COD WEEK7

1. T1

> 题目 1. 处理器！
>
> 在学习了单周期、多周期以及流水线处理器的设计之后，我们来系统分析一下这三种设计在性能上的差异。
>
> 假定访问 IM、DM 的延迟均是 3ns，ALU 的延迟为 2ns，寄存器堆的读写延迟均为 1ns，其他所有延迟忽略不计。某程序包含 1000 条指令，其中 lw 占 20%，sw 占 20%， R-type ALU 指令占 40%，beq 指令占 20%。
>
> 对于流水线处理器，50% 的 lw 指令之后紧跟 R-type ALU 指令，因此需要停顿一个时钟周期。此外，流水线对于 beq 的预测准确率为 70%，预测失败时要浪费两个时钟周期冲刷流水线。
>
> 考虑 RISC-V 32I 指令集架构的单周期 CPU、五段多周期 CPU 以及五级流水线 CPU， 根据以上信息回答下面的问题。

1.1 （1）

> 在单周期 CPU 上执行这 1000 条指令需要多长时间

单周期CPU周期：3ns+1ns+2ns+3ns+1ns = 10ns

所以总时间=$10ns \times 1000 = 10^4 ns$

1.2 （2）

> 对于多周期 CPU，其最小时钟周期是多少？在多周期 CPU 上执行这 1000 条指令需 要多长时间？

- 最小时钟周期为3ns
- lw指令3ns×5=15ns; s

sw指令没有wb阶段，3ns×4=12ns；

ALU指令没有mem阶段，3ns×4=12ns；

beq指令没有mem阶段和wb阶段，3ns×3=9ns；

所以平均为：15×0.2+12×0.2+12×0.4+9×0.2=12ns

所以总时间为$12ns \times 1000 = 12000ns$

1.3 （3）

> 对于流水线 CPU，其最小时钟周期是多少？在流水线 CPU 上执行这 1000 条指令需要多长时间？

- 最小时钟延迟为3ns；
- 不停顿时流水线CPU执行1000条指令需要1004个时钟周期；

  50% 的 lw 指令之后紧跟与之相关的 R-type ALU 指令 ：$50\% \times 1000 \times 20\% = 100$

  beq的预测准确率为70%：$30\% \times 1000 \times 20\% = 60$

  所以一共停顿100+2×60 = 220个时钟周期；
- 流水线CPU执行1000条指令需要1224个时钟周期，即$1224 \times 3ns = 3672ns$

1.4 （4）

> 结合以上内容，讨论影响流水线 CPU 性能的因素有哪些

- 数据冒险
- 控制冒险
- 各个阶段的最大延迟，决定了流水线CPU的最小时钟周期
- 分支预测准确率

2. T2

> 题目 2. 乘法！

> 某一天，神秘的外星生命体降临了 3C102 教室，在惊呆了全班同学后迅速离开，并留下了一份神秘的文件。经过翻译，这份文件竟是一份用 Verilog 编写的乘法运算模块！
>
> COD 助教们围着这个神秘的模块仔细研究。他们发现：这个模块可以时钟同步地花费两个周期（两次上升沿）计算 32bits 整数的有符号乘法（因为注释里是这么写的）。 助教们大喜过望，计划将其接入实验框架之中，满足同学们长期以来对于乘法指令的渴望。十分幸运地，正在写作业的你被助教拉来一起讨论相关事宜。我们的故事便从这里开始了…

2.1 （1）

> 两个 32bits 整数相乘，结果至多是多少位

64位

2.2 （2）

> 可以在单周期 CPU 中接入该模块，以实现乘法指令吗？请分析原因。

不可以，需要两个时钟上升沿，单周期CPU指令只能在一个周期内完成

> 助教们计划将该乘法模块接入流水线 CPU 之中。该模块在 EX 段接收到来自 ID/EX 段间寄存器的数据，经过两次上升沿的驱动，在 WB 段与 MEM/WB 段间寄存器同步输 出计算结果。注意：乘法模块同一时间只能进行一次乘法运算，其内部无法进行流水化 处理。 此外，助教们编写了如下的测试程序来检测乘法模块的工作情况：
>
> ```
> addi t1 x0 -1
> addi t2 x0 3
> mul t0 t1 t2 // - 1 × 3 = -3
> add t0 t0 t0 // - 6
> ```
>
> 请根据以上内容继续回答下面的问题。

