****

**计算机体系结构实验报告**

|  |  |
| --- | --- |
| 学生姓名： | 谭哲文 |
| 学 号： | 8202191123 |
| 专业班级： | 计科2105班 |
| 指导教师： | 余腊生 |
| 学 院： | 计算机学院 |
| 完成时间： | 2023/12/31 |

**实验4 单功能流水线调度机构模拟**

**一、实验目的**

结合数据结构的相关知识，编写流水线调度模拟程序。

**二、实验内容**

通过模拟单功能流水线调度过程，掌握流水线技术，学会计算流水线的吞吐率、加速比、效率。

1. 流水线的表示法有三种：

连接图、时空图、预约表。对于线性流水线，主要考虑前二种。

1. 流水线的主要特点：

在流水线的每一个功能部件的后面都要有一个缓冲器，称为锁存器、闸门寄存器等，它的作用是保存本流水段的执行结果。各流水段的时间应尽量相等，否则回引起阻塞、断流等。只有连续提供同类任务才能充分发挥流水线的效率。在流水线的每一个流水线段中都要设置一个流水锁存器。流水线需要有“装入时间”和“排空时间”。只有流水线完全充满时，整个流水线的效率才能得到充分发挥。

**三、设计描述**

流水线是一种工作方式，它将一个大任务分成若干个小任务，然后由若干个功能部件分别处理这些小任务。这样，就可以在同一时间内同时进行若干个小任务，提高工作效率。

在代码设计中，流水线将由 4 个功能部件组成，需要处理 5 个浮点加指令。流水线中每个功能部件都有自己的处理范围和时间，具体如下：

* NL（规格化）：处理第 1 个指令，在时间片 0 到 4 之间处理。
* MA（尾数加）：处理第 2 个指令，在时间片 1 到 4 之间处理。
* EA（对阶）：处理第 3 个指令，在时间片 2 到 4 之间处理。
* ED（阶差）：处理第 4 到第 5 个指令，在时间片 3 到 4 之间处理。

程序还需要输出流水线每个功能部件在每个时间片内处理的指令，并统计流水线的吞吐量（Throughput）、加速比（Speedup）和效率（Efficiency）。

* 吞吐量（Through Put）：指流水线能够处理的指令数量与时间的比值。计算公式：吞吐量=处理的指令数量/时间。
* 加速比（Speedup）：指流水线处理指令的速度与单个处理器处理指令的速度之比。计算公式：加速比=单个处理器处理指令的时间/流水线处理指令的时间。
* 效率（Efficiency）：指流水线的效率。计算公式：效率=吞吐量/加速比。

**具体分析：**

主函数中定义一个二维数组ts表示时空图，通过调用pipeline函数进行指令状态转换，并通过调用print函数输出时空图。最后，主函数计算并输出流水线的吞吐率、加速比和效率。

pipeline函数：将指令的状态转换到时空图中。从后往前依次处理每个功能部件，然后从起始时间片开始，依次将指令号记录到时空图中。

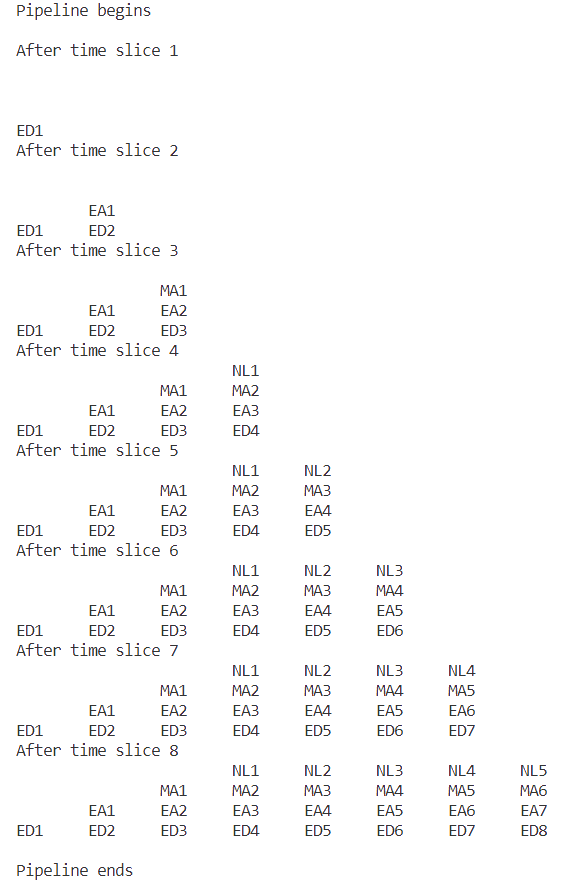
print函数：输出时空图。按照时间片的顺序，依次输出每个时间片的信息。然后按照功能部件的顺序，依次输出每个功能部件的信息。如果当前时间片没有指令，则输出换行，否则输出指令名称和指令号。

