计算机组成与结构专题实验

实验报告

第七次 K8051 单片机实验

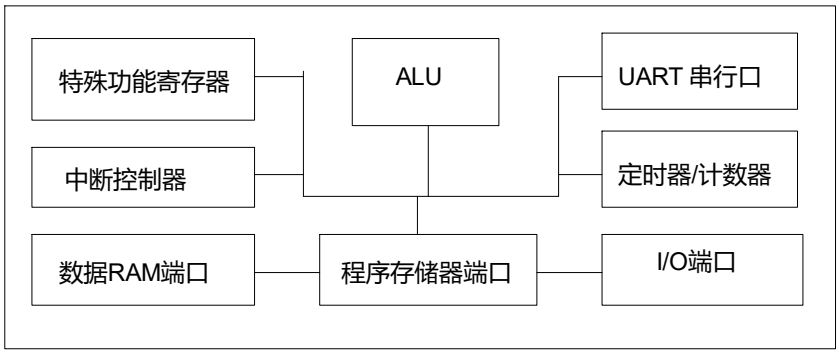
一、 实验目的

1．熟悉 51 单片机的原理和使用操作。

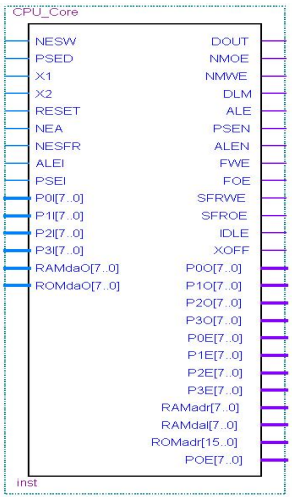
2．利用 51 单片机，自主设计实验并完成相关功能。

二、 实验原理

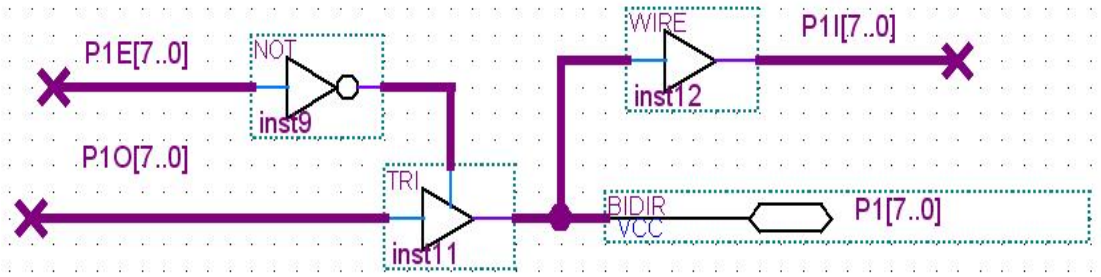
K8051 单片机软核基本功能和结构如图。



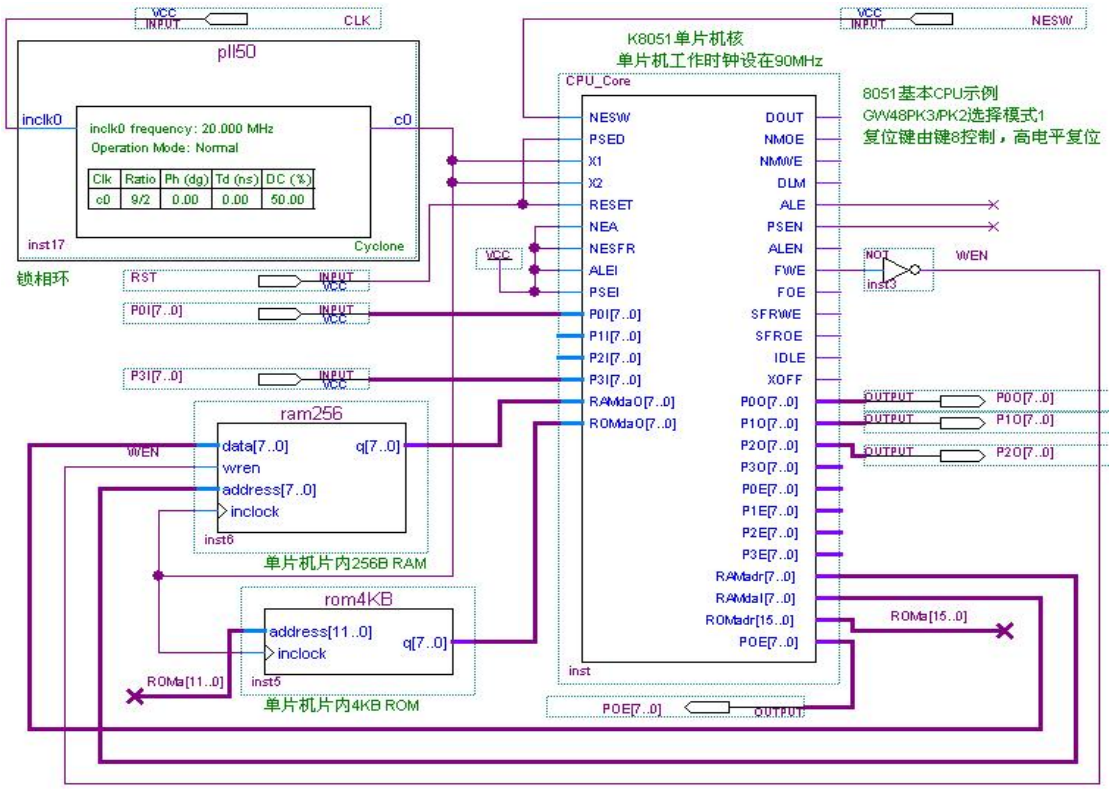
K8051 结构模块框图



K8051 原理图元件



K8051 单片机 I/O 口设置成双向口的电路



K8051 基本实用电路

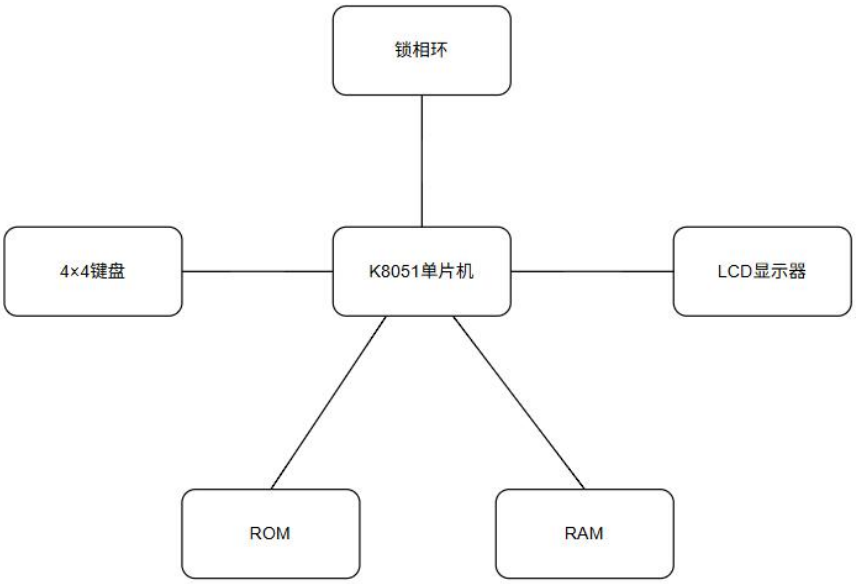
三、 实验任务设计（具体应用描述）

我们确定采用 K8051 单片机来开发一款二进制游戏。这款游戏的运行将借助 LCD 显示屏、4*4 键盘输入设备以及扬声器来实现。游戏启动之初，屏幕上会显示等级选项，分为"难""中""易"三个难度级别。玩家选择好难度后，即可开始进行二进制游戏挑战。 在游戏过程中，每答对一道题目，玩家的分数就会增加 1 分。游戏根据不同的难度设置了不同数量的题目，并且题目出现的间隔时间也有所不同。每次玩家做出回答

