厦門大學
XIAMEN UNIVERSITY

电子系统设计 Part3

简单处理器系统设计(2)

信息与通信工程系 陈凌宇

chenly@xmu.edu.cn

导言

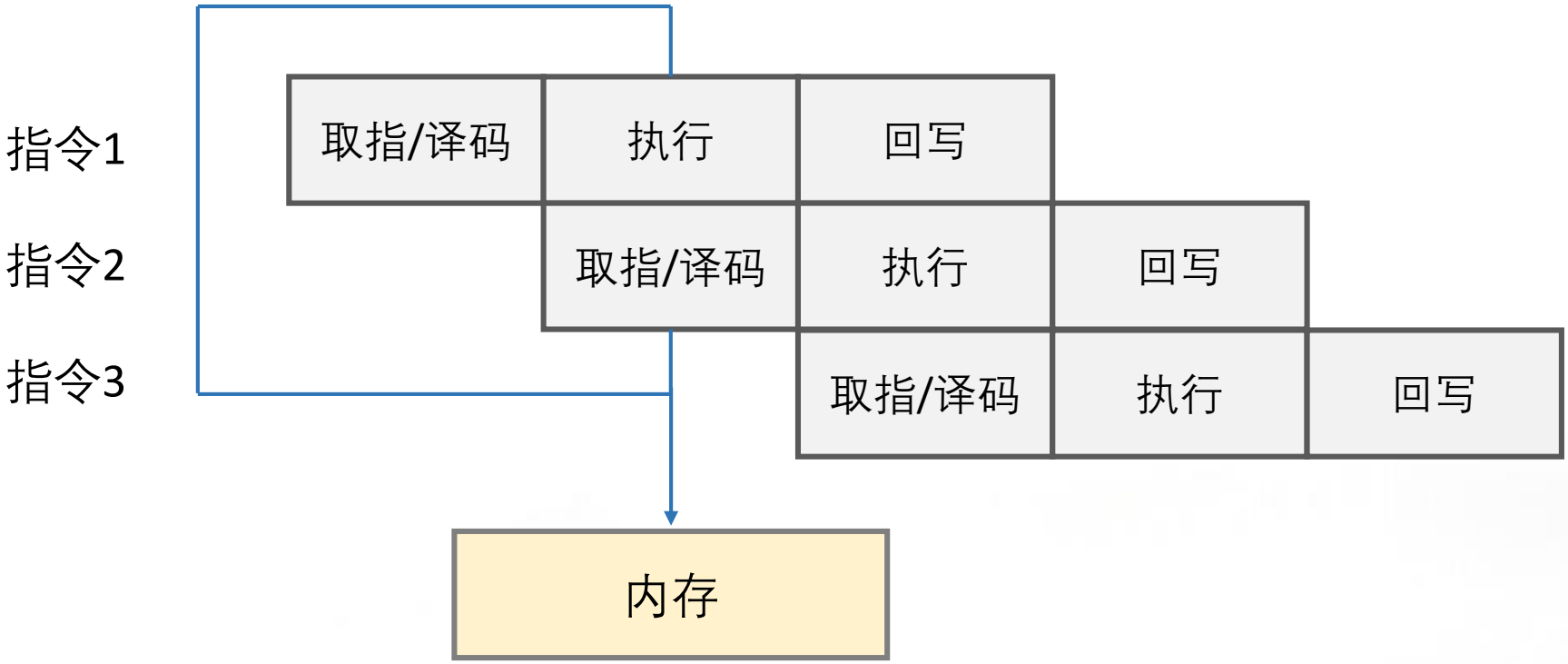
本部分介绍了CPU的三类冒险问题，以及外设及总线的设计思路。通过本节内容，应能掌握外设以及总线的设计方法。最后完成倒计数秒表的设计。

