler.c inside the if(preempt -- < 0) statement. Just like in case 1, I create an unsigned integer pointer, and then mark it to ebp, and +10 which is at the location of eip in clkdisp.

Because +1 is the return address of clkhandler, it must adds 8 standard registers and 1 eflag, which makes it to be 1+9 = 10.

Then I made a if statement, check the currpid's prcbreg = TRUE, which means it's the receiver process.

```
...creating a shell
into regcallback
inthe content of saveptr is 151373449
into xruncb_lh
into send
The callback message received is get1
sp EFD8E80 fp EFD8E98 proc->prstkbase EFD8FFC
```

As you can see, it successfully gets into xruncb_lh, and gets into callback, and my callback function received the message.

Receive test1 is for case(i), and receive test2 is for case(ii).

Bonus

My kill(pid32) receives a pid, and clear out all of its nodes using freemem(), I write the function mqextract in problem3, so I can use that function to call out a loop of certain function.

```
while (prptr->prmsgcount <= 7){
        mqextract();
}
```