

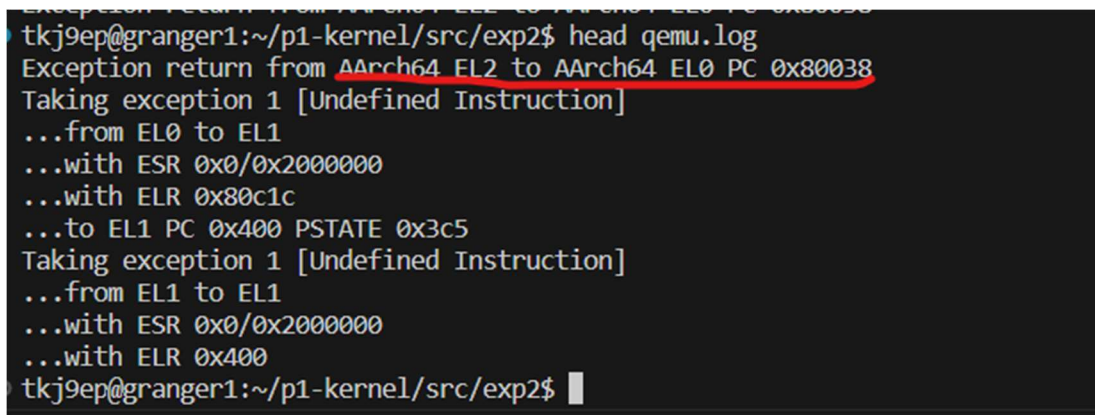
The (weights) of questions are relative and may not necessarily add up to 100.

1. Modify boot.S, so that your kernel switches to EL0 (instead of switching to EL1).

Note: Some students may want to modify other file(s), which are not necessary. But it is fine you insist that. Just submit everything you changed. See below.

Note: you may find QEMU's debug log useful, which traces the exception level.

- a. (30) (QEMU users only) Attach a screenshot here showing that your code works. The screenshot may be messages printed from your kernel execution, or a qemu log.



```
tkj9ep@granger1:~/p1-kernel/src/exp2$ head qemu.log
Exception return from AArch64 EL2 to AArch64 EL0 PC 0x80038
Taking exception 1 [Undefined Instruction]
...from EL0 to EL1
...with ESR 0x0/0x2000000
...with ELR 0x80c1c
...to EL1 PC 0x400 PSTATE 0x3c5
Taking exception 1 [Undefined Instruction]
...from EL1 to EL1
...with ESR 0x0/0x2000000
...with ELR 0x400
tkj9ep@granger1:~/p1-kernel/src/exp2$
```

- b. (20) Briefly explain: what changes have you done? What register(s) do you have to touch? What values do you put in the register(s) and why?

I have changed the sysregs.h SPSR_EL1h to (0 << 0) which would have changed the spsr_el2 register. The value 111000000 through SPSR_VALUE is stored into x0 initially (which eventually gets updated with the memory address of el1_entry) and into the system register spsr_el2.

- c. (20) Upload your code a separate tarball. **Only include the file(s) you modified in the tarball**

How to pack a code tarball (for this & future submission)

File name: [computingid].tar.gz. e.g. lg8sp.tar.gz. All lower case. Use the command 'tar -czvf' to generate.

(Optional) A file README-cs.txt, which contains anything TA should know (e.g., any caveats of your code, extra build instructions)

Absolutely no any binaries (e.g. *.elf, *.o, *.bin) or .git/ subdirectory in the tarball.

The (weights) of questions are relative and may not necessarily add up to 100.

Sample command 1: to compress all source files under a directory

Note: no trailing space after “\”

