# 24年计组考试题回忆版

Tips: 如果是左冬红老师的学生，好好做作业，好好看课件，会考原题😭

## 一. 填空题 20分

1.二进制转16进制

2.小数-8.625转化为32位浮点数

3.指令分类

4.io传输数据的方式，不通过cpu的是哪种方式

5.有符号数运算法：0x07-0x8D，写结果和溢出位

5.具体的一个beq， 闻是什么指令（I,R,J），计算beq imm的结果

6.uart协议，空闲？ 电平，启动传输要发一个？ 电平

## 二.汇编23分

1.给了一个main，里面有递归程序，c语言代码给定。填汇编代码的空（只填指令名称）

涉及宏指令(la, li)，伪指令（.asciiz），子程序调用的堆栈操作

2.给了一个汇编程序，填地址里存储的数据，汇编翻译成c语言代码，结构大概如下:

```
int a[16],t0=4;
for(t1 = 0; t1 < t0; t1++){
    for(t2 = 0; t2 < t0; t2++){
        t4 = 4 * t1 + t2;
        t3 = (t0 - 1 - t2) * t0 + t1;
        a[t3] = t4
    }
}
```

最后要求写出数组里的值

# 三.mips微处理器设计12分

给了一个mips微处理器的描述，又给了一段汇编程序的仿真电路波形，根据附录里的Verilog HDL代码和仿真波形写出原始的mips汇编指令

ps：沟槽的汇编，大概就是人脑反汇编机器码为汇编代码

附件：mips32的ALU和ControlUnit的Verilog

```
module ALU(output reg[31:0] ALUResult, output Zero,
           input[31:0] SrcA, SrcB, input[2:0] ALUControl);
assign Zero = (ALUResult == 0);

always @(*) begin
    case (ALUControl) // 4-bit in the textbook
        3'b001: ALUResult = SrcA & SrcB;
        3'b010: ALUResult = SrcA + SrcB;
        3'b011: ALUResult = SrcA | SrcB;
        3'b110: ALUResult = SrcA - SrcB;
        3'b111: ALUResult = SrcA < SrcB;
        default: ALUResult = 32'hFFFFFFFF;
    endcase
end
endmodule

module ControlUnit(output RegWrite, MemWrite, RegDst, ALUSrc, MemtoReg, Branch, Jump, output[2:0] ALUControl,input[5:0] Opcode, Funct);

reg[9:0] Ctr;
assign {RegWrite, MemWrite, RegDst, ALUSrc, MemtoReg, Branch, Jump, ALUControl} = Ctr;

always @(*) begin
    case (Opcode)
        6'h23: Ctr <= 10'h262; // lw 指令操作码
        6'h08: Ctr <= 10'h242; // addi 指令操作码
        6'h2B: Ctr <= 10'h142; // sw 指令操作码
        6'h04: Ctr <= 10'h016; // beq 指令操作码
        6'h02: Ctr <= 10'h008; // j 指令操作码
        6'h00: // R-type
            case (Funct)
                6'h00: Ctr <= 10'h000; // nop 指令功能码
```

