考试内容

一、名词解释（6 题，每题 5 分）

1、ISA

指令集体系结构，是在最底层把硬件结构抽象出来供软件编程控制的，指令集解决了最基本的软件兼容性问题。

MicroArchitecture：

微体系结构，是一款节能的新型微架构，设计的出发点是提供卓然出众的性能和能效，提高每瓦特性能，也就是所谓的能效比。

2、MIMD、SIMD

MIMD:多指令流多数据流（MultipleInstructionStreamMultipleDataStream，简称 MIMD），它使用多个控制器来异步地控制多个处理器，从而实现空间上的并行性。

SIMD:单指令多数据流(Single Instruction Multiple Data)，能够复制多个操作数，并把它们打包在大型寄存器的一组指令集。以同步方式，在同一时间内执行同一条指令。

3、资源冲突（结构冲突）

因硬件资源满足不了指令重叠执行的要求而发生的冲突。

4、数据冲突

当指令在流水线中重叠执行时，因需要用到前面指令的执行结果而发生的冲突。

5、控制冲突

流水线遇到分支指令和其他会改变 PC 值 的指令所引起的冲突。

6、Amdahl 定律

加快某部件执行速度所能获得的系统性能加速比，受限于该部件的执行时间占系统中总执行时间的百分比。

7、加速比

反映了改进后的机器速度比改进前快了多少倍。

$$加速比=\frac{系统性能_{改进后}}{系统性能_{改进前}}=\frac{总执行时间_{改进前}}{总执行时间_{改进后}}$$

8、非冯·诺依曼计算机

采用哈佛结构的计算机，将数据和指令分开存取，取数据和指令可以同时存取，有利于流水线工作，脱离了冯诺依曼结构原有模式的计算机。

9、流水线分类

静态流水线：在同一时间内，多功能流水线中的 各段只能按同一种功能的连接方式工作。

动态流水线：在同一时间内，多功能流水线中的各段可以按照不同的方式连接，同时执行多种功能。

单功能流水线：只能完成一种固定功能的流水线。

多功能流水线：流水线的各段可以进行不同的 连接，以实现不同的功能。

10、锁存技术

检测发现数据冲突，并使流水线停顿，直至冲突消失

11、寄存器换名技术

通过改变指令中操作数的名来消除名相关。

12、定向技术

解决 raw 冲突，尽可能的将结果数据从其产生的地方直接传送到所有需要该结果的功能部件。

13、ROB 技术（re-order-buffer ）

是为前瞻执行而设置的，它在指令操作完成后到指令被确认这段时间，为指令保存数据。

14、分支历史表 BHT

也被称为分支预测缓冲器，用于记录分支指令最近一次或几次的执行情况。

15、超标量机

在每个时钟周期流出的指令条数不固定，以代码具体情况而定，不过有上限的处理机。

16、向量机

在流水线处理机中，设置向量数据表示和相应的向量指令，称为向量处理机。

17、

- **时间局部性**

  程序即将用到的信息很可能就是目前正在使用的信息。

- **空间局部性**

  程序即将用到的信息很可能与目前正在使用的信息 在空间上相邻或者临近。

18、

- **数组合并技术**

  通过提高空间局部性来减少失效次数，将相互独立的数组合并成为一个复合数组，使得一个 Cache 块中能包含全部所需的元素。

```
/* 修改后 */
struct merge {
int val;
int key;
};
struct merge merged_array [ SIZE ];
```



```
/* 修改前 */
int val [ SIZE ];
int key [ SIZE ];
```

- **内外循环交换技术**

  是通过提高空间局部性来减少失效次数，在含有嵌套循环，程序没有按照数据在存储器中存储的顺序进行访问，重新排列访问顺序使得在一个 Cache 块被替换之前，能最大限度得利用块中的数据。

1. 内外循环交换

   举例：

```
/* 修改前 */
for （j = 0；j < 100；j = j+1）
  for （i = 0；i < 5000；i = i+1）
    x[i][j] = 2 * x[i][j];
/* 修改后 */
for （i = 0；i < 5000；i = i+1）
  for （j = 0；j < 100；j = j+1）
    x[i][j] = 2 * x[i][j];
```

- **循环融合技术**

  通过改进时间局部性来减少失效次数，程序含有几部分独立的程序段，它们用相同的循环访问同样的数组，对相同的数据做不同的运算，通过将它们融合为单一的循环，能使读入 Cache 的数据在被替换出去之前，得到反复的使用。

```
/* 修改前 */
    for （j = 0；j < 100；j = j+1）
       for （i = 0；i < 5000；i = i+1）
          x[i][j] = 2 * x[i][j];
/* 修改后 */
    for （i = 0；i < 5000；i = i+1）
       for （j = 0；j < 100；j = j+1）
          x[i][j] = 2 * x[i][j];
```

19、平均访存时间

评测存储系统性能的指标

$$\text{平均访存时间} = \text{命中时间}_{L1} + \text{失效率}_{L1} \times \text{失效开销}_{L1}$$
$$\text{失效开销}_{L1} = \text{命中时间}_{L2} + \text{失效率}_{L2} \times \text{失效开销}_{L2}$$
$$\text{平均访存时间} = \text{命中时间}_{L1} + \text{失效率}_{L1} \times (\text{命中时间}_{L2} + \text{失效率}_{L2} \times \text{失效开销}_{L2})$$

20、

- **强制失效**

  当第一次访问一个块时，该块不在 Cache 中，需从 下一级存储器中调入 Cache，这就是强制性失效。 (冷启动失效，首次访问失效)

