期末作业

1.与CISC架构相比RISC架构有哪些优点？

RISC（Reduced Instruction Set Computer）架构相比于 CISC（Complex Instruction Set Computer）有以下主要优点：

1. **指令集简单**：RISC 的指令集简洁、固定长度，易于解码，执行速度更快，硬件实现更简单。
2. **执行速度快**：由于指令简化，执行周期通常只有一个时钟周期，指令流水线的效率更高。
3. **高效的寄存器使用**：RISC 通常有大量通用寄存器，减少了对内存的访问，提高了性能。
4. **优化编译器设计**：RISC 的简化指令集使编译器更容易优化代码生成。
5. **硬件设计简单**：RISC 的指令集简单，硬件控制单元设计简单，功耗更低，适合嵌入式系统。
6. **扩展性强**：由于指令集设计的简单性，可以更容易地扩展或优化架构。

2.ARM体系结构支持几种类型的异常，并说明其异常处理模式和优先级状态？

**ARM 支持的异常类型：**

1. **复位（Reset）**：
   * **来源**：电源上电或手动复位信号。
   * **处理模式**：进入 Supervisor 模式。
   * **优先级**：最高。
2. **中断请求（IRQ）**：
   * **来源**：外部硬件设备的中断信号。
   * **处理模式**：进入 IRQ 模式。
   * **优先级**：低于 FIQ。
3. **快速中断请求（FIQ）**：
   * **来源**：外部硬件设备的快速中断信号。
   * **处理模式**：进入 FIQ 模式。
   * **优先级**：高于 IRQ。
4. **未定义指令（Undefined Instruction）**：
   * **来源**：执行未定义的指令。
   * **处理模式**：进入 Undefined 模式。
   * **优先级**：低于复位。
5. **数据访问异常（Data Abort）**：
   * **来源**：数据访问发生错误。
   * **处理模式**：进入 Abort 模式。
   * **优先级**：低于未定义指令。
6. **指令预取异常（Prefetch Abort）**：
   * **来源**：指令预取发生错误。
   * **处理模式**：进入 Abort 模式。
   * **优先级**：低于数据访问异常。
7. **软件中断（SWI）**：
   * **来源**：执行 SWI 指令。
   * **处理模式**：进入 Supervisor 模式。
   * **优先级**：低于指令预取异常。

**优先级从高到低：**

复位 > FIQ > 数据访问异常（Data Abort） > 指令预取异常（Prefetch Abort） > 未定义指令 > SWI > IRQ。

3、某计算机系统的存储器地址空间为0xA800~0xCFFF，数据总线宽度为16位，若采用单片容量为16\*1位的SRAM芯片，

（1）系统存储器容量是多少？

（2）组成该存储系统共需该类芯片多少个？

（3）整个系统应分为多少个芯片组？

**已知条件：**

* 存储器地址空间：0xA800 ~ 0xCFFF。
* 数据总线宽度：16 位。
* SRAM 芯片容量：16×1 位。

**(1) 系统存储器容量：**

* 地址空间大小 = 0xCFFF - 0xA800 + 1 = 0x2800 = 10,240 字节。
* 因为数据总线宽度为 16 位（即 2 字节），所以存储器容量为：

**10,240 × 2 = 20,480 字节 = 20 KB**。

**(2) 所需芯片数量：**

* 每片 SRAM 芯片容量为 16×1 位 = 16 位 = 2 字节。
* 总容量为 20,480 字节，因此所需芯片数量为：

**20,480 ÷ 2 = 10,240 片**。

**(3) 芯片组划分：**

* 每个芯片组的容量为 16 位（2 字节），而数据总线宽度也是 16 位。
* 因此，**整个系统只需 1 个芯片组**。

4、某微处理器系统有8个I/O接口芯片，每个接口芯片占用8个端口地址。若起始地址为0x9000，8个接口芯片的地址连续分布，用74LS138作为译码器，试画出端口译码电路图，并说明每个芯片的端口地址范围。

**已知条件：**

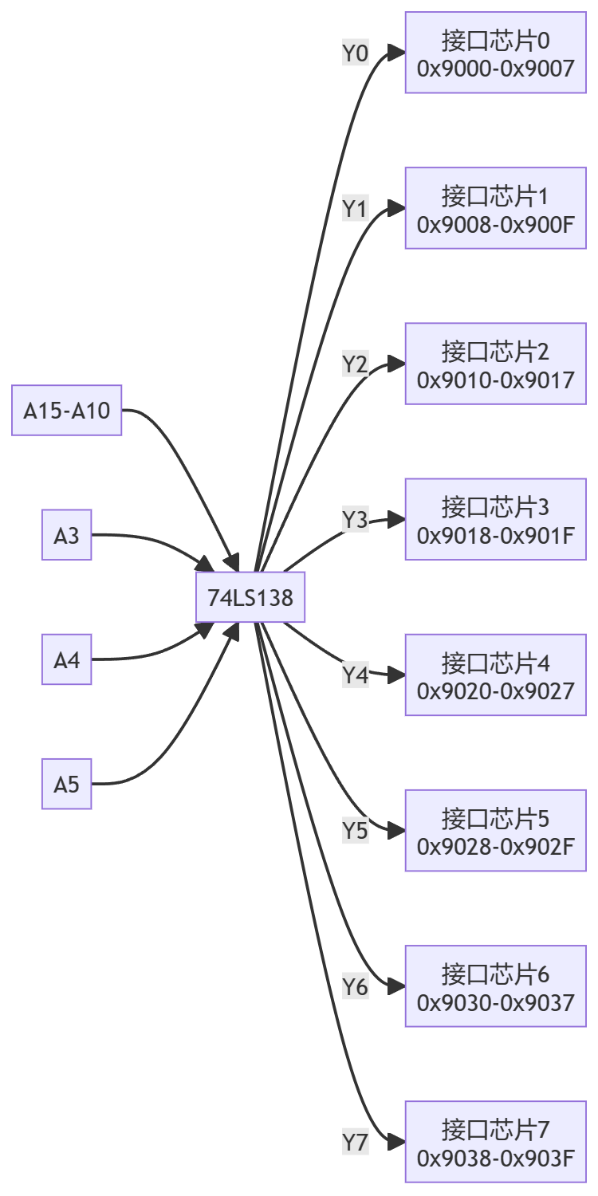
* 接口芯片数量：8 个。
* 每个芯片占用地址：8 个端口地址。
* 起始地址：0x9000。
* 地址范围连续，使用 74LS138 译码。