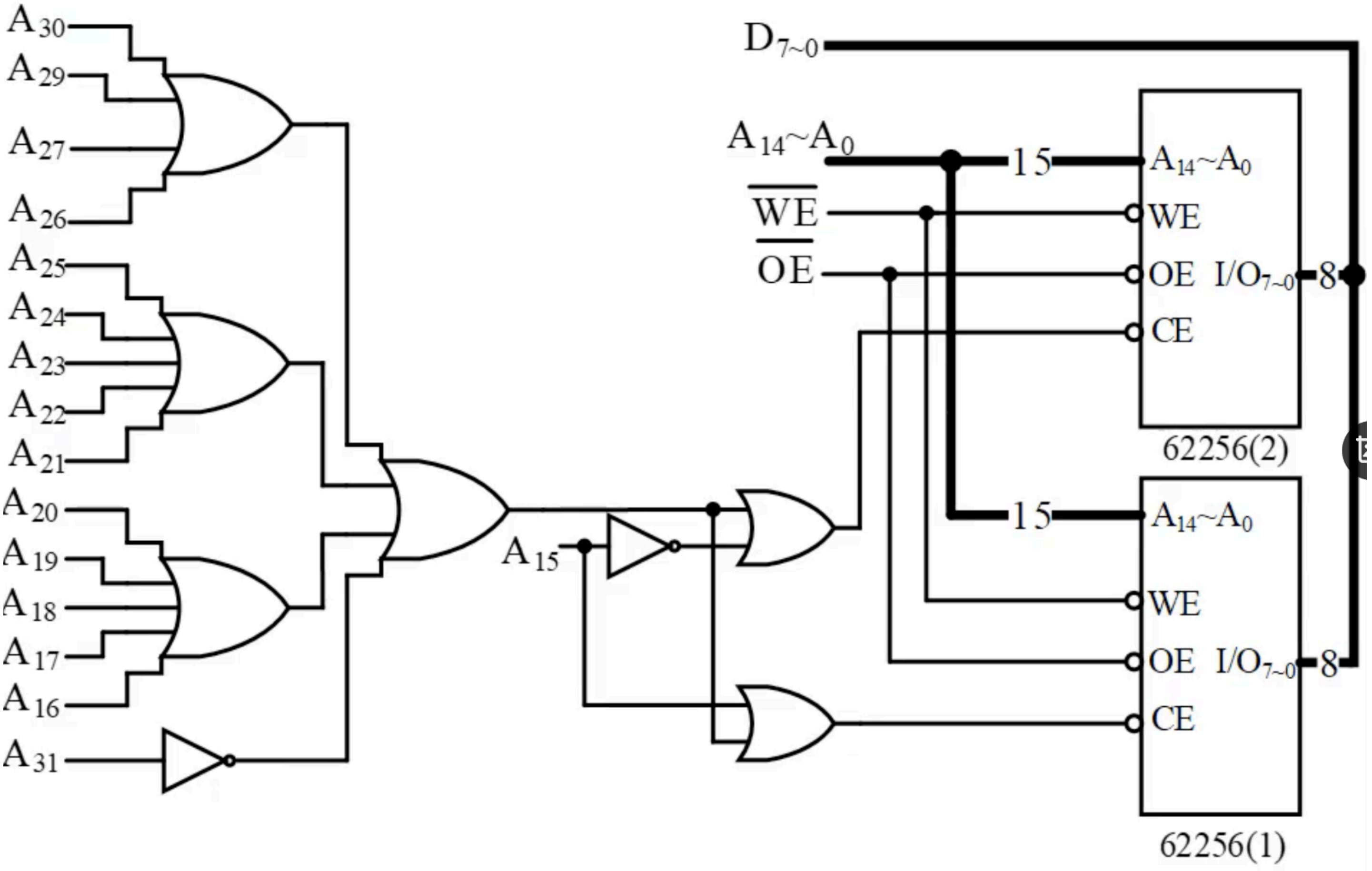
```
                6'h20: Ctr <= 10'h282; // add 指令功能码
                6'h22: Ctr <= 10'h286; // sub 指令功能码
                6'h24: Ctr <= 10'h281; // and 指令功能码
                6'h25: Ctr <= 10'h283; // or 指令功能码
                6'h2A: Ctr <= 10'h287; // slt 指令功能码
                default: Ctr <= 10'h000;
            endcase
        default: Ctr <= 10'h000;
    endcase
end
endmodule
```

# 四.高速缓存20分

1.直接映射，4GB内存，64B高速缓存，每行4B。物理内存地址位数？，行内偏移位数，对应的地址是哪些位，行偏移位数，对应的地址是哪些位，tag有多少位，对应的物理地址是哪些位