目录

- 处理器的冒险及解决方法
- GNU汇编器使用
- 总线设计
- 拓展实验（2）

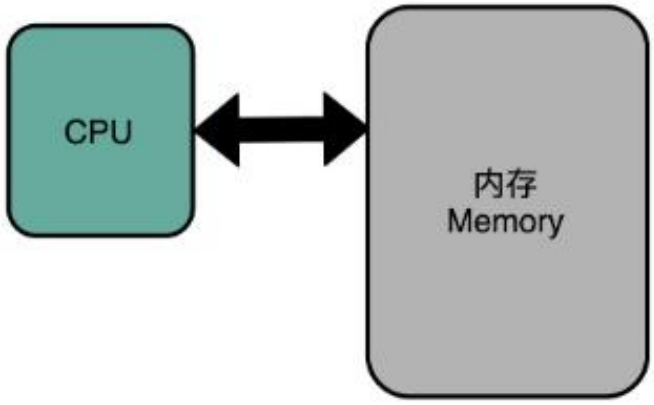
结构冒险

- CPU的流水线中，不同级(Stage)需要同时访问同个资源，造成冲突，称为结构冒险（Structural Hazard）。

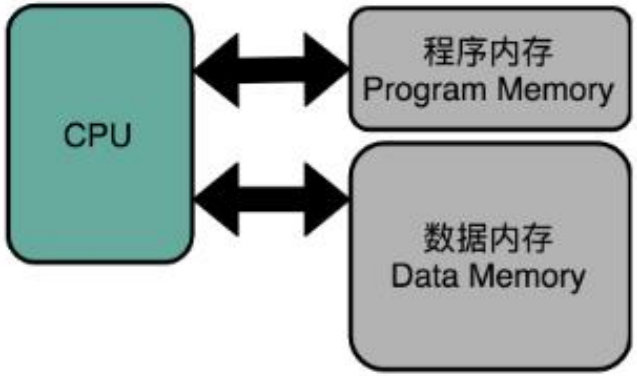


结构冒险

- 对于访问内存和取指令的冲突，一种解决方案就是把内存分成两部分，互相不冲突。即程序内存和数据内存独立。



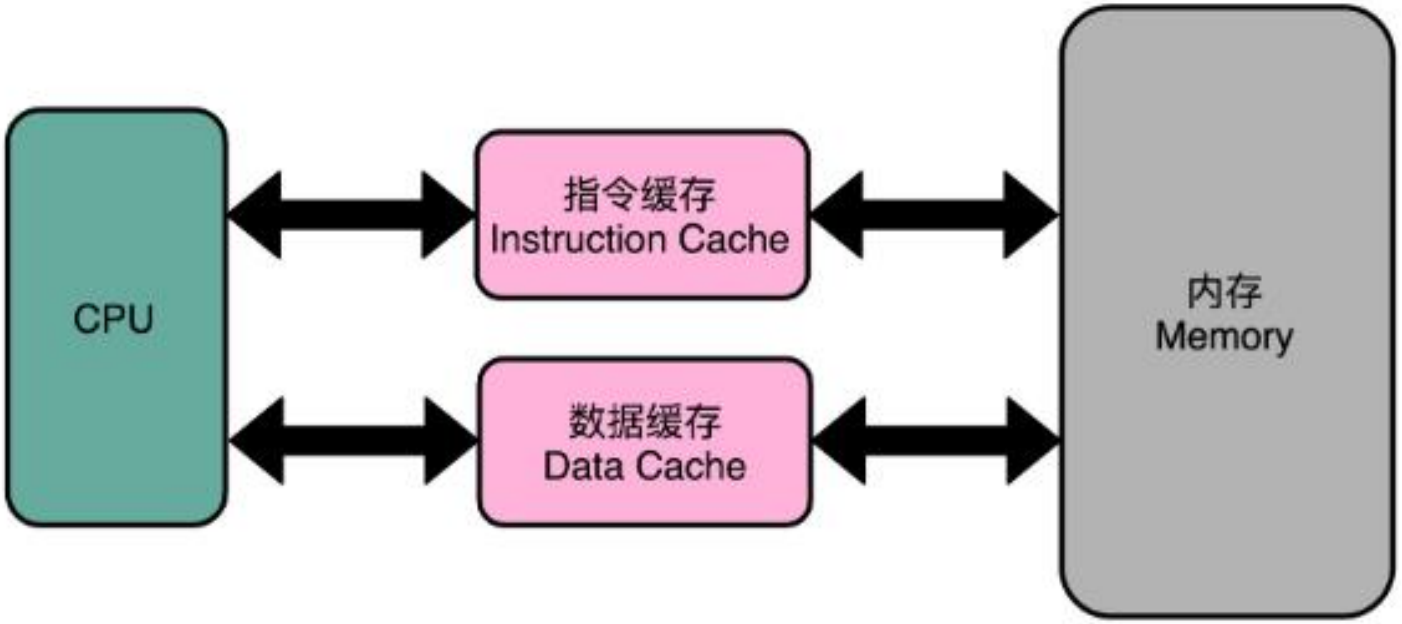
冯·诺伊曼架构/普林斯顿架构
Princeton Archicture



哈佛架构
Harvard Architecture

结构冒险

- 现代的CPU把高速缓存分成了指令缓存（Instruction Cache）和数据缓存（Data Cache）两部分



数据冒险

- 前后指令存在数据依赖，有三种情况
  - 写后读 Read after Write

```
a = b + c;   d = a + e;
```

  - 读后写 Write after Read

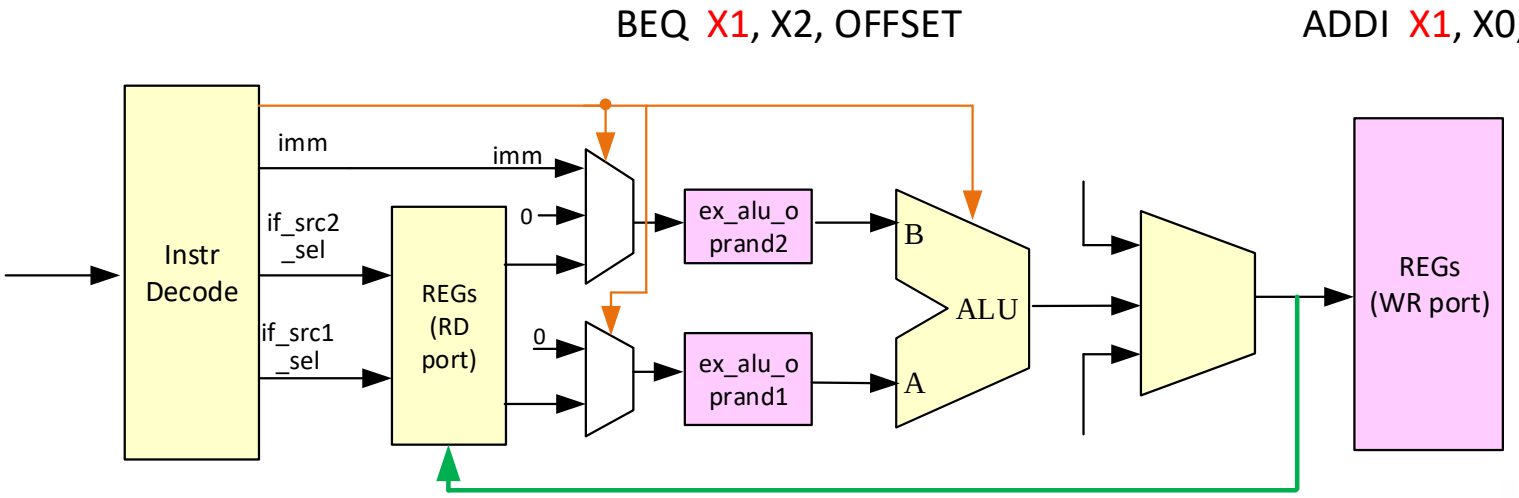
```
a = b + c;   b = d + e;
```

  - 写后再写 Write after Write

```
a = b + c;   a = d + e;
```

数据冒险

- 单线程流水线中可能存在的“写后读”顺序不满足的问题，造成数据冒险（Data Hazard）
- 解决方法：（1）停止(stalling)；（2）旁路(Forwarding)；




```
assign reg_rdata1  =  (if_src1_sel == 5'h0) ? 32'h0 :
                      ((ex_reg_update && (if_src1_sel == ex_reg_destsel))?
                       ex_reg_wdata : regs[if_src1_sel]);
```

控制冒险

- 由转移指令引起，在执行跳转指令时，后继的指令进入了流水线，导致不期望的指令被执行，称为控制冒险（Control Hazards）。
- 主要解决策略就是流水线停顿(stalling)。

	T1	T2	T3	T4	T5
取值/译码	Instr 1	NOP	NOP	Instr K	Instr K+1
执行		Instr 1	NOP	NOP	Instr K
回写			Instr 1 (跳转)	NOP	NOP

