**深 圳 大 学 实 验 报 告**

**课程名称：­ 计算机系统**

**实验项目名称： 简易四子棋**

**学院： 物理与光电工程学院**

**专业： 光电信息科学与工程**

**指导教师： 蔡晔**

**报告人： 学号：**

**实验时间： 2025年 月 5 日 15 星期四**

**实验报告提交时间： 2025年 5月 18 日 星期日**

**教务部制**

|  |
| --- |
| **一 实验目的：**  1、分析和理解试验指定的需解决问题。  2、利用LC-3的机器代码设计实现相关程序。  3、通过LC-3仿真器调试和运行相关程序并得到正确的结果。 |
| **二 实验问题**  用LC3汇编语言实现简易四子棋。  规则：  两位选手依次轮流落子；  选手不能悔棋；  有子的地方不能继续落子；  直到有一方的四个棋子能够连成一条水平线、垂直线或者是对角线；  如果棋盘已满，无人获胜，则平局。  游戏最初时打印空棋盘，用"-" （即ASCII 码 x002D）来表示该处为空，"O"(ASCII 码 x004F)表示第一位选手的棋子，"X" (ASCII 码 x0058)来表示第二位选手的棋子，为了让棋盘更易于观察，在各列间加一个空格，第6列之后不要添加，初始棋盘应该如下：  - - - - - -  - - - - - -  - - - - - -  - - - - - -  - - - - - -  - - - - - -  选手一始终先下第一步棋，然后两者轮流落子，在每次落子之后，应该打印该选手的信息，提示他落子，以选手一为例，应该打印信息如下：  Player 1, choose a column:  为了明确选手的落子的位置，该选手应该输入数字1-6，然后回车，数字1-6指示在落子所在的列，从左到右，无需输入行号，程序应默认从行号6到行号1递减的顺序填入该棋子，若前后输入的列号相同，则行号减一。例如，如果选手第一次在左起第二列落子，应该输入2，然后回车，则该棋子落在行6列2处，当后面输入的列号再次为2时，则将棋子落子行5列2处，以此类推，详情见后续示例输出。程序应该确保选手输入的数字对应正确的列的范围，如果输入不合理，应该输出一条错误信息，提示该选手继续输入，例如，如果对于选手一：  Player 1, choose a column: D  Invalid move. Try again.  Player 1, choose a column: 7  Invalid move. Try again.  Player 1, choose a column:  程序应该一直提示该选手，知道输入正确的数字，当用户输入完成，程序应通过显示回馈给选手，然后通过换行符(ASCII 码 x000A)换行。  当选手输入成功后，程序应打印更新后的棋盘，并检查是否有人获胜，如果没人获胜，则轮到下一位输入。  当其中一位获胜或平局时，游戏结束，程序显示最后的棋盘情况并终止（Halt）。例如，如果选手二有四子相连，应该输出：  Player 2 Wins.  如果平局，程序应该输出：  Tie Game.  程序首行指定第一条指令的地址，即.ORIG x3000。  最后提交的文件为connect4.asm，并提交试验报告。  现场考核要求讲解设计哪些子程序和相应的功能以及你是怎么判定赢方的算法。 |
| **实验步骤**   1. **实验思路分析：**   先对实验要求作一份大致的分析，架构如下：    **图1 思路分析**  然后根据分析，对主要部分写一份代码的大致思路，如下：    **图2 主函数伪代码**   1. **主要几个子程序的实现：**   （子程序需要保存和恢复使用的寄存器，代码如下所示，后面部分不重复说明）    **图3 保存寄存器**    **图4 恢复寄存器**  因为怕程序过长超出范围，需要通过中转解决长距离跳转问题    **图5 中转代码示例** |
| 1. **Print子程序：这个函数负责将游戏矩阵渲染到屏幕上：**     **图6 print函数c++代码思路**  使用两层循环遍历 6x6 的矩阵，根据矩阵元素值打印不同字符，  0：打印 '-' 表示空位  1：打印 'O' 表示玩家 1 棋子  2：打印 'X' 表示玩家 2 棋子   1. **寄存器保存（函数入口）：**   代码在开始已经给出   1. **初始化循环变量**     **图7 初始化循环变量** 外循环（处理每一行，标号pr\_l1）   **图8 外循环**  **外循环逻辑**：R2作为内循环计数器，每次外循环处理一行（6 列），外循环计数器 R1 递减（ADD R1, R1, #-1），直到 R1=0 时结束所有行的打印。 内循环（处理每一列，标号pr\_l2）   **图9 内循环**   1. **根据矩阵值选择打印字符（条件判断）**     **图10 打印字符** 打印具体字符（标号pr\_c1、pr\_c2、pr\_c3）   **图11打印具体字符** 列循环结束处理（标号pr\_l2ed）   **图12列循环结束处理**   1. **行循环结束处理（换行）**     **图13 换行**   1. **最后，恢复寄存器** 2. **子程序数据区：**     **图 14 子程序数据**  **矩阵存储方式**：  矩阵按行优先存储，共 6 行 6 列，每行 6 个元素。  