后，扬声器都会发出相应的提示音，以便玩家了解答题结果。当玩家完成所有题目后，游戏界面会显示出玩家在本次游戏中获得的得分以及累计的总分，同时扬声器也会播放提示音，告知玩家游戏结束。

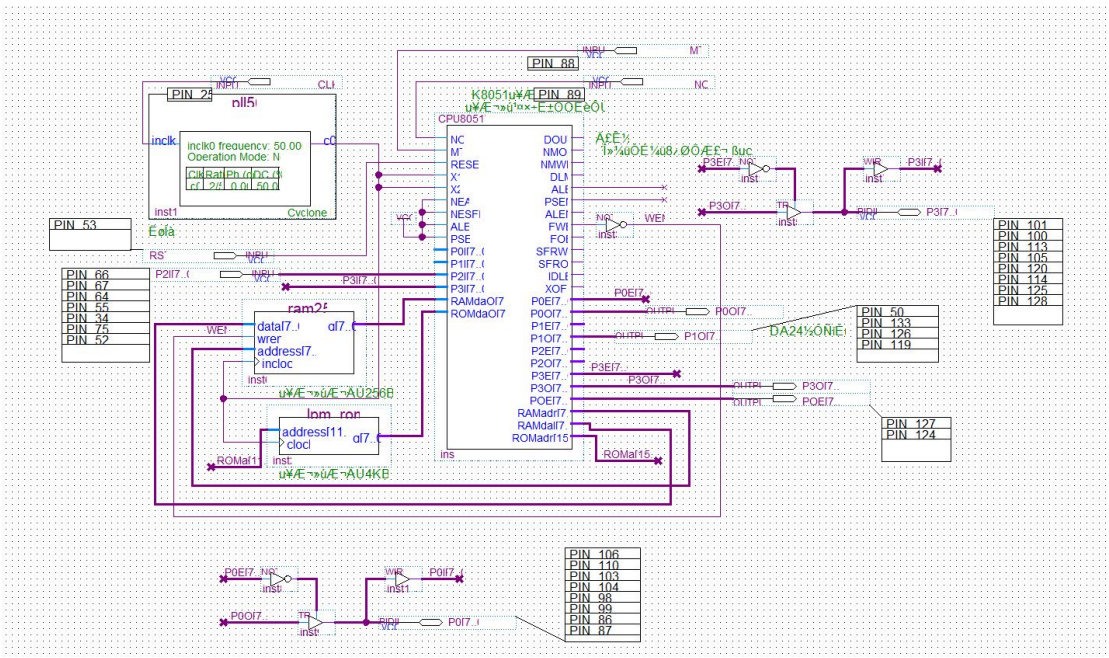
四、 实验步骤及结果

硬件设计框图及相关描述



在 ROM 中存储以 hex 格式呈现的游戏程序文件，该文件对整个应用程序以及各元件的执行逻辑起着全面的控制作用。4×4 的键盘具备 0 至 15 的输入功能，每个按键对应一个输入值。LCD 显示屏则承担着显示游戏画面的任务，同时还会展示玩家当前的所得分数以及所选择的难度等级等信息。

Quartus 下硬件设计原理图、模式及引脚说明



<<new>> Filter on node names: *

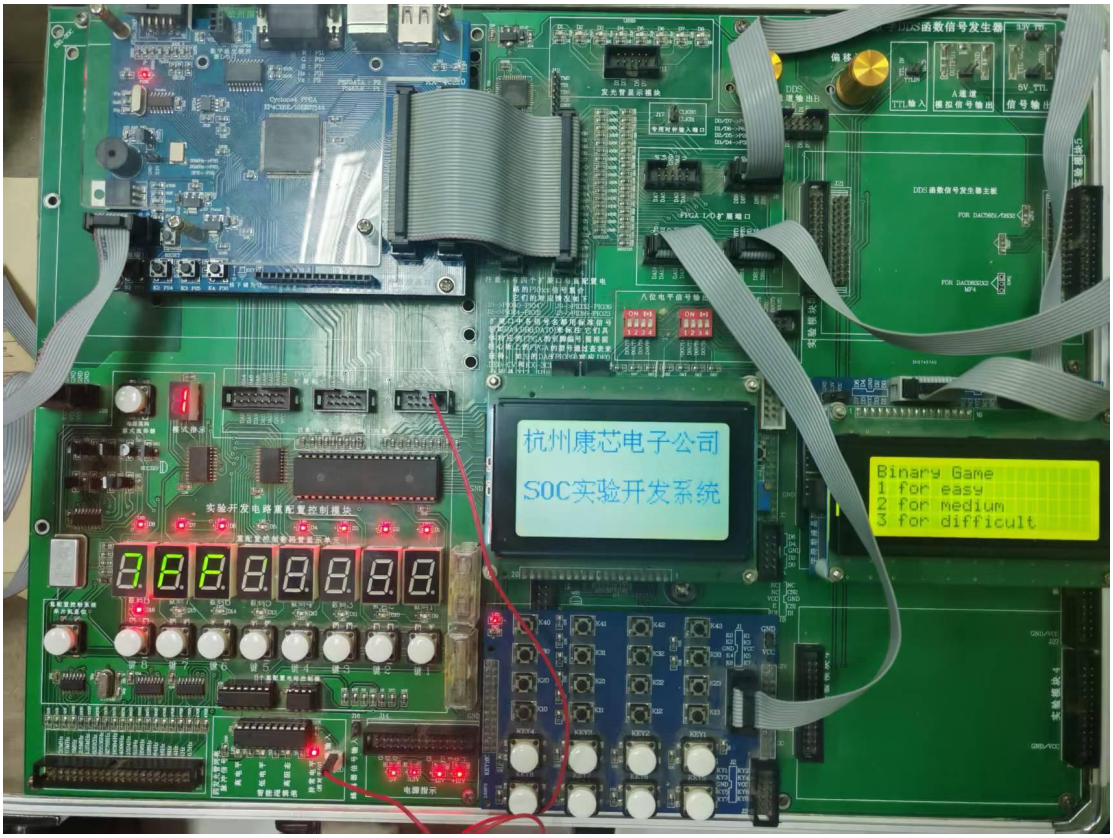
	Statu	From	To	Assignment Name	Value	Enabled	Entity	Com
1	✔	...		P1O[0]	Location	PIN_50	Yes	
2	✔	...		CLK	Location	PIN_25	Yes	
3	✔	...		RST	Location	PIN_53	Yes	
4	✔	...		POE[0]	Location	PIN_127	Yes	
5	✔	...		POE[2]	Location	PIN_124	Yes	
6	✔	...		MT	Location	PIN_88	Yes	
7	✔	...		NO	Location	PIN_89	Yes	
8	✔	...		P2I[3]	Location	PIN_66	Yes	
9	✔	...		P2I[2]	Location	PIN_67	Yes	
10	✔	...		P2I[4]	Location	PIN_64	Yes	
11	✔	...		P3[7]	Location	PIN_101	Yes	
12	✔	...		P3[6]	Location	PIN_100	Yes	
13	✔	...		P3[5]	Location	PIN_113	Yes	
14	✔	...		P3[4]	Location	PIN_105	Yes	
15	✔	...		P3[3]	Location	PIN_120	Yes	
16	✔	...		P3[2]	Location	PIN_114	Yes	
17	✔	...		P3[1]	Location	PIN_125	Yes	
18	✔	...		P3[0]	Location	PIN_128	Yes	
19	✔	...		P2I[5]	Location	PIN_55	Yes	
20	✔	...		P2I[0]	Location	PIN_34	Yes	
21	✔	...		P2I[1]	Location	PIN_75	Yes	
22	✔	...		P0[0]	Location	PIN_106	Yes	
23	✔	...		P0[1]	Location	PIN_110	Yes	
24	✔	...		P0[2]	Location	PIN_103	Yes	
25	✔	...		P0[3]	Location	PIN_104	Yes	
26	✔	...		P0[4]	Location	PIN_98	Yes	
27	✔	...		P0[5]	Location	PIN_99	Yes	
28	✔	...		P0[6]	Location	PIN_86	Yes	
29	✔	...		P0[7]	Location	PIN_87	Yes	
30	✔	...		P1O[1]	Location	PIN_133	Yes	
31	✔	...		P1O[2]	Location	PIN_126	Yes	
32	✔	...		P1O[3]	Location	PIN_119	Yes	
33	✔	...		P2I[6]	Location	PIN_52	Yes	
34		<<new>>	<<new>>	<<new>>				

