

第八章 中央处理器

第九周--课后思考：第八章 中央控制器（1）

本章思路：设计能执行 8.1 节设定的 8 条指令对应的系统，一旦设计完成，由这 8 条指令构成的任何程序（不限长度）理论上都能在该系统上运行了。

- 1、CPU 按功能模块划分有哪些模块？从设计角度侧重完成哪些部分实现（本章重点）？
- 2、CPU 主要完成哪四个基本功能？（体会控制器 CU 的重要性）
- 3、理解 CPU 指令执行中各部件和数据通路（datapath）的关系，其中各部件需要的操作控制信号的作用；本章的两个重点：**数据通路和控制信号的设计**。
- 4、理解指令执行的三个基本步骤：取指、译码和执行（可以再细分），前两步不同指令处理基本相同，差异主要在执行环节；
- 5、本教材模型机的设计需求：模型机的机器字长（注意区别 7.7MIPS 实例）？有哪几类指令？分别对应哪几条指令？它们采用哪些指令封装？操作码分别是多少？针对不同指令，各种指令封装中各字段的含义是什么？**这部分很重要，牵扯到数据通路的选择和控制等。**

运算 指令	add rd, rs, rt sub rd, rs, rt and rd, rs, rt or rd, rs, rt slt rd, rs, rt		
访存 指令		lw rt, imm16(rs) sw rt, imm16(rs)	
分支 指令		beqz rs, rt, imm16	
	R型指令	I型指令	J型指令

模型机中：
运算类指令只用了R类封装（对MIPS进行了简化）
访存类指令用I类封装（和MIPS一致）
程序控制类指令只用了I类封装（对MIPS进行了简化）

- 6、理解时序电路中触发的概念，理解时钟在时序电路中的重要性，没有时序电路，计算机就没办法“自动”执行指令了，它是存储程序计算的根本。
- 7、理解高电平有效还是低电平有效，上升沿触发还是下降沿触发是电路的设计的约定，一旦设计确定下来就按照该约定使用控制信号即可。
- 8、什么是传输时间？为什么传输周期 T 要去取那个最大值（快的等慢的），时钟周期的设定依据（不是想多快就多快的）？
- 9、什么情况下可以用时钟作控制？什么时候要设计独立的控制信号？
- 10、如何理解 CPU 的数据通路（结合城市间道路设计类比）？总线式和直连式的优缺点是什么？

理解我们采用直连方式的初衷。

11、明白书上图 8.5 上下两排分别是时序器件和组合器件，上面器件的时钟信号都省略了，我们约定都是时钟上升沿触发。我们的课程背景，图中 IM 和 DM 是分开的（哈佛结构），其他部件在 CPU 内部，注意 IM 是只读的，即对 IM 只有读操作，没有写操作。

12、熟悉图 8.5 中各部件的输入、输出及控制信号或时钟信号的关系。（接下来我们会用他们实现模型机）--重点

13、为什么图 8.5 中的通用寄存器组没用类似 DM 的 RegRead 读控制信号？

14、为什么 DM 的读写不能同时进行，而通用寄存器组的读写可以同时进行？

15、不看教材能把各指令的取指、译码和执行的数据通路分别画出来。在本节 8.4.1 学习中淡化了控制信号的作用，强调的是数据和地址的联通，单周期控制信号相对简单，很多就直接用时钟信号了，等学了多周期控制信号的设计后和单周期这部分可以做比较。（数据通路的实现是难点）

第 10 周--课后思考：第八章 中央控制器（2）

1、模型机的指令集设计中，运算类指令支持不支持寄存器和立即数的运算？如果要支持，需要做什么改动（硬件层面的）？

2、理解 CPU 指令执行中各部件和数据通路的关系，其中各部件需要的操作控制信号的作用（区别于地址线、数据线），各指令取指和译码流程都类似，但不同指令译码后的得到的结果（控制信号）和后续的执行细节不同；

