hw8

9.16 (erratum: ... b. why must a RTI instruction ...)

> *a.* How many trap service routines can be implemented in the LC-3? Why?

256.
the trap vector is 8 bits and trap's address is from x0000 to x00FF, can point to 256 subroutines.

> *b.* Why must a **RTI** instruction be used to return from a TRAP routine?
>
> Why won't a BR (Unconditional Branch) instruction work instead?

TRAP routine is done in Supervisor mode and has changed PC & PSR. So when go back to execute previous program, we need to pop PC & PSR from system stack, restore them, and if need, save R6 to Save_SSP and change R6 to user's stack.

While BR is only a simple jump instruction, don't change PC either can't finish all these above things.

> *c.* How many accesses to memory are made during the processing of a TRAP instruction? Assume the TRAP is already in the IR.

3
save PSR,PC and extract the subroutine's address

9.17 (erratum: ... d. where will the RTI of ...)

> Refer to Figure 9.14, the HALT service routine.
>
> *a.* What starts the clock after the machine is HALTed? *Hint:* How can the HALT service routine return after bit [15] of the Master Control Register is cleared?

No instructions can work when the clock stop. So we need some external hardware mechanism to start the clock again like the switch.

> *b.* Which instruction actually halts the machine?

```
STI R0, MCR ; Store R0 into MC register
```

> *c.* What is the first instruction executed when the machine is started again?

```
LD R1, SaveR1 ; Restore registers
```

> *d.* Where will the **RTI** of the HALT routine return to?

the PC of the previous program(executed the HALT instruction) which is poped back in RTI.
In other words, the address following the HALT instruction of the previous program.