Node Name	Location	Function
CLK	PIN_25	时钟输入

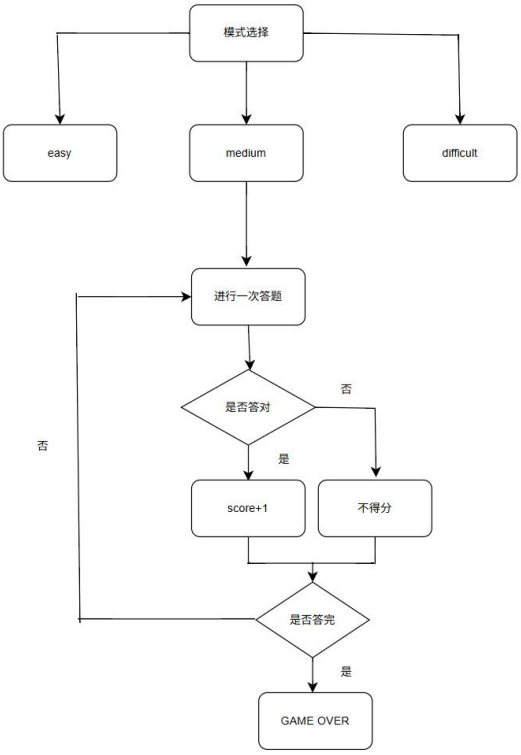
RST	PIN_53	重置
POE[0]	PIN_127	Default
POE[2]	PIN_124	Default
MT	PIN_88	Default
NO	PIN_89	Default
P0[0]	PIN_106	LCD 数据输入
P0[1]	PIN_110	
P0[2]	PIN_103	
P0[3]	PIN_104	
P0[4]	PIN_98	
P0[5]	PIN_99	
P0[6]	PIN_86	
P0[7]	PIN_87	
P10[0]	PIN_50	蜂鸣器控制
P10[1]	PIN_133	LCD 输出
P10[2]	PIN_126	
P10[3]	PIN_119	
P3[7]	PIN_101	4x4 键盘输出
P3[6]	PIN_100	
P3[5]	PIN_113	
P3[4]	PIN_105	
P3[3]	PIN_120	

P3[2]	PIN_114	
P3[1]	PIN_125	
P3[0]	PIN_128	

本次实验我们采取模式 1，便于控制扬声器、LCD 等相关元器件。在引脚绑定中，我们对 4*4 键盘的引脚进行了调整，防止其与 LCD 的引脚发生冲突，具体连线如下：