目录

- 处理器的冒险及解决方法
- GNU汇编器使用
- 总线设计
- 拓展实验（2）

GNU工具链

- 包含RISC-V汇编、C/C++编译器，GDB调试工具，libc库等等

- 可从教辅系统下载windows版本的RISC-V交叉编译工具链，或者从以下链接获取最新版本

  https://xpack.github.io/dev-tools/riscv-none-elf-gcc/

GNU工具链

- 在Windows下，解压压缩包后，设置系统环境变量Path指向 riscv-none-gcc-8.2.0\bin 目录
- Linux/MacOS等安装方法参见

  https://xpack.github.io/dev-tools/riscv-none-elf-gcc/install/
- 在终端中以下命令，应能出现汇编器的版本信息

```
riscv-none-embed-as --version
```

汇编器使用

- 一个汇编源代码文件 src.s，通过以下命令编译成ELF (Executable and Linking Format)格式文件

```
riscv-none-embed-as  -o app src.s
```

- 通过objcopy工具提取ELF文件中的程序指令部分

```
riscv-none-embed-objcopy -O binary app app.bin
```

- 通过小工具将二进制指令转成十六进制文本

```
bin2hex app.bin app.txt
```

汇编编程

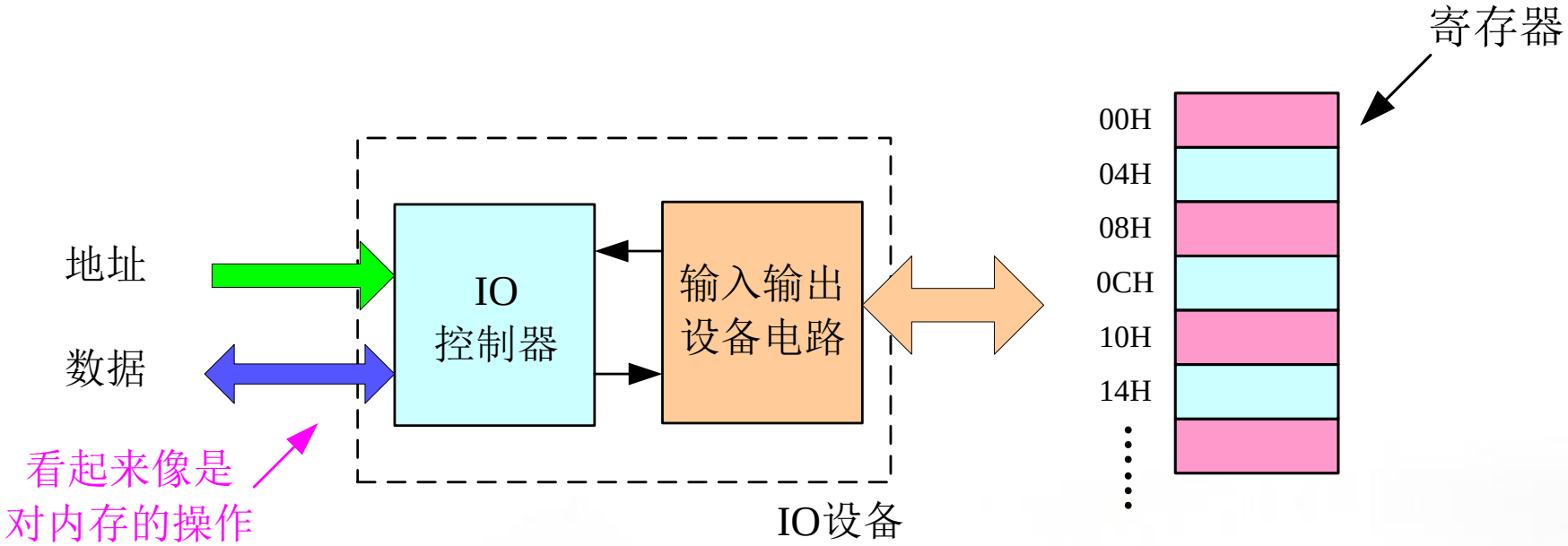
有关RISC-V汇编语言的知识可参照网络资料

- RISC-V Assembly Programmming
  https://www.robertwinkler.com/projects/riscv_book/riscv_book.pdf
- RISC-V Assembly Programmer's Manual
  https://github.com/riscv-non-isa/riscv-asm-manual/blob/main/riscv-asm.md
- 知乎两篇文章
  https://zhuanlan.zhihu.com/p/502146080
  https://zhuanlan.zhihu.com/p/540887151

