Homework 3

PB22051080 王珏

2024 年 3 月 22 日

目录

1 数据通路和 $R-type$ 指令

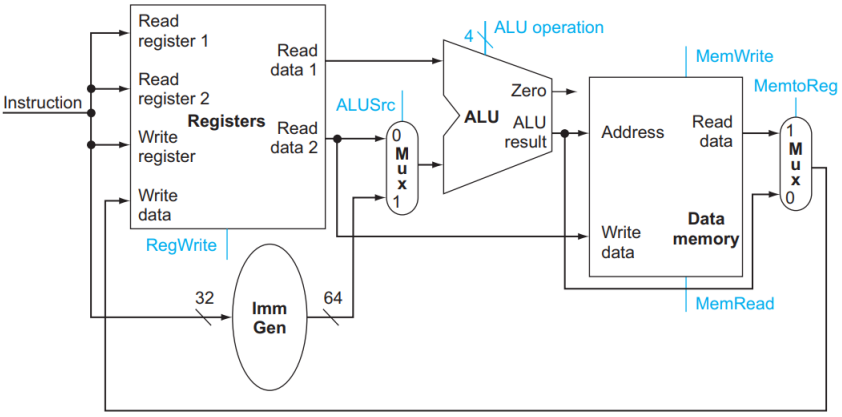

FIGURE 4.10 The datapath for the memory instructions and the R-type instructions. This example shows how a single datapath can be assembled from the pieces in Figures 4.7 and 4.8 by adding multiplexors. Two multiplexors are needed, as described in the example.

图 1: 数据通路

1.1 具体分析

4.1 考虑如下指令：

指令：`and rd, rs1, rs2`

解释：`Reg[rd] = Reg[rs1] AND Reg[rs2]`

4.1.1 [5] <4.3> 对于上述指令，图 4-10 中的控制信号各是什么数值？

4.1.2 [5] <4.3> 对于上述指令，将用到哪些功能单元？

4.1.3 [10] <4.3> 对于上述指令，哪些功能单元不产生任何输出？哪些功能单元的输出不会被用到？

图 2: 具体题目

- 1. 上述指令涉及到向寄存器内写入的操作，因此 `RegWrite` 需要在时钟边沿有效。`ALUoperation` 指示该指令为 `and` 操作。`ALUSrc` 的值为 0,`MemtoReg` 的值为 0，意味着 ALU 的计算结果直接写入寄存器堆。

- 2. 用到寄存器堆和 ALU 两个功能单元。(不太清楚多路选择器算不算是一个功能单元，如果算，此处便加上**多路选择器**)
- 3.
  - 哪些功能单元不产生输出?
    - 立即数生成单元不会产生输出。
  - 哪些功能单元输出不会被用到?
    - ALU 产生的 `Zero` 信号仅用于条件跳转，该条指令不涉及这个过程，因此该信号用不到。

2 讨论数据通路各功能模块延迟

2.1 例题

4.7 假设用来实现处理器数据通路的各功能模块延迟如下所示：

I-Mem / D-Mem	Register File	Mux	ALU	Adder	Single gate	Register Read	Register Setup	Sign extend	Control
250ps	150ps	25ps	200ps	150ps	5ps	30ps	20ps	50ps	50ps

其中，寄存器读延迟指的是，时钟上升沿到寄存器输出端稳定输出新值所需的时间。该延迟仅针对 PC 寄存器。寄存器建立时间指的是，寄存器的输入数据稳定到时钟上升沿所需的时间。该数值针对 PC 寄存器和寄存器堆。

4.7.1 [5] <4.4> R 型指令的延迟是多少？比如，如果想让这类指令工作正确，时钟周期最少为多少?

4.7.2 [10] <4.4> `ld` 指令的延迟是多少？仔细检查你的答案，许多学生会在关键路径上添加额外的寄存器。

4.7.3 [10] <4.4> `sd` 指令的延迟是多少？仔细检查你的答案，许多学生会在关键路径上添加额外的寄存器。

4.7.4 [5] <4.4> `beq` 指令的延迟是多少?

4.7.5 [5] <4.4> I 型指令的延迟是多少?

4.7.6 [5] <4.4> 该 CPU 的最小时钟周期是多少?

