电子科技大学 2020-2021 学年第 2 学期期 末 考试 A 卷

考试科目： 微处理器与嵌入式系统设计 考试形式：一本书开卷 考试日期：2021 年 6 月 16 日

成绩构成比例： 研讨班：平时 30 %， 实验 20 %， 期末 50 %
普通班：平时 20 %， 实验 20 %， 期末 60 %

本试卷由 三 部分构成，共 8 页。考试时长：120 分钟 注：**只允许带一本教材，不能夹带习题集**

题号	一	二	三(1)	三(2)	三(3)	三(4)	三(5)	三(6)	三(7)	合计
得分										

注意：请将第一、二题答案填入指定位置。

得 分

一、选择题答案（共 25 分，共 25 题，每题 1 分，注意部分为多选题。）

1	2	3	4	5	6	7	8	9	10	11	12	13
CD	D	B	A	B	ABCD	ACD	B	A	D	C	C	C

14	15	16	17	18	19	20	21	22	23	24	25
A	AB	B	A	B	D	C	B	D	D	A	C

得 分

二、填空题答案（共 20 分，共 20 空，每空 1 分）

1. ① 下一条待取指令的地址
2. ① 运算器，② 控制器（可不按顺序）
3. ① 指令周期
4. ① 立即数寻址
5. ① 1100
6. ① 总线冲突
7. ① 系统/片间
8. ① 中断设备与 CPU 不同步，② 可能有多个中断同时发生
9. ① 吞吐率，② 容量-性能-价格
10. ① 立即数方式，② 寄存器方式，③ 寄存器移位方式（可不按顺序）
11. ① 32，② 程序计数器（PC）
12. ① ENTRY
13. ① 接口初始化，② 数据收发

一、 选择题（共 25 分，共 25 题，每题 1 分，注意部分为多选题。）（答案请填入第 1 页选择题答案处。）

1. 【多选题】下面（　　）**不属于**计算机体系结构设计需要考虑的范畴。
   A.指令集内容　　B.存储器编址方式　　C. CPU 主频　　D.CPU 是否采用 5nm 工艺
2. 微处理器系统的“字长”通常是指（　　）。
   A. CPU 外部数据总线的最大位数　　B. CPU 外部控制总线的最大位数
   C. CPU 外部地址总线的最大位数　　D. CPU 内部运算器一次能处理二进制数的最大位数
3. 微处理器设计时，采用指令流水线技术的主要目的是（　　）。
   A. 提高 DMA 的传递速度　　B.提高指令处理的吞吐率
   C. 提高存储器的存取速度　　D.提高每条指令的处理速度
4. 指令流水线存在的相关性可能会引起流水线的停顿，从而影响流水线的性能和效率。其中可以采用分支预测方法来缓解的是（　　）。
   A.控制相关　　B. 数据相关　　C. 名字相关　　D.结构相关
5. 关于随机逻辑控制器，说法正确的是（　　）。
   A. 每个指令都需要一组逻辑电路实现　　B. 指令集升级改动代价大
   C. 微指令的形成需要考虑各种状态信息　　D. 微指令形成的逻辑元件在控制器中占较大比重
6. 【多选题】影响 CPU 执行现行程序时间的因素有（　　）。
   A.主机频率　　B.内存容量　　C.Cache 大小　　D.总线架构
7. 【多选题】以下符合 RISC 系统特点的是（　　）。
   A.指令编码长度固定　　B.为降低复杂度尽量不使用流水线
   C.运算类指令不能访问存储器　　D.只能使用 Load/Store 类指令存取存储单元
8. Cache 进行页替换时，利用程序局部性原理而采用的算法是（　　）。
   A.先进先出　　B.最近最少使用法　　C.随机法　　D.MMU 地址映射
9. 某 32 位微处理器系统中，字对齐存储方式下，字数据存储地址的特点是最低两位为（　　）。
   A. 00　　B. 01　　C. 10　　D. 11
10. 微处理器系统响应中断后，保护断点的目的是（　　）。
    A.查找识别中断源　　B.使 CPU 能跳转到中断服务程序开始的地方
    C.获取中断向量　　D.完成中断服务程序后，能正确返回被中断的程序
11. 采用程序控制方式的 I/O 接口中，通常微处理器可从（　　）获得外设状态信息。
    A.状态总线　　B.地址总线　　C.数据总线　　D.控制总线
12. 某外设接口中含有两个数据端口，意味着（　　）。
    A.这两个端口必须分配不同的地址　　B.系统必须采用存储器映射编址方式
    C.该接口中一般至少需要两个寄存器　　D.系统必须采用统一编址方式
13. 在多任务系统中，为提高 CPU 的工作效率，低速外设应该在准备好数据后才通知 CPU 进行数据交换。完成这种数据传输最好选用（　　）I/O 方式。
    A.无条件　　B.查询　　C.中断　　D.DMA
14. 下面二进制数的最低位是一个比特的校验位，则采用了奇校验的编码是（　　）.
    A. 10101011　　B.11010111　　C.11111111　　D. 111111110
15. 【多选题】以下**不属于**片内总线标准的是（　　）。
    A. AMBA　　B. AXI4　　C.USB　　D.SPI
16. 某微处理器系统中，64 位总线的时钟频率为 800 MHz，总线周期数为 1/2，则其带宽为（　　）。
    A. 25.6GB/s　　B. 12.8 GB/s　　C. 6.4 GB/s　　D.3.2GB/s
17. PCIE *16 总线中的“16”表示（　　）。
    A.16 个总线通道　　B. 16 个配置寄存器　　C. 总线速率 16GB/s　　D. 总线速率 16Gb/s