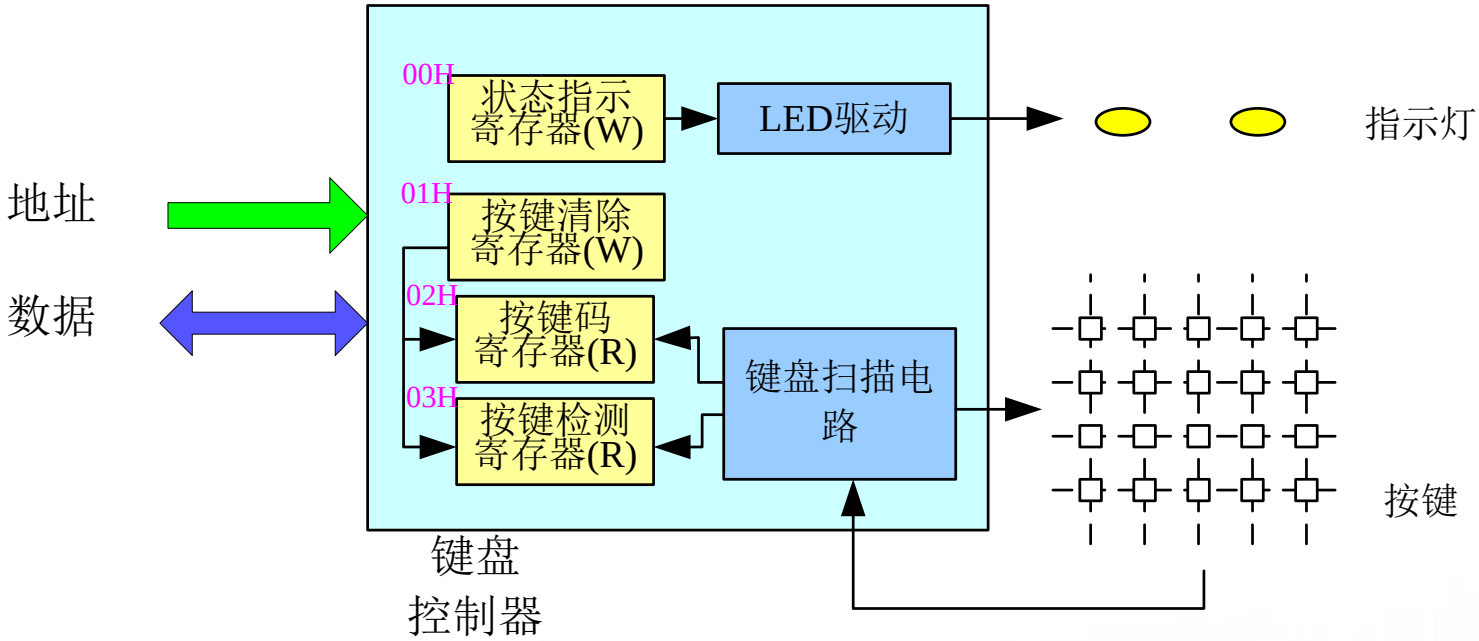
目录

外设

- CPU来仅能够通过以下方式访问外设
  - 从某个地址读取一个数/写入一个数
- 因而需要一个“转换器”