图 3: 数据通路延迟

2.2 计算

以下图为参考

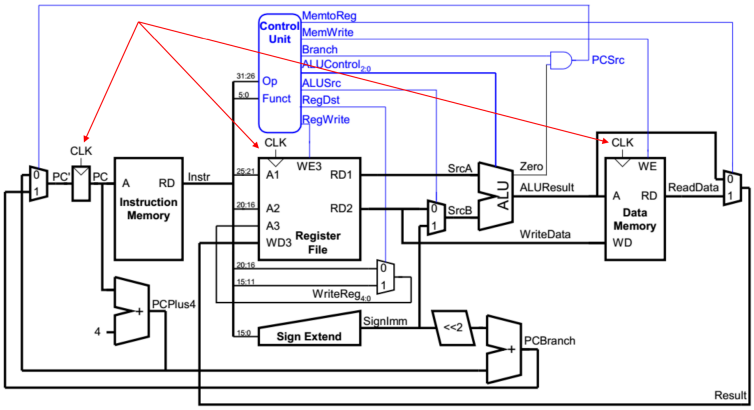

图 4: 单周期处理器

2.2.1 R 型指令

$$
\begin{aligned}
T_R &= t_{\text{Reg-read}} + t_{\text{I-Mem}} + t_{\text{RegFile}} + t_{\text{ALU}} \\
&\quad + t_{\text{Mux}} + t_{\text{Reg-setup}} \\
&= (30 + 250 + 150 + 200 + 25 + 20)\ \text{ps} \\
&= 675\ \text{ps}
\end{aligned}
$$

时钟周期最少为 250ps。

2.2.2 ld 型指令

$$\begin{aligned} T_{\text{ld}} &= t_{\text{Reg-read}} + t_{\text{I-Mem}} + t_{\text{RegFile}} + t_{\text{sign-extend}} \\ &\quad + t_{\text{ALU}} + t_{\text{D-Mem}} + t_{\text{Mux}} + t_{\text{Reg-setup}} \\ &= (30 + 250 + 150 + 50 + 200 + 250 + 25 + 20)\ \text{ps} \\ &= 975\ \text{ps} \end{aligned}$$

2.2.3 sd 型指令

$$\begin{aligned} T_{\text{sd}} &= t_{\text{Reg-read}} + t_{\text{I-Mem}} + t_{\text{RegFile}} + t_{\text{ALU}} + t_{\text{D-Mem}} + t_{\text{Reg-setup}} \\ &= (30 + 250 + 150 + 200 + 250 + 20)\ \text{ps} \\ &= 900\ \text{ps} \end{aligned}$$

2.2.4 beq 型指令

$$\begin{aligned} T_{\text{beq}} &= t_{\text{Reg-read}} + t_{\text{I-Mem}} + t_{\text{control}} + t_{\text{gate}} + t_{\text{Adder}} + t_{\text{Reg-setup}} \\ &= (30 + 250 + 50 + 5 + 150 + 20)\ \text{ps} \\ &= 505\ \text{ps} \end{aligned}$$

2.2.5 I 型指令

$$\begin{aligned} T_{R} &= t_{\text{Reg-read}} + t_{\text{I-Mem}} + t_{\text{RegFile}} + t_{\text{ALU}} + t_{\text{Mux}} + t_{\text{Reg-setup}} \\ &= (30 + 250 + 150 + 200 + 25 + 20)\ \text{ps} \\ &= 675\ \text{ps} \end{aligned}$$

注：符号扩展的时间比读取寄存器操作数时间短。

2.2.6 CPU 的最小时钟周期

应该取几个指令的时间延迟的最大值，即 $t_{min} = 975ps$

3 主控制器的 PLA 实现图

主控制器真值表

Input or output	Signal name	R-format	ld	sd	beq
Inputs	I[6]	0	0	0	1
	I[5]	1	0	1	1
	I[4]	1	0	0	0
	I[3]	0	0	0	0
	I[2]	0	0	0	0
	I[1]	1	1	1	1
	I[0]	1	1	1	1
Outputs	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