软件设计流程图及相关描述



汇编或 C 语言源代码

```
#include <reg51.h> // 51 单片机编程需要的库
#include <intrins.h> // _nop_延时函数需要的库
#include <stdio.h>
#include <stdlib.h>
#include <string.h> // 后面字符串函数中取得数组的个数中用到，调用 strlen 函数等字符串处理函数时需要此库
#define uchar unsigned char // 定义 uchar 为 unsigned char 的别名，方便代码编写
#define uint unsigned int // 定义 uint 为 unsigned int 的别名，方便代码编写
// 生日快乐歌的音符频率表，不同频率由不同的延时来决定，这里的数值对应不同音符发声的频率相关参数
uchar code song_tone[] = {
    212, 212, 190, 212, 159, 169, 212, 212, 190, 212, 142, 159, 212, 212, 106, 126, 129, 169, 190, 119, 119, 126, 159, 142, 159, 0
};
// 生日快乐歌节拍表，节拍决定每个音符的演奏长短，数值代表每个音符持续的节拍数
uchar code song_long[] = {
    9, 3, 12, 12, 12, 24, 9, 3, 12, 12, 12, 24, 9, 3, 12, 12, 12, 12, 12, 9, 3, 12, 12, 12, 24, 0
};

// 存储游戏中第一行的二进制数据，用于在 LCD 上显示相关内容
uint code row11[22] = {0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1};
// 存储游戏中第二行的二进制数据，用于在 LCD 上显示相关内容
```

```
uint code row22[22] = {1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0};
// 存储游戏中第三行的二进制数据，用于在 LCD 上显示相关内容
uint code row33[22] = {0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1};
// 存储游戏中第四行的二进制数据，用于在 LCD 上显示相关内容
uint code row44[22] = {1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1};
// 存储每个位置对应的正确答案，用于判断玩家输入的按键编号是否正确
uint code answers[22] = {5, 9, 15, 10, 6, 10, 11, 6, 15, 2, 3, 5, 6, 11, 13, 11, 9, 5, 13, 5, 7, 11};

// 用于存储游戏的相关参数，如题目数量
uint num;
// 用于存储游戏的相关参数，如题目出现的时间间隔
uint interval;
// 循环变量
uchar i;

// 重新定义 LCD 的 RS 引脚，rs 电平为 1 则传送数据，为 0 则转送指令
sbit lcdrs = P1^1;
// 重新定义 LCD 的 RW 引脚，控制 LCD 读或者写，为 1 则读 LCD，为 0 则写入 LCD
sbit lcdrw = P1^2;
// 重新定义 LCD 的 EN 引脚，LCD 行动控制，EN 为下降沿则交互执行，即 EN = 1; 跟着 EN = 0;
sbit lcden = P1^3;

// 短暂延时函数，实现毫秒级的延时，z 为延时的毫秒数
void delay_ms(uint z) {
    uchar x, y;
    // 外层循环控制延时的毫秒数
    for (x = z; x > 0; x--)
        // 内层循环实现具体的延时操作
        for (y = 124; y > 0; y--);
}

// 读忙子程序，用于判断 LCD 液晶是否忙状态，如果不进行判断可能会导致数据写入 LCD 失败
void dbusy() {
    // 把 0xff 发送给 LCD 的数据总线，准备读取 LCD 的状态
    P0 = 0xff;
    // 选指令，设置为读取 LCD 状态字的指令模式
    lcdrs = 0;
    // 选择读操作，从 LCD 读取状态信息
    lcdrw = 1;
    // 使能端置 1(高电平)，启动读取操作
    lcden = 1;
    // 循环判断忙标志位，当 P0 的最高位为 1 时，表示 LCD 忙，程序在此处等待直到 LCD 空闲
    while (P0 & 0x80);
```

```
    // 使能端置 0(低电平)，结束读取操作
    lcden = 0;
}

// 写数据或指令子程序，用于向 LCD 写入数据或指令
// x 为 0 时表示写入指令，为 1 时表示写入数据；DATA 为要写入的数据或指令内容
void tcmddata(int x, unsigned char DATA) {
    // 先判断 LCD 是否忙，确保 LCD 可以接收数据或指令
    dbusy();
    // 短暂延时，确保 LCD 准备好接收数据或指令
    delay_ms(50);
    // 如果不忙，则把要写入的数据或指令赋值给数据总线 P0
    P0 = DATA;
    // 读写端选择为写操作
    lcdrw = 0;
    // 数据或指令端选择为参数 x 的值，确定是写入数据还是指令
    lcdrs = x;
    // 使能端置 1，启动写入操作
    lcden = 1;
    // 使能端置 0，形成一个下降沿，LCD 识别到下降沿信号则读取并执行总线上的数据或指
令
    lcden = 0;
}

// LCD 初始化函数，这里 2004 的初始化和 1602 的初始化指令完全相同
void lcd_init() {
    // 使能端 EN 清零，因为上电时默认高电平，所以先清零准备初始化操作
    lcden = 0;

    // 显示模式设置：设置为 16*4 显示（对 2004）、5*7 点阵、8 位数据接口
    tcmddata(0, 0x38);
    // 显示开关及光标设置：只开显示，不显示光标
    tcmddata(0, 0x0c);
    // 清屏操作，清除 LCD 上之前显示的内容
    tcmddata(0, 0x01);
}

// 定义显示位置函数，根据输入的行 x 和列 y 参数，计算出在 LCD 上的显示地址
void set_xy(uchar x, uchar y) {
    // 根据行号进行地址计算
    switch (x) {
    case 1: y = y + 0x80; break;
    case 2: y = y + 0xc0; break;
    case 3: y = y + 0x94; break;
```

```
        case 4: y = y + 0xd4; break;
        default: y = y + 0x80;
        }
        // 发送设置显示位置的指令到 LCD
        tcmddata(0, y);
}

// 显示函数，在指定的行 x 和列 y 位置上显示字符串 s
void display(uchar x, uchar y, uchar *s) {
        // 设置显示位置
        set_xy(x, y);
        // 对于一个不定长的字符串进行显示，逐个字符发送到 LCD 显示
        while (*s) {
                P0 = *s;
                tcmddata(1, *s);
                s++;
        }
}

// 定义蜂鸣器的控制引脚，蜂鸣器接 P1.0
sbit beep = P1^0;

// 存储当前按下电子琴键对应频率的变量
unsigned int FTemp;
// Timer0_H 与 Timer0_L 共同存储当前播放音符对应的频率，Time 存储当前音符的音长
unsigned char Timer0_H, Timer0_L, Time;

// 4*4 键盘对应的音的预置数(16 位) ，对应吉他 1234 弦空弦+前三品的音高参数
unsigned int code tab[] = {
        64684, 64898, 65030, 65157, // D, G, B, E
        64732, 64934, 65058, 65178, // bE, #G, C, F
        64777, 64968, 65085, 65198, // E, A, #C, #F
        64820, 64994, 65110, 65217 // F, bB, D, G
};

// 读取当前键盘被按下的键，返回值为键的号码(0~15)，没有键按下返回 16
unsigned char Keyscan(void) {
        unsigned char i, j, temp, Buffer[4] = {0xfe, 0xfd, 0xfb, 0xf7};
        // 逐行扫描键盘
        for (j = 0; j < 4; j++) {
                // 向 P3 口输出当前行的扫描码
                P3 = Buffer[j];
                // 短暂延时，确保稳定
                _nop_();
```

```
            temp = 0x80;
            // 扫描当前行的每一列
            for (i = 0; i < 4; i++) {
                // 判断当前列是否有按键按下
                if (!(P3 & temp)) {
                    // 如果有按键按下，计算并返回按键的编号
                    return (i + j * 4);
                }
                // 右移一位，检查下一列
                temp >>= 1;
            }
        }
        // 如果没有键按下，返回 16
        return 16;
}

// 播放函数，播放“生日快乐歌”
void playmusic() {
        uint i = 0, j, k;
        // 判断是否时长或者音符长度为 0，为 0 则歌曲结束
        while (song_long[i] != 0 || song_tone[i] != 0) {
            // 时长及音符长度不为 0 时，逐一播放各个音符
            // song_long 为拍子长度，这里 20 为延时倍数，修改这个值可加快或减缓音乐的播放速度
            for (j = 0; j < song_long[i] * 20; j++) {
                // 如果检测到有按键按下，停止播放音乐
                if (Keyscan() == 0) return;
                // 电平翻转，产生不同的音调的播放效果
                beep = ~beep;
                // 这里的 5 为频率增减调节，修改该值会整体调高或者降低音调
                // 但降低该值时，应适当加大节拍延时，反之应适当将节拍延时调小
                for (k = 0; k < song_tone[i] / 5; k++);
            }
            // 短暂延时，控制音符之间的间隔
            delay_ms(10);
            // 指向下一个音符
            i++;
        }
}

// 延时 2us 的函数，t 为延时的控制参数
void DelayUs2x(unsigned char t) {
        // 通过递减参数 t 实现延时
        while (--t);
```

```
}

// 延时 tms 的函数，t 为延时的毫秒数
void DelayMs(unsigned char t) {
    // 通过多次调用 DelayUs2x 函数实现毫秒级延时
    while (t--) {
        DelayUs2x(245);
        DelayUs2x(245);
    }
}

// 延时 0.25ts 的函数，t 为延时的 0.25 秒倍数
void delay(unsigned char t) {
    unsigned char i;
    // 通过多次调用 DelayMs 函数实现 0.25 秒倍数的延时
    for (i = 0; i < t; i++) {
        DelayMs(250);
    }
}

// 音乐播放函数，设置定时器 0 并播放一个 Time 倍八分音符时长的音
// Timer0_H 和 Timer0_L 为定时器 0 的计数初值高 8 位和低 8 位，Time 为音符持续时间参数
void Song(unsigned char Timer0_H, unsigned char Timer0_L, uchar Time) {
    TH0 = Timer0_H;
    TL0 = Timer0_L;
    TR0 = 1;
    delay(Time);
    TR0 = 0;
}

void main(void) {
    // 默认 Key_Value 为 16(键盘未按下)，Key_Temp1 存放 4*4 键盘按键的位置，Key_Temp2
作校验用
    unsigned char Key_Value = 16, Key_Temp1 = 16, Key_Temp2 = 16;
    // 表示当前节拍数，初始值为-2
    int persent_beat = -2;
    // 玩家的得分，初始值为 0
    int score = 0;
    // 循环变量
    int ii;
    // 当前行号
    int present_row;
    // 用于判断玩家输入是否正确的标志，0 表示错误，1 表示正确
    int correct;
```

```
    // 用于存储格式化输出的字符串
    char printstr[5];
    // 用于存储格式化输出的字符串，如显示得分
    char printstr1[10];
    // 用于存储格式化输出的字符串，如显示总题目数
    char printstr2[10];
    // 游戏是否正在进行的标志，0 表示未开始或已结束，1 表示正在进行
    int is_playing = 0;

    // 工作模式赋值为 00010001，前 4 位表示定时器 1 工作，后四位表示定时器 0 工作
    TMOD |= 0x11;
    // 允许中断
    EA = 1;
    // 定时器 0 允许中断
    ET0 = 1;
    // 高电平触发
    IT0 = 1;
    // 定时器 1 允许中断
    ET1 = 1;

    // 初始化 LCD
    lcd_init();

    while (1) {
        // 重新初始化 LCD，确保显示正常
        lcd_init();

        // 判断当前节拍数是否大于 0 且是间隔的整数倍，即判断是否到了判断玩家输入的时
机
        if (persent_beat > 0 && persent_beat % interval == 0) {
            // 判断玩家输入的按键编号是否与当前位置的正确答案一致
            if (Key_Value == answers[persent_beat / interval - 1]) {
                correct = 1;
            }
            // 如果答案错误，在 LCD 上显示"F"
            if (correct == 0) {
                display(1, 0, "F");
            }
            // 如果答案正确且有按键按下，在 LCD 上显示"T"并增加得分
            else if (correct == 1 && Key_Value != 16) {
                display(1, 0, "T");
                score += 1;
            }
        }
```

```
// 如果游戏未开始
if (!is_playing) {
    // 在 LCD 上显示游戏标题和难度选择信息
    display(1, 0, "Binary Game");
    display(2, 0, "1 for easy");
    display(3, 0, "2 for medium");
    display(4, 0, "3 for difficult");
    // 等待玩家按下按键，直到有按键按下
    while (Key_Temp1 == 16) {
        // 读取按键的编号
        Key_Temp1 = Keyscan();
        // 再读一遍进行校验
        Key_Temp2 = Keyscan();
        // 如果校验无误且按下的是 1 键，设置游戏为简单难度并开始游戏
        if (Key_Temp1 == 1) {
            is_playing = 1;
            num = 6;
            interval = 3;
        }
        // 如果校验无误且按下的是 2 键，设置游戏为中等难度并开始游戏
        else if (Key_Temp1 == 2) {
            is_playing = 1;
            num = 14;
            interval = 2;
        }
        // 如果校验无误且按下的是 3 键，设置游戏为困难难度并开始游戏
        else if (Key_Temp1 == 3) {
            is_playing = 1;
            num = 22;
            interval = 1;
        }
        // 如果按键无效，重置 Key_Temp1
        else {
            Key_Temp1 = 16;
        }
    }
}
// 如果游戏正在进行
else {
    // 如果当前节拍数为-2，显示"Ready"并等待一段时间，然后增加节拍数
    if (persent_beat == -2) {
        display(2, 8, "Ready");
        persent_beat++;
```

```
        delay(10);
    }
    // 如果当前节拍数为-1，显示"Go"并等待一段时间，然后增加节拍数
    else if (persent_beat == -1) {
        display(2, 8, "Go");
        persent_beat++;
        delay(10);
    }
    // 游戏正式开始阶段
    else {
        unsigned char persent_delay = 245;
        unsigned char delay_times = 15;
        Key_Temp1 = 16;

        // 循环显示游戏的二进制数据
        for (ii = 1; ii < num + 1; ii++) {
            present_row = ii * interval - persent_beat;
            // 判断当前行号是否在有效显示范围内
            if (present_row >= 1 && present_row <= 19) {
                // 将第一行的二进制数据转换为字符串并显示在 LCD 上
                sprintf(printstr, "%d", row11[ii - 1]);
                display(1, present_row, printstr);
                // 将第二行的二进制数据转换为字符串并显示在 LCD 上
                sprintf(printstr, "%d", row22[ii - 1]);
                display(2, present_row, printstr);
                // 将第三行的二进制数据转换为字符串并显示在 LCD 上
                sprintf(printstr, "%d", row33[ii - 1]);
                display(3, present_row, printstr);
                // 将第四行的二进制数据转换为字符串并显示在 LCD 上
                sprintf(printstr, "%d", row44[ii - 1]);
                display(4, present_row, printstr);
            }
        }

        // 增加当前节拍数
        persent_beat++;

        // 等待玩家按键或者时间耗尽
        while (Key_Temp1 == 16 && persent_delay) {
            Key_Value = 16;
            correct = 0;
            // 减少剩余等待时间
            --persent_delay;
```

```
// 如果还有额外的延时次数且当前等待时间耗尽
if (delay_times > 0 && persent_delay == 1) {
    // 重置等待时间
    persent_delay = 245;
    // 减少额外的延时次数
    delay_times--;
}

// 停止定时器 0
TR0 = 0;
// 停止定时器 1
TR1 = 0;
// 读取按键的编号
Key_Temp1 = Keyscan();
// 如果有键按下
if (Key_Temp1 != 16) {
    // 再读一遍进行校验
    Key_Temp2 = Keyscan();
    // 如果校验无误
    if (Key_Temp1 == Key_Temp2) {
        // 按键编号赋给 Key_Value
        Key_Value = Key_Temp1;
        // 获取对应按键的频率预置数
        FTemp = tab[Key_Value];
        // 启动定时器 1
        TR1 = 1;
        // 等待按键松开
        while (Keyscan() < 16);
        // 停止定时器 1
        TR1 = 0;
        // 关闭蜂鸣器
        beep = 1;
    }
}

// 判断游戏是否结束
if (persent_beat > num * interval + 1) {
    // 格式化得分信息
    sprintf(printstr1, "score %d", score);
    // 格式化总题目数信息
    sprintf(printstr2, "total %d", num);
    // 在 LCD 上显示游戏结束信息
    display(1, 2, "Finish!");
    // 在 LCD 上显示得分信息
```

