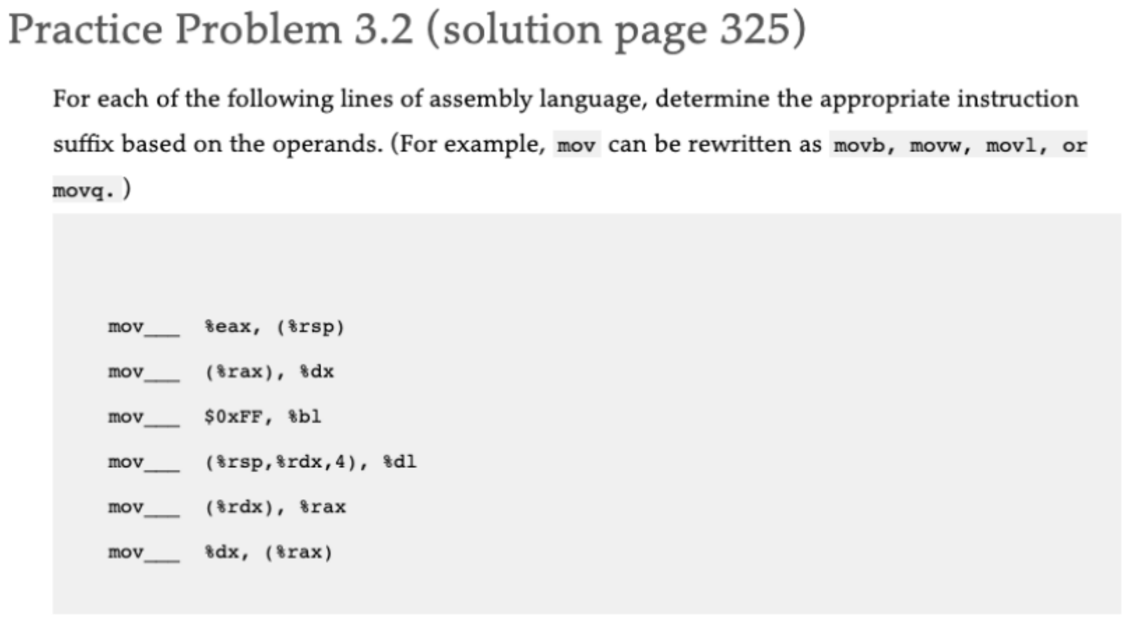
1. Answers

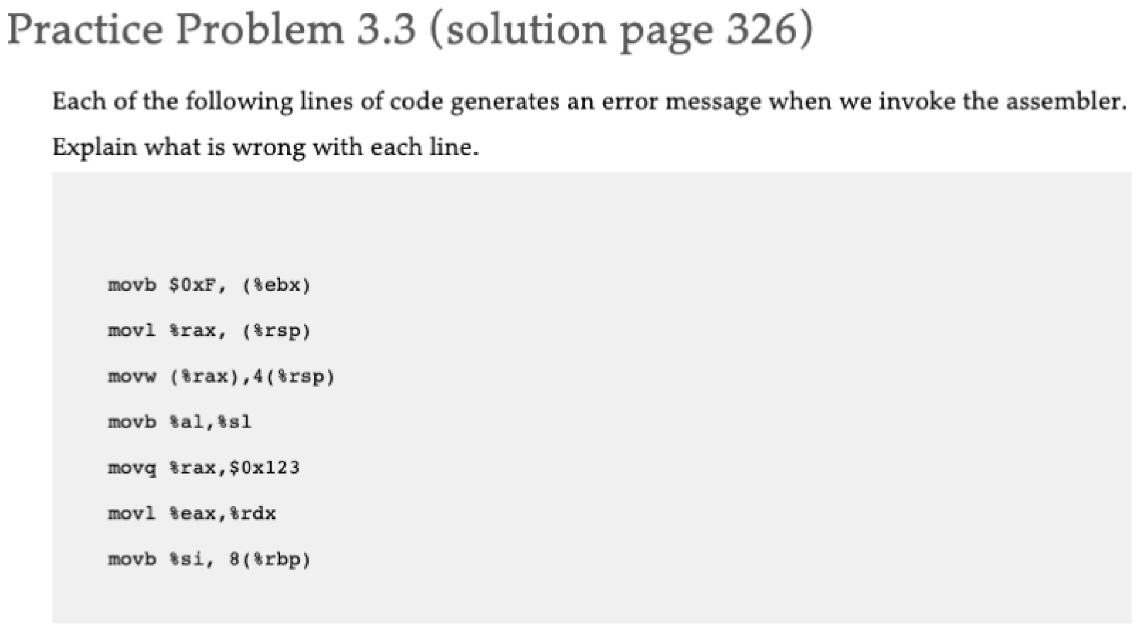


Practice Problem 3.2 (solution page 325)

