Lab 3 Report

刘翰文

522030910109

本次实验要求用 C 语言实现 LC3 的模拟器，模拟器分为 Shell 和 Simulation routines 两部分，要求完成 Simulation routines 这部分的实现。这部分主要需要实现的功能包括：Fetch 指令、Decode、Execute 以及各种 opcode 对应的指令的实现等。

在 Instruction_process 函数中，主要包括三个功能：Fetch、Decode 和 Execute，Fetch 指令实现了 PC+1，将二进制指令加载到定义的数组中；Decode 指令实现了获得加载指令的 opcode，Execute 指令实现了将不同的 opcode 对应到不同的指令中并执行该指令。

CC、Regs 的值以及 PC 的变化主要根据不同的指令在对应指令的实现函数中进行，以便在指令的实现过程中完成对应值的改变，最后通过 CURRENT_LATCHES<-NEXT_LATCHES 完成各种值的传递。在各种指令的实现函数中，也通过调用 CURRENT_LATCHES 和 NEXT_LATCHES 获得当前的状态以便指令的准确执行。

在指令的实现具体过程中，我认为相对重要的有 CC 的调整、按位与和 NOT、BR 等。在 CC 的调整中，需要先将十进制数转换为二进制数之后再进行 CC 的调整，其原因是 16 位的负数对应到 32 位则成为正数，所以需要先转为 16 为二进制数后再根据最高位进行判断。在按位与和 NOT 中，是对每一位进行 AND 或 NOT 操作。在 BR 指令中，先判断 NZP 条件和对应 CC 是否符合再对+1 后的 PC 进行与 Offset 求和的操作。对于加载和存储指令，区分 3 种模式按照正确的逻辑进行实现。

在对 simulator 进行验证的过程中，主要通过反复执行 run 1 和 rdump 并和 LC3simulator 进行对比通过不同的那一步判断 bug 的函数进行修改直到二者的结果相同。例如在一个 BR 指令执行之后 PC 值的不同判断 BR 指令的执行发生错误，检查过后发现用了+1 前的 PC 与 offset 进行求和从而得到了错误的结果。

通过本次实验，加强了我对 C 语言以及 LC3 基本指令的实现逻辑的理解，对 debug 过程和 linux 系统实现转化运行有了实际操作加深了熟悉程度，通过对 shell 代码的解读也一定的提高了我对 C 语言以及 simulator 的实现逻辑。