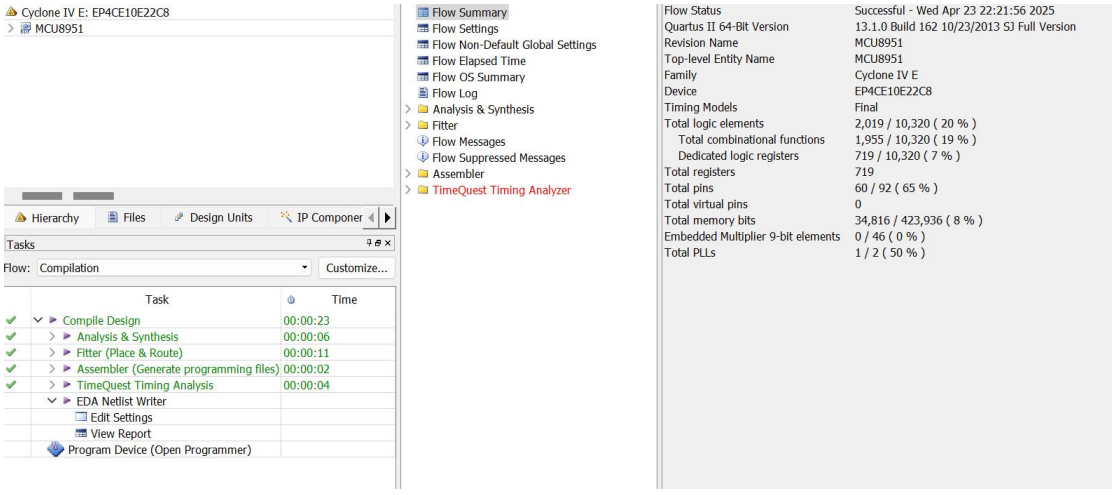
```
                                display(2, 2, printstr1);
                                // 在 LCD 上显示总题目数信息
                                display(3, 2, printstr2);
                                // 在 LCD 上显示重新开始提示信息
                                display(4, 2, "0 to restart");
                                // 播放生日快乐歌
                                playmusic();
                                // 重置得分
                                score = 0;
                                // 重置当前节拍数
                                persent_beat = -2;
                                // 标记游戏结束
                                is_playing = 0;
                                // 延时一段时间
                                delay(15);
                                break;
                            }
                        }
                    }
                }
            }
        }

// 定时器 T0 中断函数，用于产生特定频率的蜂鸣器声音
void TIM0_ISR() interrupt 1 {
    // 停止定时器 T0
    TR0 = 0;
    // 反转蜂鸣器电平，产生声音
    beep = !beep;
    // 重新加载定时器 0 的高 8 位计数初值
    TH0 = Timer0_H;
    // 重新加载定时器 0 的低 8 位计数初值
    TL0 = Timer0_L;
    // 启动定时器 T0
    TR0 = 1;
}

// 定时器 T1 中断函数，用于处理键盘按键对应的声音
void T1_INT(void) interrupt 3 {
    // 停止定时器 T1
    TR1 = 0;
    // 加载定时器 1 的低 8 位计数初值
    TL1 = FTemp;
    // 加载定时器 1 的高 8 位计数初值
```

```
    TH1 = FTemp >> 8;
    // 反转蜂鸣器电平，产生声音
    beep = !beep;
    // 启动定时器 T1
    TR1 = 1;
}
```