- **容量失效**

  如果程序执行时所需的块不能全部调入 Cache 中， 则当某些块被替换后，若又重新被访问，就会发生失效。这种失效称为容量失效。

- **冲突失效**

  在组相联或直接映象 Cache 中，若太多的块映象到 同一组(块)中，则会出现该组中某个块被别的块替 换(即使别的组或块有空闲位置)，然后又被重新访 问的情况。这就是发生了冲突失效。(碰撞失效，干扰失效)

21、

- **全相联映像**

  主存中的任一块可以被放置到 Cache 中的任意一个位置。

- **直接映像**

  主存中的每一块只能被放置到 Cache 中唯一的一个位置。

- **组相联映像**

  主存中的每一块可以被放置到 Cache 中 唯一的一个组中的任何一个位置。

22、非阻塞 CACHE 技术

Cache 失效时仍允许 CPU 进行其他的命中访问。即允许“失效下命中”。

23、TLB（(传输后备缓冲器)）

也称快表、一个内存管理单元用于改进虚拟地址到物理地址转换速度的缓存。

23、平均 CPI

CPI = 执行程序所需的时钟周期数 / IC

IC：所执行的指令条数

24、

- **乱序发射**

  CPU 允许将多条指令不按程序规定的顺序分开发送给各相应电路单元处理的技术

- **乱序执行**
- **顺序完成**

25、指令动态调度

在程序的执行过程中，依靠专门硬件对代码进行调度，减少数据相关导致的停顿。

26、LRU 算法

选择近期最少被访问的块作为被替换的块。

27、

- **RISC**

  精简指令集计算机，尽可能地把指令集简化，不仅指令的条数少，而且 指令的功能也比较简单。
- **CISC**

  复杂指令集计算机，增强指令功能，把越来越多的功能交由硬件来实现，并且指令的数量也是越来越多。

28、循环展开

是一种牺牲程序的尺寸来加快程序的执行速度的优化方法。

二、简答题（4 题，每题 8 分）

1、对流水线的冲突处理通常有两种方式：静态调度方式和动态调度方式；说明这两种方式，并分别举出其实现方式。

- 静态调度 **(实现方式：依靠编译器)**

  静态指令调度是指编译程序通过调整指令的顺序来减少流水线的停顿,提高程序的执行速度;
- 动态调度**（实现方式：以硬件复杂性的显著增加为代价）**

  动态指令调度用硬件方法调度指令的执行以减少流水线停顿。

2、Tomasulo 采取了什么方法避免三种数据冲突。

- 记录和检测指令相关，操作数一旦就绪就立即执行，把发生 RAW 冲突的可能性减少到最小;
- 通过寄存器换名来消除 WAR 冲突和 WAW 冲突。

3、给出一段有相关性的指令，分析相关性、请重新设计指令顺序（编译器方式），消除相关性

1、add R3,R1,R2

2、add R4,R3,R2

3、add R4,R1,R5

4、计算机系统结构、计算机组成和计算机实现的概念与关系。

- **计算机系统结构：**

  就是程序设计者所看到的计算机的基本属性，即概念性结构与功能特性。
- **计算机组成：**

  计算机系统结构的逻辑实现。即根据计算机系统结构所制订的功能，从逻辑上完成计算机的设计。这里包括各部件的逻辑实现，部件之间的互相连接以及物理机器级中的数据流和控制流的组成以及逻辑设计等。
- **计算机实现：**

  计算机组成的物理实现。包括处理机、主存等部件的物理结构，器件的集成度和速度，模块、插件、底板的划分与连接，信号传输，电源、冷却及整机装配技术等。

一种体系结构可以有多种组成。

一种组成可以有多种物理实现。

5、论述 RISC 与的 CISC 技术；讨论 RISC 从哪些方面提高了指令的执行效率，并举例说明。

- **CISC（复杂指令集计算机）**

  增强指令功能，把越来越多的功能交由硬件来实现，并且指令的数量也是越来越多。

- **RISC（精简指令集计算机）**

  尽可能地把指令集简化，不仅指令的条数少，而且 指令的功能也比较简单。

- **RISC 从哪些方面提高了指令的执行效率：**
  - 指令条数少而简单。只选取使用频度很高的指令， 在此基础上补充一些最有用的指令。
  - 采用简单而又统一的指令格式，并减少寻址方式； 指令字长都为 32 位或 64 位。
  - 指令的执行在单个机器周期内完成。(采用流水线机制)
  - 只有 load 和 store 指令才能访问存储器，其他指令 的操作都是在寄存器之间进行。 （即采用 load-store 结构）
  - 大多数指令都采用硬连逻辑来实现。
  - 强调优化编译器的作用，为高级语言程序生成优 化的代码。
  - 充分利用流水技术来提高性能。

6、简要画出 DLX 多周期（经典 5 段流水）流水线的数据通路图；说明 load 和 store 在每个周期的表现。

指令编号	1	2	3	4	5	6	7	8	9
指令 i	IF	ID	EX	MEM	WB				
指令 i+1		IF	ID	EX	MEM	WB			
指令 i+2			IF	ID	EX	MEM	WB		
指令 i+3				IF	ID	EX	MEM	WB	
指令 i+4					IF	ID	EX	MEM	WB

- **load 指令**
  1、访存有效地址：Reg[rs] + immediate
  2、从存储器取来的数据放入寄存器 rt
- **store 指令**
  1、访存有效地址：Regs[rs] + immediate
  2、要存入存储器的数据放在寄存器中