18. 下列关于 AHB 总线的说法，**不正确**的是（　　）。
A. AHB 总线支持突发传输　　B. AHB 总线提供了丰富的外设接口
C. AHB 总线支持多从机　　D. AHB 总线支持多主机

19. ARM 微处理器中，链接寄存器 LR 通常用于（　　）。
A.指示待取指令的地址　　B.存放指令
C.指示正在执行指令的地址　　D.保存子程序返回地址

20. ARM 微处理器中进位/借位标志 C=1 时，表示（　　）。
A.两个带符号数相加溢出　　B.两个带符号数相减溢出
C.两个无符号数相加进位　　D.两个无符号数相减借位

21. 下列描述**不符合**事实的是（　　）。
A.ARM 公司不直接从事芯片生产　　B.ARM 公司最初成立于美国
C.ARM 公司是一家 IP 供应商　　D.RISC-V 和 ARM 都属于 RISC 指令集架构

22. ATPCS 规定的子程序调用规则**不包括**（　　）。
A.寄存器使用规则　B.数据栈使用规则　C.参数传递规则　D.Cache 读写规则

23. 宏和子程序有许多相似之处，相比子程序调用，宏的优点在于（　　）。
A. 可以提供模块化的程序设计支持　　B. 宏可以减小最终生成的映像文件的大小
C. 支持参数传递　　D. 宏调用时不需要保护现场，系统开销较小

24. 复位模块是微处理器最小硬件系统中的重要部分。常见的复位模式**不包括**（　　）。
A.自动复位　B.手动复位　C.内部复位　D. 加电复位

25. Linux 驱动程序可以在运行时以模块的形式进行动态加载/卸载。卸载模块使用的命令是(　　)。
A.lsmod　B.insmod　C.rmmod　D.delmod

二、填空题（共 20 分，共 20 空，每空 1 分）（答案请填入第 1 页填空题答案处）

1. 微处理器中，程序计数器 PC 存放的内容一般是①_____。
2. 冯·诺依曼计算机硬件由①_____、②_____、存储器、输入设备和输出设备组成。
3. CPU 从主存取出一条指令并执行该指令的时间称为①_____。
4. 微处理器从指令中直接获取操作数本身的寻址方式一般称为①_____。
5. 异步串行通信方式下，设置波特率因子为 32，字符长度为 8 位（含 1 位奇校验位），起始位 1 位，停止位为 2 位，若每秒传输 100 个字符，则该总线的传输速率为①___bps。
6. 同一总线上，多个主设备同时发送信息导致的异常一般称为①_____。
7. 将微处理器芯片、存储器芯片及 I/O 接口芯片连接起来总线一般称为①_____。
8. 设置中断触发器保存外设提出的中断请求，主要是因为①_____和②_____。后者也是中断优先级、中断排队等概念提出的缘由。
9. 多体交叉编址技术可用于提高存储器的①___，而分层技术则主要是为了解决②_____问题。
10. 三操作数的 ARM 指令中，第二源操作数支持①_____、②_____和③_____三种形式。
11. ARM 转移指令（B）可以实现从当前指令向前或向后①_____Mbyte 地址空间的转移；而通过向寄存器②_____中写入转移地址值，则可实现在 4GByte 地址空间中的任意转移。
12. 伪指令①_____用于指定 ARM 汇编源程序的入口地址。
13. 对 UART 串行通信接口的编程通常包括①_____和②_____两个部分，其中前一个部分主要用于配置工作模式、数据帧格式、通信速率等参数。