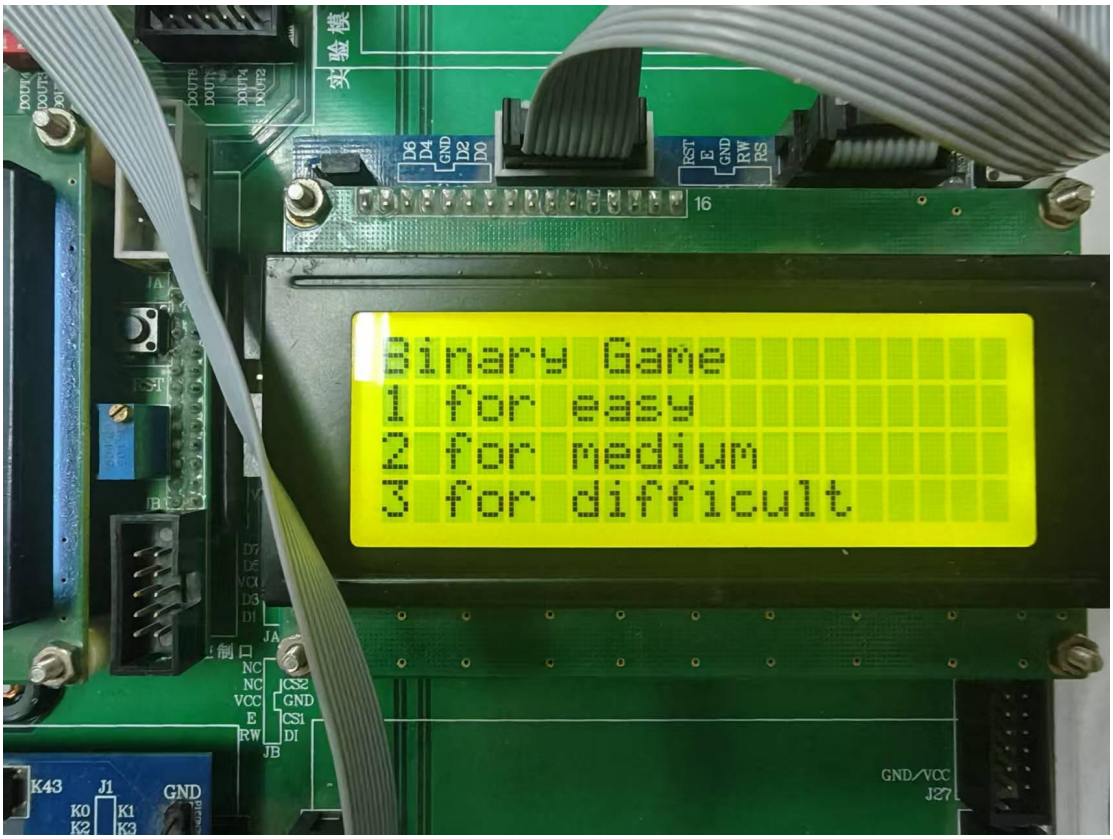
该源代码的核心功能主要在 main 函数中按照一定的顺序逐步实现。其中，display 函数专门负责 LCD 显示屏的显示操作，包括在指定的位置显示游戏相关的文字信息、题目内容以及得分等数据；playmusic 函数用于实现音乐的播放，能够根据预设的音符频率和节拍信息，通过控制蜂鸣器的电平翻转来演奏“生日快乐歌”；Song 函数则实现了提示音功能，在游戏过程中适时发出提示音，为玩家提供反馈。这些函数均可供 main 函数进行调用，以完成整个游戏的功能逻辑。 当代码编写完毕后，借助 Keil 软件对其进行编译处理，最终生成可被单片机识别的 hex 文件，随后将该 hex 文件导入到单片机的 RAM 中，以便单片机能够执行相应的程序指令，运行该二进制游戏。