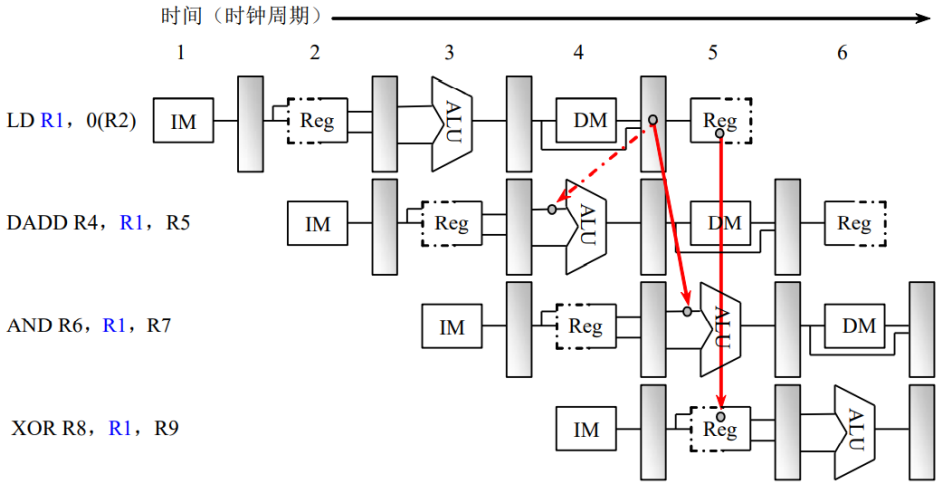
	ALU 指令	LOAD/STORE
IF	取指	取指
ID	译码，读寄存器堆	译码，读寄存器堆
EX	执行	计算有效地址
MEM	-	访存（读或写）
WB	结果写回寄存器堆	读出数据写入寄存器堆

6、简述流水线中的定向技术，并举例说明其用途和缺陷（不能解决的问题）。

简述：在某条指令产生计算结果之前，其他指 令并不真正立即需要该计算结果，如果能够将该计算结果从其产生的地方直接送到其他指令需要它的地方，那么就可以避免停顿。

用途：通过定向技术减少数据冲突引起的停顿

缺陷：



无法将LD指令的结果定向到DADD指令

7、流水线冲突有哪三种？请简述每种流水线冲突。

- **资源冲突（结构冲突）**

  因硬件资源满足不了指令重叠执行的要 求而发生的冲突。如果某种指令组合因为资源冲突而不能正常执行，则称该处理机有结构冲突。

- **数据冲突**

  当指令在流水线中重叠执行时，因需要用到前面指令的执行结果而发生的冲突。

- **控制冲突**

  流水线遇到分支指令和其他会改变 PC 值的指令所引起的冲突。

8、如果某计算机系统有 3 个部件可以同时改进，则这 3 个部件经改进后达到的加速比分别为：*S*1=30,*S*2=20,*S*3=10。如果部件 1 和部件 2 改进前的执行时间占整个系统执行时间的比例都为 30%，那么，部件 3 改进前的执行时间占整个系统执行时间的比例为多少，才能使 3 个部件都改进后的整个系统的加速比 *Sn* 达到 10?

解：

	S1	S2	S3	OTHER
改进前	0.3T	0.3T	XT	0.4-XT
改进后	0.3T/30	0.3T/20	XT/10	0.4-XT

0.01T + 0.015T + XT/10 + 0.4 – XT = T/10

解得：X = 36.1%

9、GPU 采用了哪种处理器设计方式作为原型，请简述并画出这种处理器的体系结构原理图。

TODO:

10、名相关和数据相关会产生写读冲突、读写冲突、写写冲突。简述这三种冲突，并举例说明是如何造成的。

- **RAW（读超前于写）**

  原程序要求对同一单元进行先写后读的操作，可能因为非按序执行成为先读后写，造成出错。
- **WAR（写超前于读）**

  原程序要求对同一单元进行先读后写的操作，可能因为非按序执行成为先写后读，造成出错。
- **WAW（写后写）**

  原程序中如果两条指令都要对同一单元进行写数操作，可能因为非按序执行的原因，改变了两条指令写入的次序。

11、简述并举例说明层次化存储系统存在的理论依据。

计算机系统中存储层次可分为**高速缓冲存储器**、**主存储器**、**辅助存储器**三级

- **高速缓冲存储器**用来改善主存储器与中央处理器的速度匹配问题
- **辅助存储器**用于扩大存储空,即硬盘，光盘等，容量大，但存取数据慢，计算机都是先把辅存中要读的东西放到主存后处理，然后在依据情况是否写回。
- **主存即为内存**，断电信息丢失，但存取数据块，他的容量大小直接影响计算机运行速度。

12、写出平均访存时间的公式，从公式的三个变量出发，分别举出一个优化（减少）平均访存时间的技术方案。

公式：平均访存时间 = 命中时间 + 失效率×失效开销

减少失效率

- 增加 Cache 块大小，
- 提高相联度，
- 增加 Cache 的容量，
- Victim Cache，
- 伪相联 Cache，
- 硬件预取，
- 编译器控制的预取，
- 编译器优化

