北京大学 - 智能硬件体系结构

2024年秋季 期末考试

1、数据依赖与寄存器重命名（25分）

```
1. R1 = R2 / R3
2. R0 = MEM[R1]
3. R1 = R2 + R3
4. R0 = R0 + R1
5. R1 = MEM[R1]
6. R1 = R0 * R1
7. R4 = R0 + R1
8. MEM[R3] = R2 // a store instruction
```

1）假设指令并行度仅受到真实依赖（true dependencies）的限制，那么这段代码中可用的指令级并行（instruction-level parallelism, ILP）是多少？请画出依赖关系图来证明你的答案（使用指令编号表示指令）。

1 3 8　　ILP=5/8
2 5
4
6
7

2）将上述汇编代码进行寄存器重命名，以消除代码段中的所有伪依赖（false dependencies），同时保持相同的功能。要求如下：第一、不能重新排序、删除或添加指令（只能更改寄存器编号）；第二、确保代码段结束时，寄存器 R0-R4 中的值与上述代码段执行后完全相同。例如，如果上述代码段执行后 R0=5，那么在你重写的代码段结束后，R0 仍然必须是 5。任何内存MEM的写操作结果也必须保持一致；第三、可使用的寄存器为 R0-R8。

1. R5=R2/R3
2. R6=MEM[R5]
3. R7=R2+R3
4. R0=R6+R7
5. R8=MEM[R7]
6. R1=R0*R8
7. R4=R0+R1
8. MEM[R3]=R2

2、缓存与一致性（25分）

1）在课堂上讲解的 MESI 协议中，以下哪些状态转换可能是由于另一个处理器的事务引起的?

M -> E　　S -> E　　I -> S　　M -> S　　S -> M

2) 考虑以下C代码：

```
char A[4096]; // each element is 1 byte
for(j=0;j<100000;j++){
    for(i=0;i<Y;i=i+X){
        A[i]=A[i]+1;
    }
}
```

假设只有对数组 A 的访问会进入数据缓存（其他值存储在寄存器中）。对于这段代码，如果数据缓存大小为 1 KB，缓存行（cache line）大小为 32 字节，并且是直接映射的，那么对于不同的 X 和 Y 值，预期的命中率是多少？完成以下表格并简要说明计算过程。

	X=2	X=4	X=64
Y=2048	15/16	7/8	0
Y=1025	511/513	N/A	15/17

3、乱序执行微架构（25分）

假设我们有一个乱序执行处理器，RS可容纳3条指令，ROB可容纳6条指令。由于第一个Load指令需要很长时间停顿，以下两个程序A和B都会因为等待Load完成而停滞。对于每个程序，请指出加载完成之前，最后一条将被放入 ROB 的指令，同时给出简要解释。（指令会一直停留在RS，直到它被发射到计算单元。）

Program A	Program B
R1=MEM[R2+0]	R1=MEM[R2+0]
R2=R4+4	R2=R1+4
R3=R2+R7	R3=R4+R4
R4=R1+6	R4=R2+6
R1=R2+R3	R1=R2+R3
R6=R2+R6	R6=R9+R10
R7=R6+19	R7=R3+19
R8=R3+R6	R8=R9+R3

由于Load操作会导致长时间停顿，我们需要检查是否首先遇到结构性hazard，原因可能是RS满了或者ROB满了。一

旦到达第6条指令，ROB就会满；如果有3条指令因为依赖于Load操作的数据而无法执行，RS就会满。

在程序A中，前6条指令中只有一条指令依赖于Load（R4 = R1 + 6）。因此，ROB会在RS之前填满，最后进入ROB的指令是第6条指令：R6 = R2 + R6。在程序B中，第二条指令（写入R2的指令）依赖于Load，第4条和第5条指令则依赖于第2条指令写入的R2。因此，RS在第5条指令时就会满，最后进入ROB的指令是在加载完成之前的第5条指令：R1 = R2 + R3。

4、AI处理器架构（25分）

1）在神经网络加速器设计中，通常会用到多级的乘加树结构来计算卷积或矩阵乘法。我们利用课堂中所学的Radix-4布斯编码乘法器，构建了一个可用于神经网络加速器的乘加单元。该乘加单元输入为4个8-bit的有符号补码数（2's complement）X1、X2、X3、X4，输出Y = X1*X2 + X3*X4。假设X1 = -2，X2 = 8，X3 = -15，X4 = -9。参考完成第三讲第40页，完成X1*X2和X3*X4的布斯编码乘法器步骤。

-2 * 8：

$(-2)_{10} = (11111110)_2$　　$(8)_{10} = (00001000)_2$

假设对-2做Radix-4 booth encoding：

1．对-2补零得到: (111111100)

2．拆分为4组：(111), (111), (111), (100)

3．做booth encoding得到：0, 0, 0, -2Y

4．将不同部分和相加得到：$-2 * 8 = -2(00001000)_2 = (11110000)_2 = (-16)_{10}$

-15 * -9情况相同，此处不多做赘述

2）神经网络加速器架构一般可简单分为Output Stationary、Weight Stationary、Input Stationary，请简要说明这3种架构设计方式的区别。

架构种类	特点	优点	缺点	设计思路
Ouput Stationary	中间结果固定	减少中间结果的访问次数，提高能效	权重和输入需要频繁更新	PE局部存储部分和结果，权重输入被广播，不断进行运算
Weight Stationary	权重固定	高效复用权重，减少权重的移动	输入需要频繁更新	将权重存储在PE中，输入广播至不同PE和不同权重进行运算

Input Stationary	输入固定	复用输入	权重需要频繁更新	将输入存储在PE中，权重被广播

3）残差是当前神经网络中极为常用的机制，广泛应用于各类AI模型中。假设一个神经网络加速器采用Weight Stationary进行设计，如何应对残差等需要跳跃多个神经网络层级的数据通信？请给出1-2个思路，并简单描述该思路的可行性和优劣（200-300字以内，配合1-2个简单图示说明即可）。

开放式回答，以下提供一个参考回答：

1. 专用残差缓冲区

在加速器的设计中增加一块专用的残差缓冲区，存储跳跃层级所需的残差数据。

- 优点
  - 缓冲区可以高效地存储和读取残差数据，避免频繁访问主存储器。
  - 减少数据通信延迟，提升加速器的整体性能。
- 缺点
  - 需要额外的硬件资源，可能增加设计复杂性。
  - 缓冲区大小需合理设计，否则可能导致资源浪费或容量不足。

2. 数据复用与流水化处理

通过调度优化，将残差存储在当前计算单元的部分资源中，并利用流水线机制实现残差的并行处理。

- 优点
  - 减少数据传输，降低总线带宽需求。
  - 在充分利用现有资源的情况下提高吞吐量。
- 缺点
  - 调度算法设计复杂，可能会降低硬件设计的灵活性。
  - 对层间依赖的网络可能不适用，需严格分析数据复用的可行性。

总结

两种方法都能有效解决残差通信问题。专用缓冲区适合资源较为宽裕，需要实时性的设计，而流水化适合资源受限的场景。