调试总结

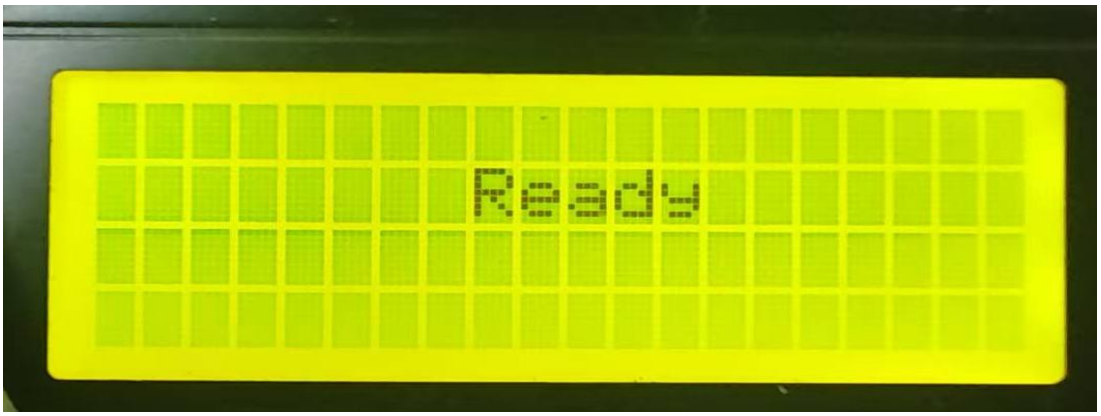
经调试，可以在单片机上实现所需的二进制游戏功能。

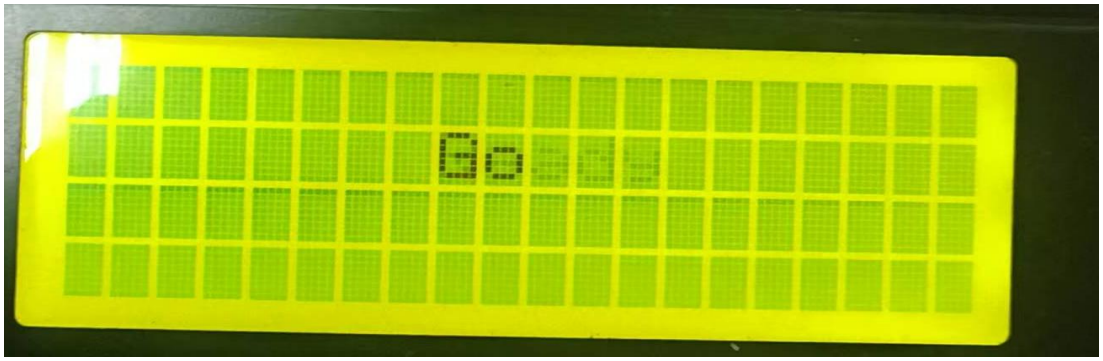
具体流程如下：



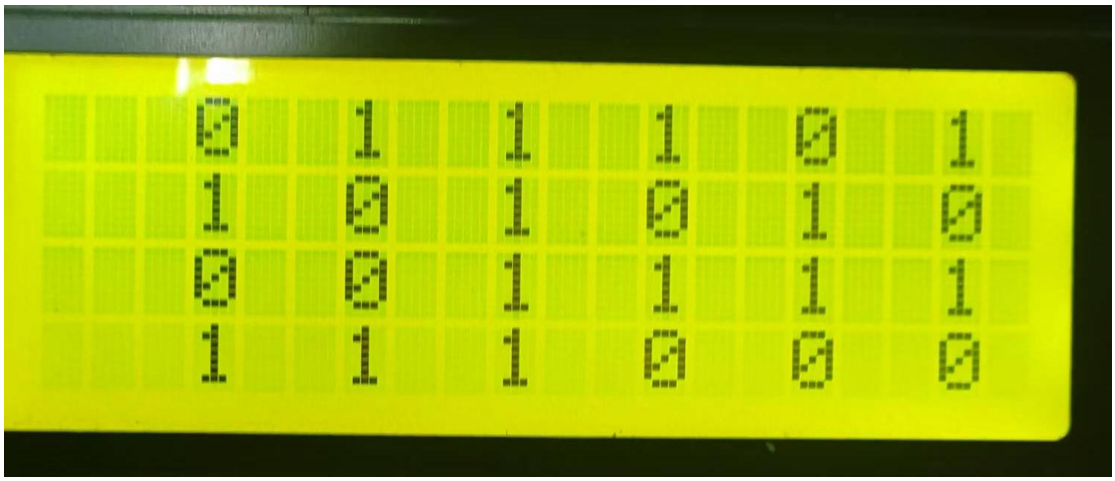
游戏开始显示游戏名字“Binary Game”并给出模式选项。

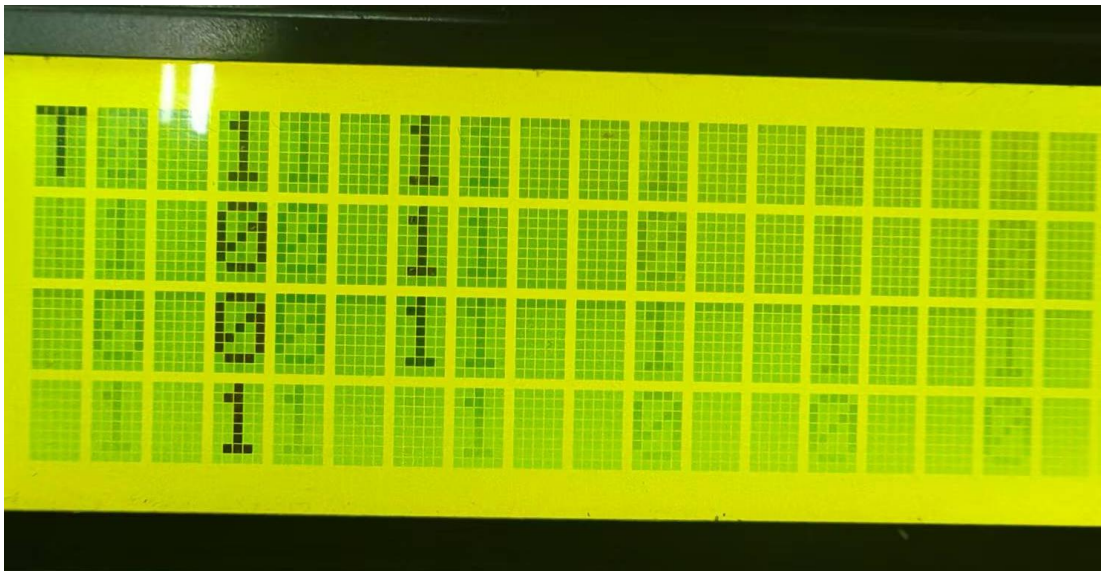
选择模式后开始游戏。

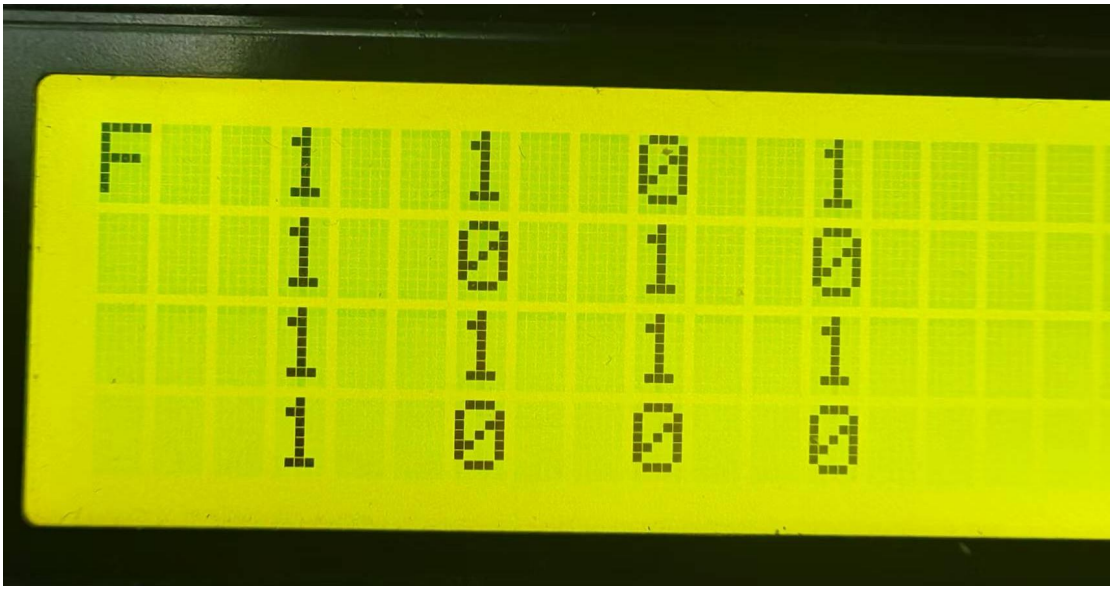




Easy 模式：





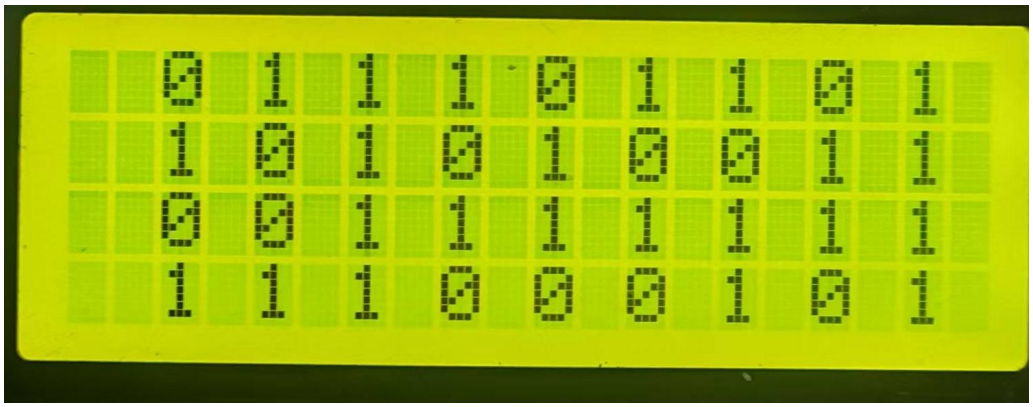


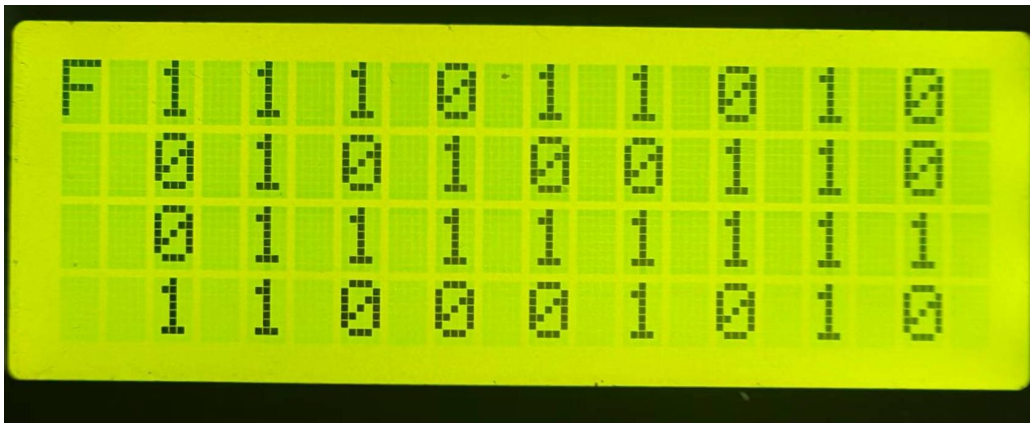
回答正确屏幕上显示“T”并有成功的提示音。

回答错误屏幕上显示“F”并有错误的提示音。

游戏结束，显示本次作答成绩和总题数

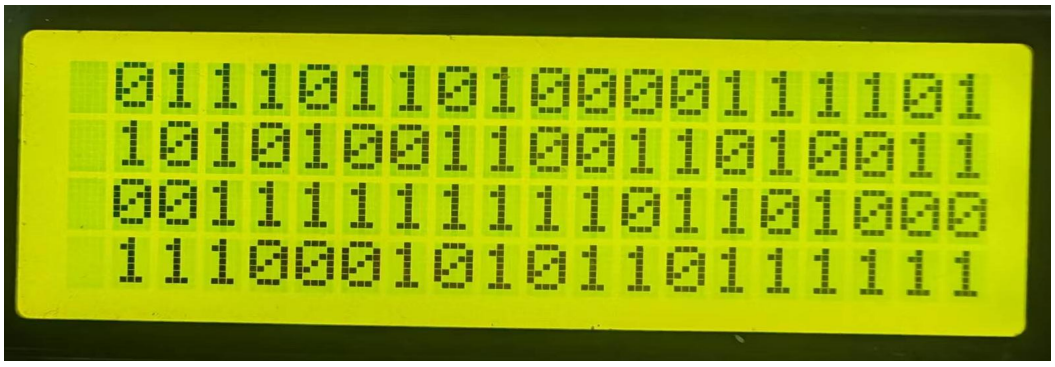
Medium(题数更多，时间更短):

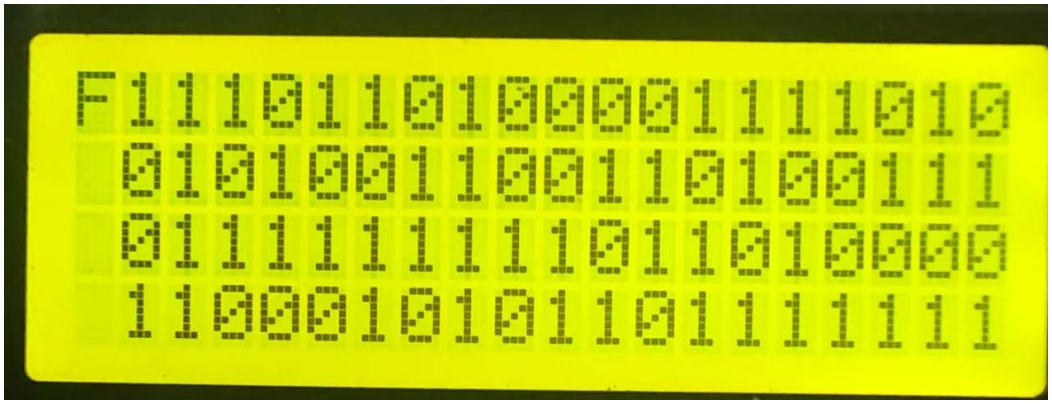


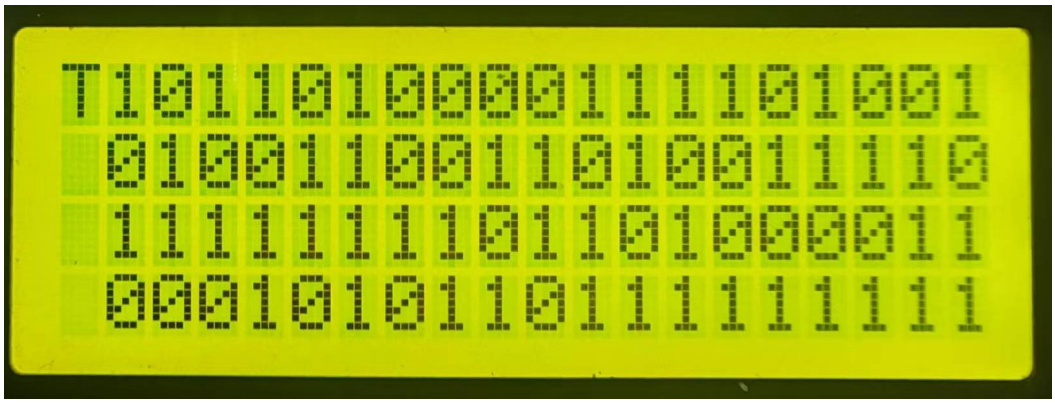

F 1 1 1 0 1 1 0 1 0
0 1 0 1 0 0 1 1 0
0 1 1 1 1 1 1 1 1