减少 Cache 失效开销

- 让读失效优先于写
- 写缓冲合并
- 请求字处理技术
- 非阻塞 Cache 技术
- 采用两级 Cache

减少命中时间

- 容量小、结构简单的 Cache
- 虚拟 Cache
- Cache 访问流水化
- Trace Cache

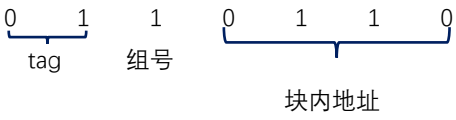
总结：

优化技术	失效率	失效开销	命中时间	硬件复杂度	说 明
增加块大小	+	–		0	实现容易；Pentium 4 的第二级Cache采用了128 B的块
增加Cache容量	+				被广泛采用，特别是第二级Cache
提高相联度	+		–	1	被广泛采用
Victim Cache	+			2	AMD Athlon采用了8个项的Victim Cache
伪相联Cache	+			2	MIPS R10000的第二级Cache采用
硬件预取指令和数据	+			2～3	许多机器预取指令，UltraSPARC Ⅲ预取数据
编译器控制的预取	+			3	需同时采用非阻塞Cache；有几种微处理器提供了对这种预取的支持
用编译技术减少Cache失效次数	+			0	向软件提出了新要求；有些机器提供了编译器选项
使读失效优先于写		+	–	1	在单处理机上实现容易，被广泛采用
写缓冲归并		+		1	与写直达合用，广泛应用，例如21164，UltraSPARC Ⅲ
尽早重启动和关键字优先		+		2	被广泛采用
非阻塞Cache		+		3	在支持乱序执行的CPU中使用
两级Cache			+	2	硬件代价大；两级Cache的块大小不同时实现困难；被广泛采用
容量小且结构简单的Cache	–		+	0	实现容易，被广泛采用
对Cache进行索引时不必进行地址变换			+	2	对于小容量Cache来说实现容易，已被Alpha 21164和UltraSPARC Ⅲ采用
流水化Cache访问			+	1	被广泛采用
Trace Cache			+	3	Pentium 4 采用

13、CACHE 的地址映像规则有三种：全相联、直接映像与组相联。阐述这三种规则，并用图示法说明三种规则的优缺点。

	全相联映像	直接映像	组相联映像
规则	主存中的任一块可以被放置到 Cache 中的任意一个位置。	主存中的每一块只能被放置到 Cache 中唯一的一个位置。	主存中的每一块可以被放置到 Cache 中 唯一的一个组中的任何一个位置。
优点	块冲突小，控制简单，Cache 的利用率高	空间利用率最低，冲突概率最高，	是直接映像与全相映的一个折中．块的冲突率大大降低，块的利用率大大提高，并且实现比全相联方式容易。
缺点	实现最复杂。	实现最简单，不需相联存储器，并且只需比较区号.	

14、有一个 Cache 存储器，主存有 8 块(0-7)，Cache 有 4 块(0-3)，采用组相联映像，组内块数为 2 块，每块大小为 16 个字节。某程序运行时，要访存主存地址（二进制）为 0110110 的字节，则访问 Cache 的哪一块?



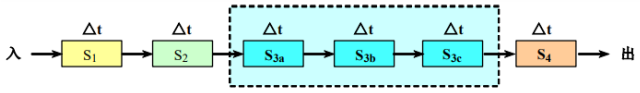
在主存的第 0110110 mod 16 = 3 块，映射至第 (3 mod 2 = 1) 组，所以被映射至第 2 或 3 块

组号 = 主存块号 mod cache 组数

15、解决流水线瓶颈问题有哪些方法，应用场合和效果有何异同?

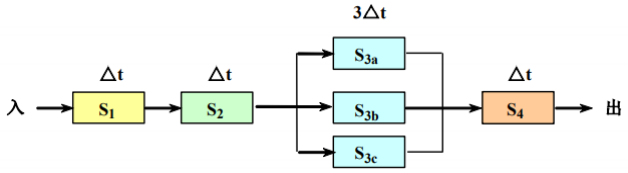
- **细分瓶颈段**

  改进后的流水线的吞吐率 ： $TP_{\max}=\frac{1}{\Delta t}$



- **重复设置瓶颈段**

  缺点：控制逻辑比较复杂，所需的硬件增加了。



16、简述通过软件（编译器）来减少分支延迟的 3 种静态方法及它们的共同特点。

预测分支失败

- 允许分支指令后的指令继续在流水线中流动，就好象什么都没发生似的。
- 若确定分支失败，将分支指令看作是一条普通指令流水线正常流动。

预测分支成功

- 假设分支转移成功，并从分支目标地址处取指令执行。
- 起作用的前题：先知道分支目标地址，后知道分支是否成功。

延迟分支

从逻辑上“延长”分支指令的执行时间。把延迟分支 看成是由原来的分支指令和若干个延迟槽构成，不管分支是否成功，都要按顺序执行延迟槽中的指令。

共同点：

- 对分支的处理方法在程序的执行过程中始终是不变的，是静态的。
- 要么总是预测分支成功，要么总是预测分支失败。

17、在降低 Cache 失效率的方法中，对于给定的 Cache 容量，当块大小增加时，失效率开始是下降，后来反而上升了。解释 Cache 失效率为什么出现这样的变化?

- 一方面它减少了强制性失效;
- 另一方面，由于增加块大小会减少 Cache 中块的数目， 所以有可能会增加冲突失效。

18、简要说明提高计算机系统并行性的 3 种技术途径，并各举一例

- **时间重叠**

  引入时间因素，让多个处理过程在时间上相互错开，轮流重叠地使用同一套硬件设备的各个部分，以加快硬件周转而赢得速度。

- **资源重复**

  引入空间因素，以数量取胜。通过重复设置硬件资源，大幅度地提高计算机系统的性能。

- **资源共享**

  这是一种软件方法，它使多个任务按一定时间顺序轮流使用同一套硬件设备

19、采用动态分支预测的目的是什么？常见有哪些动态分支预测方法？

目的：

- 预测分支是否成功 。
- 尽快找到分支目标地址（或指令） （避免控制相关造成流水线停顿）。

常见方法：

- 采用分支历史表 BHT
- 采用分支目标缓冲器 BTB
- 基于硬件的前瞻执行

  基于硬件的前瞻执行的三种思想：
  - ✧ 动态分支预测。用来选择后续执行的指令。
  - ✧ 在控制相关的结果尚未出来之前，前瞻地执行后续指令。
  - ✧ 用动态调度对基本块的各种组合进行跨基本块的调度。

