题目1d

1.1写出CPU运行时间公式，哪些因素会影响以上分量？结合计算机体系结构的发展趋势分析哪些技术通过影响以上分量降低了CPU的运行时间。（6分）

答：

1. CPU运行时间可以用以下公式表示：

CPU运行时间 = (CPI × 执行指令数 × 时钟周期)

其中，CPI（Cycles Per Instruction）表示每条指令需要的时钟周期数，执行指令数表示需要执行的指令数量，时钟周期表示CPU时钟周期。

以上公式中的各个分量受到不同因素的影响：

CPI受到CPU体系结构和指令集的影响。不同的CPU体系结构和指令集对于同样的操作可能需要不同数量的时钟周期，因此CPI也会不同。

执行指令数受到应用程序的特性和优化程度的影响。相同的应用程序在不同的实现中可能需要执行不同数量的指令。

时钟周期受到CPU的制造工艺和微架构设计的影响。更先进的制造工艺和更高效的微架构设计可以使CPU的时钟周期更短。

2. 随着计算机体系结构的发展，有许多技术通过影响以上分量降低了CPU的运行时间，例如：

超标量技术：通过在同一时钟周期内同时执行多个指令，降低了CPI。

乱序执行技术：允许CPU以任意顺序执行指令，从而提高了指令级并行度，降低了CPI。

预测执行技术：通过预测程序的分支和数据依赖关系，使CPU能够提前执行指令，从而降低了CPI。

高速缓存技术：将常用的数据和指令存储在高速缓存中，减少了内存访问的次数，降低了执行指令数。

动态电压调节技术：根据CPU的工作负载调节供电电压，提高了CPU的能效和稳定性，从而提高了时钟频率，降低时钟周期。

1.2量化方法的三个原则。流水线使用了哪些原则。（4分）

量化方法的三个原则是：

1、充分利用并行。

2、局部性原理。

3、重点关注常见情形

流水线使用了充分利用并行原则。

题目2

2.1 假设一个任务的运行时间包括3部分，分别为Fraction1, Fraction2, Fraction3, 对应的加速比为Speedup1,Speedup2, Speedup3部分。求总的加速比Speed-up-overall。（5分）存疑

答

总的加速比Speed-up-overall可以通过以下公式计算：

Speed-up-overall = 1 / ((1 - Fraction1) + Fraction1/Speedup1 + Fraction2/Speedup2 + Fraction3/Speedup3)

2.2 土豆悖论与Amdahl’s定律。假设有一个程序有两个部分，分别耗时99s和1s。职员甲用了2个小时把第一部分的耗时降低50%，职员乙则用1个小时把第二部分的耗时降低90%。甲和乙都认为自己的贡献更大，分析他们的理由。（5分）

**职员甲的观点：** Amdahl’s定律：当提升系统的一部分性能时，对整个系统性能的影响取决于:1、这一部分有多重要 2、这一部分性能提升了多少。

假设原来在一个系统中执行一个程序需要时间 ，其中某一个部分占的时间百分比为，然后，把这一部分的性能提升 倍。即这一部分原来需要的时间为 ，现在需要的时间变为。则整个系统执行此程序需要的时间变为：

故可得，系统性能提速的倍数为:

根据Amdahl’s定律， 职员甲优化第一部分后，程序整体性能提升的倍数为1.98；职员乙优化第二部分后，程序整体性能提升的倍数1.009，尽管职员乙将第二部分加速了10倍，但对整体程序帮助不大，所以甲的贡献较大。

**职员乙观点：**土豆悖论: 假设有1000斤的土豆，然后这些土豆的水分含量是99%，只有1%是干货，把土豆风干一下，使它的含水量下降到98%，要风干500斤的水。

职员甲花了10个小时，其优化的部分由原来的99%变成97.8%。职员乙则花了2个小时，其优化部分由原来的1%变成0.1%，时间短，效果显著。

题目3

3.1 简述数据并行和任务并行的区别。举一个你身边的例子，分别给出基于数据并行和任务并行的解决方案。（5分）

答：数据并行和任务并行是两种常见的并行计算方法。它们的区别在于并行处理的对象不同。

数据并行：将数据划分为多个部分，同时将每个部分分配给不同的处理器进行处理，最后将结果合并。在数据并行中，每个处理器都处理相同的操作，但处理的数据不同。数据并行常用于处理大规模的数据集，例如机器学习的训练过程。