2.0x11,0x13,0x20,0x21依此访问，写出对应的缓存行以及是否命中

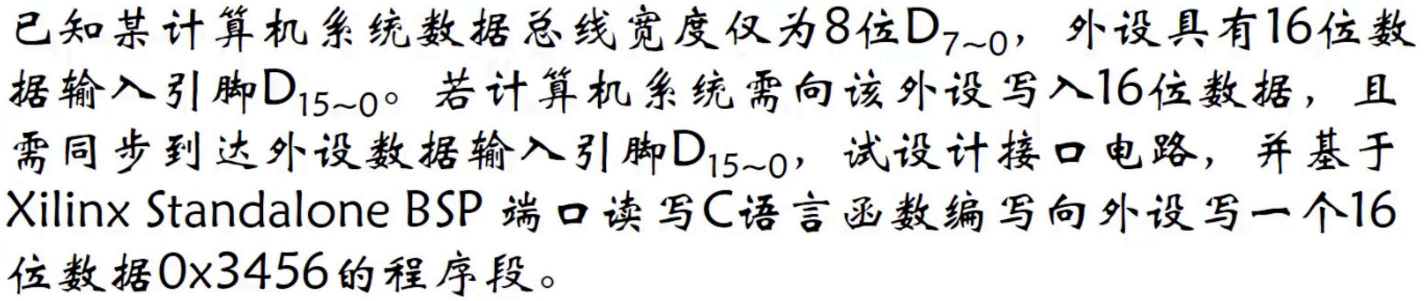
# 五.存储器5分



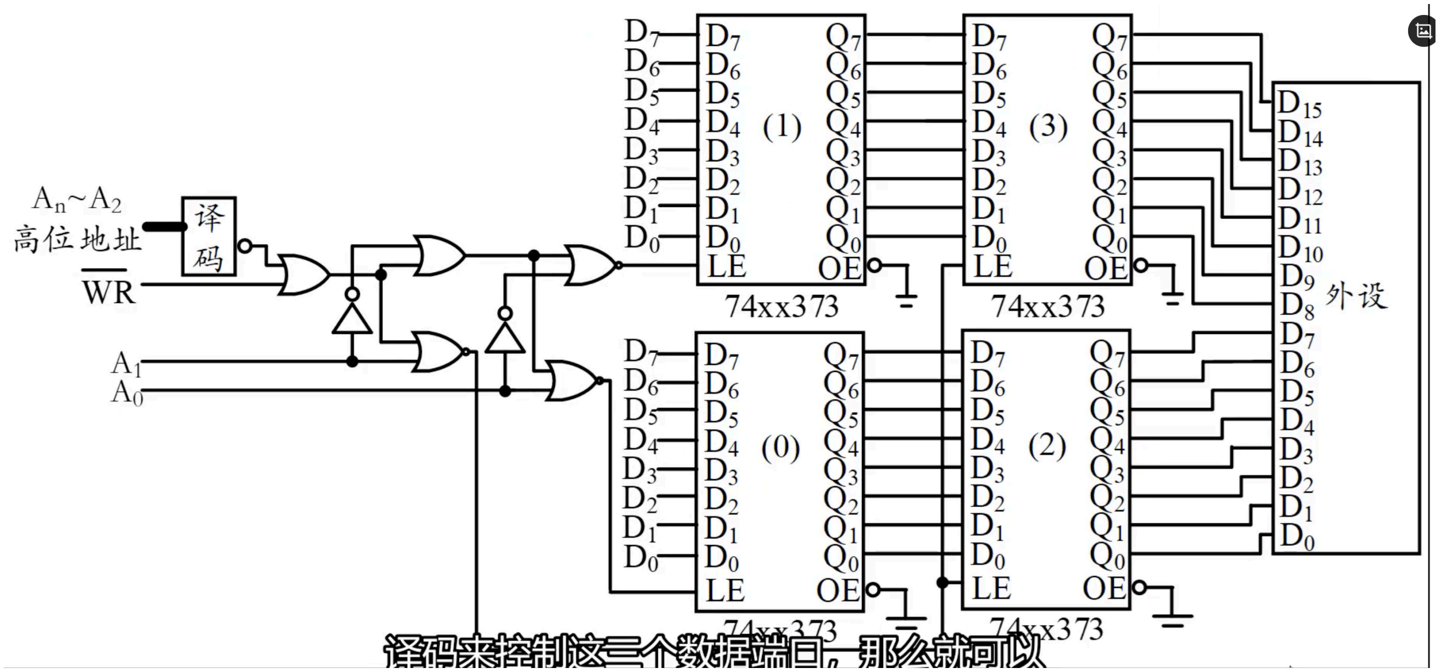
1.一个存储器的容量是多少

2.存储器映射到的物理地址是那些（hint：不止以一个物理地址区间，需要全部写出）