20、计算机系统结构中常用 4 个定量原理是什么？请说出他们的含义

- **以经常性事件为重点。**

  在计算机系统的设计中，对经常发生的情况，赋予它优先的处理权和资源使用权，以得到更多的总体上的改进。
- **Amdahl 定律**

  加快某部件执行速度所获得的系统性能加速比，受限于该部件在系统中所占的重要性。
- **CPU 性能公式**

  执行一个程序所需的 CPU 时间 = IC ×CPI ×时钟周期时间；IC：所执行的指令条数
- **程序的局部性原理**

  程序在执行时所访问地址的分布不是随机的，而是相对地簇聚。

21、组相联 Cache 的不命中率比相同容量直接映像 Cache 的不命中率低，由此能否得出结论：采用组相联一定能带来性能上的提高，为什么？

答：不一定。因为组相联命中率的提高是以增加命中时间为代价的，组相联需要增加多路选择开关。

22、描述基于硬件的前瞻执行的思想和实现方法。

关键思想：

允许指令乱序执行，但必须顺序确认。

基本思想：

- 对分支指令的结果进行猜测，并假设这个猜测总是对的，然后按这个猜测结果继续取、流出和执行后续的指令。
- 执行指令的结果不是写回到寄存器或存储器，而是写入一个称为再定序缓冲器 ROB（ReOrder Buffer）中 。等到相应的指令得到“确认”（commit） （即确实是应该执行的）之后，才将结果写入寄存器或存储器
- ROB 中的每一项由以下 3 个字段组成：
  - ✧ **指令类型**

指出该指令是分支指令、 store 指令或寄存器操作指令。

✧ **目标地址**

给出指令执行结果应写入的目标寄存器号（如果是 load 和 ALU 指令）或存储器单元的地址（如果 store 指令）。

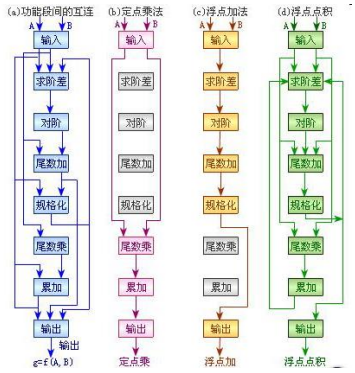
✧ **数据值**

用来保存指令前瞻执行的结果，直到指令得到确认。

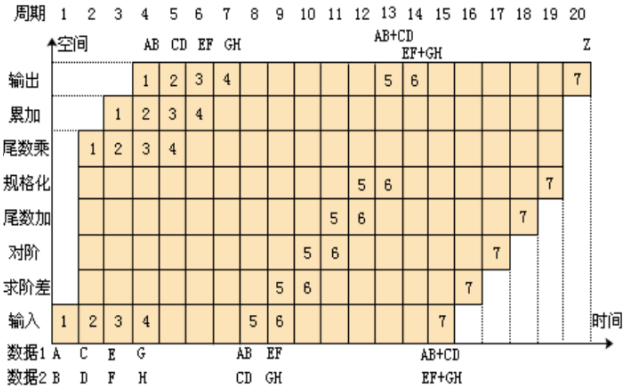
三、综合题（4 题　38 分）

1、计算机运行以下指令：

线性多功能静态流水线，输入任务是不连续的情况，画出该计算的时空图,并计算流水线的吞吐率、加速比和效率。 用 TI－ASC 计算机的多功能静态流水线计算两个向量的点积：Z＝AB＋CD＋EF＋GH



解:为了尽量减少数据相关性，充分发挥流水线的作用。计算的顺序应该是先做4个乘法：AB、CD、EF和GH，然后做两个加法AB+CD和EF+GH，最后求总的结果Z。流水线的时空图如图：



从流水线时空图中看到，用 20 个周期完成了 7 个运算。当每一个流水段的延迟时间都为△t 时，有：流水线的吞吐率：7/20△t

若采用顺序执行方式，完成一次乘法要用 4 个△t，完成一次加法要用 6 个△t，则完成全部运算要用

T0 = 4 * 4△t + 3 * 6△t = 34△t

加速比 S 为： $S=\frac{T_0}{T_k}=\frac{34\cdot\Delta t}{20\cdot\Delta t}=1\cdot70$

整个流水线共 8 段，流水线效率 E 为： $E=\frac{T_0}{k\cdot T_k}=\frac{34\cdot\Delta t}{8\times20\cdot\Delta t}=0\cdot2$

整个流水线的效率很低，其原因主要有如下四个。

○ 一是多功能流水线在做某一种运算时，总有一些流水段是空闲的;

- 二是静态流水线必须等待前-种运算全部排出流水线之后，才能重新进行连接;
- 三是题目本身存在有数据相关，当发生数据相关时，必须等待前一个运算结果产生之后，下一个运算才开始;
- 四是流水线有装入与排空部分，当输入到流水线中的任务不多时，装入与排空部分所占的比例比较大。

2、（12 分）有一个 4 段流水线，各段执行时间均为 Δ *t*，其预约表如下表所示。

	T1	T2	T3	T4	T5	T6	T7
S4				√		√	
S3			√				
S2		√					
S1	√				√		√

（1）写出禁止向量，初始冲突向量，画出流水线调度的状态有向图。（6 分）

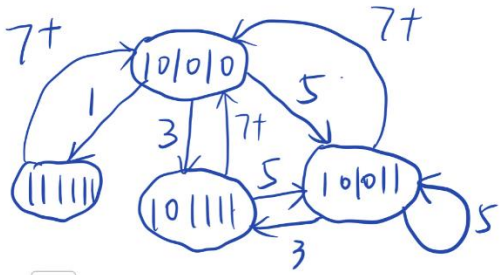
（2）求出流水线执行 8 个任务时的最优调度策略。（2 分）

（3）求出按最优调度策略连续输入 8 个任务时的流水线实际吞吐率、加速比、效率。（4 分）

解:

（1）禁止向量：（2,4,6） 初始冲突向量：101010

状态有向图如下：



（2）最优调度策略

1,7	(1+7)/2 = 4
3,7	(3+7)/2 = 5
5	5
3,5,7	(3+5+7)/3 = 5

因为 4 最小，所以最优策略选择 1,7 (引入新的任务延迟的拍数)

（3）画出时空图：

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
S1	1	2			1	2	1	2	3	4			3	4	3	4
S2		1	2							3	4					
S3			1	2							3	4				
S4				1	2	1	2					3	4	3	4	

由上面的时空图可知:8 个△t 完成了 2 个任务，则完成 8 个任务，需要 32△t

实际吞吐率：8 / (32△t)

加速比 S：(8 * 7△t) / (32△t) = 56/32 = 1.75

效率 E：56△t/(4 * 32△t) = 56/128 = 0.4375

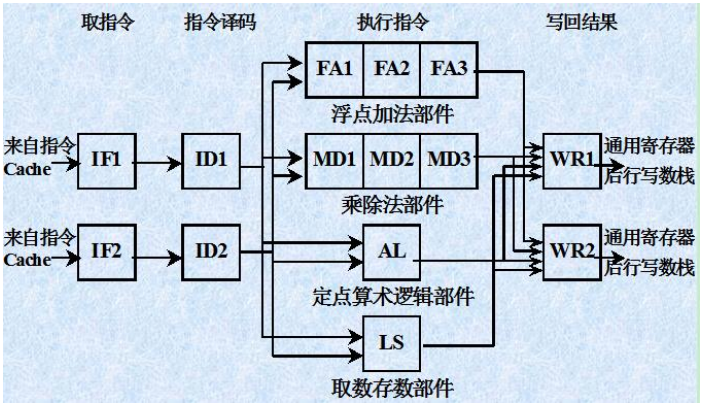
PS：新方法，不用画时空图（感谢zjh同学的公式总结）：

实际吞吐率 = $\frac{N}{S+n}$，其中N表示任务数，n表示流水线的长度，S表示最优调度策略的时钟间隔序列的前n-1项之和，例如最优调度策略为(1,7)，n = 8，则S = 1+7+1+7+1+7+1 = 25，n = 7，

所以实际吞吐率=8/(8+25) = 8/32

8-1=7项

3、超标量机的相关性问题以及调度



计算机运行以下指令：

I1：LOAD R1, A ；R1←(A)

I2：FADD R2, R1 ；R2←(R2)+(R1) //写后读，数据相关

I3：FMUL R3, R4 ；R3←(R3)×(R4)

I4：FADD R4, R5 ；R4←(R4)+(R5) //读后写 ，反相关

I5：DEC R6 ；R6←(R6)－1

I6：FMUL R6, R7 ；R6←(R6)×(R7) //写后写，输出相关

（1）请列出程序代码中可能出现的数据相关及相关类型。

（2）当程序通过下图的双发射超标量机时，请采用顺序发射乱序完成的方式画出指令流水时空图。

(流水线没有使用定向技术。)

解:

（1） **写后读，数据相关**

R1←(A)

R2←(R2)+(R1)

读后写，反相关

R3←(R3)×(R4)

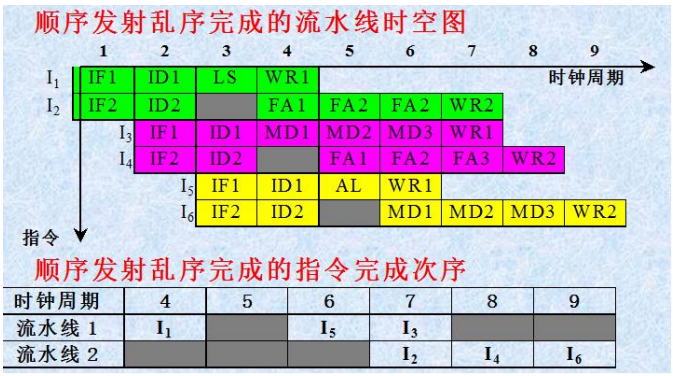
R4←(R4)+(R5)

写后写，输出相关

R6←(R6)－1

R6←(R6)×(R7)

（2）



4、CACHE 映像算法

有一个 Cache 存储器，主存有 8 块(0-7)，Cache 有 4 块(0-3)，采用组相联映像，组内块数为 2 块。采用 LRU（近期最久未使用）替换算法。(12 分，(1）题 4 分，(2）题 8 分)

(1)指出主存各块与 Cache 各块之间的映像关系。

(2)某程序运行过程中，访存的主存块地址流为：

2， 3， 4， 1， 0， 7， 5， 3， 6， 1， 5， 2， 3， 7， 1

说明该程序访存对 Cache 的块位置的使用情况，指出发生块失效且块争用的时刻，计算 Cache 命中率。

解:(答案未知，注意甄别)

1） 第 0 组：0,2,4,6

第 1 组：1,3,5,7

eg：主存第 7 块映射在 cache 的第 7 mod 2 = 1 组上

(2)