三、综合题（共 7 题，共 55 分）

得 分

1. （8 分）冯・诺伊曼计算机的基本设计思想是什么？ARM 芯片早期采用冯・诺伊曼架构，后来为改进性能采用了哈佛架构，试简述其改进思路。

【参考答案】

1）冯・诺伊曼计算机的基本设计思想主要包括以下几点：

- 采用二进制表达数据和指令信息；（2 分）
- 将数据和指令事先保存在存储器中；（2 分）
- 控制器自动按顺序取指、执行。（2 分）

2）指令和数据分开存储，并采用分离总线，有利于并行处理提高存储器的吞吐率。（2 分）

得 分

2. （8 分）某 CPU 采用 m 级流水线，其各级时长（即拍长）均为 Δt。则该 CPU 连续处理 n 条指令时的加速比 Sp 为可表达为：

$$S_p = \frac{T_{串行}}{T_{流水}} = \frac{n \cdot m\Delta t}{m\Delta t + (n-1)\Delta t} = \frac{nm}{m+n-1} = \frac{m}{1+(m-1)/n}$$

从该公式可以看出，当指令条数 n 趋近无穷大时，加速比 S_p=m。

试分析：我们是否可以由此得到结论，即通过不断增加流水线级数就可以提高 CPU 的性能？为什么？

【参考答案】

CPU 性能并不能单纯通过不断增加指令流水线级数来持续提高；（2 分）

主要原因包括：

1）上述理想加速比公式并未考虑流水线效率问题：流水线本身也会带来读写延迟，制约了流水线的吞吐量，即流水线段数增加，会导致流水线效率下降。（2 分）

2）增加流水线级数会导致各种相关性问题更加严重。（2 分）

3）增加流水线级数会导致 CPU 硬件复杂度上升，从而带来其它问题，如可靠性、测试复杂度等等。（2 分）

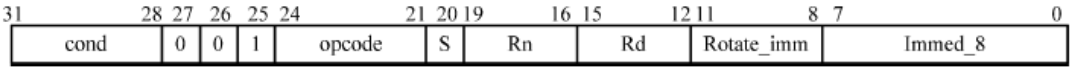
得 分

3. （6 分）ARM 指令可以使用立即数，但某些立即数（如 0x01、0xF000000F 等）合法， 某些立即数（如 0x101、0xF000001F 等）则不合法，试解释为什么。

【参考答案】

从下面 ARM 的机器指令编码格式可以看出，立即数只能用最低 12 位表示。（2 分）

其中 Immed_8 表示 8 位无符号常量，Rotate_imm 表示将该 8 位数循环右移 2*Rotate_imm 次后恢复出 32 位数值常量。（2 分）



只有能用上述方法恢复的立即数才是合法的，而无法用上述方法恢复的立即数即为不合法数据。究其根本，是机器指令编码长度有限造成的。（2 分）

得 分

4. （8 分）请从硬件设计开销和性能方面比较微码结构与随机逻辑结构

【参考答案】

1. 硬件设计开销方面：
- 随机逻辑 CPU 的硬件和指令集必须同步进行设计和优化，因此比较复杂。（2 分）
- 微码 CPU 的指令集并不直接影响现有硬件，修改指令集并不需要重新设计新的硬件。（2 分）