3、在 beqz 指令的数据通路中为什么设计一个左移两位的处理？

注意我们的前提，内存是字节编址（一个字节内存单元分配一个内存地址），指令在 IM 中是 4 字节对齐（字对齐）存放的，为了保证生成的新目标地址是 4 字节对齐的，所以要进行左移两位（保证 PC 寄存器的后两位始终为 0）。

4、理解图 8.11 的数据通路整合过程，其中几个多路选择器的作用。在此基础上理解图 8.12（我们第一个较完整的数据通路设计完成）的整合过程，注意该图中 9 路（11bits）控制信号的作用。

5、从两次整合过程理解，理解 CPU 每增加一种指令，甚至是一种指令封装（如模型机中要支持寄存器和立即数的运算）的实现都会增加 CPU 的设计复杂度和成本。

6、图 8.12 是整合后的数据通路，上、下两排的控制信号是核心，理解这些控制信号在不同指令执行时的作用，用于生成这些控制信号的部件就是控制单元（CU），它是 CPU 的控制中心。

7、理解为什么我们要采用二级译码来实现控制器，控制信号主控制器和 ALU 控制器的作用，它们的输入输出信号都有哪些？

- 8、为什么需要 ALUOp？此处的 ALUOp 可不可以改为 1 位？体会指令译码的过程，最主要的作用就是生成图中的小辫子信号。
- 9、图 8.13 是在图 8.12 数据通路设计基础上增加了控制器设计后更完善的一版---单周期数据通路设计（单周期的本质是所有指令在一个时钟周期内完成，那就又存在快的等慢的问题，此处的时间粒度是什么？如何改进以弥补单周期的不足？）。能将我们设计的三类功能、两种封装、8 条指令的执行过程在图 8.13 中的执行过程走一遍，注意各自经历的数据通路、组件和需要的控制信号、状态描述清楚。（不光能看懂，要求会设计）--- 考点!!!
- 10、学到这里，再去体会 RISC 处理器为什么一般体积小、价格和功耗低？当然它也有缺点，是什么？
- 11、了解主控制器和运算控制器设计思路，着重关心他们的输出（也就是各控制信号）在数据通路中的作用。
- 12、为什么要采用多周期实现？它是怎么提高 CPU 的指令执行效率的？
- 13、在多周期背景下，指令平均执行周期数 CPI 引出的意义是什么？
- 14、理解多周期实现机制后，思考 CPU 的主频提升依据是什么，主频仍然存在快的等慢的，这里时间粒度是什么？
- 15、理解系统性能和 ISA 设计、数据通路实现、主频大小等有关，这些又相互制约。

第 11 周--课后思考：第八章 中央控制器（3）

1. 比较单周期数据通路图 8.13 和多周期实现图 8.14 的差异，增加了哪些临时寄存器和控制信号？
2. 在提高主频上，理解由“单周期”→“多周期”的设计思路，在多周期下“快的等慢的”依然存在，它的粒度单位是什么？理解“微操作”的原子性特点，微操作是针对多周期数据通路而言的，单周期数据通路中一条指令就是一个不可分割的基本操作（相当于一条指令本身具有原子性）。
3. 多周期实现解决了单周期相应问题的同时，增加了哪些成本？周期的划分要考虑什么因素？
4. 多周期实现中为什么增加了 IRWrite 和 PCWrite 两个控制信号？
5. 结合表 8.6（重点）和图 8.14，对各指令的多周期的执行过程所经历过的数据通路、组件和需要的控制信号状态能描述清楚。（**重要考点**）

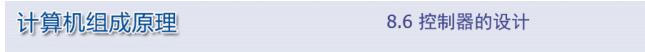
思考题 1：为什么译码环节不区分指令，R 类封装为什么也截取 imm 字段？体会 RISC 固定字段译码的好处。

思考题 2：Load 的写回能不能整合到上一个节拍（理由：这样所有指令就都只需要 4 个节拍，PCWrite 控制就可以用主频时钟 4 分频的时钟信号直接控制了）？

思考题 3: Load 指令的 PCwrite 写有效能不能放到写回节拍去实现。(理由: 其他指令都是在最后一个节拍实现的)