任务并行：将任务分解为多个子任务，并将每个子任务分配给不同的处理器执行。在任务并行中，每个处理器执行不同的操作，但所有操作都是为了完成同一个任务。任务并行通常用于复杂的计算任务，例如在一个大型项目中并行执行多个计算任务。

例子言之有理即可。

3.2 阐述图1.9所表示的计算机的发展趋势，分析为什么延迟的发展速度远远落后于吞吐量的发展速度。（5分）

答：延迟的发展速度远远落后于吞吐量的发展速度，主要是因为计算机的吞吐量可以通过并行化来提高，而延迟却很难通过并行化来解决。并行化可以同时执行多个任务，提高吞吐量，但是延迟是指执行单个任务所需的时间，而单个任务很难通过并行化来缩短时间。

此外，延迟还受到物理限制和技术瓶颈的影响，例如电子信号在电路中传输的速度是有限的，而光信号的传输速度更快但技术复杂度和成本也很高，因此要突破延迟的瓶颈需要在技术、算法等方面做出更大的创新和突破。

题目4

4. 列举5种操作数的寻址方式。假设寄存器Ri的初始化为i + 2，地址a的内存初始化为(a + 3) mod 10。依次执行以下指令后，给出每条指令执行后寄存器的值，仅回答R1-R4寄存器即可。（10分）

寻址方式（4分）：1、立即寻址：将常数直接作为操作数，常用于一些简单的算术运算。例如：ADD 5, R1 表示将寄存器 R1 中的值加上常数 5。

2、直接寻址：直接给出操作数的地址，例如：ADD MEM[100], R1 表示将存储在地址 100 的数据与寄存器 R1 中的值相加。

3、间接寻址：操作数为存储器中的地址，例如：ADD (R2), R1 表示将存储在 R2 中的地址所指向的数据与寄存器 R1 中的值相加。

4、寄存器寻址：将寄存器中的值作为操作数，例如：ADD R2, R1 表示将寄存器 R2 中的值与寄存器 R1 中的值相加。

5、寄存器间接寻址：操作数为寄存器中存储的地址，例如：ADD (R2), R1 表示将存储在 R2 中的地址所指向的数据与寄存器 R1 中的值相加。

LB指令将存储在R1中的地址偏移100的内存读取一个字节，将其符号扩展为32位，并将结果存回在R1中，R1的原始值为3，所以要读取的地址值为3+100。103地址的初始值为6，R1的值变为6。

R1的值为6，而它的偏移量为0， 所以读取的是地址6的内容，该内容为9，同理，R2的值为9。

R3为5，R2为9，去14地址的数字(14+3)mod10=7放入R3，所以R3是7

R4为(1001+3)mod10 = 4。

综上：R1:6 R2:9 R3:7 R4:4

题目5

5.基于RISC-V的5级流水线机制设计支持Add R1，R2，100（R3）的7级流水线示意图。分析为什么RISC-V设计为只有访存指令才能读写内存。（10分）

答：1.设计的7级流水线步骤包含核心的6个步骤即可。IF，ID，EXE1（计算有效地址），MEM，EXE2（计算Add），WB。2. 在RISC-V 指令集中，对内存的读写只能通过LOAD 和STORE 指令实现。而其他的指令只能以寄存器为操作对象。这样设计使得5级流水线的数据冲突和结构冲突更少，有利于提高流水线的效率。