2. 性能方面：
- 如果采用相同指令集，则随机逻辑 CPU 操作会更快。（2 分）
- 如果执行相同的计算任务，微码 CPU 能够通过使用更少的指令达到更高性能。当系统整体受限于存储器的速度时，微码 CPU 对性能提高的优势更加明显。（2 分）

得 分

5. （6 分）简述总线仲裁的两种方式及其特点。

【参考答案】

1. 分布式（对等式）仲裁（3 分）

控制逻辑分散在连接于总线上的各个部件或设备中，协议复杂且昂贵，效率高；

2. 集中式（主从式）仲裁（3 分）

采用专门的控制器或仲裁器；总线控制器或仲裁器可以是独立的模块或集成在 CPU 中；协议简单而有效，但总体系统性能较低

得 分

6. （12 分）某微处理器系统地址总线宽度为 16bit，字长为 8bit，请问：

1）（2 分）该系统的最大寻址空间是多少？

【参考答案】 最大寻址空间为 2^{16}=64KB

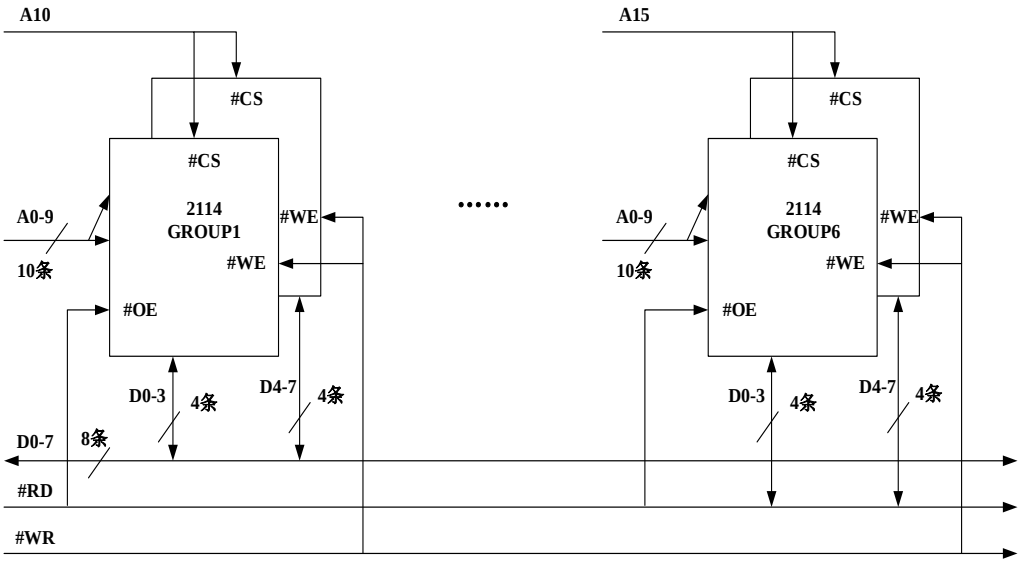
2）（2 分）用 SRAM2114（1K×4bits）芯片组成存储系统，若采用线选法，最大可扩充多少存储容量？

【参考答案】 可扩充存储容量为（16-10）*1KB= 6KB

3）（8 分）绘出相应的存储系统连接图，并写出每组的地址范围。

【参考答案】 参考原理图（低位地址线 1 分、数据线 1 分），

地址范围（每行 1 分）：



存储组号	首地址	末地址
1	0xF800	0xFBFF
2	0xF400	0xF7FF
3	0xEC00	0xEFFF
4	0xDC00	0xDFFF
5	0xBC00	0xBFFF
6	0x7C00	0x7FFF

得 分

7. （7 分）下面图（a）~（b）给出了在 Zynq 7020 FPGA 芯片上搭建 SoC 系统、实现流水灯显示控制实验的几个关键步骤，请据此完成下面几个问题。

1）（2 分）图（c）中 offset address 和 high address 分别代表什么含义？

【参考答案】

这两个地址确定了各模块占用的系统地址空间, offset address 指低端地址（1 分），high address 确定了高端地址（1 分）：

- ✓ LED 接口模块占用的系统地址空间为 0x41200000 到 0x4120FFFF
- ✓ SW 接口模块占用的系统地址空间为 0x41210000 到 0x4121FFFF
- ✓ 这两个模块均分别占用了 64k 个地址。