2.3 （3）

> 当 mul 指令运行到 ID 段时，t1、t2 的正确结果位于什么位置（精确到某一段的连线或模块）

- t1：EX/MEM段间寄存器的alures_out输出连线
- t2：ALU输出连线

2.4 （4）

> 当 add 指令运行到 ID 段时，t0 的正确结果位于什么位置（精确到某一段的连线或模块）？此时流水线应当进行怎样的操作以保证指令的正确执行

- t0：乘法输出连线；
- 在EX段停顿一个时钟周期

3. T3

> 题目 3. RISC-VI！
>
> 注：本题中增加的部分指令均不是 RV-32I 标准指令，请大家注意甄别
>
> 又是某一天，WJD 助教突发奇想：能不能自己添加一些指令，得到属于自己的 RISC-VI 指令集呢？于是他设计了如下的几条指令：
>
> • addmi：存储器-立即数加法指令。 指令格式为 addmi rd, rs, imm，操作为 x[rd]←MEM[x[rs]]+sext(imm)。
>
> • addmr：存储器-寄存器加法指令。 指令格式为 addmr rd, rs1, rs2，操作为 x[rd]←MEM[x[rs1]]+x[rs2]。
>
> • addmm：存储器-存储器加法指令。 指令格式为 addmm rd, rs1, rs2，操作为 x[rd]←MEM[x[rs1]]+MEM[x[rs2]]。
>
> 为了保证指令能够正常执行，我们的数据存储器有多个只读端口，但仅有一个读写 端口。假定我们的程序可以保证地址都是字节对齐的。请根据以上内容回答下面的问题

3.1 (1)

> 将下面的 RISC-VI 指令序列用 RISC-V 32I 指令改写。你可以自己指定程序所需要的临时寄存器。

```
addi x5, x0, 0x14
addi x6, x0, 0x18
addmm x6, x6, x5
```

```
addi x5, x0, 0x14
addi x6, x0, 0x18
ld x11, 0(x5)
ld x12, 0(x6)
add x6, x11, x12
```

3.2 (2)

> 修改 RISC-V 32I 五级流水线的数据通路，使其支持这三条指令的正确运行（不考虑 这三条指令涉及的冒险，支持指令功能即可）。提示：三条指令的处理流程可以是一致的，你可以以 addmr 为例介绍通路中的改动，以及该指令在各个阶段所需要进行的操作。

- ID/EX段间寄存器增加EX2_op2_sel接口
- 在MEM和WB阶段之间增加一个EXm阶段,同时原MEM/WB段间寄存器改为MEM/EXm段间寄存器和EXm/WB段间寄存器，那么MEM/EXm段间寄存器需要增加MEM[x[rs1]]、MEM[x[rs2]]、x[rs2]、imm、alu_res1， MEM[alu_res1]、EX2_op2_sel接口，EXm/WB段间寄存器需要增加alu_res1、alu_res2、 MEM[alu_res1]、pc_add4、wb_sel接口
- EX/MEM的段间寄存器上增加输入x[rs1]、x[rs2]、imm、 alu_res1、EX2_op2_sel接口
- 在EXm里需要一个ADD模块，op2为MEM[x[rs2]]，x[rs2]，imm，sel为EX2_op2_sel，ADD模块的输出为alu_res2

3.3 (3)

> 考虑如下的指令序列：

```
addi x5, x0, 1
addi x6, x0, 0
sw x5, 0(x6)
addmi x6, x0, 1
```

为了让流水线 CPU 能够正确执行该程序，在上一问数据通路的基础上应当增加怎样的设计？

将data_memory前递给EX阶段

相较 CISC 指令集，RISC 指令集通常采用 LOAD-STORE 体系结构，即存储器和存器之间的数据交互由专门的 load 和 store 指令负责，其他指令均不能访问存储器， 所有的算数与逻辑操作均在寄存器中进行。请根据以上内容分析 RISC 指令集这种操作模式的优缺点。

优点：

1. 简单和规范：RISC指令集精简而规范，指令的数量较少且操作简单，便于指令的设计、解码和执行
2. 更好的流水线技术支持：由于RISC指令集的指令简单，指令之间的依赖性较少，流水线可以同时执行多条指令的不同阶段，提高了指令的并行度和整体性能

缺点：

1. 一些复杂的操作可能需要多条指令来完成，占用更多的存储空间
2. 无法迅速地读写任意地址的内容，RISC指令集的LOAD-STORE体系结构要求数据必须先加载到寄存器中，然后才能进行操作，最后再存回内存，并且会导致对存储器的访问频率增加