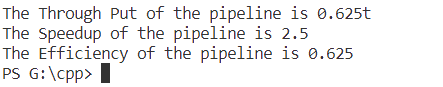
**四、程序清单**

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  // 功能部件数目  const int SPACE = 4;  // 需要流水处理的浮点加指令数目  const int NUM = 5;  // 存储不同时间段各个功能部件内指令值  const int TIME = NUM + SPACE - 1;  // ED：求阶差 EA：对阶MA：尾数加 NL：规格化  const string INSTRUCTIONS[] = {"NL", "MA", "EA", "ED"};  // 初始化时空图  int ts[SPACE][TIME] = {0};  // 输出时空图  void print();  // 流水线中指令状态转换算法  void pipeline(int ts[SPACE][TIME]);  int main()  {      cout << "Pipeline begins" << endl           << endl;      pipeline(ts);      print();      cout << endl           << "Pipeline ends" << endl           << endl;      // 输出流水线吞吐率      cout << "The Through Put of the pipeline is " << (double)NUM / TIME << "t" << endl;      // 输出流水线加速比      cout << "The Speedup of the pipeline is " << ((double)NUM \* SPACE) / TIME << endl;      // 输出流水线效率      cout << "The Efficiency of the pipeline is " << ((double)NUM \* SPACE) / (TIME \* SPACE) << endl;      return 0;  }  // 输出时空图  void print()  {      for (int i = 0; i < TIME; ++i)      {          cout << "After time slice " << i + 1 << endl;          for (int j = 0; j < SPACE; ++j)          {              // 当前时间片没有指令时，输出换行              if (i < NUM && ts[j][i] == 0)              {                  cout << endl;              }              else              {                  for (int k = 0; k < i + 1; ++k)                  {                      // 当前功能部件存在指令时，输出指令名称和指令号                      if (ts[j][k] != 0)                      {                          cout << INSTRUCTIONS[j] << ts[j][k];                      }                      cout << "\t";                  }                  cout << endl;              }          }      }  }  // 流水线中指令状态转换算法  void pipeline(int ts[SPACE][TIME])  {      int tempSpace = 0; // 记录处理的指令号      int tempTime = 0;  // 记录时间轴的变化      // 从后往前依次处理每个功能部件      for (int s = SPACE - 1; s >= 0; s--)      {          // 初始化指令号          tempSpace = 1;          // 从tempTime开始，依次处理时间片          for (int t = tempTime; t < TIME; t++)          {              // 将当前时间片的指令号记录到时空图中              ts[s][t] = tempSpace++;          }          // 时间轴向后移动一位          tempTime++;      }  } |

**五、结果分析**

**输出结果：**





**调试分析：**

在调试上面的代码时，可以通过打印输出来查看代码的执行情况。比如在函数pipeline中加入打印输出，来查看每个时间片中功能部件所处理的指令号：

|  |
| --- |
| // 流水线中指令状态转换算法  void pipeline(int ts[SPACE][TIME])  {      int tempSpace = 0; // 记录处理的指令号      int tempTime = 0;  // 记录时间轴的变化      // 从后往前依次处理每个功能部件      for (int s = SPACE - 1; s >= 0; s--)      {          // 初始化指令号          tempSpace = 1;          // 从tempTime开始，依次处理时间片          for (int t = tempTime; t < TIME; t++)          {              //加入调试分析              cout << "time slice " << t << ": " << tempSpace << endl;              // 将当前时间片的指令号记录到时空图中              ts[s][t] = tempSpace++;          }          // 时间轴向后移动一位          tempTime++;      }  } |

这样就可以查看每个时间片中功能部件所处理的指令号了。如果发现指令号有误，就可以从输出的信息中找出错误的原因，并进行修正。

同理，在函数print中也可以加入打印输出，来查看时空图的输出情况。

经过调试分析，输出结果正确。

**六、实验心得**

流水线处理指令是一种提高处理器性能的高效方式。它的工作原理是：将指令分成若干个部分，每个部分由一个功能部件来处理。每个功能部件有自己的处理范围和时间，每个指令都会在流水线中流动，最终达到高效处理的目的。

在设计流水线时，我们需要考虑如何平衡功能部件的数量、指令的数量以及流水线的时间，这些因素都会对流水线的吞吐量、加速比和效率产生影响。

在调试流水线代码时，可以通过打印输出或使用调试工具GDB等来查找代码中的错误，并进行修正。通过这一次的实验，我对流水线处我对流水线处理指令的工作原理有了更深的了解，并且能够使用代码实现流水线处理指令的功能。