2）（4 分）为图（d）中的代码①~④添加注释。

【参考答案】① 对 LED 口外接的 8 个 LED 灯轮流做如下处理（1 分）

【参考答案】② 控制指定引脚的输出，点亮对应的 LED 灯（1 分）

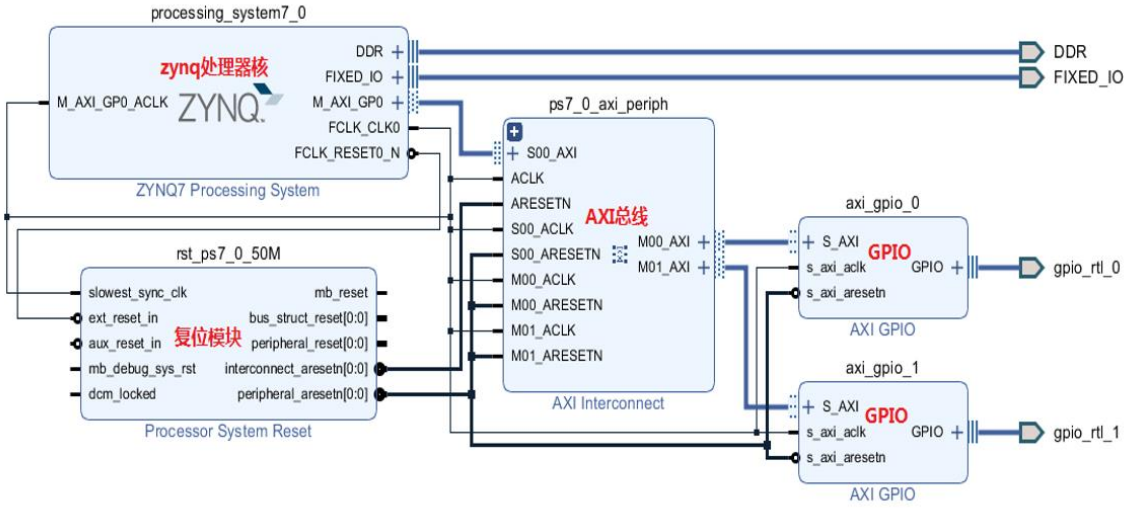
【参考答案】③ 延迟至指定时间（LED_DELAY）（1 分）

【参考答案】④ 从 SW 口读入外接拨码开关的状态（1 分）

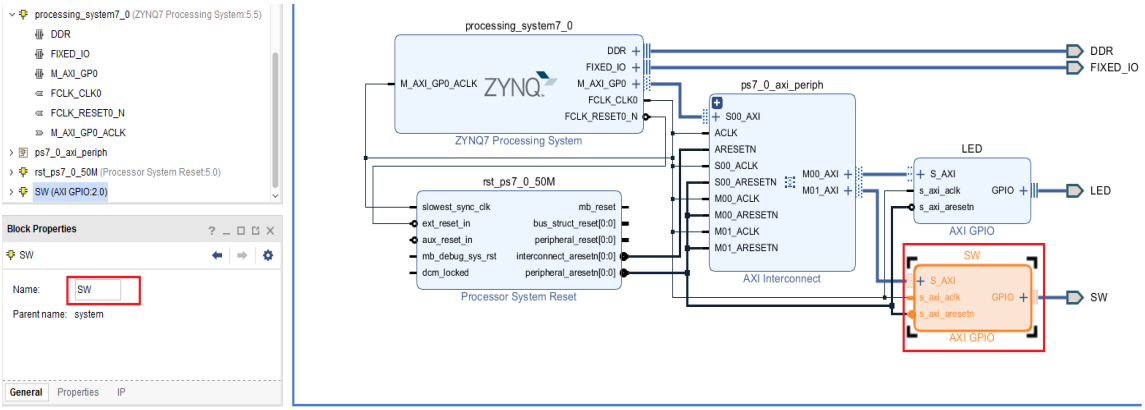
3）（1 分）如想调整流水灯的闪烁速度，可修改哪一个“#define”语句？如何修改？