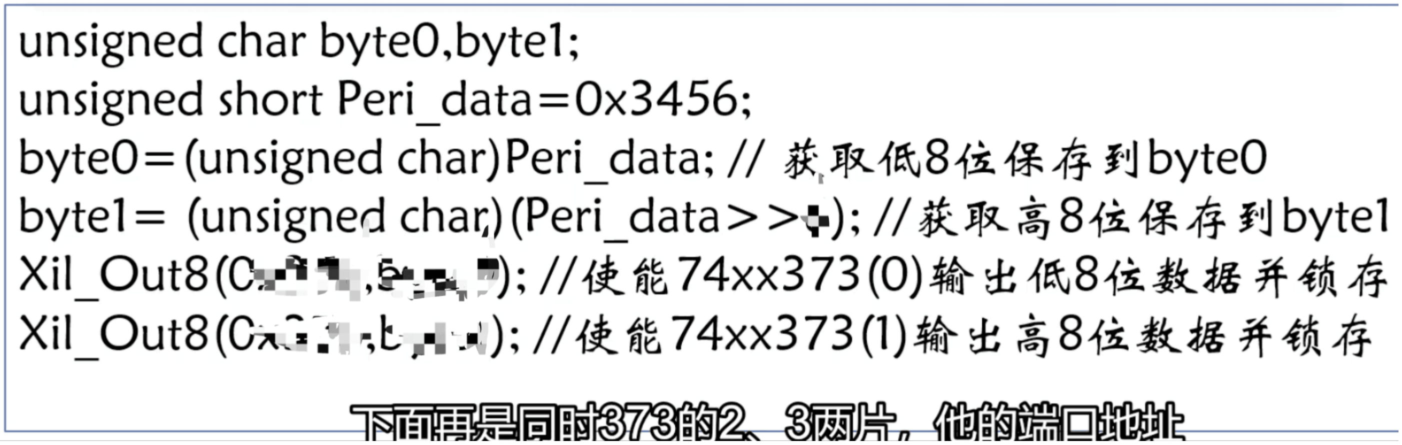
## 六.IO接口设计



已知某计算机系统数据总线宽度仅为8位$D_{7\sim0}$，外设具有16位数据输入引脚$D_{15\sim0}$。若计算机系统需向该外设写入16位数据，且需同步到达外设数据输入引脚$D_{15\sim0}$，试设计接口电路，并基于Xilinx Standalone BSP 端口读写C语言函数编写向外设写一个16位数据0x3456的程序段。