题目6

6.分析下列7条指令的流水线执行示意图（10分）

--条件跳转指令在EX阶段才知道最终的跳转指令位置，

--预测条件跳转指令不会选中

--当条件跳转指令执行时，$1中的值等于$2中的值

门上写着字

中度可信度描述已自动生成

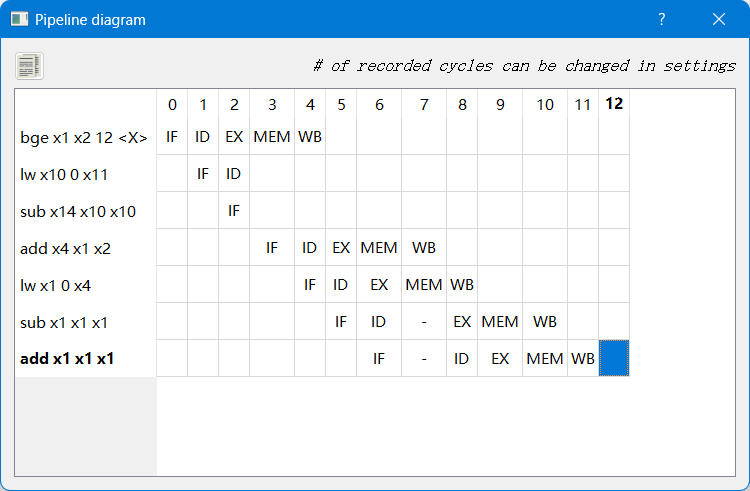
仅使用了stall技术的

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|  | F | D | E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | F | D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | F |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | F | D | D | D | E | M | W |  |  |  |  |  |  |
|  |  |  |  |  |  | F | F | F | D | D | D | E | M | W |  |  |  |
|  |  |  |  |  |  |  |  |  | F | F | F | D | D | D | E | W | M |

考虑stall 和 forwarding

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|  | F | D | E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | F | D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | F |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |

表格

中度可信度描述已自动生成

Table, whiteboard

Description automatically generated

考虑了转发和stall技术的

Table, whiteboard

Description automatically generated

题目7

7.下图是教材的图C.25,读图后（10分）

1)解释每个部件的作用。

2)分析指令是怎么执行的。

Diagram

Description automatically generated

答

第一问：

PC:程序计数器

ADD（左）：产生下一个指令

MUX（左）:多路选择器：选择跳转或PC+

Instruction memory:根据PC值读出指令

Registers:根据寄存器地址读写寄存器

ADD(中)：计算PC跳转有效地址

Sign-extend:立即数符号扩展

ALU：逻辑计算单元

MUX（中）：多路选择器：选择操作数

=？：分支选择器

Data memory: 根据地址读出/写入内存

MUX（右）：多路选择器：选择写回数据

第二问：

讲清楚IF，ID，EXE，MEM，WB的动作即可。

题目8

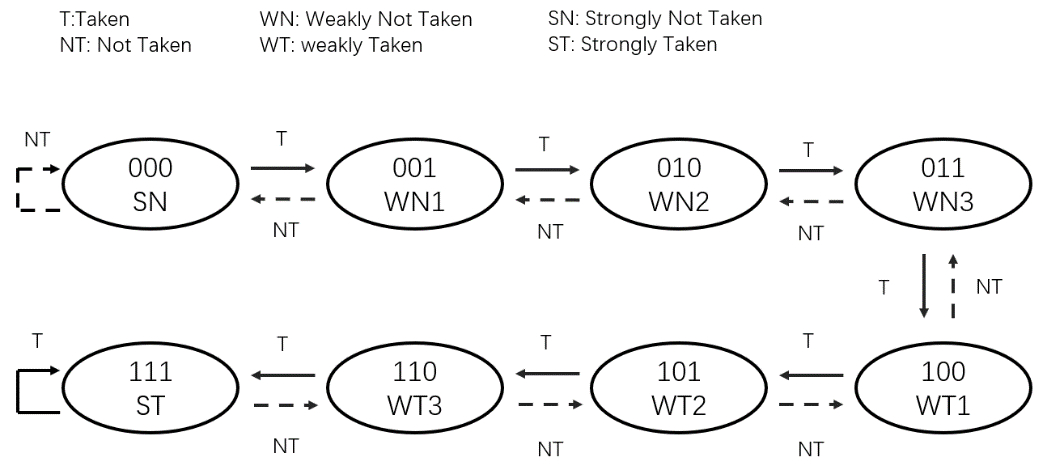
8.以下是教材C.15

1）图中所表示的机制是什么，解决什么问题？

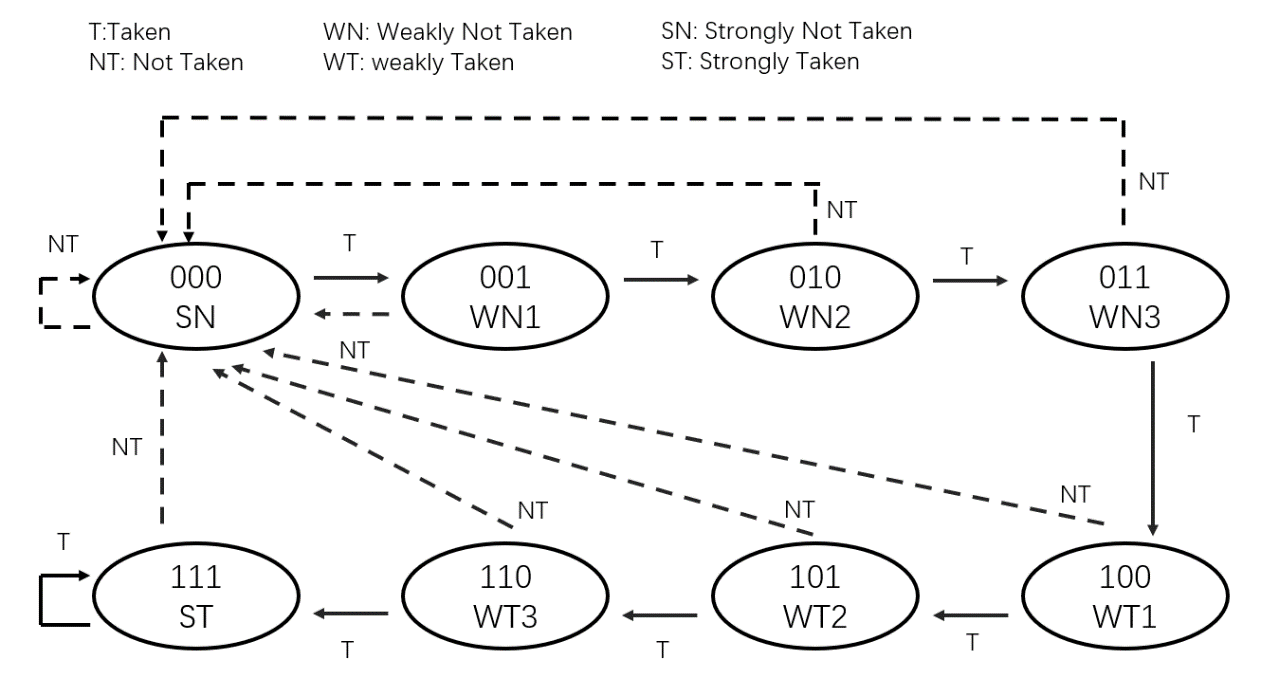
答

2位预测机制。动态分支预测机制中，简单的1位预测机制在性能上有一处短板：即使某个分支几乎总是被选中，在其未被选中时，我们也可能会得到两次错误的预测，而不是一次，因为错误预测会导致该预测位反转，2位预测机制就是为了弥补这个缺点。

2）画出类似的3bit 8状态的指令预测跳转图.



**或**：



3)分析现代计算机为什么只有选用2bit而不是更多的预测机制。

答

因为2位预测器的效果几乎与n位预测器相同，所以大多数系统采用2位分支预测器，而不是更具一般性的n位预测器。

Diagram

Description automatically generated

题目9

9.根据下图提供的RISC-V 32I手册，写出以下指令的二进制码表示：

Graphical user interface, application, table, Excel

Description automatically generated

所需指令对应opcode：

表格

描述已自动生成

0000000 00011 00010 000 00001 0110011

0000000 R3 R2 000 R1 0110011

000000000100 00010 000 00001 0010011

00000000100 R2 000 R1 0010011

0000011 00001 00010 010 00100 0100011

0000011 R2 R1 010 00100 0100011

1 111101 01110 01111 101 1110 1 1100011

1 111101 R14 R15 101 1110 1 1100011

题目10

1) 找出下面代码中所有的数据冒险（data hazard）,找出的所有数据冒险。具体格式如I5指令和I1指令在r2上存在RAW。

文本

描述已自动生成

I1和I2在r2上存在WAR,

I1和I3在r2上存在WAR

I1和I4在r2上存在WAR

I2和I4在r42上存在WAR

2) 画出没有转发（no forwarding）的流水线

3) 画出有转发（forwarding）的流水线

无转发，无停顿

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |

有停顿，无转发

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | F | D | D | D | E | M | W |  |  |  |  |  |  |  |  |  |
|  |  |  | F | F | F | D | E | M | W |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | F | D | D | E | M | W |  |  |  |  |  |  |
|  |  |  |  |  |  |  | F | F | D | E | M | W |  |  |  |  |  |

表格

描述已自动生成

有转发

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |

表格

描述已自动生成