【参考答案】 可修改语句：#define LED_DELAY 10000000

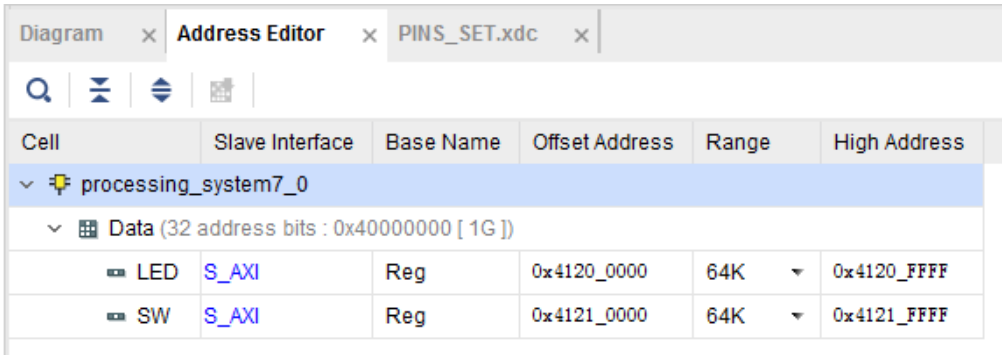
- ✓ 如想闪烁的更快，则将 LED_DELAY 的值减小；
- ✓ 如想闪烁的更慢，则将 LED_DELAY 的值加大。



图（a）



图（b）



图（c）

```
/*流水灯功能：拨动任意一个拨码开关时，8个LED灯轮流点亮一次*/
#include"xparameters.h"
#include"xgpio.h"
#include"xil_printf.h
#include"xil_cache.h"
#define GPIO_BITWIDTH 8
#define GPIO_LED_DEVICE_ID 0
#define GPIO_SW_DEVICE_ID 1
#define LED_DELAY 10000000
#define LED_MAX_BLINK 1
#define LED_CHANNEL 1
#define printf xil_printf
XGpio GpioOutput;
XGpio GpioInput;
u32 flag=0;
u32 DataRead;

int Gpio_led(u16 Deviceid,u32 Gpio_Width)
{   volatile int delay;
    u32 Ledbit;
    u32 Ledloop;
    int status;
    status = XGpio_Initialize(&GpioOutput,Deviceid);
    if(status !=XST_SUCCESS)
        return XST_FAILURE;
    XGpio_SetDataDirection(&GpioOutput,LED_CHANNEL,0x0);
    for(Ledbit =0x0;Ledbit < Gpio_Width;Ledbit++)                    //  ①
    {     for(Ledloop=0x0;Ledloop<LED_MAX_BLINK;Ledloop++)
          {     XGpio_DiscreteWrite(&GpioOutput,LED_CHANNEL,1<<Ledbit);    //  ②
                for(delay=0;delay<LED_DELAY;delay++);                      //  ③
                XGpio_DiscreteClear(&GpioOutput,LED_CHANNEL,0x00);
                for(delay=0;delay<LED_DELAY;delay++);   }
    }
    return XST_SUCCESS;   }

u32 Gpio_sw(u16 Deviceid,u32 *DataRead)
{   int Status;
   Status = XGpio_Initialize(&GpioInput, Deviceid);
    if (Status != XST_SUCCESS)
        return XST_FAILURE;
    XGpio_SetDataDirection(&GpioInput, LED_CHANNEL, 0xFFFFFFFF);
    *DataRead = XGpio_DiscreteRead(&GpioInput, LED_CHANNEL);          //  ④

    u32 data;
    data=*DataRead;
  return data;      }

int main( )
{   while(1)
    {     flag =Gpio_sw(GPIO_SW_DEVICE_ID, &DataRead);
          if(flag == 0)
          {     u32 status;
                status = Gpio_led(GPIO_LED_DEVICE_ID,GPIO_BITWIDTH);
                if(status==0)
                      printf("SUCCESS.\r\n");
                else   printf("FAIL.\r\n");      }
          else
          {      u32 Ledbit;
                for(Ledbit =0x0;Ledbit < GPIO_BITWIDTH;Ledbit++)
                    XGpio_DiscreteClear(&GpioOutput,LED_CHANNEL,0x00);   }
    }
}
```

图（d）