译码来控制这三个数据端口，那么就可以



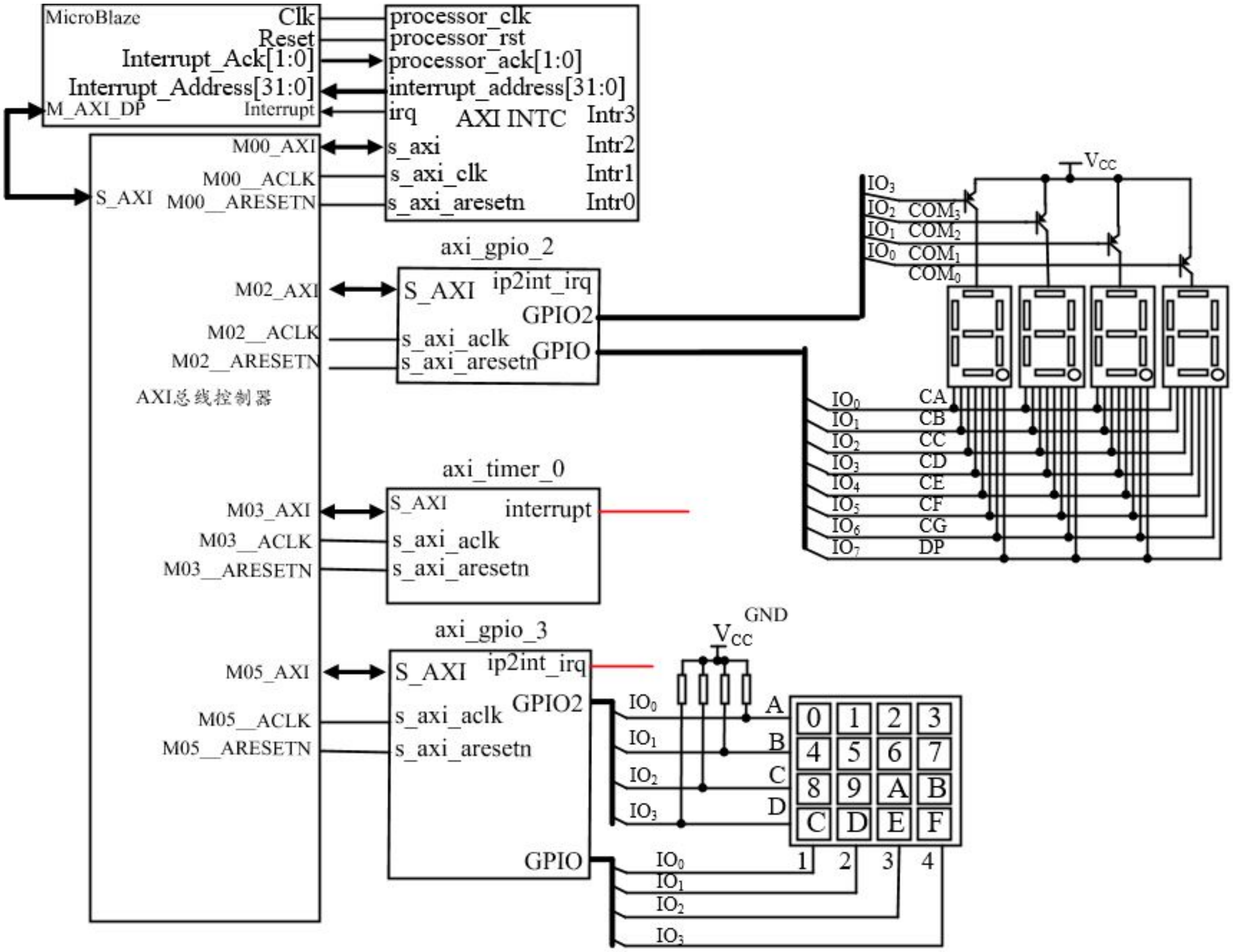
```
unsigned char byte0,byte1;
unsigned short Peri_data=0x3456;
byte0=(unsigned char)Peri_data; // 获取低8位保存到byte0
byte1= (unsigned char)(Peri_data>>[redacted]); //获取高8位保存到byte1
Xil_Out8(0[redacted],[redacted]); //使能74xx373(0)输出低8位数据并锁存
Xil_Out8(0x[redacted],b[redacted]); //使能74xx373(1)输出高8位数据并锁存
```