1 1 0 0 0 1 0 1 0


Finish!
score 4
total 14
0 to restart

Difficult:


0 1 1 1 0 1 1 0 1 0 0 0 0 1 1 1 1 0 1
1 0 1 0 1 0 0 1 1 0 0 1 1 0 1 0 0 1 1
0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0 0 0
1 1 1 0 0 0 1 0 1 0 1 1 0 1 1 1 1 1 1


F 1 1 1 0 1 1 0 1 0 0 0 0 1 1 1 1 0 1 0
0 1 0 1 0 0 1 1 0 0 1 1 0 1 0 0 1 1 1
0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0 0 0
1 1 0 0 0 1 0 1 0 1 1 0 1 1 1 1 1 1 1



三个模式均成功实现。

五、 实验总结及问题分析

在本次实验里，我们成功实现了预期功能。不过，过程并非一帆风顺，像是引脚绑定错误、C 语言程序编写完成后却无法正常运作等状况频出。幸得一同参与实验的同学们热心相助，在此向他们致以诚挚感谢。通过这次实验，我在软件操作技能方面收获了显著成长，对计算机组成原理以及 51 单片机的运用，也有了更为深刻的领悟。在此，特别感恩三位老师，在计算机组成实验全程给予我们专业的讲解与悉心的指导。这场实验，为我的计算机组成实验之旅画上了圆满的句号。