组号	0		1	
块号	0	1	2	3
初始	null	null	null	null
2	2	null	null	null
3	2	null	3	null
4	2	4	3	null
1	2	4	3	1
0	0	4	3	1
7	0	4	7	1
5	0	4	7	5
3	0	4	3	5
6	0	6	3	5
1	0	6	3	1
5	0	6	5	1
2	2	6	5	1
3	2	6	5	3
7	2	6	7	3
1	2	6	7	1

命中率：0/15 = 0

5、举三个例子说明系统中采用软件来提高性能的方法和效果。

1、用编译技术减少 Cache 失效次数

2、静态调度，通过编译器调整指令顺序。

6、举三个例子说明系统中采用硬件来提高性能的方法和效果。

1、动态调度：在程序的执行过程中，依靠专门硬件对代码进行调度，减少数据相关导致的停顿。

2、Cache

3、等

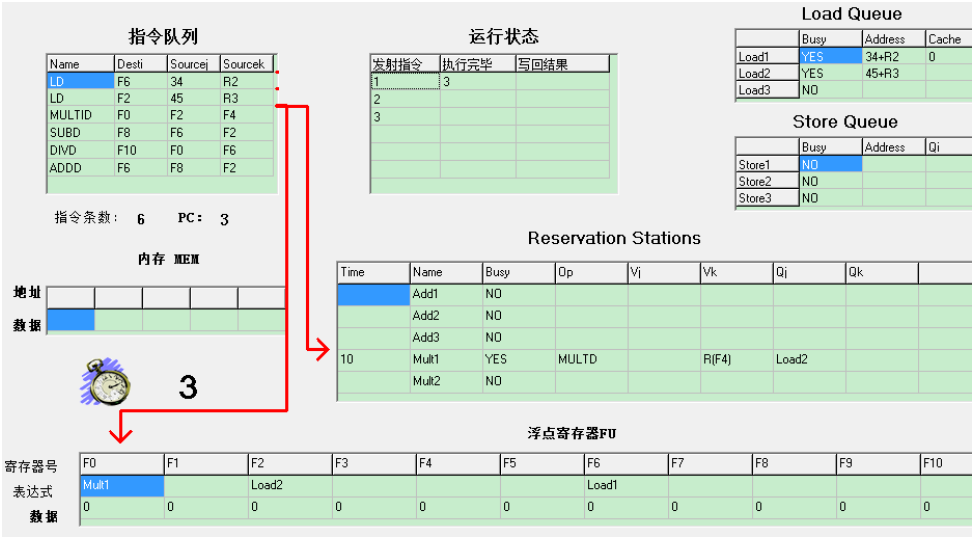
7、Tomasulo 算法的第 3 个时钟周期的指令状态，保留站状态，和寄存器结果状态如下图所示；

（其中 Op 表示现在保留站中正在工作的指令,Vj，Vk 表示已经准备好的操作数，Qj,Qk 表示已发射但未准备好的操作数）。已知 load 执行延时 2 个 cycles，add（sub）执行延时 2 个 cycles，mul 执行延时 10 个 cycles，div 执行延时 40 个 cycles。

要求：

(1)写出 tomasulo 算法的核心思想。

(2)写出第 5 个时钟周期的指令运行状态，保留站状态，和寄存器结果状态，并说明原因。



解:

(1) **核心思想：**

- 记录和检测指令相关，操作数一旦就绪就立即执行，把发生 RAW 冲突的可能性减少到最小；
- 通过寄存器换名来消除 WAR 冲突和 WAW 冲突。

(2)

- 指令运行状态：

序号	指令	D	J	K	发射指令	执行完毕	写回结果
1	LD	F6	34	R2	1	3	4
2	LD	F2	45	R3	2	4	5
3	MULTID	F0	F2	F4	3	13	14
4	SUBD	F8	F6	F2	4	6	7
5	DIVD	F10	F0	F6	5	45	46

6	**ADDD**	**F6**	**F8**	**F2**	**6**	**8**	**9**

1 号指令在第四周期执行完毕

2 号指令在第五周期开始写回结果

3 号指令在第五周期正在执行

4 号指令也正在执行

5 号指令开始发射

6 号指令还未来得及发射

○ 浮点寄存器:

寄存器	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
表达式	Mult1		45(R3)				34(R2)		Add1		
数据											

○ 保留站

Name	Busy	Op	Vj	Vk	Qj	Qk
Load1	no					
Load2	no					
Add1	Yes	Sub	Mem[34+Regs[R2]]	Mem[45+Regs[R3]]		
Add2	No					
Mult1	yes	mul	Mem[45+Regs[R3]]	Regs[F4]		
Div1	Yes	Div		Mem[34+Regs[R2]]	Mult1	

8、某台主频为 400MHz 的计算机执行标准测试程序，程序中指令类型、执行数量和平均时钟周期数如下：

指令类型	指令执行数量	平均时钟周期数
整数	45000	1
数据传送	75000	2
浮点	8000	5
分支	2000	2

求该计算机的平均 CPI、MIPS 和程序执行时间(单位：us)。

解:

(1) CPI = (45000 × 1 + 75000×2 + 8000× 5 + 20 00×2) / 1 300 00 = 1. 84

(2) **MIPS 速率 = f / CPI** = 400/1. 84 = 2 17 MIPS

(3) 程序执行时间= (45000 × 1 + 75000×2 + 8000× 5 + 20 00×2) / 400 M = 597. 5 us

9、假设机器的时钟周期为 10 纳秒, Cache 命中时间为 1 个时钟，失效时的开销为 20 个时钟周期，

(1)设失效率为 0.05,忽略写操作时的其他延迟,求机器的平均访存时间。

(2)假设通过增加 Cache 容量一倍而使失效率降低到 0.03,同时使得 Cache 命中时间增加到了 1.2 时钟周期,从平均访存时钟周期来看，这样的改动设计是否有利?