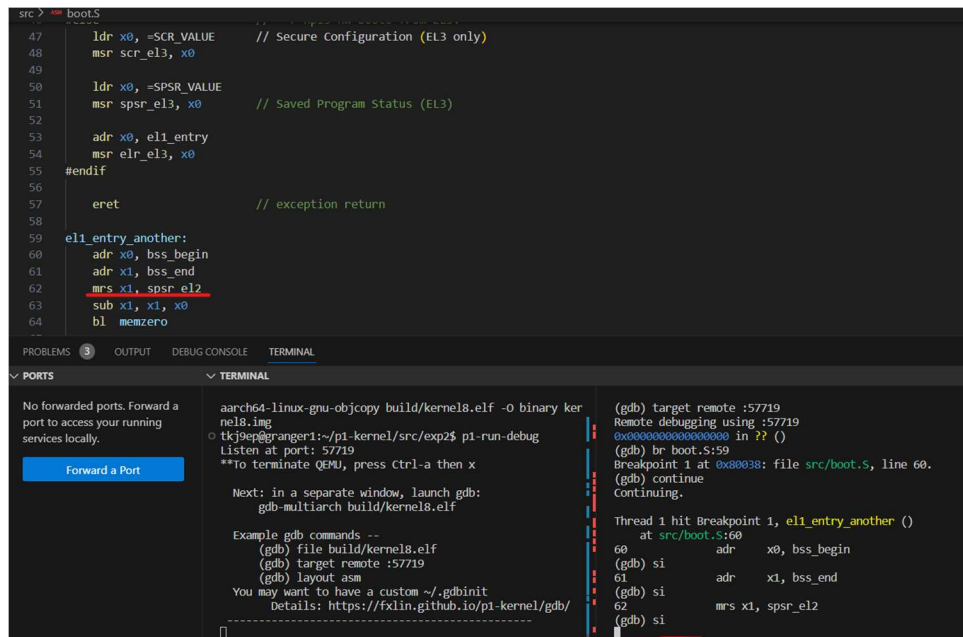
```
cd YOURDIR
tar czvf ../[computingid].tar.gz \
--exclude='*.o' \
--exclude='*.d' \
--exclude='*.bin' \
--exclude='*.elf' \
--exclude='*.img' \
--exclude='*.git/*' \
*
```

Sample command 2: to compress specific files in a directory

Note: no trailing space after “\”

```
cd YOURDIR
tar czvf ../[computingid].tar.gz \
my1.c my1.h my2.c my2.h
```

d. (20) Can you demonstrate that the kernel ACTUALLY reaches EL0? For instance, can you execute some instructions disallowed at EL0 before/after the switch, and reason about the results? Explain your choice and observation. Attach screenshot(s) if needed.



```
src > boot.S
47      ldr x0, =SCR_VALUE      // Secure Configuration (EL3 only)
48      msr scr_el3, x0
49
50      ldr x0, =SPSR_VALUE
51      msr spsr_el3, x0      // Saved Program Status (EL3)
52
53      adr x0, el1_entry
54      msr elr_el3, x0
55  #endif
56
57      eret      // exception return
58
59  el1_entry_another:
60      adr x0, bss_begin
61      adr x1, bss_end
62      mrs x1, spsr_el2
63      sub x1, x1, x0
64      bl  memzero

(gdb) target remote :57719
Remote debugging using :57719
0x0000000000000000 in ?? ()
(gdb) br boot.S:59
Breakpoint 1 at 0x00038: file src/boot.S, line 60.
(gdb) continue
Continuing.

Thread 1 hit Breakpoint 1, el1_entry_another ()
at src/boot.S:60
60      adr    x0, bss_begin
(gdb) si
61      adr    x1, bss_end
(gdb) si
62      mrs    x1, spsr_el2
(gdb) si
```

To test whether it was in EL0, I simply added a privileged instruction in `el1_entry_another` and redirected the address to `el1_entry_another`. When running the gdb, the system freezes because the instruction runs on EL0 and that instruction is illegal. In addition, since there is no exception handling, the OS will panic. A similar instruction was executed prior to the `eret` which means that those instructions executed with minimal issue.

The (weights) of questions are relative and may not necessarily add up to 100.

2. (30) After landing in EL0, can your kernel print out the current exception level?

If so, attach a screenshot showing the printout.

If not, explain why.

It will not be able to because in order to get the current exception level, the “get_el()” needs to call the “mrs” instruction which is illegal in EL0.

3. (10) What does an “eret” instruction do?

The instruction “eret” returns from the exception. More specifically, it restores the ALU flags, execution state, exception level, and processor branches of SPSR_ELn. In addition, it jumps to the address stored in ELR_ELn by modifying the PC.

(10) Does an “eret” instruction must correspond to an earlier exception?

No, because we can simply modify the system register SPSR_ELn to store whichever exception level we want.

(10) What will happen if we execute an `eret` instruction at EL0?

It would be pointless but “eret” would return to EL0 and continue through the rest of the process. That is, moving PC to the memory address stored in ELR_ELn.