For each of the following lines of assembly language, determine the appropriate instruction suffix based on the operands. (For example, `mov` can be rewritten as `movb, movw, movl, or movq.`)

```
mov___  %eax, (%rsp)
mov___  (%rax), %dx
mov___  $0xFF, %bl
mov___  (%rsp,%rdx,4), %dl
mov___  (%rdx), %rax
mov___  %dx, (%rax)
```

   a. movl
   b. movw
   c. movb
   d. movb
   e. movq
   f. movw
2. Answers



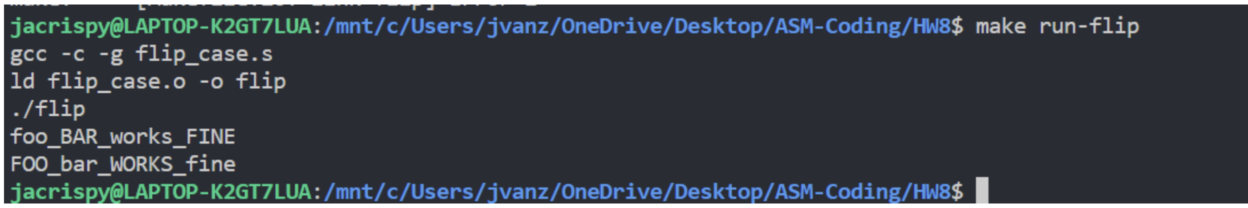
Practice Problem 3.3 (solution page 326)

Each of the following lines of code generates an error message when we invoke the assembler. Explain what is wrong with each line.

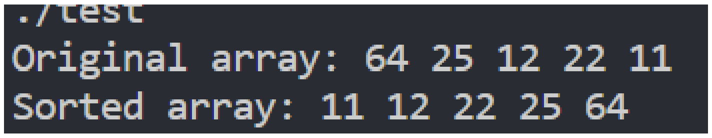
```
movb $0xF, (%ebx)
movl %rax, (%rsp)
movw (%rax),4(%rsp)
movb %al,%sl
movq %rax,$0x123
movl %eax,%rdx
movb %si, 8(%rbp)
```

   a. Trying to move a byte (0xF) to a value, not a register.

b. movl should be movq since your using %rax
c. Trying to move 4 bytes into the stack (rsp)
d. Trying to move a 32bit register into a decimal value.
e. Eax and rdx are mismatched sizes.
f. Trying to move a register into a callee?

3. .
   a. 

```
jacrispy@LAPTOP-K2GT7LUA:/mnt/c/Users/jvanz/OneDrive/Desktop/ASM-Coding/HW8$ make run-flip
gcc -c -g flip_case.s
ld flip_case.o -o flip
./flip
foo_BAR_works_FINE
FOO_bar_WORKS_fine
jacrispy@LAPTOP-K2GT7LUA:/mnt/c/Users/jvanz/OneDrive/Desktop/ASM-Coding/HW8$
```

4.
   a. 

```
./test
Original array: 64 25 12 22 11
Sorted array: 11 12 22 25 64
```

5. When comparing my function (selection_sort_asm) to the object-dumped version of my C function (selection_sort_c), I noticed that the program did some things similarly, and others drastically different. For example there seems to be a lot of random jumps within the C version, while my calls and jumps in my assembly version seem fairly organized. This may be a side effect of the -Og flag when compiling that stops assembly optimizations. I also noticed the C version uses pops and pushes to manage stack memory. This is interesting since we have just started to dig into how to manage those. I'm not sure how I would implement that within my own assembly, but I can start to follow why the C version did it.