(3)如果改动后，CPU 时钟周期时间也随 Cache 的命中时间增加,从平均访存时间看，上述改动设计是否有利?

解:

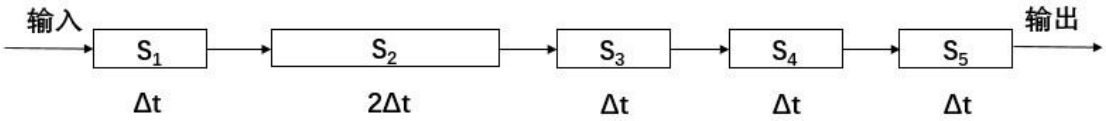
（1）(0.05 * 20 + 1) * 10 ns = 20ns

（2）(0.03 * 20 + 1.2) * 10ns = 18ns < 20ns ，这样改动有利

（3）1.8X < 20ns → x = 11.11ns

当 CPU 时钟周期时间大于 11.11ns 时，改动设计就不利，反之，有利。

10、一条各流水段执行时间不完全相等的 5 段线性流水线，假设其第 1、3、4、5 段的执行时间为Δt，第 2 段的执行时间为 2Δt，其连续执行了 5 个任务，且不考虑数据与控制冲突，则其实际效率为__5/(14Δt)__。(精确到小数点后 2 位)



- 各段时间不等的流水线的实际吞吐率：

（ Δt_j为第j段的时间，共有k个段 ）

$$TP=\frac{n}{\sum_{i=1}^{k}\Delta t_i+(n-1)\max(\Delta t_1,\Delta t_2,\cdots,\Delta t_k)}$$

- 流水线的最大吞吐率为

$$TP_{\max}=\frac{1}{\max(\Delta t_1,\Delta t_2,\cdots\Delta t_k)}$$

解:

由公式：TP = 5/(4Δt+2Δt+4 * 2Δt) = 5/(14Δt)

11、假设分支目标缓冲的命中率为 95%，程序中无条件转移指令的比例为 5%，没有无条件转移指令的程序的 CPI 值为 1。假设分支目标缓冲中包含分支目标指令，允许无条件转移指令进入分支目标缓冲，则程序的 CPI 值为？假设原来的 CPI=1.2。(精确到小数点后 2 位)

解:

无条件分支指令的特点是只要执行肯定分支成功。因此，对于进入分支目标缓冲器的无条件分支指令，分支预测的精度 100%，也就不会带来分支延迟。而没有进入分支目标缓冲器的无条件分支指令会带来一定分支延迟。首先要求出一条无条件分支指令的分支延迟是多少，不妨设为 x 个时钟周期。

$CPI_{原来} = CPI_{基本} + CPI_{无条件分支延迟} = 1 + 5\% x = 1.2$ 得到 x = 4

因此允许无条件分支指令进入分支目标缓冲后，

$CPI_{改变} = CPI_{基本} + 5\% * 100\% * (1 - 95\%) * 4 = 1 + 0.05 * 4 * 0.05 = 1.01$

12、假设有一条长流水线，仅仅对条件转移指令使用分支目标缓冲。假设分支预测错误的开销为 4 个时钟周期，缓冲不命中的开销为 3 个时钟周期。假设命中率为 95%，预测精度

为 95%，分支频率为 10%，没有分支的基本 CPI 为 1。程序执行的 CPI 为 ？（精确到小数点后 3 位）

解:

CPI = CPI 基本 + 分支延迟 = 1 + 0.1 * (0.95 * (1 – 0.95) * 4 + (1 – 0.95) * 3) = 1.034
分支延迟 = 分支指令的占比 *(命中率 * 分支预测失败率 * 失败开销 + 不命中率 * 不命中开销)

13、考虑某两级 cache，第一级为 L1，第二级为 L2，两级 cache 的全局不命中率分别是 5%和 2%，假设 L2 的命中时间是 10 个时钟周期，L2 的不命中开销是 200 时钟周期，L1 的命中时间是 1 个时钟周期，平均每条指令访存 1.4 次。问：每条指令的平均停顿时间是多少个时钟周期?

解:

一级：局部不命中率 = 全局不命中率 = 5%
二级：局部不命中率 = 全局不命中率 / 一级局部不命中率 = 2% / 5% = 40%
T 停 =1.4 * [1 + 0.05 * (0.4 * 200 +10)]= 7.7

14、某个程序共访问存储器 1 000 000 次，该程序在某个系统中运行，系统中 Cache 的不命中率为 7% ，其中，强制性不命中和容量不命中各占 25%，冲突不命中占 50%。问:
(1)当允许对该 Cache 所作的唯一改变是提高相联度时，此时期望能够消除的最大不命中次数是多少?
(2)当允许能够同时提高 Cache 的容量大小和相联度时，此时期望能够消除的最大不命中次数是多少?
说明原因并给出计算过程和结果。

解:

(1) 提高 cache 的相联度，可以降低冲突不命中的次数，但不会影响强制性不命中和容量不命中的次数。已知 cache 的不命中率为 7%，程序共访问存储器 1 000 000 次，所以总的不命中次数为 70 000， 其中 50%为冲突不命中的次数。因此，提高 cache 的相联度能够消除的最大不命中次数是:
70000 x 50%= 35 000。
(2)当同时提高 cache 的容量大小和相联度时，可以消除容量不命中和冲突不命中的次数。而这两种不命中占总不命中次数的 75%，所以，能够消除的最大不命中次数是:
70000 X 75% = 52 500。