行 i 的首地址为mat0x + i×6，例如：  行 0：mat0x, mat01, mat02, mat03, mat04, mat05  行 1：mat1x, mat11, ..., mat15  行 5：mat50, mat51, ..., mat55    **图15矩阵存储示意图**    **图16 矩阵存储**  注：矩阵全初始化为0，代码太长，不全部展示。  **（2）胜负判断函数（judge1-judge4）：四个子程序分别检查四种获胜情况：**  **① 大致判断思路，四个子程序具有相同的结构：**  寄存器保护：保存所有通用寄存器，避免函数调用破坏上下文。  循环初始化：设置外循环（行）和内循环（列）的计数器，定位矩阵首地址。  连续四子检查：通过多重条件判断，检查指定方向上的四个连续棋子。  胜负判定：根据计数器值判断玩家 1 或玩家 2 是否获胜。  寄存器恢复：恢复寄存器状态并返回。  **② 纵向四连判断（judge1 函数，方向：|）**    **图17 judge1子程序c++代码思路**  检查同一列中连续四行是否为同色棋子  **行循环次数：**外循环计数器 R1=3，因为要检查前 3 行（行 0-2），加上后续三次下移，共检查行 0-3（4 行）。  **列循环次数：**内循环遍历 6 列（列 0-5）。  **地址计算：**同一列的行偏移为 6（每行 6 个元素），因此ADD R4, R4, #6实现下移一行。  列地址 = 当前行基地址（R3） + 列偏移（R2）。  **胜负判定：**  R5=4 → 玩家 1 四连。  R5=0 → 玩家 2 四连（因四个位置都不为 0 且非玩家 1，故必为玩家 2）    **图18 judge1 函数**  注意：R1设为3，只需检查前 3 行（行 0、行 1、行 2），因为纵向四连需要 4 个棋子，若从行 3 开始检查，最多只能覆盖到行 6（越界）  **③ 斜向四连判断（judge2 函数，方向：\）**  检查从左上到右下的斜向（\）是否有四连棋  逻辑和judge1类似，这里只介绍思路和贴代码  **斜向地址偏移**：每下移一行，列偏移 + 1，因此：第 1 个位置：行 i，列 j，第 2 个位置：行 i+1，列 j+1，第 3 个位置：行 i+2，列 j+2，第 4 个位置：行 i+3，列 j+3。    **图19 judge2 函数**   1. **斜向四连判断（judge3 函数，方向：/）**   同理，代码如下  **斜向地址偏移**：每下移一行，列偏移 - 1，因此：第 1 个位置：行 i，列 j，第 2 个位置：行 i+1，列 j-1，第 3 个位置：行 i+2，列 j-2，第 4 个位置：行 i+3，列 j-3。    **图20 judge3 函数**  **⑤ 横向四连判断（judge4 函数，方向：-）**  **横向地址偏移：**同一行内，列偏移 + 1，因此：第 1 个位置：行 i，列 j第 2 个位置：行 i，列 j+1第 3 个位置：行 i，列 j+2第 4 个位置：行 i，列 j+3    **图21 judge4 函数**   1. **主函数（main）：控制游戏流程，如玩家输入和检测等：** 2. **在进入循环前，先初始化：**    1. R1作为循环计数器，初始化为18，表示游戏最多进行 18 轮     **图22 初始化变量**   * 1. **列指针初始化（ptr0-ptr5）**     **图23列指针初始化**  **列指针的作用**：四子棋的棋盘是 6 行 6 列的矩阵（mat0x到mat55），棋子从底部（行号大的位置）开始堆叠。每个列（0-5）有一个指针（ptr0-ptr5），指向该列当前**可落子的最高位置**（初始时指向该列的最底部）  **（2）然后进行玩家回合流程（这里以玩家1为例）**  **① 输入处理（in1 部分）**    **图24输入处理**   * 1. **ASCII 转数字（关键转换）**     **图25 ASCII 转数字**  如输入字符 '1' 的 ASCII 码是 0x31（十进制 49），减去 48 后得到 1（49-48=1）。  **（3）玩家输入检测：**    **图26玩家输入检测**  （4）**棋子放置与指针更新**    **图27 棋子放置**  （5）**棋盘显示与胜负检查**    图28 **胜负检查**  每次落子后，调用print函数刷新棋盘，并调用四个judge函数检查是否形成四连棋。若某函数检测到胜利条件，会跳转到winc1（玩家 1 胜）或winc2（玩家 2 胜）  （6）**玩家 2 回合（in2 部分）**  玩家 2 的流程与玩家 1 完全一致，区别仅在于：  提示字符串为p2inp（“player2 choose a cloumn:”）。  棋子值为 2（ADD R2, R2, #2）。  （7）**循环控制与结束条件**    **图29 循环结束**    **图30 跳转哪个玩家胜利**  （8）数据段定义    **图31数据段定义**  **3．运行示例**    **图32 检测输入是否合法**    **图33 player胜利**    **图34 平局** |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。