下面再是同时373的2、3两片，他的端口地址

填马赛克处的值

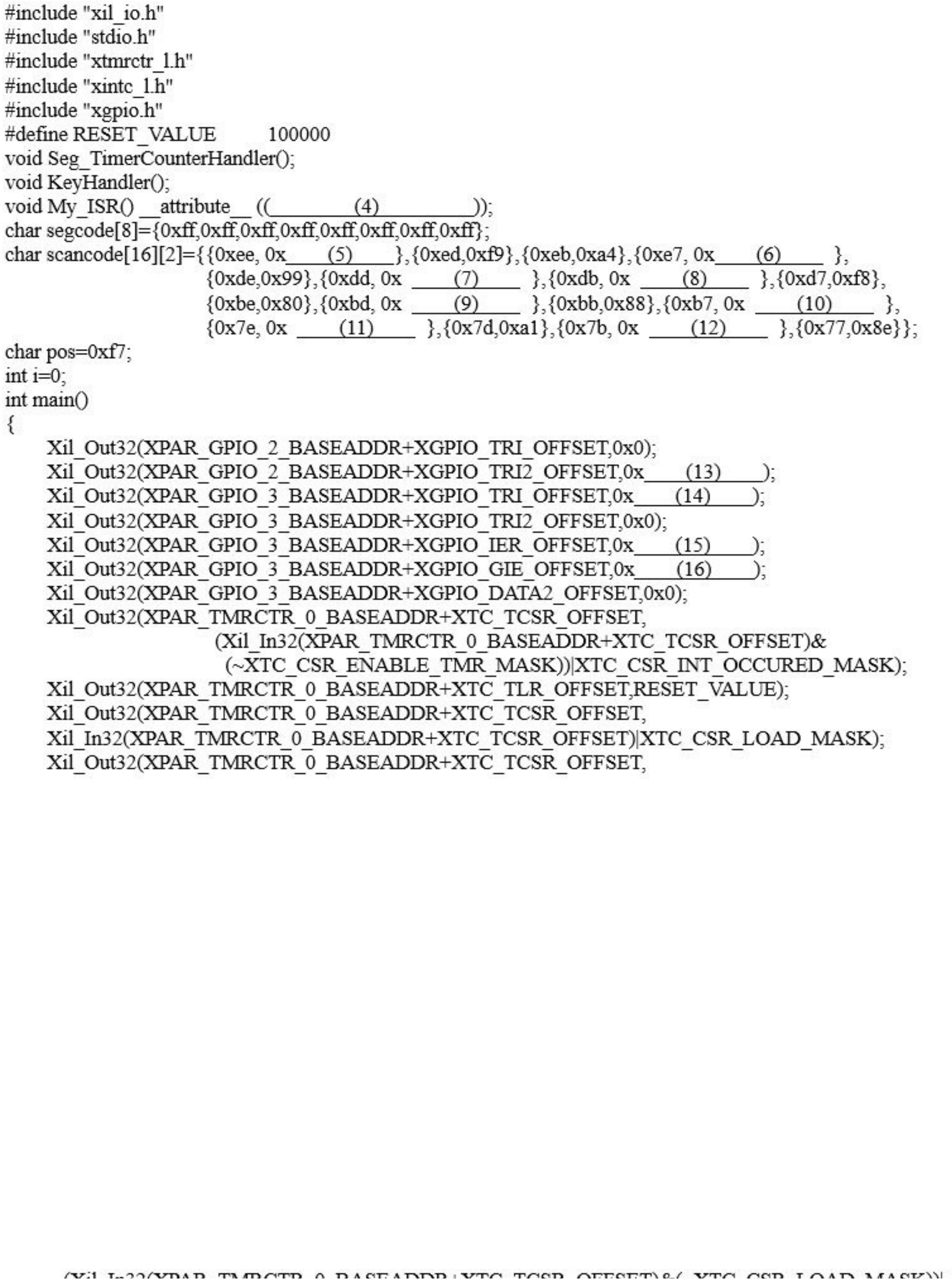
# 七.中断24分

已知通过 GPIO 中断方式获取矩阵键盘按键编码并显示在 4 位七段数码管上的接口电路如图 1 所示。按键对应的十六进制数字在七段数码管上的字型码如图 2 所示。控制程序将最近按下的 4 个按键依次显示在 4 位七段数码管上且最后按下的按键显示在最左边，每按一个按键其余按键对应的字符依次向右移一位。



1.给了电路图，连线（只需要连几根中断的线）

2.