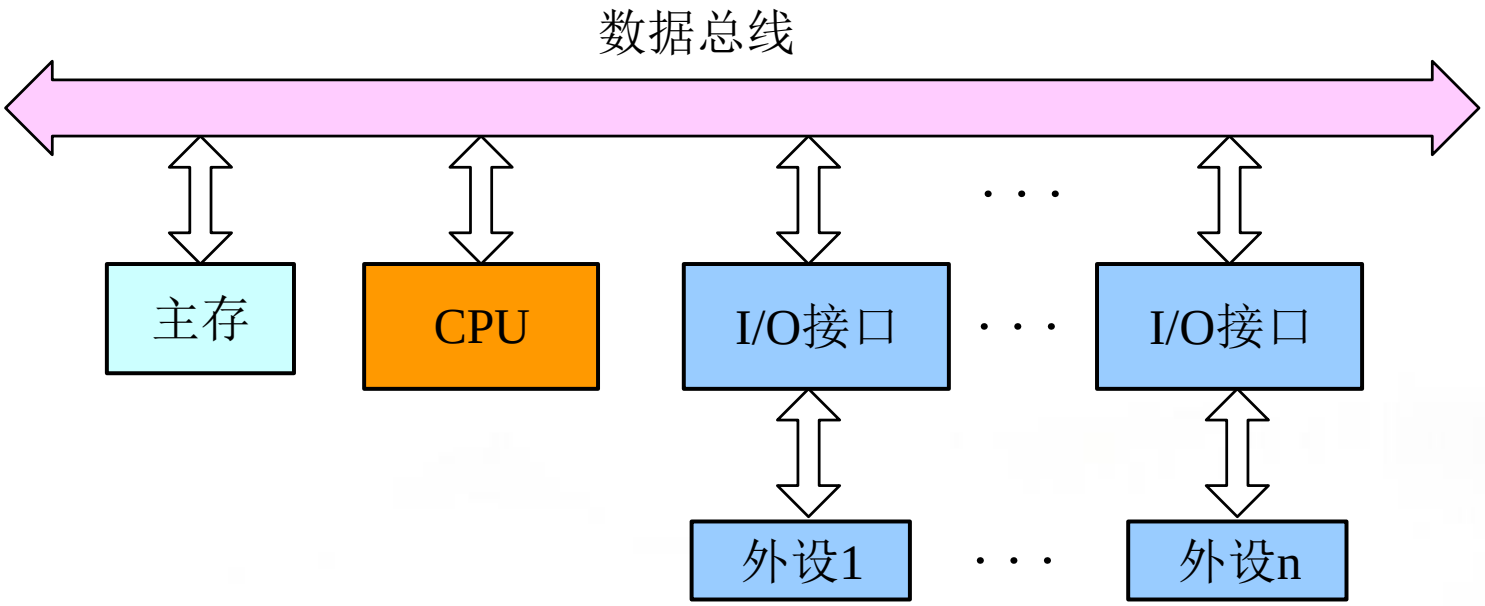


- 例子：简单的键盘


00H
状态指示寄存器(W)
LED驱动
指示灯
地址
01H
按键清除寄存器(W)
数据
02H
按键码寄存器(R)
键盘扫描电路
03H
按键检测寄存器(R)
按键
键盘控制器

数据总线

- CPU仅有1个数据读写端口，但是外设的数量可能很多，因此需要“数据总线”来实现CPU与多个外设的互联。
- CPU是总线操作的发起者，称为“主设备”，而其他部件为“从设备”。

