

### Hw3 单周期

4.1.1 Reg Write = 1 ALUSrc = 0 ALU operation : "AND" = 0010.

Mem Write = 0, Mem Read = 0, Mem to Reg = 0, PCSrc = 0

4.1.2 寄存器 Register, ALUSrc Mux, ALU, Mem to Reg Mux.

4.1.3 所有的功能单元都会有输出

但由于控制信号：ImmGen 生成的立即数，PC 与偏移量 ADD. 算出的 Sum  
Data Memory 读到的输出不会被用到。

4.7.1 R类 op rd rs1 rs2

$$30(\text{PC read}) + 250(\text{I-M}) + 150(\text{Reg File}) + 25(\text{mux}) + 200(\text{ALU}) + 25(\text{mux}) + 20(\text{Setup}) = 700 \text{ ps.}$$

这里 Adder: PC + I-M. 这里 Control, reg-read, Imm-Gen 是并行的, 取时间较长的一部分. 这里也有 PC mux + pc setup = 25+20.

4.7.2 ld rd offset(rs1).

$$30(\text{PC read}) + 250(\text{I-M}) + 150(\text{reg File}) + 25(\text{mux}) + 200(\text{ALU}) + 25(\text{D-Mem}) + 25(\text{mux}) + 20(\text{Setup}) = 950 \text{ ps.}$$

这里 to D-Mem.

4.7.3. sd. rs2 offset(rs1).

$$30(\text{PC read}) + 250(\text{I-M}) + 150(\text{reg File}) + 25(\text{mux}) + 200(\text{ALU}) + 250(\text{D-Mem}) = 905 \text{ ps.}$$

4.7.4 beq. rs<sub>2</sub>, rs, label.

$$30(\text{PC read}) + 250(\text{I-M}) + 150(\text{Reg File}) + 25(\text{mux}) + 200(\text{ALU}) + 5(\text{Single Gate}) + 25(\text{mux}) + 20(\text{PC Setup})$$

这里 Adder: PC + extend 是并行的. = 705 ps.

4.7.5. I类. 如果是: addi rd, rs1, Imm.

$$30(\text{PC}) + 250(\text{I-M}) + 150(\text{Reg File}) + 25(\text{mux}) + 200(\text{ALU}) + 25(\text{mux}) + 20(\text{Setup}) = 700 \text{ ps}$$

如果是 Id. Lw 类, 则是 950 ps

4.7.6 最小时钟周期指最长指令由记为 950 ps

## 思考题：

### 1. 寻址实现方式

① PC相对寻址：则有通过  $PC+4$  和  $PC+立即数的偏移量$  的偏移寻址，主要是通过 Addr+Mux 来实现。

② 基址寻址：lw, sw 指令：是通过读寄存器得基址+立即数偏移量实现内存寻址。

另外还有 jalr 类型也利用基址寻址改变 PC 值。

③ 寄存器寻址：直接利用寄存器存储的地址在 register\_file 中寻值。如 add.

④ 立即数寻址：操作数本身就是常量，直接获得值。

### 2. 周期宽度确定

寻找关键路径，即找执行时间最长的指令，由该时间为周期宽度

3. 单周期可以在一个 CLK 内完成所有指令，只要  $CLK \geq$  周期宽度即可。

4. 在 Adder 之前增加 MUX 多路选择器，根据指令选择  $PC+4$  or  $PC+Imm$ .

即可把 2 个 Adder 合并

5. 不可以把两个 Memory 合并，因为存储器不能同时提供数据和指令。还要考虑外设 I/O。  
如果不考虑外设的 I/O，且 Memory 中可以提供两个读口，一个写口，且实现先读操作。  
在存储器初始化条件下 指令只在机器内运行，实现了两者合并，不能在外设中显示。缺陷太多！！

## 多周期

### 1. 每一类指令含有时钟周期数不同

① R 型： IF + ID + EX + WB                  4 个

② I 型 (lw)    IF + ID + EX + MEM + WB                  5 个

③ S 型 (sw)    IF + ID + EX + MEM                  4 个

④ SB 型 (bq)    IF + ID + EX                  3 个

⑤ U 型    : IF + ID + WB.                  3 个

⑥ UJ 型    : IF + ID + WB                  3 个。

2. R-Type: IF: 从 MEM 中取指令, 存在中间寄存器 PC+4,

ID: 从 Reg 中读值 rs1, rs2

EX: 从 ALU 中运算结果: rs1 op rs2

WB: 将结果送回 Reg 中 rd

I-type IF: MEM 取指, PC+4.

ID: 从 Reg 取值, 生成立即数 imm.

EX: ALU 计算 rs1 + imm.

MEM: 从 Mem 读数据

WB: 向 Reg 中写回 rd

S-type: IF: 从 MEM 取指, PC+4

ID: 从 Reg 取值 rs1, rs2, 生成立即数 imm.

EX: ALU 计算 rs1 + imm. = ALUout

MEM: 把 ALUout 存到 Mem 中.

B-Type: IF: 从 MEM 取指, PC+4

ID: 从 Reg 取值 rs1, rs2; 同时 ALU 计算 PC + (exten-imm).

EX: 计算比较 rs1, rs2, 以控制 PC 是否跳转