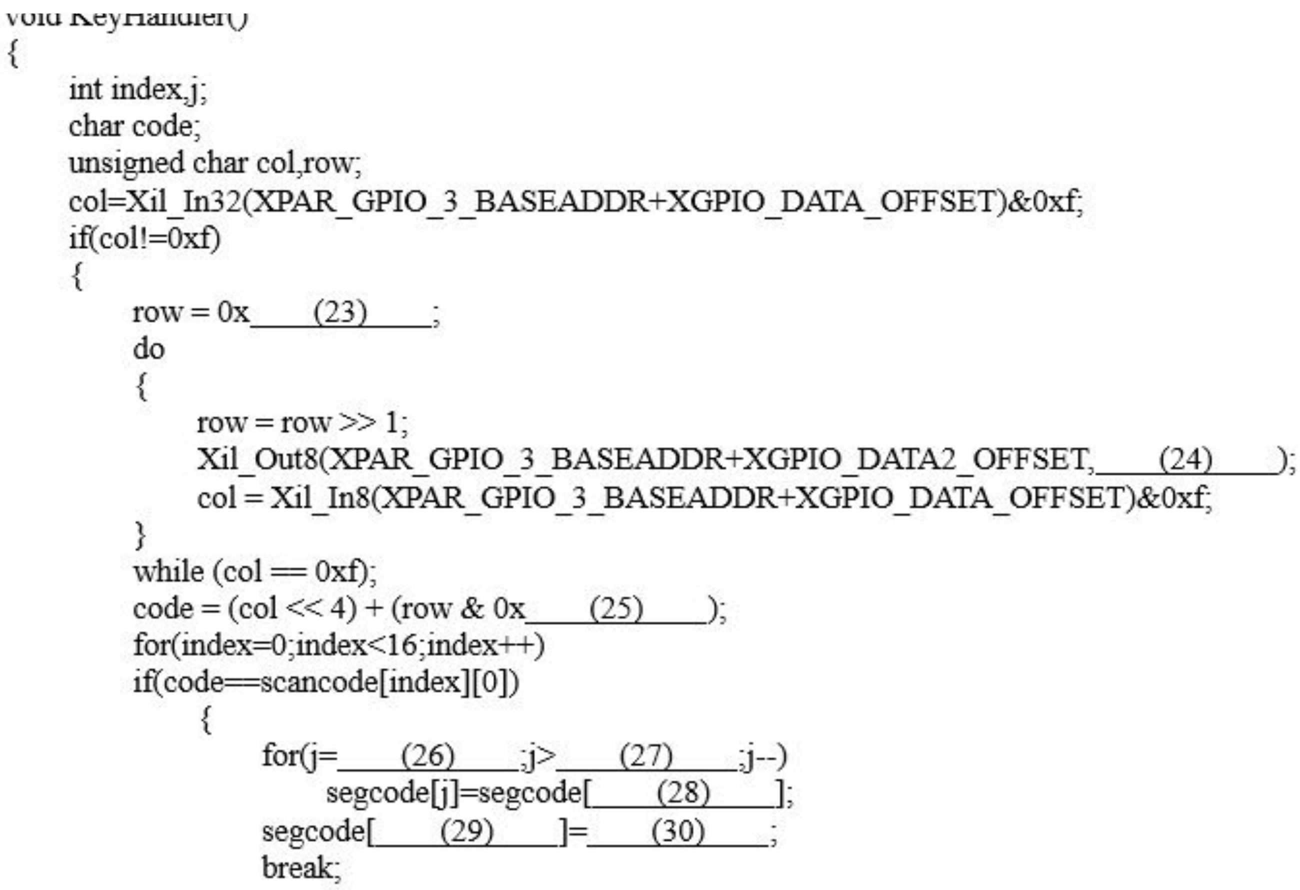


```
#include "xil_io.h"
#include "stdio.h"
#include "xtmrctr_l.h"
#include "xintc_l.h"
#include "xgpio.h"
#define RESET_VALUE        100000
void Seg_TimerCounterHandler();
void KeyHandler();
void My_ISR() __attribute__ ((________(4)__________));
char segcode[8]={0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff};
char scancode[16][2]={{0xee, 0x____(5)____},{0xed,0xf9},{0xeb,0xa4},{0xe7, 0x____(6)_____ },
                      {0xde,0x99},{0xdd, 0x _____(7)_____ },{0xdb, 0x _____(8)_____ },{0xd7,0xf8},
                      {0xbe,0x80},{0xbd, 0x _____(9)_____ },{0xbb,0x88},{0xb7, 0x _____(10)_____ },
                      {0x7e, 0x _____(11)_____ },{0x7d,0xa1},{0x7b, 0x _____(12)_____ },{0x77,0x8e}};
char pos=0xf7;
int i=0;
int main()
{
    Xil_Out32(XPAR_GPIO_2_BASEADDR+XGPIO_TRI_OFFSET,0x0);
    Xil_Out32(XPAR_GPIO_2_BASEADDR+XGPIO_TRI2_OFFSET,0x____(13)____);
    Xil_Out32(XPAR_GPIO_3_BASEADDR+XGPIO_TRI_OFFSET,0x____(14)____);
    Xil_Out32(XPAR_GPIO_3_BASEADDR+XGPIO_TRI2_OFFSET,0x0);
    Xil_Out32(XPAR_GPIO_3_BASEADDR+XGPIO_IER_OFFSET,0x____(15)____);
    Xil_Out32(XPAR_GPIO_3_BASEADDR+XGPIO_GIE_OFFSET,0x____(16)____);
    Xil_Out32(XPAR_GPIO_3_BASEADDR+XGPIO_DATA2_OFFSET,0x0);
    Xil_Out32(XPAR_TMRCTR_0_BASEADDR+XTC_TCSR_OFFSET,
                    (Xil_In32(XPAR_TMRCTR_0_BASEADDR+XTC_TCSR_OFFSET)&
                     (~XTC_CSR_ENABLE_TMR_MASK))|XTC_CSR_INT_OCCURED_MASK);
    Xil_Out32(XPAR_TMRCTR_0_BASEADDR+XTC_TLR_OFFSET,RESET_VALUE);
    Xil_Out32(XPAR_TMRCTR_0_BASEADDR+XTC_TCSR_OFFSET,
    Xil_In32(XPAR_TMRCTR_0_BASEADDR+XTC_TCSR_OFFSET)|XTC_CSR_LOAD_MASK);
    Xil_Out32(XPAR_TMRCTR_0_BASEADDR+XTC_TCSR_OFFSET,
        (Xil_In32(XPAR_TMRCTR_0_BASEADDR+XTC_TCSR_OFFSET)&(~XTC_CSR_LOAD_MASK))|
```

```
    (Xil_In32(XPAR_TMRCTR_0_BASEADDR+XTC_TCSR_OFFSET)&(~XTC_CSR_LOAD_MASK))|