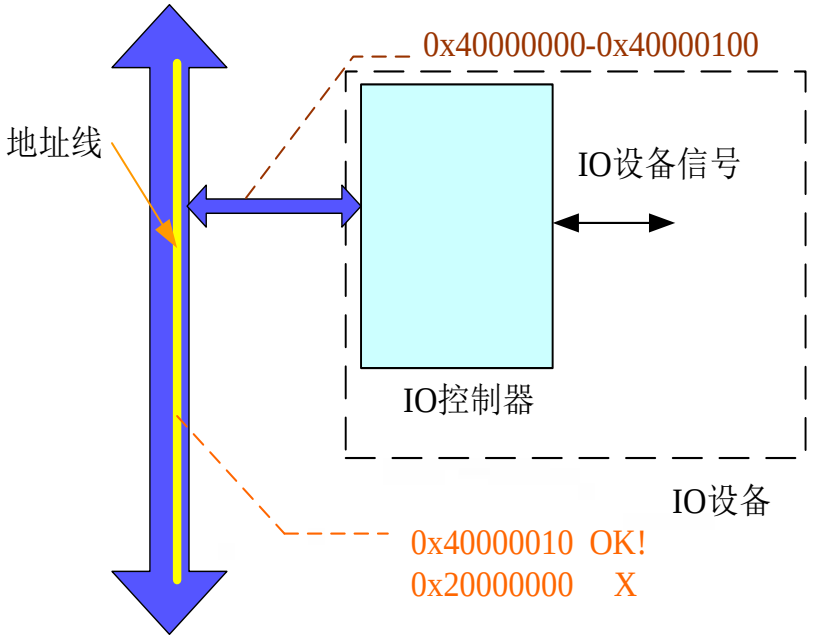


数据总线

- CPU数据总线的基本信号

  ADDR(WADDR/RADDR), WE, RD, WDATA, RDATA

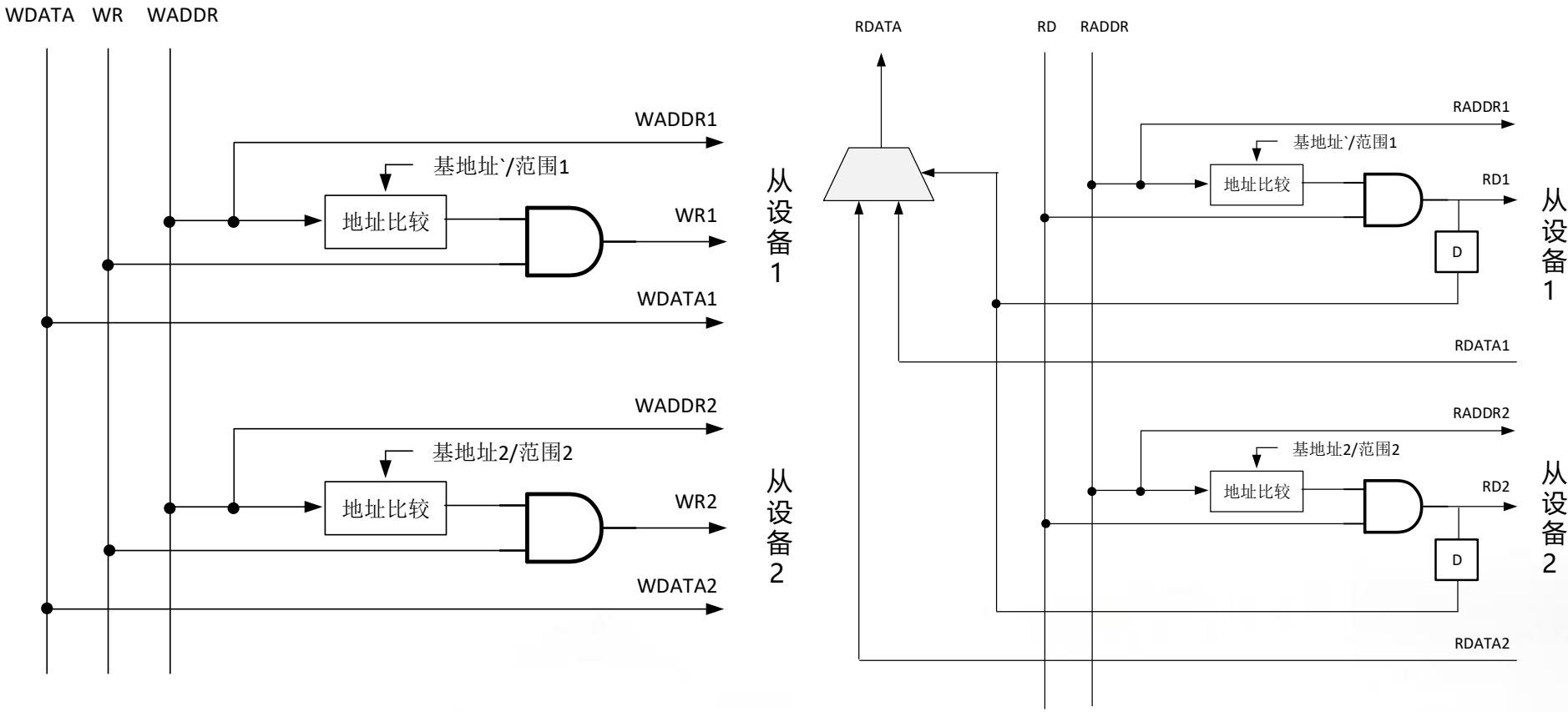
- 如何区分总线上的不同设备？为每个设备分配固定的地址空间。



考察总线地址是否落在Base Address至Base Address + Size中

数据总线

- 硬件实现结构



具体HDL实现参考示例代码

目录

- 处理器指令集
- 处理器体系设计与实现
- 软件编程
- 拓展实验

拓展项目

- 实现一个倒计数秒表，具体功能如下
  - 用SW0开关作为秒表的控制，即当SW0处于上拨状态，数码管显示20，代表20秒；
  - 当SW0下拨后，数码管开始倒计数，当倒计数至5秒时，蜂鸣器开始发出声音，直至0秒停止；
  - 重新上拨开关SW0，复位秒表；

Question ?