6. 什么是时钟周期、机器周期、和指令周期? 理解计算机系统的时序系统设计。

(说明: 教材这部分描述存在问题, 对教材这部分做了修正。)



2. 时序系统

- 时钟 (节拍) 和其电位变化构成了计算机的时序系统的基础。为指令的执行提供各种定时信号。
- **指令周期:** 从取指令、分析指令到执行完该指令所需的全部时间。
 - 指令周期一般由若干个机器周期组成
- **机器周期** (又叫CPU周期)
 - 把指令周期划分为若干大小相等 (定长) 或大小不等 (变长) 的时间段, 每个时间段对应一个机器周期, 如取指、译码、执行等
 - 机器周期一般由若干个时钟周期 (时钟节拍) 组成
- **时钟周期** (也称脉冲、节拍、T周期或CPU时钟周期): 用于寄存器的复位或打入脉冲等, 是CPU操作的最基本单位。
 - 一个机器周期需要完成一序列的微操作, 每个微操作都必须在一个节拍内完成, 如各指令的不同机器周期由若干不同微操作完成
 - 这些微操作必须按一定的先后次序进行。

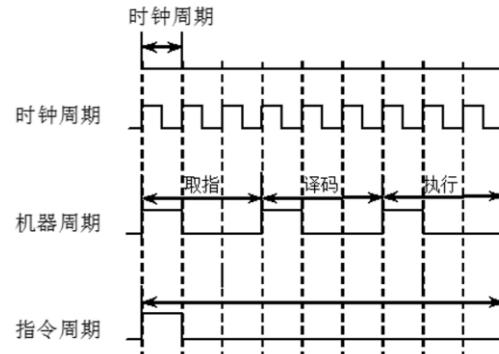


图 8.16 3 级时序系统。

7. 结合图 8.15 理解控制器工作原理, 它的输入有哪些信号? (如程序状态寄存器 PSR 中的程序状态字 PSW、中断信号、译码结果、时序信号等) 输出有哪些信号? (微操作控制信号)
8. 比较多周期下的控制信号 (P213) 与单周期下的控制信号 (P201) 有什么不同? (本质区别: 一个是时序逻辑, 一个是组合逻辑), 为了好理解对图 8.22 进行了修改:

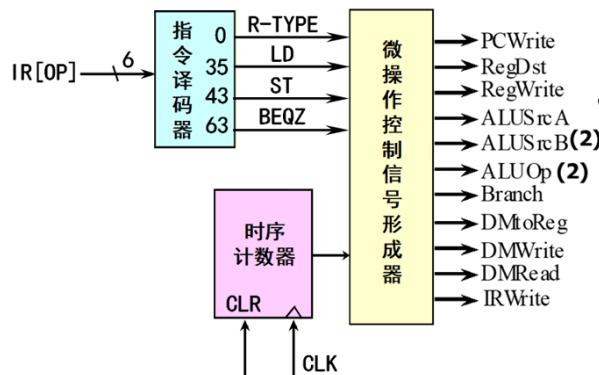


图8.22 模型机的硬连逻辑控制器

9. 什么是同步和异步时序控制? 比较优缺点, CPU 内部为什么采用同步控制?
10. 理解硬连接方式实现控制器, 本质就是通过实现输入 (指令操作码字段) 和输出 (微操作控制信号) 的组合和时序电路的实现。
11. 流水的数据通路本学期不要求掌握, 会在下学期系统结构学习细节, 但要理解“流水一定是多周期的, 多周期不一定是流水的”(理解黑板上画的从单周期到多周期到流水示意图, 理解指令在三种场景下执行的差异), 流水机制本质是“指令级并行”。体会多周期是为了降低系统时钟提出的方案, 流水线是为了实现指令并行从而降低 CPI 提出的方案, 现阶段主流处理器基本都是支持流

水执行的，所以大多数资料把多周期和流水数据通路画等号了。

第 8 章总作业： 8.1、8.4、8.5、8.8（提交时间待定）