                  XTC_CSR_ENABLE_TMR_MASK|XTC_CSR_ENABLE_INT_MASK|
                     XTC_CSR_DOWN_COUNT_MASK|XTC_CSR_AUTO_RELOAD_MASK);
    Xil_Out32(XPAR_AXI_INTC_0_BASEADDR+XIN_IER_OFFSET,0x____(17)____);
    Xil_Out32(XPAR_AXI_INTC_0_BASEADDR+XIN_MER_OFFSET, 0x____(18)____);
    microblaze_enable_interrupts();
    while(1);
return 0;
}
void My_ISR()
{
    int status;
    status=Xil_In32(XPAR_AXI_INTC_0_BASEADDR+XIN_ISR_OFFSET);
    if((status&0x8)==0x8)
        Seg_TimerCounterHandler();
    else if((status&0x4)==0x4)
        KeyHandler();
    Xil_Out32(XPAR_AXI_INTC_0_BASEADDR+XIN_IAR_OFFSET,status);
}
void Seg_TimerCounterHandler()
{
    Xil_Out8(XPAR_GPIO_2_BASEADDR+XGPIO_DATA_OFFSET,____(19)____);
    Xil_Out8(XPAR_GPIO_2_BASEADDR+XGPIO_DATA2_OFFSET,____(20)____);
    pos=pos>>1;
    i++;
    if(i==____(21)____)
    {
        i=0;
        pos=0x____(22)____;
    }
    Xil_Out32(XPAR_TMRCTR_0_BASEADDR+XTC_TCSR_OFFSET,
            Xil_In32(XPAR_TMRCTR_0_BASEADDR+XTC_TCSR_OFFSET));
}




void KeyHandler()
```



```
void KeyHandler()
{
    int index,j;
    char code;
    unsigned char col,row;
    col=Xil_In32(XPAR_GPIO_3_BASEADDR+XGPIO_DATA_OFFSET)&0xf;
    if(col!=0xf)
    {
        row = 0x____(23)____;
        do
        {
            row = row >> 1;
            Xil_Out8(XPAR_GPIO_3_BASEADDR+XGPIO_DATA2_OFFSET,____(24)____);
            col = Xil_In8(XPAR_GPIO_3_BASEADDR+XGPIO_DATA_OFFSET)&0xf;
        }
        while (col == 0xf);
        code = (col << 4) + (row & 0x____(25)____);
        for(index=0;index<16;index++)
        if(code==scancode[index][0])
            {
                for(j=____(26)____;j>____(27)____;j--)
                    segcode[j]=segcode[____(28)____];
                segcode[____(29)____]=____(30)____;
                break;
```

填空，实际考题和这个差不多，只不过挖的空不一样