Homework3

张子康 PB22020660

2024 年 04 月 15 日

1

1.1

单精度浮点数：符号位：1 位, 阶码：8 位, 尾数：22 位。
双精度浮点数：符号位：1 位, 阶码：11 位, 尾数：51 位。

1.2

单精度浮点数：0.2：cdcc4c3e，0.7：3333333f，0.9：6666663f，0.2 + 0.7 的结果：6666663f。

双精度浮点数：0.2：9a9999999999c93f, 0.7：666666666666e63f, 0.9：cdcccccccccccc3f，0.2 + 0.7 的结果：ccccccccccccec3f。

在计算机中小数无法精确表示,0.2,0.7 和 0.9 在单精度下恰好 0.2+0.7=0.9，而在双精度下不满足。

2

2.1

2.1.1

1. PC 元件：给出指令所在的地址；
2. 指令存储器：根据 PC 的值取出指令；
3. 控制器：生成控制信号，包括 ALU 的 opcode，寄存器写使能信号，控制输入 ALU 单元的数据，控制写回寄存器的数据来源；
4. 寄存器：根据给出的地址取出操作数，向制定的地址写入结果；
5. MUX：控制输入 ALU 单元的数据时来自寄存器还是立即数，控制写回寄存器的数据来源，控制下一个 PC 的来源；
6. ALU：进行加法运算并输出结果；

7. ALU 控制器：产生 ALU 的 opcode；

8. ImmGen：根据指令对立即数做符号扩展，产生操作数。

2.1.2

1. PC 元件：给出指令所在的地址；

2. 指令存储器：根据 PC 的值取出指令；

3. 控制器：生成控制信号，包括 ALU 的 opcode，寄存器写使能信号，控制输入 ALU 单元的数据，控制写回寄存器的数据来源；

4. 寄存器：根据给出的地址取出操作数，向制定的地址写入结果；

5. MUX：控制输入 ALU 单元的数据时来自寄存器还是立即数，控制写回寄存器的数据来源；

6. ALU：进行比较运算并输出结果；

7. ImmGen：根据指令对立即数做符号扩展，产生操作数。

8. Add：计算需要跳转到的 pc；

9. MUX：控制输入 ALU 单元的数据时来自寄存器还是立即数，控制写回寄存器的数据来源，控制下一个 PC 的来源；

2.2

2.2.1

pc+4 无法写回寄存器

2.2.2

将 pc+4 的结果连接到控制写回到寄存器数据的 MUX 上，同时将其改为 4 选 1 的数据选择器。

2.2.3

没有改变（应该吧）

3

3.1

1. (假设是 A-B)，对 B 取反与 A 相加，再将结果 +1；
2. 设计算结果为 C，需要将 A 右移 B 位。先计算'd32-B 的结果存储在 B 中 (减法计算见上)，C 左移一位，然后加上 A+32'h80000000 的进位，将 A 左移一位，同时 B-1。重复上述运算直到 B==0。

3.2

设该操作数为 src，则操作为 src+0

4

4.1

Loongarch32 中 opcode 和 function 部分是连在一起的且 opcode 不是定长（相当于每个 opcode 对应一个运算），但是在 RISCV 中是分开的且 opcode 定长。

在译码器设计时，Loongarch32 的每个 opcode 对应一种运算，在 RISCV 中除了要处理 opcode，还要根据 func 段来判断具体运算。

4.2

RISC-V 中的 rs1，rs2，rd 位置固定，可以直接读取，译码和寄存器堆读取不存在依赖关系。但是在 Loongarch32 中对于每条指令需要具体判断，译码和寄存器堆读取存在依赖关系，可能会导致延迟增加。