图 5: 主控制器的真值表

3.1 根据图控制真值表，画出主控制器的 PLA 实现图

3.1.1 先写布尔表达式

3.1.2 参考英文版教材附录 C

3.1.3 由于 PLA 规模不小，可以手绘，也可以用软件画完后打印出来

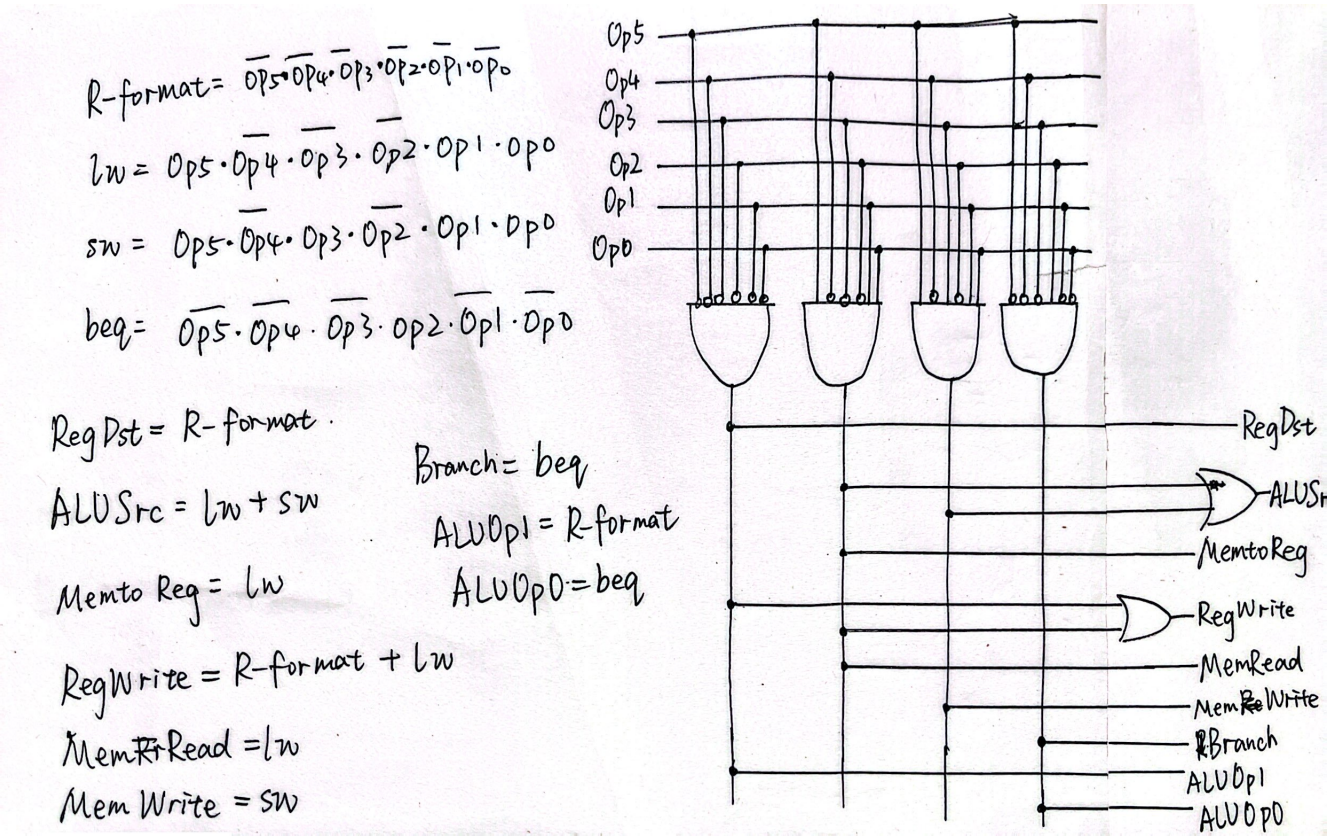

图 6: 主控制器的 PLA 实现图

4 Thinking

4.1 本书有哪些逻辑设计惯例 (约定)？

–功能部件，时钟方法

4.2 单周期处理器在一个周期内完成指令所有的微操作的思考

对于单周期处理器：

1. **寻址方式如何实现：**
   在单周期处理器中，寻址通常是通过地址总线实现的。根据指令中的地址字段，CPU 将地址发送到存储器中，然后接收存储器返回的数据。每个周期都能完成一次完整的寻址操作。

2. **周期宽度如何确定：**
   周期宽度由最慢的指令决定。周期宽度要足够长，才能确保需要执行时间最长的指令有足够时间执行。

3. **能否 “在一个 clk 内完成”：**
   单周期处理器的定义就是在一个时钟周期内完成一条指令的所有微操作。每个阶段（如取指、译码、执行等）必须在一个时钟周期内完成。

4. **能否将两个加法器合二为一：**
   在单周期处理器中一般不可以，多周期可行是因为其中一个 `adder` 可以在 `ALU` 中进行运算。

5. **能否将两个存储器合二为一：**
   在单周期处理器中，存储器通常是分开的指令存储器和数据存储器。两个存储器的读和写都应该独立进行，分别有各自的控制信号，因此不能，或者说，笔者不建议。多周期反而可以考虑合并以复用。

4.3 控制逻辑有哪些实现方式？

1. 组合逻辑控制：

这种方式通过组合逻辑电路来实现控制逻辑。组合逻辑控制器根据当前指令的操作码以及处理器的状态来产生控制信号，直接控制数据通路的各个部件。但对于复杂的指令集架构可能会导致逻辑电路过于庞大。

2. 状态机控制：

控制逻辑可以通过状态机来实现。状态机可以根据当前状态和输入信号来确定下一个状态以及需要执行的动作。状态机控制器通常用于需要处理时序逻辑的情况，例如时序电路或者时钟驱动的系统。