**端口地址范围：**

* 每个芯片占用 8 个端口地址，所以地址范围为：
  + 接口芯片 0：0x9000 ~ 0x9007。
  + 接口芯片 1：0x9008 ~ 0x900F。
  + 接口芯片 2：0x9010 ~ 0x9017。
  + 接口芯片 3：0x9018 ~ 0x901F。
  + 接口芯片 4：0x9020 ~ 0x9027。
  + 接口芯片 5：0x9028 ~ 0x902F。
  + 接口芯片 6：0x9030 ~ 0x9037。
  + 接口芯片 7：0x9038 ~ 0x903F。

**电路图设计：**

以下为端口译码电路的设计，包括 74LS138 的连接。



5、指出下列指令操作数的寻址方式：

(1) MOV R1，R2 (2)SUBS R0, R0, #2

(3) SWP R1, R1, [R2] (4) STR R1, [R0, #-4]!

(5) LDMFD SP!, {R1~R4, LR} (6) ANDS R0, R0, R1, LSL R2

(7) STMIA R1!, {R2~R5, R8} (8) BL AGAIN

(1) **MOV R1, R2**：寄存器直接寻址。  
(2) **SUBS R0, R0, #2**：立即数寻址（#2 为立即数）。  
(3) **SWP R1, R1, [R2]**：寄存器间接寻址（通过 R2 指向的内存地址）。  
(4) **STR R1, [R0, #-4]!**：基址变址寻址，带写回。  
(5) **LDMFD SP!, {R1~R4, LR}**：块传输寻址（递增堆栈寻址）。  
(6) **ANDS R0, R0, R1, LSL R2**：寄存器移位寻址（R1 左移 R2 位）。  
(7) **STMIA R1!, {R2~R5, R8}**：基址递增寻址，带写回。  
(8) **BL AGAIN**：PC 相对寻址。

6、判断下列指令的正误，并说明理由。

（1）ADD R1, R2, #4! （2）LDMFD R13!, {R2, R4}

（3）LDR R1, [R3]! （4）MVN R5, #0x2F100

（5）SBC R15, R6, LSR R4 （6）MUL R2, R2, R5

（7）MSR CPSR, #0x001 （8）LDRB PC, [R3]

(1) **ADD R1, R2, #4!**：错误，ARM 中没有 #4! 的立即数格式。  
(2) **LDMFD R13!, {R2, R4}**：正确，递增堆栈寻址。  
(3) **LDR R1, [R3]!**：正确，带写回的基址寻址。  
(4) **MVN R5, #0x2F100**：错误，立即数格式超出范围，ARM 的立即数限制为 8 位旋转数。  
(5) **SBC R15, R6, LSR R4**：错误，PC（R15）不能作为目标寄存器。  
(6) **MUL R2, R2, R5**：正确，乘法指令格式正确。  
(7) **MSR CPSR, #0x001**：错误，不能直接向 CPSR 写入立即数。  
(8) **LDRB PC, [R3]**：错误，不能通过字节加载指令设置 PC。

7、用汇编语言实现128位数的减法。

SUB128:

LDR R0, [R1] ; 读取第一个数的低 32 位

LDR R2, [R3] ; 读取第二个数的低 32 位

SUBS R4, R0, R2 ; 低 32 位相减，结果存 R4

LDR R0, [R1, #4] ; 读取第一个数的次低 32 位

LDR R2, [R3, #4] ; 读取第二个数的次低 32 位

SBC R5, R0, R2 ; 次低 32 位减法，带借位

; 继续处理次高 32 位和高 32 位

LDR R0, [R1, #8]

LDR R2, [R3, #8]

SBC R6, R0, R2

LDR R0, [R1, #12]

LDR R2, [R3, #12]

SBC R7, R0, R2

STR R4, [R1]

STR R5, [R1, #4]

STR R6, [R1, #8]

STR R7, [R1, #12]

BX LR

8、分别编写一个主程序和子程序，实现主程序对子程序的调用。要求子程序完成两个数的加法运算功能；主程序完成对变量的初始化赋值后调用子程序，实现两个数的加法运算。按照如下两种方式来完成程序设计：

（1）主程序采用C语言程序，子程序采用汇编语言程序设计；

（2）主程序采用汇编语言程序，子程序采用C语言程序设计。

**(1) 主程序：C，子程序：汇编**

**C 主程序**：

#include <stdio.h>

extern int add(int a, int b);

int main() {

int num1 = 5, num2 = 10;

int result = add(num1, num2);

printf("Result: %d\n", result);

return 0;

}

**汇编子程序**：

.global add

add:

ADD R0, R0, R1 ; R0 = R0 + R1

BX LR ; 返回

**(2) 主程序：汇编，子程序：C**

**汇编主程序**：

.extern add

.global main

main:

LDR R0, =5 ; 初始化第一个参数

LDR R1, =10 ; 初始化第二个参数

BL add ; 调用子程序

B .

**C 子程序**：

int add(int a, int b) {

return a + b;

}